Wei Qi Yan

Computational Methods for Deep Learning

Theory, Algorithms, and Implementations

Second Edition

Springer

Texts in Computer Science

Series Editor

Orit Hazzan , Faculty of Education in Technology and Science, Technion—Israel Institute of Technology, Haifa, Israel

Titles in this series now included in the Thomson Reuters Book Citation Index!

'Texts in Computer Science' (TCS) delivers high-quality instructional content for undergraduates and graduates in all areas of computing and information science, with a strong emphasis on core foundational and theoretical material but inclusive of some prominent applications-related content. TCS books should be reasonably self-contained and aim to provide students with modern and clear accounts of topics ranging across the computing curriculum. As a result, the books are ideal for semester courses or for individual self-study in cases where people need to expand their knowledge. All texts are authored by established experts in their fields, reviewed internally and by the series editors, and provide numerous examples, problems, and other pedagogical tools; many contain fully worked solutions.

The TCS series is comprised of high-quality, self-contained books that have broad and comprehensive coverage and are generally in hardback format and sometimes contain color. For undergraduate textbooks that are likely to be more brief and modular in their approach, require only black and white, and are under 275 pages, Springer offers the flexibly designed Undergraduate Topics in Computer Science series, to which we refer potential authors.

Wei Qi Yan

Computational Methods for Deep Learning

Theory, Algorithms, and Implementations

Second Edition

Springer

Wei Qi Yan
Auckland University of Technology
Auckland, New Zealand

Preface

This book has been drafted based on my lectures and seminars from recent years for postgraduate students at Auckland University of Technology (AUT), New Zealand. We have integrated materials on deep learning, machine learning, and computational mathematics, refining the contents to publish this book. Our aim is to provide a resource that benefits postgraduate students, particularly those working on their theses, by sharing our research outputs and teaching work to enhance their projects.

In this book, we organize our material and present the story of deep learning in a progression from easy to difficult concepts in mathematics. We have structured the content with a focus on knowledge transfer from the perspective of machine intelligence. We begin by explaining artificial neural networks, including neuron design and activation functions. We then delve into the mechanics of deep learning using advanced mathematical concepts. At the end of each chapter, we emphasize the practical implementation of deep learning algorithms using Python-based platforms and the latest MATLAB toolboxes. Additionally, we provide a list of questions for reflection and discussion.

Before reading this book, we strongly encourage our readers to have a solid foundation in postgraduate mathematics, including subjects such as basic algebra, functional analysis, graphical models, and other fundamental topics like mathematical analysis, linear algebra, probability theory, mathematical statistics, optimization theory, computational methods, differential geometry, manifold, and information theory. Developing computational knowledge will not only help readers understand this book but also enable them to engage with relevant journal articles and conference papers in the field of deep learning.

This book is written for research students, engineers, computer scientists, and anyone interested in computational methods of deep learning for both theoretical analysis and practical applications. Additionally, it is relevant for researchers in fields such as machine intelligence, pattern analysis, computer vision, natural language processing (NLP), computational linguistics, robotics, and control theory.

Auckland, New Zealand   Wei Qi Yan
June 2023

Acknowledgements

The first edition of this book has been published in 2021. In the past two years, we have endowed all aspects of deep learning methods to make the book full and perfect. Following the feedback from our readers and audiences, the author has further omitted mistakes and typos, detailed the mathematical descriptions, flowcharts, data structures, pipelines, deployments, and algorithms, as well as updated each chapter with the latest references. The first edition of this book emphasized computational methods in computational mathematics related to convolutional neural networks (ConvNets or CNNs) and recurrent neural networks (RNNs), reinforcement learning, and ensemble learning.

The second edition of this book showcases our collected datasets, programming language R, control theory, transformer models, and generative pre-trained transformer models (GPT), graph neural networks (GNN), and knowledge distillation in deep learning. We integrate the latest development of algorithms and large deep learning models into this book for matching today's research trend. The second edition of this book demonstrates how computational methods and algorithms are playing a powerful role as the energetic engine in this era of artificial intelligence (AI).

Thanks to our peer colleagues and students whose materials were referenced and who have given invaluable comments on this book; special thanks to my supervised students: Mr. J. Wang, Dr. Y. Zhang, Dr. J. Lu, Mr. D. Shen, Mr. K. Zheng, Ms. Y. Ren, Mr. R. Li, Mr. P. Li, Mr. Z. Liu, Ms. Y. Shen, Ms. H. Wang, Mr. C. Xin, Ms. Q. Zhang, Mr. C. Liu, Ms. B. Xiao, Ms. X. Liu, Mr. C. Song, Mr. X. Ma, Mr. S. Sun, Ms. Y. Fu, Ms. N. An, Ms. L. Zhang, Dr. Q. Gu, Mr. Z. Yu, Mr. K. Zhao, Ms. Y. Wang, Mr. S. Liang, Mr. J. Xing, Mr. C. Li, Mr. J. Liu, Ms. X. Cao, Mr. X. Gao, S. Z. Luo, Mr. Y. Liu, Mr. C. Liang, Mr. Y. Xia, my colleagues Dr. X. Wang, Dr. Y. Li, Dr. M. Nguyen, Dr. P. Nand, Dr. S. Zhou, Prof. C. Pan, Prof. Y. Li, Prof. R. Klette, and Prof. N. Kasabov.

Many thanks to our colleagues who have translated the first edition of this book into Chinese.

Special thanks to MathWorks® who listed this book in the MATLAB website.

Contents

About the Author

Wei Qi Yan is the Director of the Institute of Robotics and Vision (IoRV) at Auckland University of Technology (AUT), New Zealand; his expertise covers intelligent surveillance, deep learning, computer vision, and multimedia computing. Dr. Yan has served as the Editor-in-Chief (EIC) of the *International Journal of Digital Crime and Forensics* (IJDCF) and has worked as an exchange computer scientist between the Royal Society Te Apārangi (RSNZ) and the Chinese Academy of Sciences (CAS) in China. He is a guest (adjunct) professor at the Chinese Academy of Sciences and has been a visiting professor at the University of Auckland in New Zealand and the National University of Singapore. In 2022, Dr. Yan was recognized as one of the world's top 2% cited scientists by Stanford University. He currently holds the position of chair of the ACM Multimedia Chapter of New Zealand and a member of the ACM, and a senior member of the IEEE, as well as a TC member of the IEEE.

Acronyms

ACM	Association for Computing Machinery
AdaBoost	Adaptive Boosting
AI	Artificial Intelligence
ANN	Artificial Neural Networks
ASCII	American Standard Code for Information Interchange
Bagging	Bootstrap Aggregating
CapsNet	Capsule Neural Network
CNN	Convolutional Neural Network
ConvLSTM	Convolutional Long Short-Term Memory
ConvNet	Convolutional Neural Network
CVPR	International Conference on Computer Vision and Pattern Recognition
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
DETR	Detection Transformer
DMRF	Deep Markov Random Field
DNN	Deep Neural Network
DQN	Deep Q-Network
EKF	Extended Kalman Filter
FAIR	Facebook AI Research
FC	Layers Fully Connected Layers
FCN	Fully Connected Network
FCNN	Fully Connected Neural Network
FN	False Negative
FP	False Positive
FRU	Fully Gated Unit
GAN	Generative Adversarial Network
GNN	Graph Neural Network
GPT	Generative Pre-training Transformer
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HOG	Histogram of Oriented Gradients
ICCV	International Conference on Computer Vision
LBP	Local Binary Patterns

LSTM	Long Short-Term Memory
MC	Monte Carlo (method)
MCNN	Multichannel Convolutional Neural Network
MDP	Markov Decision Process
MGU	Minimal Gated Unit
MNIST	Modified NIST Database
MRF	Markov Random Field
MRI	Magnetic Resonance Imaging
MRP	Markov Random Process
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
NPU	Neural Processing Unit
PCA	Principal Component Analysis
PDF	Probability Density Function
PPO	Proximal Policy Optimization
R-CNN	Region-Based CNN
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
RNN	Recurrent Neural Network
ROI	Region of Interest
RPN	Region Proposal Network
SARSA	State–Action–Reward–State–Action
SGD	Stochastic Gradient Descent
SSD	Single Shot Multibox Detector
STGCN	Spatial–Temporal Graph Convolutional Network
SVM	Support Vector Machine
TD	Temporal Difference
TN	True Negative
TP	True Positive
TPU	Tensor Processing Unit
VAE	Variational Autoencoder
WCSS	Within-Cluster Sum of Squares
YOLO	You Only Look Once

Symbols

$\sigma(\cdot)$	Activation function
$\arg\max(\cdot)$	Argument of the maxima
$p(\cdot\|\cdot)$	Conditional probability
$J(\cdot)$	Cost function
$\underline{\underline{\Delta}}$	Definitions
$\frac{d\,f(x)}{dx}$ or $f'(x)$	Derivative
$det(\cdot)$	Determinant
$(b_i)_{n\times 1}$	Element b_i of vector $\mathbf{b}_{n\times 1}$
$(w_{ij})_{m\times n}$	Element w_{ij} of $m\times n$ matrix $\mathbf{W}_{m\times n}$
$\mathscr{E}$	Euclidean space
$\exists$	Exist
$\mathbf{E}(\cdot)$	Expected value function
$\exp(\cdot)$	Exponential function
$\mathbf{C}^1$	First-order parametric continuity
$\forall$	For all
$\overline{1,n}$	From 1 to n, i.e., $1, 2, \ldots, n$
$\mathbf{C}$	Function continuity
$\mathbf{N}(\cdot)$	Gaussian or normal distribution
$\tanh(\cdot)$	Hyperbolic tangent function
$\mathbf{C}^\infty$	Infinite continuity
∞	Infinity
$<\cdot>$	Inner or dot product
$\int$	Integral
$\cap$	Intersection of sets
$\|\cdot\|_0$	$\mathbf{L}^0$ Norm
$\|\cdot\|_1$	$\mathbf{L}^1$ Norm
$\|\cdot\|_2$	$\mathbf{L}^2$ Norm
$\|\cdot\|_\infty$	$\mathbf{L}^\infty$ Norm
$\log(\cdot)$	Logarithm base 10
$\mathscr{L}$	Loss function
$\mapsto$	Mapping
$\mathbf{W}^\mathrm{T}$	Matrix $\mathbf{W}$ transpose
$\max(\cdot)$	Max function
μ	Mean

$\in$	Member
$\ln(\cdot)$	Natural logarithm
$\lVert \cdot \rVert$	Norm
$\frac{\partial f}{\partial x}$	Partial derivative
$\perp$	Perpendicular
$\pm$	Plus or minus
$\mathbf{P}$	Point
Π	Product
$\subset$	Proper subset
$\mathbf{C}^2$	Second-order parametric continuity
$\mathbf{S}$	Set
$\mathscr{C}$	Set of complex numbers
$\mathscr{Z}$	Set of integer numbers
$\mathscr{Z}^+$	Set of positive integer numbers
$\mathscr{R}$	Set of real numbers
$\mathbf{b}$	Shift vector
$sgn(\cdot)$	Sign function
$\subseteq$	Subset equal
$\sum$	Sum
$\mathscr{T}$	Tensor space
$\cup$	Union of sets
$\boldsymbol{\sigma}$	Variance
$\mathbf{b}^{\mathrm{T}}$	Vector transpose
$\mathbf{W}$	Weight matrix

Introduction

<div style="text-align: right">**1**</div>

This chapter covers the fundamentals of deep learning, therefore, we present relevant knowledge in chronological order so as to fully introduce the history of deep learning development; meanwhile, we review how to use MATLAB, TensorFlow, software R, and WEKA from New Zealand, etc., as typical platforms for developing deep learning applications. In this chapter, we expect our readers could understand the definitions and concepts well, and grasp the knowledge points of deep learning implementations. Specifically, we will provide an overview of the core ideas, and demonstrate our advanced understandings of the state-of-the-art theory and practice of deep learning and machine intelligence.

1.1 Deep Learning as a Prominent Component of AI

Artificial Intelligence (AI) is a synonym for Machine Intelligence or Computational Intelligence which relies on computable algorithms in computational mathematics and applied mathematics for decision-making, following the methodology having the components such as mathematical definitions, concepts, equations, computational algorithms, Boolean logic, coding and functions, evaluations, and comparisons, especially in evolutionary learning and optimization. Deep learning, inspired by bioinformatics in computer science, has been thought as the jewel in the AI crown which attained remarkable results comparable to or surpassing human expert performance. Nowadays, deep learning has been successfully employed to robotics, autonomous vehicles, speech recognition, computational linguistics, natural language processing, biometrics, medical image analysis, etc.

Deep learning turns up after a number of years of evolution of information technology such as sensor networks, big data, World Wide Web(WWW), mobile technology, and supercomputing. Sensor networks capture and provide tremendous data for us to fully touch and understand the cyberspace; cloud computing accommodates storage space for tremendous data. The big data could be visualized and analyzed by

using various methods related to data volume, velocity, and variety, while mobile phones are prone to viewing or operating the data processing using our thumbs. After knowledge accumulation and computational evolution in past decades, deep learning emerges and becomes an iconic landmark of this digital era. Deep learning is thought as the historical necessity of modern computing technology.

Deep learning is a redhot technology at present, which is thought as a core part in artificial neural networks (ANNs) and artificial intelligence (AI). Deep learning is also named as deep neural networks (DNNs), which is the core content of AI. The latest development of AI has been reflected in the surge of deep learning.

We eyewitnessed that ACM 2018 Turing Award has been conferred to a trio of computer scientists: Yoshua Bengio, Geoffrey Hinton, and Yann LeCun for their conceptual and engineering breakthroughs that have made deep neural networks as a critical component of computing in 2019. Turing Award, named after Alan Turing (1912–1954), which usually is thought as the Nobel Prize of Computing, is an annual prize given by the Association for Computing Machinery (ACM) to the persons selected for contributions of lasting and major technical importance to the computer field.

The articles published by this group of computer science scientists in the journals Nature [73] and Science [52] have shown their distinctive contributions to the field of deep learning. The publications have been regarded as the classical work of this area. The book entitled Deep Learning published by the MIT Press [43] inspired tremendous number of young scientists, students, engineers, and enthusiasts.

As stated in the book Deep Learning, pertaining to deep neural networks, usually the input data is imported and the outputs from those activation functions for simulating the stimuli of neurons are calculated; the activation functions [61] include ridge activation functions like ReLU function, sigmoid function, and logistic function; and radial activation functions like Gaussian function, multiquadratics, polyharmonic splines, etc. Activation functions are usually nonlinear, continuous, and differentiable.

Transfer function comes from the name transformation which is employed for the purposes of transferring human brain signals, i.e., from input nodes to the output. On the other hand, the activation function checks the output of neurons whether it meets a threshold and outputs between zero and one. As a result, the difference between imported data and exported data of a neural network should be minimized. In 2011, ReLU activation function

$$y = x^+ = \max(x, 0) \in \mathscr{R}, \tag{1.1}$$

where $x, y \in \mathscr{R}$, $x \in (-\infty, +\infty)$, was found much better than Tanh activation function or hyperbolic tangent function

$$y = \tanh(x) \in \mathscr{R}, \tag{1.2}$$

where $x, y \in \mathscr{R}$, $x \in (-\infty, +\infty)$ in resolving the vanishing gradient problem, which paves the way for further development of deeper neural networks [41]. The shapes of these two activation functions are shown in Fig. 1.1.

Usually, we use backpropagation including forward pass and backward pass to adjust the weights of convolutional neural works. The algorithms of deep neural
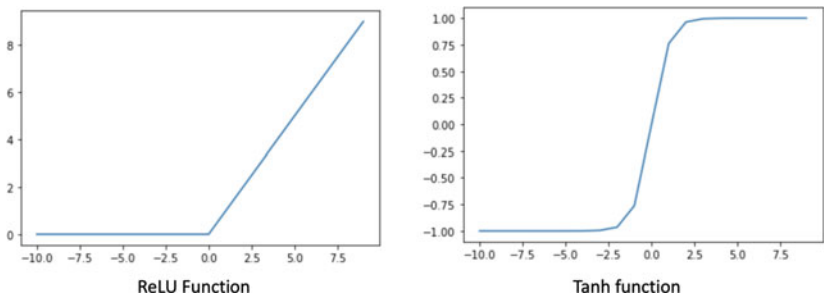
Fig. 1.1 ReLU function and Tanh function

Fig. 1.2 An example of dataset MNIST

networks from machine learning have been grouped into supervised learning and unsupervised learning. Supervised learning is related to labeled data or ground truth. Public websites such as NIST (i.e., National Institute of Standards and Technology) also provide verified dataset MNIST (i.e., Modified NIST database) for training and testing deep learning models [71]. The MNIST database contains 60,000 training images and 10,000 testing images. MNIST dataset is a large database of handwritten digits. The black and white images from NIST were normalized to fit into a 28×28 pixel bounding box and anti-aliased as shown in Fig. 1.2 (http://yann.lecun.com/exdb/mnist/).

On the contrary, unsupervised learning [138] is subject to similarity functions; for instance, clustering is a typical unsupervised learning. In deep learning, the unsupervised learning methods include principal component analysis (PCA), autoencoder [181], manifold learning [1], etc. The unsupervised learning has been applied to dimensionality reduction or data embedding. Dimensionality reduction, or dimension reduction [52], is the transformation of data from a high-dimensional space into
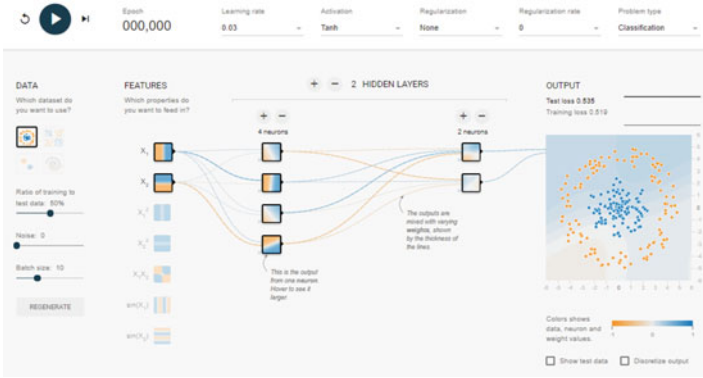
Fig. 1.3 The playground software Tink for understanding neural networks

a low-dimensional space so that the low-dimensional representation retains principle properties of the original data, ideally close to its intrinsic dimension.

The famous playground software Tinker has been applied to understand artificial neural networks for beginners as shown in Fig. 1.3 which renders how the deep neural networks work; the relevant parameters and outcomes are explicitly linked to one web page. Four types of exemplar datasets are provided for demonstrations. Any adjustments of the input parameters will reflect the changes of visualized results of classification and regression. The network architecture can be manually adjusted; the nodes of the network could be freely added or removed by a network designer. L^1 and L^2 regularizations have been provided for optimization. The other options include four activation functions, learning rates, and epoch numbers.

In the course of backpropagation, we need to calculate the stochastic gradient based on optimization; usually SGD (i.e., Stochastic Gradient Descent) will be adopted. SGD [78] is an iterative method for optimizing an objective function with suitable smoothness properties (e.g., differentiable or subdifferentiable). Thus, the chain rule for function differentials is required. Meanwhile, mini-batch has been taken into account for various parametric optimizations. In pseudocode, stochastic gradient descent can be presented as Algorithm 1.

Data: Initial weight: $\mathbf{w}$, Learning rate: η
Result: Optimized weight: $\mathbf{w}$
Choose an initial vector of parameters $\mathbf{w}$ and learning rate η;
Repeat until an approximate minimum is obtained by using the gradient $\nabla F(\mathbf{w})$;
　Randomly shuffle samples in the training set;
　　For $i = 1, 2, 3, \cdots, n$ do:
　　$\mathbf{w} := \mathbf{w} - \eta \nabla F(\mathbf{w})$;
　　　　Algorithm 1: Stochastic gradient descent (SGD) algorithm

In deep learning, the hottest research topics now are distributed in the fields of manifold learning [1], reinforcement learning [143], generative adversarial network (GAN) [42], Transformer [106], graph neural network (GNN) [134], etc. These typical approaches have been applied to automation or automatic control, robotics, machine vision, natural language processing, computational linguistics, intelligent surveillance, recommender or recommendation system, etc. In the remaining part of this book, we will emphasize these computational methods one by one.

Deep learning is being updated rather rapidly. The site Github.com provides relevant source codes and datasets for various projects. From a technological perspective, there are two very popular software platforms: MATLAB and Python-based TensorFlow which could be applied to implement deep learning projects intuitively. Software R [123] is expected to be a free environment for statistical computing and computer graphics if other software is not available.

1.2   Theory and Foundations of Deep Learning

Deep learning is also called deep neural networks which is originated from modeling biological vision and brain-inspired information processing. Deep learning is one part of machine learning or machine intelligence. AlexNet was the first model of deep learning, which has been applied to handwriting recognition with the famous MNIST dataset, i.e., AlexNet was designed by Alex Krizhevsky and published with Ilya Sutskever and Geoffrey Hinton. AlexNet was competed with other deep learning models in [132] Large Scale Visual Recognition Challenge (ILSVRC) on September 2012 [66]. The network achieved a top 5 error rate 15.30%, lower than that of the runner up [67, 68].

ImageNet is an image database which was designed for use in visual object recognition and organized according to the WordNet hierarchy from Princeton University (US); each node of the hierarchy is depicted by hundreds and thousands of images. More than 14 million images have been annotated, and corresponding bounding boxes are also provided. ImageNet contains more than 20,000 object classes. Since 2010, the ImageNet project runs an annual software contest, which is called ILSVRC, where software programs compete to correctly classify and detect objects and scenes. The challenge uses a "trimmed" list of one thousand non-overlapping classes.

Deep learning is closely related to mathematics, especially optimization, graph theory, numerical analysis, functional analysis, probability theory and mathematical statistics, information theory, etc. These subjects provide the analysis for neural network models. Usually, when we measure a neural network or evaluate an algorithm, we take into consideration its error analysis as well as robustness, stability, and convergence in numerical analysis [142]. This ensures the algorithms to have few errors in practical applications.

In deep learning, we take advantage of gradient descent to update the parameters of deep learning models [15]. Gradient descent is the first-order iterative optimization algorithm for finding the local minimum of a function. For example, gradient

descent is harnessed to solve a system of linear equations as a quadratic minimization problem, e.g., using linear least squares. The solution of the linear system in Eq. (1.3):

$$\mathbf{A}\mathbf{x} - \mathbf{b} = 0 \tag{1.3}$$

is defined as minimizing the function as in Eq. (1.4):

$$F(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2. \tag{1.4}$$

In linear least squares for real matrix $\mathbf{A}$ and vector $\mathbf{b}$, Euclidean distance is applied as in Eq. (1.5):

$$\nabla F(\mathbf{x}) = 2\mathbf{A}^\top (\mathbf{A}\mathbf{x} - \mathbf{b}). \tag{1.5}$$

With this observation in mind, one starts with a guess $\mathbf{x}_0$ for a local minimum of F, and considers the sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$, described as in Eq. (1.6):

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma \nabla F(\mathbf{x}_n), n \geq 0, \tag{1.6}$$

where $\nabla F(\cdot)$ is the gradient, $\gamma \in \mathscr{R}^+$ is small enough $|\gamma| < 1$, and the step size γ is allowed to be adjusted at each iteration.

If we have a loss or cost function $F(\mathbf{w})$ that needs to be minimized, the gradient descent tells us to update the weights in the direction of the steepest descent in $F(\mathbf{w})$; the weight decay equation is shown as Eq. (1.7):

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \gamma \nabla F(\mathbf{w}_n), \ n \geq 0, \tag{1.7}$$

where γ is learning rate, and $\mathbf{w}_n$ represents weights of deep neural networks. In numerical analysis [142], the termination conditions for the iterations in Eq. (1.7) are a preset number of loops or running time as well as a given resultant estimation and the errors between two adjacent loops. The termination condition is an expression or a mathematical equation that consists of variables, constants, or operators that define movement.

In deep learning, our focus is still on the optimization problems such as solution existence, error analysis, stability, robustness, and convergence for weight decay like most of the existing ones in computational methods. These two prominent problems in deep learning are related to the vanishing gradient problem and the exploding gradient problem.

At present, vanishing gradient problem and exploding gradient problem in SGD could be resolved by making use of multilevel hierarchy of networks, restricted Boltzmann machine, generative model, long short-term memory (LSTM), residual networks (ResNets) while exploding gradient problems [19] could be eschewed by using redesigned networks, ReLU activation functions, LSTM in RNN, gradient clipping, weight regularization, etc.

Regularization is applied to avoid exploding gradient problem and vanishing gradient problem [43, 152]. Regularization is defined as a modification to deep learning algorithms that is to reduce its generalization errors. Regularization assists us to

reduce overfitting and drive the weights to lower errors. The regularized objective function is shown as in Eq. (1.8):

$$\hat{J}(\theta; \mathbf{X}, \mathbf{y}) = J(\theta; \mathbf{X}, \mathbf{y}) + \alpha \cdot \Omega(\theta), \qquad (1.8)$$

where $\alpha \in [0, \infty)$ is a hyperparameter or regularization rate; θ denotes all of the latent parameters, and $\Omega(\cdot)$ is the latent function. The optimized parameter θ^* is obtained by using Eq. (1.9):

$$\theta^* = \arg\min_{\theta} \nabla_{\theta} \hat{J}(\theta; \mathbf{X}, \mathbf{y}). \qquad (1.9)$$

The typical regularizations [152] include Tikhonov regularization, sparse regularization, Lagrangian regularization, etc. Tikhonov regularization or $\mathbf{L}^2$ regularization is shown as in Eq. (1.10):

$$\Omega(\theta) = \frac{1}{2}\|\mathbf{w}\|_2^2. \qquad (1.10)$$

Therefore,

$$\hat{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = J(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \frac{\alpha}{2}\mathbf{w}^\top \mathbf{w}. \qquad (1.11)$$

The gradient is shown as Eq. (1.12):

$$\nabla_{\mathbf{w}} \hat{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \nabla_{\mathbf{w}} J(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \cdot \mathbf{w}. \qquad (1.12)$$

Update the weights as shown in Eq. (1.13):

$$\mathbf{w} \leftarrow \mathbf{w} - \varepsilon \cdot \nabla_{\mathbf{w}} \hat{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}), \varepsilon \in (0, 1). \qquad (1.13)$$

Sparse regularization or $\mathbf{L}^1$ regularization refers to Eq. (1.14):

$$\Omega(\theta) = \|\mathbf{w}\|_1 = \sum_i |\omega_i|. \qquad (1.14)$$

$\mathbf{L}^1$ regularization is shown as Eq. (1.15):

$$\hat{\mathbf{J}}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha\|\mathbf{w}\|_1 + J(\mathbf{w}; \mathbf{X}, \mathbf{y}). \qquad (1.15)$$

With regard to the gradients in Eqs. (1.16) and (1.17):

$$\nabla_{\omega}\hat{\mathbf{J}}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha \cdot Sign(\mathbf{w}) + \nabla_{\omega} J(\mathbf{w}; \mathbf{X}, \mathbf{y}), \qquad (1.16)$$

and

$$\mathbf{w} \leftarrow \mathbf{w} - \varepsilon \cdot \nabla_{\mathbf{w}} \hat{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}), \varepsilon \in (0, 1). \qquad (1.17)$$

Lagrangian regularization means a generalized Lagrange function (or multiplier) and a constant k satisfies Eq. (1.18):

$$\mathcal{L}(\theta, \alpha; X, y) = J(\theta; X, y) + \alpha \cdot (\Omega(\theta) - k). \qquad (1.18)$$

The solution is shown as Eq. (1.19):

$$\theta^* = \arg\min_{\theta} \mathscr{L}(\theta, \alpha). \tag{1.19}$$

If we have a fixed α^* as shown in Eq. (1.20), then

$$\theta^* = \arg\min_{\theta} \mathscr{L}(\theta, \alpha^*) = \arg\min_{\theta} J(\theta; X, y) + \alpha^* \cdot \arg\min_{\theta} (\Omega(\theta) - k). \tag{1.20}$$

While we seek the maximum in weight decay, we cannot guarantee all the weights exist that could be found directly. But after the regularization, we are able to better obtain the peak point and solve the tough problems if we use stochastic gradient descent (SGD).

The mathematical regularization has a plenty of merits; by using increased bias for reduced variance, regularization can reduce overfitting and drive the weights to lower values. The regularization has shown the performance as effective as dropout.

1.3 The Chronicle of Deep Learning

Deep learning has shown its effectiveness and superiority for resolving practical problems. It has been particularly successful in surpassing human visual system in visual object detection and recognition, image segmentation, speech recognition, natural language processing, and robot control if the technology related to big data is employed. We summarize the timeline of deep learning progress in Table 1.1.

In 1995, CNN (i.e., convolutional neural network, or ConvNet) as a typical neural network has been employed for postcode recognition [69,70,145] on an envelope or handwriting recognition on a bank check. CNNs assist us to find ROI (i.e., region of interest) and salient regions, which are based on emulating the mechanism of our human neural system. The end-to-end structure and fine-tuning merits inspire us in recognizing tiny objects with details at the pixel level [74]. Typically, GoogLeNet, a pioneering 7-level convolutional neural network created by LeCun Yann et al. in 1998 [71] that classifies digits, was applied by numerous banks to recognize handwritten numbers on checks digitized and normalized in 32×32 images.

In 1997, the AdaBoost algorithm was proposed for boosting a strong classifier from a weak learner [30,118]. This makes ensemble learning one of the most important parts in machine learning [5] or deep learning [43]. Ensemble learning methods, like Bayes optimal classifier, Bayesian model averaging, Bayesian model combination, etc. have been employed for enhancing the weak learners to stronger.

Random forest is an ensemble learning method for classification and regression by constructing a multitude of decision trees at training time and outputting the class that is the mode of classification with the regression of individual trees. Random forest [65] is based on decision trees. A random forest [65] as shown in Fig. 1.4 is formed if multiple trees are ensembled together [44]. Decision trees usually are employed for decision-making [5].

Since 1995, SVM (i.e., support vector machine) [176] has become a very popular machine learning algorithm for pattern classification based on specific features as

Table 1.1 The timeline of deep learning progress

Years	Acronyms	Descriptions
2023	YOLOv8	You only look once
2022	ChatGPT	OpenAI generative pre-trained transformer
2022	YOLOv7	You only look once
2022	YOLOv6	You only look once
2021	ViT	Vision transformer
2020	YOLOv5	You only look once
2020	YOLOv4	You only look once
2020	GPT-3	OpenAI text-generating language model
2020	DERTR	Object detection transformers
2019	ACM Turing Award	
2018	YOLOv3	You only look once
2018	GPT	OpenAI text-generating language model
2017	YOLO9000	You only look once
2017	YOLOv2	You only look once
2018	Transformer	
2016	YOLO	You only look once
2015	SSD	Single shot multibox detector
2015	Faster R-CNN	Region-based CNN
2015	Fast R-CNN	Region-based CNN
2014	GAN	Generative adversarial networks
2013	R-CNN	Region-based CNN
2012	AlexNet	A CNN architecture designed by Krizhevsky et al.
2009	GNN	Graph neural network
2006	DBN	Deep belief network
2001	RF	Random forest
1995	CNN	Convolutional neural network

well as hyperplane and hyperparameters [5]. Different from SVM [176] in machine learning, deep learning algorithms take multiple classes into account; the class with the highest probability is thought as the output of a neural network [179]. Deep learning [43] is based on the end-to-end framework of neural networks, which is a well-designed subject not only in programming and data collection but also in mathematical theories and network structures.
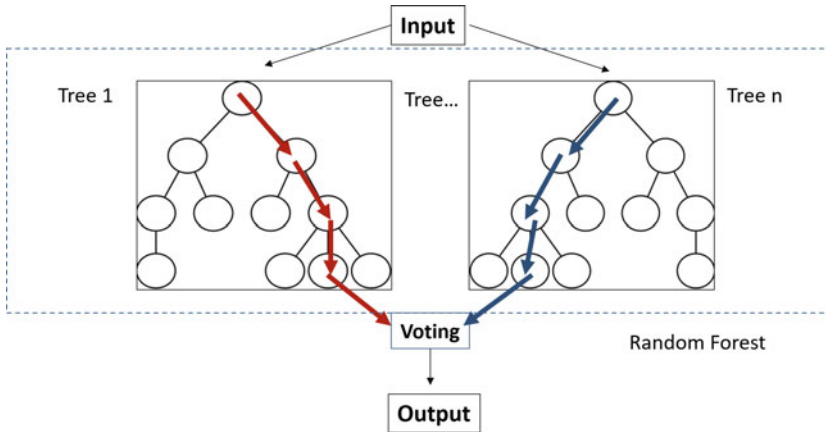
Fig. 1.4 A random forest

Deep belief network (DBN) is a directed network [51,133], where the edges and nodes have different weights; on the other hand, deep Markov random field [17] is an undirected network, where all the cliques are connected and all edges are bidirectional. A deep Boltzmann machine (DBM) is a type of binary pairwise Markov random field (i.e., undirected probabilistic graphical model) with multiple layers of hidden random variables. It is a network of symmetrically coupled stochastic binary units [3,31]. DBM has been successfully applied to pattern classification, regression, and time series analysis.

AlexNet is an early simple neural network [68]. AlexNet contains eight layers; the first five ones are convolutional layers, followed by max pooling layers, and the last three ones are fully connected layers. As a milestone of machine learning and machine intelligence, AlexNet won the ImageNet challenge 2012. In its further improved version, deep learning surpasses our human visual system in the test pertaining to computer vision.

AlexNet was implemented by using MATLAB in its early deep learning toolbox and has been further developed as an example for transfer learning [119]. In MAT-LAB, AlexNet is trained based on more than a million of images from the ImageNet dataset. The network is eight layers deep that can classify images into 1,000 classes of visual classes, including keyboards, mouses, pencils, and animals. The network needs input images with the resolution 227×227.

R-CNN is the region-based CNN. Based on traditional CNNs, the region proposals have been added into the structure of neural networks. The proposed region of interest (ROI) has been recommended to reduce the processing time.

Fast R-CNN [38–40] and Faster R-CNN [50,50,127] are region-based CNN (R-CNN) which originated from CNN (i.e., ConvNet), but R-CNN has been applied to quickly find visual objects based on region segmentation and ROI [127]. If the region could be given much earlier, that will speed up the computations of visual object location and classification.

The problem of R-CNN is that its training time is very long because it needs to get $1,000 \sim 2,000$ proposals at first and save them; these proposals need to be calculated in all the former layers. In addition, the fully connected layer is expected that all the vectors will have the same size, all the proposals need to be resized by using cropping or wrapping operations, both strategies are not suitable because the cropping operations may cause that the proposals are not fully extracted, and the wrapping operation may change scales of the visual objects.

Fast R-CNN was proposed in 2015 which overcomes the problems of R-CNN. What Fast R-CNN has completed is to replace ROI pooling layer; softmax is applied to the classification. Softmax is one extension of the logistic regression function to the multiclassification problem.

After Fast R-CNN, Faster R-CNN was proposed to improve the training speed of Fast R-CNN. From R-CNN to Faster R-CNN, four steps of visual object detection are finally unified into one network. Faster R-CNN does not use selective search to get region proposals. Instead, it makes use of a region proposal to carry out the same task. There is not a repetition, and all the calculations are possible to be performed by using Graphics Processing Unit (GPU) [50, 127].

In recent years, YOLO [125] (Darknet) has become a very popular deep net. Darknet is an open-source framework. It is fast, and easy to be installed, which supports CPU and GPU computations. YOLOv3 is the third version of the YOLO family (https://pjreddie.com/darknet/). Before YOLOv3, YOLO and YOLOv2 already have been developed for visual object detection in deep learning, especially for pedestrian detection [113].

YOLO is one of the fast object detectors, which creates grid cells; each cell predicts the bounding box and the confidence score of this box. For evaluating YOLO models, a 7×7 bounding box and 20 labeled classes are employed. Generally, YOLO is faster than R-CNN model.

YOLO makes use of the whole image instead of a region proposal which has a lower rate of background errors. Compared with another real-time system based on PASCAL VOC 2007, YOLO has an overwhelming advantage. Fast R-CNN takes around 2 s for each image to generate region proposals of a bounding box. Faster R-CNN as the most accurate model reaches 7 fps while a smaller model, which has lower mAP 62.1%, achieves 18 fps. But YOLO could reach 45 fps, which is twice faster than R-CNN and even has a higher mAP 63.4%. The limitation of YOLO is that each cell could predict bounding boxes, which makes small objects hardly to be detected.

In 2020, YOLOv4 and YOLOv5 have been proposed for optimal speed and accuracy of visual object detection, especially small object detection has been well-resolved. In 2022, YOLOv6 was developed as the next-generation object detection from the Chinese company Meituan or MT-YOLOv6 which provides a better mean Average Precision (mAP) than all the previous versions of YOLOv5, with approximately two times faster inference time based on COCO val2017 dataset. YOLOv7 [153] makes YOLO great again by using transformers and multi-task training. YOLOv7 achieves mAP 43 and exceeds Mask R-CNN; it's more accurate and even more lighter. Now, YOLOv8 has been online for further study.

Fig. 1.5 SSD for visual object recognition

SSD (i.e., single shot multibox detector) [89,117] is famous for balancing the resolution and speed as shown in Fig. 1.5. Meanwhile, YOLO [125] and YOLOv2 are excellent in achieving the fast speed to detect visual objects based on operations by using 7×7 blocks. Now, YOLOv3 has been thought to overcome the shortcomings and become a very excellent method for visual object detection.

Deep residual network (ResNet) [48,49] was designed for avoiding the problems of vanishing gradients and exploding gradients, reusing activation functions from a previous layer until the adjacent layer obtains its weights. ResNets utilize skip connections, or shortcuts to jump over a few of layers.

In deep learning, there exists the degradation problem, namely with the network depth increasing, accuracy gets saturated. However, ResNets easily enjoy accuracy gained from the increased depth:

$$\mathbf{y} = \mathscr{F}(\mathbf{x}, \{\mathbf{W}_i\}) + \mathbf{x}, \qquad (1.21)$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output vectors of the layers, and $\mathscr{F}(\cdot)$ is the residual mapping.

GAN as a contrastive net [16] refers to generative adversarial network [42,138] that is applied to identify the differences between a fake object and the real one. GAN is a type of deep learning networks that can generate data with similar characteristics as the input training data. A GAN consists of two networks that are trained together: Generator and discriminator. In order to train a GAN, it needs to train the generator

that generates data which fools the discriminator; the discriminator is applied to distinguish real data and generated data. The objective of a generator is to generate data that the discriminator classifies as the real one. The objective of a discriminator is not to be fooled by the generator. These strategies result in the generator that generates convincingly realistic data and a discriminator that has learned strong feature representations.

In GANs [34], the generative network and the discriminative network are playing a game following the min-max decision rule and utilize mathematical expectation as the loss output. The function is also called min-max loss function as shown in Eq. (1.22), where function $D(I, M_d)$ represents the discriminator network, M_d is a randomly generated image, $G(I, M_i)$ refers to the generative network, and I is the input image:

$$\mathcal{L}_{GAN} = \min_{G} \max_{D} \mathbf{E}[\log D(I, M_d) + \log(1 - D(G(I, M_i), M_i))]. \qquad (1.22)$$

The joint loss is further subdivided into discriminator loss and generator loss, which is determined by the structure of GAN as shown in Eq. (1.23). The generator makes this function as small as possible, while the discriminator boosts the value as great as possible. It is worth noting that this function is not used alone in this experiment which is only one part of the total loss function. The total loss is based on a combination of MSE loss $\mathcal{L}_{MSE}(\cdot)$ and GAN loss $\mathcal{L}_{GAN}(\cdot)$ so as to obtain better training results, where α refers to the weight of nets:

$$\mathcal{L}_{Joint} = \min_{G} \max_{D} \mathbf{E}[\mathcal{L}_{mse} + \log D(I, M_d) + \alpha \log(1 - D(G(I, M_i), M_i))]. \quad (1.23)$$

Autoencoder as another contrastive net [16] is trained to minimize reconstruction errors as the loss

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')\|^2, \qquad (1.24)$$

where $\mathbf{x}$ is usually averaged over the input training set. Mathematically, this loss function is a square loss function. A family of loss functions also include $0 \sim 1$ loss function, absolute loss function, average loss function, etc.

Autoencoders are assisted by using the output as its input iteratively, which leads to the famous fixed-point theorem. If the fixed point could be found or converged, that means we can use autoencoders for noise removal and dimension reduction regarding the given data. Autoencoders have been successfully applied to image artifact removal such as inpainting and image denoising [171]. Autoencoders belong to unsupervised learning which generally has a strong ability for data dimensionality reduction.

Reinforcement learning is different from supervised learning and unsupervised learning. Reinforcement learning [84, 112] is suitable for pendulum, dynamic control, and intelligent navigation. An environment and relevant states as well as rewards are established for an agent to control the system. Once the agent takes a step, we need to calculate the rewards and assess the feedback so as to decide whether the action is with positive or negative reward [5].

In Markov decision processes, a Bellman equation is a recursion for expected rewards in control theory by using dynamic programming [59],

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^\pi(s'), \qquad (1.25)$$

where s is a state and the policy is π, $V(\cdot)$ is the value function, and $R(\cdot)$ is the reward function. The Bellman optimality equation is

$$V^{\pi^*}(s) = \max_a \{ R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^{\pi^*}(s') \}, \qquad (1.26)$$

where π^* is the optimal policy and V^{π^*} refers to the value function of the optimal policy.

In recent years, Q-learning algorithm is proposed for solving the problem given by the Bellman equation. The goal of the Q-learning algorithm [147] is to learn a policy, which tells an AI agent what action will be taken under which circumstance.

An autoencoder is employed to explore knowledge from efficient data coding in an unsupervised manner [20]. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction and data embedding, by training the network to ignore signal noises [181].

Given one hidden layer, the encoder of an autoencoder takes the input $\mathbf{x} \in \mathscr{R}^d = \mathbf{X}$ and maps it to $\mathbf{h} \in \mathscr{R}^p = \mathbf{F}$,

$$\mathbf{h} = \sigma(\mathbf{Wx} + \mathbf{b}), \qquad (1.27)$$

where $\mathbf{h}$ usually is a latent variable, $\sigma(\cdot)$ is an activation function, $\mathbf{W}$ is a weight matrix, and $\mathbf{b}$ is a bias vector.

Transfer learning [178] is to apply the well-trained weights of an existing neural network to a new model. We modify the weights or parameters and make minor corrections. It saves training time, however, does not affect the average precision of pattern classification [53] too much.

Transfer learning applies stored parameters or knowledge to a different but relevant problem. From a practical standpoint, reusing or transferring information from previously trained nets for new tasks has the potential to significantly improve the training and testing efficiency in deep learning.

In the new version of MATLAB, there is a toolbox especially for implementing transfer learning [66]. In transfer learning, a deep learning approach, in which a model has been trained for one task, is used as a starting point to train a model for similar tasks. Fine-tuning for a deep neural network with transfer learning is usually much faster and easier than training a network from scratch. Transfer learning is usually offered for visual object detection and recognition, speech recognition, and other applications.

As an instance, ChatGPT, which means a generative pre-trained Transformer, was fine-tuned by using supervised learning as well as reinforcement learning. ChatGPT was developed by OpenAI and launched in November 2022. In reinforcement learning, human trainers firstly ranked responses that the model had created in a previous conversation. These rankings were employed to create "reward models"
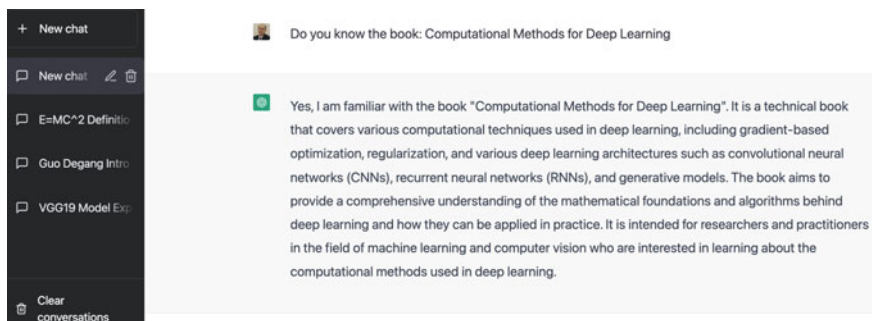
Fig. 1.6 The interface of ChatGPT on February 2023

that the model was further fine-tuned on using several iterations of Proximal Policy Optimization (PPO). ChatGPT has successfully been applied to the conversions related to mathematical equations, chemical reaction formula, VBA programming, etc. The interface of ChatGPT's comments on the book Computational Methods on Deep Learning is shown in Fig. 1.6.

Markov random field (MRF) [77], Markov network, or undirected graphical model is a set of random variables having a Markov property described by a undirected graph. MRF satisfies Markov properties; the Markov random process [64] only has a relationship with the state of the next time. A class of Markov random fields are those that can be factorized according to the cliques of the graph.

In graphical models [64], MRF and DBN are two typical networks. Dynamic Bayesian networks, such as influence diagrams [26], are thought as the directed network; the probability of each node is completely dependent on its neighbors. Therefore, conditional probability and joint probability are especially required.

SqueezeNet [166] and compressed network (CompressedNets) are employed for mobile phones or portable devices. If we trim a network or compress the network, we can use it for the small devices which do not have extremely big memory or need GPUs to support the computations though most of the tablets and mobiles now have been facilitated with this powerful hardware.

Ensemble methods make use of multiple learning algorithms to obtain better predictive performance than what could be obtained from any of the constituent learning algorithms alone. Ensemble learning [46,150] integrates all learners together. By using ensemble learning, we boost a weak classifier to a strong one. The typical ones are AdaBoost and Bagging algorithms. We have employed this algorithm in OpenCV for visual object detection.

Entropy is a measure of the unpredictability of the state, or equivalently, of its average information. The measure of information entropy associated with each possible data value is the negative logarithm of the probability mass function for the value

$$S = -\sum_i P_i \log P_i = -\mathbf{E}_P(\log P), \tag{1.28}$$

where $\mathbf{E}_P(X) = \sum_i P_i X_i$ is mathematical expectation defined by the probability P. After normalization, the normalized entropy will fall in the interval [0, 1] [45].

Function $\mathbf{E}_P[X]$ has the properties:

- $\mathbf{E}_P(c) = c$, c is a constant;
- $\mathbf{E}_P(cX) = c\mathbf{E}_P(X)$, c is a constant;
- $\mathbf{E}_P[\mathbf{E}_P(X)] = \mathbf{E}_P(X)$;
- $\mathbf{E}_P(X \pm Y) = \mathbf{E}_P(X) \pm \mathbf{E}_P(Y)$;
- $\mathbf{E}_P(aX \pm b) = a\mathbf{E}_P(X) \pm b, a, b \in R$;
- If X and Y are independent, $\mathbf{E}_P(XY) = \mathbf{E}_P(X)\mathbf{E}_P(Y)$.

Entropy [25, 28] is a fundamental concept in information theory [22]. All kinds of entropy, such as joint entropy, conditional entropy, and mutual information (i.e., KL divergence) construct the main framework of information theory. Information theory [22] has been applied to information retrieval [13, 107] previously, now it has been employed to deep learning [43].

Mathematics usually refers to calculus [108] and algebra [55] at the undergraduate level. Calculus denotes courses of elementary mathematical analysis, which are mainly devoted to the study of functions and limits. Calculus is related to numerical analysis; linear algebra and tensor algebra are employed in deep learning. Linear algebra deals with vectors and matrices, more generally, with vector spaces and linear transformations. Tensor algebra [55] is needed for better understanding TensorFlow [2] from DeepMind of Google Inc. In mathematics, a tensor is an algebraic object that describes a multilinear relationship between sets of algebraic objects related to a vector space. Tensor algebra is a powerful tool with applications in machine learning, data science, engineering, and computer sciences.

In deep learning, it is really true that we need to study probability theory and mathematical statistics at the undergraduate level; we also should know functional analysis [114] and abstract algebra [56] at the postgraduate level for fully understanding the algorithms and mechanism of the loss functions in a normed space and polynomial rings. Finite fields [82] are especially recommended if polynomial theory is needed in computations.

Since we have known deep learning did not fall from the sky, it accumulates human computing experience in the past decades from sensor networks, big data, cloud computing, image processing, computer vision, pattern classification, machine learning, etc.

We strongly suggest that a beginner should start deep learning study from a well-written journal article [72, 73, 135] or a good book [43], download the relevant source codes for implementation, and experience the differences of distinguished deep learning and conventional machine learning methods. Based on the first-hand experience, further deep learning study or research work is encouraged. Mathematical knowledge and network structure are especially recommended as the fundamental knowledge.

1.4   Sample Projects for Deep Learning

In recent years, a myriad of projects have been developed based on deep learning [7, 9, 10, 32, 62, 80, 94, 95, 98, 104, 128, 140, 154, 162, 170, 177, 182, 183, 186]. The features of these developed projects are different from conventional machine learning or pattern classification. We would like to list a few of them as follows.

A project has been developed for human face detection and recognition from a distance [23, 154, 155, 158]. Face recognition is an important biometric in video surveillance. In this project, we took a spate of human face photos from multiple views and trained the Inception network [144] by using data augmentation. In the distances, the influence from the camera to a face and the size of the face in the images are diverse. The results indicate that the deep learning method is able to recognize human faces with partial occlusions and various distances. If a face could not be detected, we will quickly switch to human gait recognition.

In human age estimation based on face recognition [139], we propose an improved end-to-end learning algorithm to address the aggregation of multiclass classification and regression for age estimation by using CNNs. Our contribution is to work for an updated algorithm regarding age estimation by adopting the attention and normalization mechanisms for balancing the efficiency and accuracy of the proposed model. In addition, our model suits to be deployed on mobile phones owing to its compact size and superior performance. In future, we will explore this mechanism for other applications related to facial information.

One of our projects is related to human facial expressions of emotion [4]; we investigate the reliability of facial emotion recognition (FER) based on the seven classes of human facial expressions of emotion: Neutral, scared, angry, disgusted, sad, happy, and surprised. We classify human facial emotions from digital images. Convolutional Neural Network (CNN), Xception, Vision Transformer (ViT), Simple Deep Neural Network (SDNN), and Graph Convolutional Neural Network (GCN) have been taken into consideration with two types of methods: Non-facial landmarking and facial landmarking. With the popularity of deep learning algorithms, our SDNN model was employed as a baseline to test the proposed method. With this model, we are able to achieve 96.0% accuracy in our real-time testing experiments.

In the throes of the COVID-19 pandemic, people were asked to wear masks in public. The current face recognition method is well established and quite straightforward. However, if people wear face masks, it will reduce the accuracy of face recognition. In our project [79, 88], we propose a masked face recognition method and solve the problem that many of the models cannot be applied to portable devices or mobile terminals.

Human action recognition from digital videos is a hot topic in the field of computer vision [177]. Deep learning methods have gained great attainments in this field. The goal of human action recognition is to classify patterns so as to understand human actions from visual data and export corresponding tags. In addition to spatial correlation existing in 2D images, human actions in a video are explored based on the correlation in temporal domain. Based on CNN, Two-Stream CNN, CNN+LSTM,

and 3D-CNN are harnessed to identify human actions. HMDB-51 dataset is employed to test these algorithms and gain the best results.

Human behavior such as running, jumping, and skipping could be detected using deep learning [97]. Based on this work, we quickly detect pedestrians, especially abnormal behaviors for anomaly detection [10, 11]. The differences from previous work based on Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG) are that we applied a big training dataset with YOLOv3 as the classifier.

Furthermore, we investigate the state-of-the-art deep learning methods for sign language recognition. Capsule Network (CapsNet) and Selective Kernel Network (SKNet) with attention mechanism are applied to extract spatial features. Sign language recognition is one of the fundamental ways to assist deaf people to communicate with others. Sign language being an important means of communications, the problems of recognizing sign language from digital videos in real time have become the new challenge of this research field [93]. Recently, Vision Transformer and other Transformers have shown apparent advantages in object recognition compared to traditional computer vision models such as Faster R-CNN, YOLO, SSD, and other deep learning models. We use a Vision Transformer-based sign language recognition method called DETR (Detection Transformer) to improve the current state-of-the-art sign language recognition accuracy. The DETR method is able to recognize sign language from digital videos with high accuracy using the model: ResNet152 + FPN (i.e., Feature Pyramid Network), which is based on Detection Transformer. ResNet152 + FPN is able to enhance the detection accuracy based on the test dataset of sign language compared to the standard Detection Transformer models.

A model of dynamic skeletons called spatial temporal graph convolutional networks (STGCN) was proposed by automatically seeking both the spatial and temporal patterns from visual data to achieve the human behavior recognition [86]. It performs pose estimation based on videos and constructs a spatiotemporal graph of skeleton sequences. Multiple layers of the spatiotemporal graph network gradually generate high-level feature maps from the videos. It is classified by applying the standard softmax classifier to obtain the corresponding pattern class.

Human gait recognition is one of the most promising biometric technologies, especially for unobtrusive video surveillance and human identification from a distance [85, 157–161]. Aiming at improving gait recognition rate, we conduct gait recognition by using deep learning methods and proposed a framework based on multichannel convolutional neural networks (MCNN) and convolutional long short-term memory (ConvLSTM), and manifold learning known as GaitManifold.

Human finger motion was detected for Morse code enter [76, 185]. In a circumstance, if we are not allowed to speak loudly, we can use gesture or Morse codes to contact others. Writing Morse codes on a table surface could not attract much attention. Computer vision by using human gesture recognition could work in mutual silent communications in mute mode.

Our motivation for image colorization is inspired by deep learning, especially video analogies [174] and transfer learning in the domain of deep learning [162]. In this project, we experimented with deep learning networks for colorizing the CT

lung images. For hybrid colorization, we select appropriate reference images so as to colorize the target CT lung grayscale images. Pertaining to the results, we consider numerous methods such as human visual analysis, PSNR, and SSIM to evaluate the proposed deep neural networks. The results of rendering the CT lung images by using deep learning are significantly prominent.

In recent years, deep neural networks (DNNs) have achieved a remarkable progression for resolving complicated problems. DNNs are suitable for dealing with the problems related to time series analysis, such as speech recognition and natural language processing. Video dynamics detection, as an instance, is time-dependent. Apparently, video dynamics detection needs to utilize the present, previous, and next frames of a given video. If a frame change occurs, it triggers whether a video event happens or not. We are able to achieve high-precision and real-time video dynamics detection by using RNN and GRU as well as apply CNN to reduce the video size and extract critical information. We integrate CNN and RNN together to make a significant reduction in the size of video data and the training time [188].

Visual object detection in blindspots [136] of moving vehicles has been developed as a research project in unmanned vehicles. The frequent look-back actions of a driver could be reduced, and a monitoring system could automatically and timely count moving objects in the blindspots and report potential hazards to the driver in all time. We also developed a project for predicting the distance between current car and the front car, or current car and the car behind, the three-second distance is kept for monitoring the safe distance between vehicles [110].

Flame detection [137,170] is a series of projects we have developed for a number of years in intelligent surveillance; the flames from burning fires of a torch and combustible materials in a fireplace could be recognized. In this project, we detect the fire flare region with fine-tuning operations from deep learning. The features of natural fire have been identified to discriminate the correct frames from the wrong ones.

Currency and coin detection, recognition, and forensics are still very valuable [104,105,167,183,184] in our banking business. So far, we can quickly find the currency from the aspect of visual object detection. We mainly use single shot multibox detector (SSD) based on deep learning as the framework, and employ CNN to extract visual features of paper currency, so that we are able to accurately recognize the denomination of currency and coin, both front and back faces. Moreover, YOLOv5 was implemented to detect paper banknote. These methods not only slash human labor but also promote the precision of currency recognition. By comparing multiple versions of YOLOv5 in terms of YOLOv5s, YOLOv5m, YOLOv5l, and their variants, we find that more network layers propel higher precision and a better GIoU loss with the sacrifice of training speed [146].

Deep learning methods have been offered for removing noises from an image [94]. After trained a neural network model, we obtained the parameters to make any picture smooth including the picture after JPEG lossy compression. Deep learning has the capability to remove artifacts and reconstruct an image including image inpainting [34].

In general, face image inpainting is composed of generators and discriminators in GANs [34,35]. In this project, we have designed and implemented a new model for face image inpainting. We take advantage of CNNs in deep learning as the basis net; the weighted mean squared error (MSE) loss and the GAN loss functions are combined to improve the stability of model training; the complete network of the entire structure of convolutional nets is employed for the image inpainting. In addition, global and local discriminator nets are also put forward to improve the quality of image inpainting.

Alzheimer's disease (AD) is a neurodegenerative disorder which leads to memory and behavior impairment [57,58,140]. Early diagnosis and therapy can delay the progress of this disease. Deep learning methods have been applied to Alzheimer's disease diagnosis. Selective kernel network with attention (SKANet) for early diagnosis of AD using magnetic resonance imaging (MRI) is proposed. Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, and allow to model the dependencies with regard to the distance in the input or output sequences. The attention mechanism was added to the bottom of the block to emphasize important features and suppress unnecessary ones for accurate representation of the network [148].

Deep learning aids fruit recognition and allows a computer to detect a fruit and find its freshness and ripeness automatically [32,33]. A number of algorithms have been reviewed, including YOLOv5 for detecting region of interest with considerations of digital images, ResNet, VGG, GoogLeNet, and AlexNet as the base networks for fruit freshness grading. Fruit decaying occurs in a gradual manner, this characteristic is included for freshness grading by interpreting chronologically-related fruit decaying information [32].

Apple ripeness identification is such a type of pattern classification. In this project, the ripeness of apples was detected with deep learning [168]. We make use of a set of various apple and pear datasets to train, test, and evaluate the performance of different transformer models. The Transformer model is essentially a process of encoding and decoding. Encoder is to stack the encoder block, like a regular CNN. Vision Transformer (ViT) and Swin Transformer are considered. We carefully compare and examine the advantages of these two models for accurate fruit ripeness analysis. We find out that Swin Transformer achieves significantly better results than Vision Transformer for both pears and apples [169].

We deploy a model based on CenterNet for visual object detection to resolve the problem of fruit detection. By comparing those models with different backbones, the deep learning-based model with DLA-34 was chosen as the final model to detect fruits from an image. Meanwhile, our dataset with four classes and 1,690 images were collected. By evaluating the performance of the models, we eventually design a CenterNet based on DLA-34 to detect multiclass fruits from the given images [186, 187]. Furthermore, deep learning could be applied to food safety assessment; we have employed deep learning to meat quality analysis [6–9].

In traffic-sign recognition, we propose a deep learning algorithm based on Faster R-CNN and YOLOv5. Firstly, we conduct image preprocessing by using guided image filtering for the input image to remove noises. The processed images are

Fig. 1.7 Traffic signs from Google satellite view

imported into the neural networks for training and testing. The outcomes of traffic-sign recognition are promising. The improved YOLOv5 model was applied to Google satellite images for accurately detecting road signs on ground [172] as shown in Fig. 1.7.

In the sailboat and kayak detection [103], we search for a set of best parameters for YOLOv5 model. We find out the best structure for the kayak detection using our training dataset. We verify our model and compare it with a well-trained model by using ensemble learning. We construct a realistic sailboat detection dataset [101], which allows us to evaluate the robustness of our model in real-world applications. Facing with the diversity of data samples collected by using various devices, we create the model by automating the search and design which makes the model much robust. Our model combines and improves the YOLO family with the attention mechanisms which are validated based on real datasets [102].

The pose estimation of swimmers is a basic problem to be solved in the tasks of relevant computer vision. In this project [18], we implemented a method for the pose estimation of swimmers based on deep learning, which fits scenarios containing multiple swimmers. We combined the HRNet with YOLOv5 to implement our model. Our method achieved ideal accuracy; the model is easy to be trained and deployed. In addition, a dataset is annotated with keypoints for swimmers and a slew of datasets are explored for swimmer detection. Our dataset is composed of the underwater view of swimmers. Compared with the side view, the torso of swimmers collected by the underwater view is much suitable for a broad spectrum of deep learning applications.

In order to describe a journey, storytelling could be automatically generated from a group of digital photographs. Most of the existing methods focus on descriptions of specific content of a single image, such as image captioning, which lack correlation between the images and the spatiotemporal relationships. In this project, we make use of visual object detection from digital images. Combining the changes in spatiotemporal domain and filling in the predetermined template, we automatically generate

a text-based travel diary. Compared with conventional image captioning, our methods are to effectively connect correlation between digital images and background information [190].

Speech recognition is an important field in natural language processing. In this project, we implemented a hybrid model of CTC and attention (CTC + Attention) model for multilanguage speech recognition. In order to compare the methods, we designed and created three datasets: Chinese, English, and Code-Switch. Throughout our experiments, we find that the hybrid CTC + Attention model based on end-to-end framework achieves better performance compared with the HMM-DNN model in a single language and Code-Switch speaking environment. The CER (i.e., Character Error Rate) of the proposed hybrid CTC + Attention model based on the Chinese dataset defeated the traditional model [80,81].

In order to achieve automatically litter detection in residential area, machine vision has been applied to monitor the environment of surveillance. Based on our observations and comparative analysis of the current algorithms, we proposed an improved object detection method based on Faster R-CNN algorithm and achieved more than 98% accuracy of litter detection in surveillance [87]. ConvNeXt was also selected as a model for waste classification from digital images. We made use of ConvNeXt as the backbone to obtain an efficient waste classification model [120].

While current visual object detection algorithms focus on the exploration of larger objects, the development of small object detection is being expanded relatively slowly due to the inability to acquire more visual information. We proposed a method combining contextual information and multiscale learning to improve small object detection performance in waste classification. Furthermore, based on the advantages of parallel computing in Transformers, we utilize DETR model to explore our method [121]. The experimental results show that our method achieves high accuracy in the detection of a small object in waste.

1.5   The Databases for Deep Learning Projects

In machine learning and deep learning [103], PASCAL VOC is a benchmark dataset for pattern classification and visual object detection. By starting with only four classes in 2005, the dataset has grown into 20 classes. Since 2009, the number of photos has increased year after year, but all previous images have been retained to be able to compare the test results. ILSVRC [132] is a derivative of ImageNet, which extends the standard algorithm of visual object recognition as well as evaluations. ImageNet1000, a subset of ImageNet images with 1,000 object classes and a total of 1.2 million images, has been modified to provide a standard test for the ILSVRC object classification. MS COCO database contains complex scenes with visual objects in natural situations. All of these datasets use bounding boxes as output or object instances as the segmented output.

In vehicle-related scene understanding [90,91], the first step is to label data or obtain labeled data; it is imperative to collect images of the relevant environment

extensively. In this project, we use an in-vehicle camera to obtain video files of traffic scenes. We define pixel-level labels for project object classification, including "sky", "building", "buslane", "road", "lane", "tree", "trafficsign", "turnsign", "vehicle", "pedestrian", and "traffic light". This dataset guarantees the sharpness of images, which contains visual objects taken from various angles and locations. The same object has different distances from the camera, and the same objects in different images have different sizes.

In scene segmentation [91], the labels as the ground truth were manually annotated including "vehicle", "sky", "tree", "building", "road", and "traffic sign", leading to six classes. We split the dataset into a training set and a test set in a ratio 3:1. In addition, we also perform intensity preprocessing on the dataset. In order to make the pixels of each image within interval [–1.0, 1.0], we resized the input image. We conducted image enhancement based on the dataset with transformations (e.g., translation, scaling, rotation, and elastic deformation) and intensity (e.g., translation, scaling, and equalization) to avert overfitting.

The self-driving dataset KITTI is a benchmark for performance comparison among models [109]. KITTI dataset provides right and left view cameras for acquiring color images, LiDAR's 360° point clouds, and GPS coordinates, and all are synchronized. Scenes have been recorded from well-structured highways, complex urban areas, and narrow countryside roads. The dataset contains 15,000 frames: 7,500 labeled and 7,500 unlabeled. The various sensors and labeling information are provided for each frame, along with the orientation. Meanwhile, Waymo dataset has resolution 1980×1280 with continuous labeled frames. Waymo dataset provided an enormous amount of data in the form of 900 segments, with each segment of around 1 GB size. It was found to contain the night, rainy as well as foggy scenes with car objects in the downloaded data segments.

In traffic-sign recognition, we recorded the realistic traffic-sign images [122]. The resolution is 1080×1440. Our dataset consists of 3,436 images and 3,545 instances in total, labeled as "Stop" (236 instances), "Keep left" (536 instances), "Road diverges" (505 instances), "Road bump" (619 instances), "Crosswalk ahead" (636 instances), "Give way at roundabout" (533 instances), and "Roundabout ahead" (480 instances). We utilized data augmentation to expand our dataset; the basic manipulations for data augmentation include flipping, rotation, shearing, and adding noise as well as blurring images. In this case, we merely applied two augmentation operations, including adding noises and blurring images, because these methods could not deal with the distorted objects that may impact the quality of our dataset and even degrade the accuracy of our training models.

In another project [191], we collected traffic-sign images. Our dataset is composed of 2,182 traffic-sign images which are labeled as "No U-turn" (271 images), "Road bump" (329 images), "Road works" (294 images), "Watch for children" (176 images), "Crosswalk ahead" (313 images), "Give way" (317 images), "Stop" (286 images), and "No entry" (196 images). We resized the images while keeping the same aspect ratio between width and height. Thus, we normalized all images in our dataset to be 1128×2016 and 1536×2048.

In the traffic-sign recognition with image defogging by using guided image filtering [172, 173], our dataset consists of 3,105 images and 5,536 instances in total. First of all, a dataset includes 12 traffic signs. The resolution of the images is 1920×1080. We have added three datasets: FROSI, FRIDA, and FRIDA2. The fogs in the FROSI dataset are grouped according to the visibility from 50 to 400 m; the traffic signs in the dataset are similar to real ones. These three datasets assist us to improve the accuracy of these two deep learning methods for traffic-sign recognition on foggy days.

In the project for tree leaf recognition [156], the dataset includes five classes of tree leaves: (1) Magnolia grandiflora, (2) Boehmeria nivea, (3) Clausena lansium, (4) Euphoria longan, and (5) Hibiscus. We made use of the software LabelMe to label the acquired images. The method of labeling is to label each class by manually selecting a rectangle box as the region of interest. In the process of labeling, multiple classes of leaf abbreviations are represented by combining different numbers and initials of leaf names. The data augmentation includes specific operations such as flipping, zooming in, zooming out, clipping, and combining. The number of images was expanded to 1,676. The data is split into training set and verification set according to the ratio 8:2. The final training set has 1,340 images.

The dataset in the project related to sailboat recognition [101–103] includes 600 images of kayaks and 400 images of sailboats. The labels of each picture are tagged manually, the coordinates of each sailboat and kayak location are marked, and the visual objects are indexed. Each photo contains at least one index; there are total more than 2,700 indexed images of sailboats and kayaks. The training set has 80% of the data from the dataset, the other 20% is of the data for testing, and validation set takes 20% of the set.

There are plenty of public datasets for human behavior recognition, such as Weizmann dataset, KTH dataset, UCF dataset, CAVIAR dataset, CASIA dataset, and BEHAVE dataset [98]. Weizmann dataset encapsulates 10 classes, each of the classes has nine videos which were acquired by using a static camera with individual behaviors; the resolution of the image samples is 180×144. KTH dataset includes a total of 2,391 videos with six classes (i.e., "walking", "running", "boxing", "hand clapping", "jogging", and "hand waving") of human actions from 25 participants in four scenarios: Indoor, outdoor, outdoor with amplification, and outdoor with different clothing. The videos in this dataset enclose scaling changes, clothing changes, and lighting changes which were captured by using static cameras. The resolution of this dataset is 160×120 with the 5,667 video frames [97, 100].

UCF101 [86] is a collection of real-time action videos from YouTube that are grouped into 101 action classes. It has the most diverse collection of 13,320 videos by 101 action classes, as well as a broad range of camera movements, object appearance, poses, cluttered backgrounds, lighting conditions, and more. In the same group of videos, backgrounds and perspectives are the same. A class of actions include human interactions, body movements, playing musical instruments, and movement.

HMDB51 [86] contains 6,849 samples, which was grouped into 51 classes, each class contains at least 101 samples, with a resolution 320×240. The actions mainly include the following: (1) General human facial actions: Smiling, laughing, chewing,

and talking; (2) Human facial actions with objects: Smoking, eating, and drinking; (3) General human body actions: Cartwheeling, clapping, climbing, climbing stairs, jumping, landing on the floor, backhand flip, handstand, jumping, pulling, pushing, running, sitting down, sitting up, heeling, standing up, turning, walking, and waving; (4) Human actions interacting with objects; (5) Human actions.

Our dataset for swimmer detection [18] was collected by using video footages of swimming events. The dataset includes 2,500 images with 3,615 annotated swimmers having the swimming styles such as breaststroke, freestyle, butterfly, and backstroke. In data augmentation, we added 20% noisy images, including water reflection, bubbles, dim light, occlusion, and other factors. In addition, we also added empty swimming lanes as negative samples.

In the project related to human facial expressions of emotion [4], the dataset for our experiments is FER2013, which contains approximately 30,000 grayscale facial images with seven classes of facial expressions of emotion having the resolution 48×48. In this project, we make use of 20% samples for training. This dataset contains 35,887 images, all of which are grayscale images. Human facial expressions of emotion are from Class '0' to Class '6' to represent the seven classes of emotions: Angry (Class '0'), disgust (Class '1'), fear (Class '2'), happy (Class '3'), sad (Class '4'), surprise (Class '5'), and neutral (Class '6'). Regarding the images of human facial expressions of emotion, we returned the number as a label that each image is classified as the emotions with the use of '0' and '1', '1' represents the image that is classified as a facial expression of emotion.

In the research project related to human facial image inpainting [34, 35], we made use of the CelebA dataset with 202,599 face images. CelebA is a large-scale dataset dedicated to human face-related experiments. The image size in the CelebA dataset is 178×218. To make the experiments easier, we resize all the image sizes to 128×128. During model training, we randomly add a masked image with the size ranging from 24×24 to 48×48 to generate the training dataset.

The dataset for Braille text recognition comprises four types of characters [75], including 1,404 images of Braille characters, there are 27 classes of Braille characters corresponding to 26 English alphabets and special symbols. The sizes of the images are various from 91×96 to 509×598. The characters are printed having various colors, brightness of backgrounds, and added noises. For preprocessing, we have completed data augmentation for enhancing the robustness of the model, rotating each image in random angles within the range $[-45°, 45°]$, combining the rotated images with the original dataset; the final dataset contains 2,808 images, resized into 52×52, then the pixels are normalized between 0 and 1.0 (i.e., pixel intensity is divided by 255). The processed data will be randomly shuffled and separated into training set, test set, and validation set.

Since there is no existing fruit freshness dataset available, this project encompasses the work for data collection [32, 33]. The collected dataset consists of six classes of fruits: "Apple", "banana", "dragon fruit", "orange", "pear", and "kiwifruit", derived from a variety of locations with different ambient noises, irrelevant adjacent objects, and light conditions. In total, there are (approximately) 4,000 images collected with each class of fruit images up to 700. The dataset was split into training and validation

sets at the ratio 1:9 (90% for training and 10% for validation). The freshness grading is scaled from 0.0 to 10.0 with 0.0 indicating total corruption and 10.0 for total freshness. The image augmentation includes scaling, rotation, cropping, and adding random noises. All images were added with random noises consisting of random changes of brightness, contrast, saturation, and erosion of 10 image regions. The added random noises follow the sequential order: Random brightness adjustment, random contrast, and random erosion of 10 image regions.

There are various types of domestic wastes [120,121], which are classified into four main classes according to the waste classification criteria, namely "dry waste", "wet waste", "hazardous waste", and "recyclable waste". Within each class, there is a consortium of subcategories. For example, recyclable waste includes "cardboard", "glass", and "plastic bottles". The hazardous waste includes "batteries", "nail polish bottles", and "medicine bottles". In our experiment, we collected a total of 1,660 images. For each class, the number of samples is around 400. Besides, during the training process, our dataset consists of training, validation, and test sets with sample sizes 1,328, 166, and 166, respectively.

In speech recognition [80,81], THCHS-30 is an open-source Chinese corpus released by the Tsinghua University China. THCHS-30 contains over 30 h Mandarin speech footages recorded by a single carbon microphone at a silent office with a sampling rate of 1.6 kHz and a sample size of 16 bits. The training dataset of THCHS-30 contains 29.23 h 10,893 utterances from 8 male speakers and 22 female speakers. The test dataset of THCHS-30 contains 6.24 h 2,496 utterances from 1 male speaker and 9 female speakers.

Meanwhile, LibriSpeech corpus is a large English corpus containing 1,000 h of speech sampled at 16 kHz based on the LibriVox project. LibriSpeech is split into data subsets based on recording quality. Furthermore, TAL-CSASR corpus is the audio in English class teaching environment. This dataset contains 587 h of audio clips from over 200 teachers, sampled at 16 KHz and 16 bits. Considering the teachers teach bilingually in both Chinese and English, TAL-CSASR corpus is used in the Code-Switch environment. In the TAL-CSASR corpus, there is a transcript file called label. Each row of the transcript file consists of a unique ID and a mixed Mandarin–English sentence including Chinese characters and Capitalized English words.

1.6   Awarded Papers on Deep Learning

In this section, we list the awarded work published in IEEE CVPR (International Conference on Computer Vision and Pattern Recognition) and IEEE ICCV (International Conference on Computer Vision). IEEE ICCV (1987–present) has the best paper award—Marr prize. The prize is regarded as one of the top honors for computer vision scientists.

A good paper is reflected in many aspects such as its idea, writing, algorithms and results, references, equations, tables, and figures. The core is how the paper idea

attracts readers and what impact is generated from this work. In recent years, a great deal of deep learning papers have been awarded in IEEE CVPR conferences.

In 2015, the award was conferred to the paper "Deep Neural Decision Forests" [65] from Microsoft Research Cambridge (U.K.) which is an approach that unifies classification trees with the representation learning functionality known from deep convolutional networks, by training them in an end-to-end manner. Conventional approaches for decision tree training typically operate in a greedy and local manner. To overcome the shortcoming, stochastic routing for decision trees, enabling split node parameter learning via backpropagation, the ways to populate leaf nodes with their optimal predictors are introduced. The decision forest model was successfully validated based on ImageNet without any form of dataset augmentation.

In 2017, the paper "Mask R-CNN" from Facebook AI Research (FAIR) has been selected. Mask R-CNN extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding-box regression. Mask R-CNN is simple to be trained [50]. Mask R-CNN allows to estimate human poses in the same framework and outperforms in instance segmentation, bounding-box object detection, and person keypoint detection.

In 2019, the paper "SinGAN: Learning a Generative Model from a Single Natural Image" has been selected and awarded in ICCV'19. SinGAN has the capability to generate diverse realistic samples for natural complex images. SinGAN contains a pyramid of fully convolutional GANs, which allows generating the samples having arbitrary size and aspect ratio that maintain both the global structure and the fine textures of the training image. This approach is not limited to texture images.

Swin Transformer [96] refers to a hierarchical Transformer [27] whose representation is computed with shifted windows. The shifted windows bridge the windows of the preceding layer, providing connections among them that significantly enhance modeling power. Swin Transformer constructs hierarchical feature maps . With these hierarchical feature maps, Swin Transformer can generate dense prediction such as feature pyramid networks (FPN) [83] or U-Net [129,175]. The hierarchical architecture has linear computational complexity with respect to image size while the previous ones have quadratic complexity [27]. The paper was awarded as the best paper award in IEEE ICCV 2021.

IEEE CVPR (1985–present) conference is regarded as the biggest academic conference of the world, which is highly selective with generally less than 30% acceptance rates for all papers and less than 5% for oral presentations. The conference is usually held in June and rotated around the U.S. generally West, Central, and East.

In 2016, the paper "Deep Residual Learning for Image Recognition" from Microsoft Research has been awarded. ResNet [150] was thought as a prominent contribution in deep learning. The ResNets are easy to be optimized and acquire accuracy gains from a greatly increased depth.

DenseNet (i.e., Dense convolutional network) [54] was awarded as one of the best works in IEEE CVPR 2017, which connects each layer to every other layer in a feedforward fashion. DenseNets naturally integrate the properties of identity mappings, deep supervision, and diversified depth. The specific design takes

advantage of the merits: Alleviating the vanishing gradient problem, strengthening feature propagation, encouraging feature reuse, and substantially reducing the number of parameters.

In IEEE CVPR 2017, another awarded work was SimGAN [138]. SimGAN refines the output of a simulator with a refiner neural network and minimizes the combination of a local adversarial loss and self-regularization. The discriminator network classifies an image as real or refined. The refiner network and the discriminator network are updated alternately. Using synthetic images without any labeled real data is the prominent merit of this published work.

In IEEE ICCV 2017, Mask R-CNN [50] has been awarded, which was extended from Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (ROI). Because of masks, i.e., binary images, Mask R-CNN is easily generalized to instance segmentation, bounding-box object detection, and person keypoint detection.

In 2018, the best paper of CVPR was "Taskonomy: Disentangling Task Transfer Learning" [178]. The work presented a method for modeling the space of visual tasks by using transfer learning to reduce the needs for supervision. Taskonomy [178] is a fully computational approach for modeling the structure of visual tasks. The outcome is a computational taxonomic map for task transfer learning. The outputs are stable with no change in the top of taxonomy list. This remarkable work has been awarded as the paper of IEEE CVPR 2018.

1.7   Deep Learning Papers Published with Nature and Science

In this section, we will detail the work published in the famous journals: Nature and Science. Nature is a British weekly scientific journal based in UK, which features peer-reviewed research from a variety of academic disciplines, mainly in science and technology (ISSN: 0028-0836, impact factor: 69.504, 2021). Science is a peer-reviewed academic journal of the American Association for the Advancement of Science (AAAS). The major focus of the journal is publishing important original scientific research and research reviews (ISSN: 0036-8075, impact factor: 63.714, 2021).

Meanwhile, the famous academic journals Nature and Science have published multiple papers related to deep learning [73,84,111,131,163,189] and artificial intelligence [37,52,60]. The articles related to deep learning, especially reinforcement learning [84] and manifold learning [189], have been turned up in these top-stream journals. These publications ramp up and lead the deep learning research work much deeper. Deep convolutional nets have brought breakthroughs [73], which have turned out to be very good at discovering intricate structures in high-dimensional data by using the backpropagation algorithm. Deep learning is still promising because it can easily take advantage of available computations and a large amount of data.

Reinforcement learning concerns with the experience gained by interacting with real world and evaluative feedback to make behavioral decisions [84]. The feedback

provides the learners with an assessment of the effectiveness of the decisions. The central problem of reinforcement learning is the challenge of interactions, evaluations, and awards.

Reinforcement learning is one of the three basic machine learning paradigms, alongside supervised learning and unsupervised learning. In reinforcement learning, the information which is available from training data is intermediate between supervised and unsupervised learning. The training data in reinforcement learning are assumed to provide only an indication whether an action is correct or not; if an action is incorrect, there remains the problem of finding the correct action.

Reinforcement learning generally makes use of ideas that are familiar with control theory [115], such as policy iteration, value iteration, rollouts, and variance reduction, with innovations arising to address the specific needs of machine learning.

Our human brains are use of distributional reinforcement learning to select appropriate actions [24]. Distributional reinforcement learning is a computational framework, which is based on the principle of asymmetric regression. In conventional reinforcement learning [84, 141], the reward prediction is represented as a single quantity: The average over all potential reward outcomes, weighted by their respective probabilities. By contrast, distributional reinforcement learning uses a multiplicity of predictions. These predictions vary in the degree of optimism about upcoming reward. Compared with traditional procedures, distributional reinforcement learning can increase performance in deep learning systems, and yield a performance advantage greater than double Q-learning (DQN) and standard Q-learning.

In the game of StarCraft II (AlphaStar) [151], a multi-agent reinforcement learning algorithm makes use of data from both human and agent games within a diverse league of continually adapting strategies and counter-strategies. The reinforcement learning was applied to improve the performance of AlphaStar based on agent-versus-agent games.

Reinforcement learning has been applied to train a glider autonomously [124]. Using the reinforcement learning framework, the behavior of a glider is described as an agent traversing different states by taking actions while receiving a local reward. This flight policy was established through experiments, numerical simulations, and estimates of the noise in measurements. Measurements from multiple instruments are combined by an extended Kalman filter [29, 180] (EKF) to give an estimate of relevant quantities. The navigational component of the glider is modeled as a Markov decision process [14].

Reinforcement learning is concerned with experience gained by interacting with the world and evaluative feedback to improve a system ability to make behavioral decisions, which includes generalization, observation, planning, exploration, and empirical methodology [84]. Reinforcement learning has contended with all three forms of feedback simultaneously: Sampled, evaluative, and sequential feedback.

A deep Q-network (DQN) can learn policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning which is able to combine reinforcement learning with a class of deep neural networks [112]. Reinforcement learning makes use of a reward to continuously shape representations within the convolutional network toward salient features of the environment that facilitate value

estimation. The integration of reinforcement learning with deep network architectures is dependent on the incorporation of a replay algorithm on the storage and representation of experience. The Q-learning algorithm updates the weights after every time step, and replaces the expectations by using single samples.

Image reconstruction is treated as a data-driven supervised learning task that allows a mapping between the sensor and the image domain to emerge from an appropriate corpus of training data [189]. Manifold learning results in sparse representations of domain transforms along low-dimensional data manifolds, and observe superior immunity to noise and a reduction compared with conventional image reconstruction methods.

Deep learning [73] was thought that it could be applied to computer vision [63], natural language processing [21], robot control, and other applications [60]. The supervised learning methods include decision trees/forests [36], logistic regression [12], deep neural networks, Bayesian classifiers [116], Boosting [130], etc.

Deep neural networks are multilayer networks with threshold units. Deep learning makes use of gradient-based optimization algorithms to adjust parameters throughout a multilayer neural network based on errors at its output. The internal layers of deep networks can be viewed as providing learned representations of the input data.

The aim of deep neural networks [131] is to find a set of weights that ensure for each input vector the output vector produced by the network is same as the desired output vector. If there is a fixed and finite set of input–output cases, the total error of the network with a particular set of weights can be computed by comparing the actual and desired output vectors for every case. For a given case, the partial derivatives of the error with respect to each weight are computed in two passes: Forward pass and backward pass which propagates derivatives from the top layer back to the bottom one.

Unsupervised learning is mainly applied to solve the problems such as data clustering and dimensionality reduction; these mathematical approaches include PCA (i.e., a linear dimensionality reduction method), manifold learning [14] (i.e., a nonlinear dimension reduction method), autoencoders [20, 165], etc.

In order to circumvent the limited amount of observation data, transfer learning is applied to train a convolutional neural network (CNN) based on CMIP5 dataset and multi-year ENSO (i.e., El Niño/Southern Oscillation) forecasts [47]. Using transfer learning and heat map analysis for predicting helps understand a climate phenomenon. A heat map analysis indicates that the CNN model predicts ENSO events using physically reasonable precursors. The CNN model is powerful for both the prediction of ENSO events and for the analysis of their associated complex mechanisms.

The key problems in geosciences include pattern classification, anomaly detection, regression, and space- or time-dependent state prediction. Deep learning approach is able to extract spatiotemporal features automatically and gain further process understanding of geoscience problems, by improving the predictive ability of seasonal forecasting and modeling of long-range spatial connections across multiple timescales [126].

Deep learning requires extremely large datasets that are well annotated so that the algorithms can learn to distinguish features and categorize patterns. After an algorithm has been well trained, it can apply that trained model to analyze other data [163]. In biology, deep learning algorithms extract features that others cannot catch. Transfer learning has the ability to apply classification prowess acquired from one data type to another. The algorithms make both accurate and explainable predictions, but remain black boxes.

CAPTCHA is regarded as deciphering letters that may be distorted, partially obscured, or shown against a busy background. CAPTCHAs add clutter and crowd letters together to create a problem for algorithmic classifiers. It is tricky from the viewpoint of computer vision, but for our humans it's not. Recursive cortical network (RCN) is a probabilistic generative model for object recognition, segmentation, tracking, and reasoning in a unified manner. The model demonstrates excellent generalization and occlusion-reasoning capabilities.

A recommendation system [164] is based on the data that indicates the links between a set of users (e.g., people) and a set of items (e.g., products). The problem [60] is to suggest a solution to a given user based on the data across all users. Human data was employed to aid in exploration and to preserve strategic diversity. Machine learning, i.e., supervised learning, unsupervised learning, and reinforcement learning, has been employed for solving this problem.

High-dimensional data can be converted to low-dimensional codes [52]. Gradient descent can be used for fine-tuning the weights by using autoencoder. The autoencoder consists of an encoder and a symmetric decoder. Deep autoencoders would be very effective for nonlinear dimensionality reduction, provided that computers were fast enough, datasets were big enough, and the initial weights were close enough to a solution.

A probabilistic generative model—Recursive Cortical Network (RCN)—for computer vision is able to handle visual object recognition, segmentation, and reasoning in a unified way [37]. RCN outperforms deep neural networks, however, unlike neural networks, RCN algorithms need clean training data.

A powerful framework [189] for image reconstruction was implemented with a deep neural network that learns an optimal reconstruction function. This reconstruction is based upon low-dimensional manifolds defined by real-world data. The framework with a feedforward architecture is composed of fully connected layers followed by a sparse convolutional autoencoder. The fully connected layers approximate the between-manifold projection from the sensor domain to the image domain.

Deep learning takes features from an extremely large and annotated dataset, and classifies patterns buried inside [163]. Deep learning is able to see the structure in too complex data which is hard for human brains. Humans can't detect features that are impossible to catch. The deep learning algorithms make both accurate and explainable predictions, but remain as black boxes in explanations.

In the work related to dimensionality reduction of data with deep neural networks [52], deep autoencoder networks as nonlinear models work much better and effective than principal components analysis (PCA) (linear model) to reduce the dimensionality by transforming the high-dimensional data into a low-dimensional

one. Based on the MNIST handwritten digit recognition task, the best reported error rates are 1.6% for randomly initialized backpropagation. After using the steepest descent and a small learning rate, it achieves 1.2%.

1.8 Organization of This Book

At the end of this chapter, we would like to offer the organization of this book. In this chapter, we outline the development history of deep learning; we also depict our research work and relevant projects as well as the datasets we have collected.

In Chap. 2, we detail deep learning platforms, which include MathWorks® MATLAB system and TensorFlow system for Python-based programming and project development. We also provide the description of programming language R in this chapter for further exploration. We illustrate on data labeling and argumentation as the main outcome of this chapter. This chapter provides the first-hand information for a newcomer to embark on their research projects in deep learning.

In Chap. 3, we review the fundamentals of deep learning, namely Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). The state-of-the-art models such as YOLOv7 and YOLOv8 models and Transformer models are stressed. The Generative Pre-Trained Transformer (GPT) model from OpenAI Laboratory is narrated as the hot-spot subject.

In Chap. 4, we examine Generative Adversarial Network (GAN) and Siamese network as well as Autoencoder as contrastive neural networks. We emphasize the fixed point theory as the base of computational iterations in deep learning.

In Chap. 5, we start from reinforcement learning and detail optimization theory as the main part of our control theory; we also expound other approaches such as dynamic programming for robotic control.

In Chap. 6, we explicate manifold learning and Graph Neural Network (GNN). The general way for dealing with graph problems has been listed. Furthermore, GNNs have been unfolded after manifold learning is delivered.

In Chap. 7, transfer learning is elucidated and ensemble learning has been justified. As a typical method in machine learning, we expect to obtain a strong learner from many weak learners after reading this book.

Among all chapters, a great number of computational methods and algorithms in deep learning are spelled out from a mathematical viewpoint. We anticipate the unique characteristic of mathematical language could provide the best way to convey our ideas and commutable solutions of the research problems in deep learning.

Exercises

Question 1.1. Where does deep learning come from?

Question 1.2. Why is deep learning so important in AI study?

Question 1.3. What are gradient vanishing problem and gradient exploding problem in deep learning? How to avoid them?

Question 1.4. What are the differences between deep learning methods and conventional machine learning methods such as SVM (support vector machine)?

Question 1.5. Why does deep learning affect computer vision, image and video technology, and natural language processing (NLP) so much?

Question 1.6. Which work is the most important one in deep learning history?

Question 1.7. Where will deep learning go?

References

1. Tu L (2011) Introduction to manifold, 2nd Edn
2. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Kudlur M (2016) TensorFlow: a system for large-scale machine learning. In: USENIX symposium on operating systems design and implementation (OSDI), USA, vol 16, pp 265–283
3. Ackley DH, Hinton GE, Sejnowski TJ (1987) A learning algorithm for Boltzmann machines. In: Readings in computer vision, pp 522–533
4. Alexander R (2022) Human facial emotion recognition from digital images using deep learning. Master's thesis, Auckland University of Technology, New Zealand
5. Alpaydin E (2009) Introduction to machine learning. MIT Press
6. Al-Sarayreh M, Reis M, Yan W, Klette R (2018) Detection of red-meat adulteration by deep spectral-spatial features in hyperspectral images. J Imaging 4(5):63
7. Al-Sarayreh M, Reis M, Yan W, Klette R (2019) A sequential CNN approach for foreign object detection in hyperspectral images. In: CAIP, pp 271–283
8. Al-Sarayreh M, Reis M, Yan W, Klette R (2020) Potential of deep learning and snapshot hyperspectral imaging for classification of species in meat. Food Control 117:107332
9. Al-Sarayreh M (2020) Hyperspectral imaging and deep learning for food safety assessment. PhD thesis, Auckland University of Technology, New Zealand
10. An N (2020) Anomalies detection and tracking using Siamese neural networks. Master's thesis, Auckland University of Technology, New Zealand
11. An N, Yan W (2021) Multitarget tracking using Siamese neural networks. ACM Trans Multimed Comput, Commun, Appl 17:1–16
12. Andrews PL, Loftsgaarden DO, Bradshaw LS (2003) Evaluation of fire danger rating indexes using logistic regression and percentile analysis. Int J Wildland Fire 12(2):213–226
13. Baeza-Yates R, Ribeiro-Neto B (2011) Modern information retrieval: the concepts and technology behind search, 2nd edn. Addison-Wesley, U.K

14. Baum LE, Petrie T (1966) Statistical inference for probabilistic functions of finite state Markov chains. Ann Math Stat 37(6):1554–1563

15. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Networks 5(2):157–166

16. Bengio Y, Lecun Y, Hinton G (2021) Deep learning for AI. Commun ACM 64(7):58–65

17. Blake A, Rother C, Brown M, Perez P, Torr P (2004) Interactive image segmentation using an adaptive GMMRF model. In: European conference on computer vision, pp 428–441

18. Cao X (2021) Pose estimation of swimmers from digital images using deep learning. Master's thesis, Auckland University of Technology, New Zealand

19. Caruana R, Lawrence S, Giles CL (2001) Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. In: Advances in neural information processing systems, pp 402–408

20. Cho K (2013) Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images. In: International conference on machine learning, pp 432–440

21. Collobert R, Weston J (2008) A unified architecture for natural language processing: deep neural networks with multitask learning. In: International conference on machine learning, pp 160–167

22. Cover T, Thomas J (1991) Elements of information theory. Wiley

23. Cui W (2014) A scheme of human face recognition in complex environments. Master's thesis, Auckland University of Technology, New Zealand

24. Dabney W et al (2020) A distributional code for value in dopamine-based reinforcement learning. Nature 577:671–675

25. De Boer PT, Kroese DP, Mannor S, Rubinstein RY (2005) A tutorial on the cross-entropy method. Ann Oper Res 134(1):19–67

26. Detwarasiti A, Shachter RD (2005) Influence diagrams for team decision analysis. Decis Anal 2(4):207–228

27. Dosovitskiy A et al (2021) An image is worth 16×16 words: transformers for image recognition at scale. In: International conference on learning representations

28. Dunne RA, Campbell NA (1997) On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In: Australian conference on neural networks, vol 181, p 185

29. Einicke G, White L (1999) Robust extended Kalman filtering. IEEE Trans Signal Process 47(9):2596–2599

30. Ertel W (2017) Introduction to artificial intelligence. Springer International Publishing

31. Fischer A, Igel C (2012) An introduction to restricted Boltzmann machines. In: Iberoamerican congress on pattern recognition, pp 14–36

32. Fu Y (2020) Fruit freshness grading using deep learning. Master's thesis, Auckland University, New Zealand

33. Fu Y (2020) Fruit freshness grading using deep learning. Springer Nat Comput Sci

34. Gao X, Nguyen M, Yan W (2021) Face image inpainting based on generative adversarial network. In: IEEE IVCNZ

35. Gao X (2021) A method for face image inpainting based on generative adversarial networks. Master's thesis, Auckland University of Technology, New Zealand

36. Gashler M, Giraud-Carrier C, Martinez T (2008) Decision tree ensemble: small heterogeneous is better than large homogeneous. In: International conference on machine learning and applications, pp 900–905

37. George D et al (2017) A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs. Science 358(6368)

38. Girshick R (2015) Fast R-CNN. In: IEEE international conference on computer vision, pp 1440–1448

39. Girshick R, Donahue J, Darrell T, Malik J (2016) Region-based convolutional networks for accurate object detection and segmentation. IEEE Trans Pattern Anal Mach Intell 38(1):142–158

40. Gkioxari G, Girshick R, Malik J (2015) Contextual action recognition with R-CNN. In: IEEE ICCV, pp 1080–1088
41. Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: International conference on artificial intelligence and statistics, pp 315–323
42. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. In: International conference on neural information processing systems, pp 2672–2680
43. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press
44. Gottschalk S, Lin MC, Manocha D (1996) OBBTree: a hierarchical structure for rapid interference detection. In: Conference on computer graphics and interactive techniques, pp 171–180
45. Gowdra N, Sinha R, MacDonell S, Yan W (2021) Maximum categorical cross entropy (MCCE): a noise-robust alternative loss function to mitigate racial bias in convolutional neural networks (CNNs) by reducing overfitting. Pattern Recognition
46. Guan Y, Li C, Roli F (2015) On reducing the effect of covariate factors in gait recognition: a classifier ensemble method. IEEE Trans Pattern Anal Mach Intell 37(7):1521–1529
47. Ham Y, Kim J, Luo J (2019) Deep learning for multi-year ENSO forecasts. Nature 573:568–572
48. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: IEEE CVPR, pp 770–778
49. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. In: European conference on computer vision, pp 630–645
50. He K, Gkioxari G, Dollar P, Girshick R (2017) Mask R-CNN. In: IEEE ICCV, pp 2980–2988
51. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
52. Hinton G, Salakhutdinov R (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
53. Hoo-Chang S, Roth HR, Gao M, Lu L, Xu Z, Nogues I, Summers RM (2016) Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE Trans Med Imaging 35(5):1285
54. Huang G, Liu Z, Weinberger KQ, van der Maaten L (2017) Densely connected convolutional networks. In: IEEE CVPR, vol 1, no 2, p 3
55. Itskov M (2011) Tensor algebra and tensor analysis for engineers, 4th edn. Springer, Berlin
56. Jacobson N (2009) Abstract algebra, 2nd edn. Dover Publications
57. Ji H, Liu Z, Yan W, Klette R (2019) Early diagnosis of Alzheimer's disease using deep learning. In: ACM ICCCV, pp 87–91
58. Ji H, Liu Z, Yan W, Klette R (2019) Early diagnosis of Alzheimer's disease based on selective kernel network with spatial attention. In: IAPR ACPR, pp 503–515
59. Jones M, Peet M (2021) A generalization of Bellman's equation with application to path planning, obstacle avoidance and invariant set estimation. Automatica 127:109510
60. Jordan MI, Mitchell TM (2015) Machine learning: trends, perspectives, and prospects. Science 349(6245):255–260
61. Kasabov N (1996) Foundations of neural networks, fuzzy systems, and knowledge engineering. The MIT Press
62. Kim Y (2014) Convolutional neural networks for sentence classification. In: Conference on empirical methods in natural language processing, pp 1746–1751
63. Klette R (2014) Concise computer vision: an introduction into theory and algorithms. Springer, London, U.K
64. Koller D, Friedman N (2009) Probabilistic graphical models. MIT Press
65. Kontschieder P et al (2015) Deep neural decision forests. In: IEEE ICCV
66. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105

67. Kriegeskorte N (2015) Deep neural networks: a new framework for modelling biological vision and brain information processing. Annu Rev Vis Sci 417–446
68. Krizhevsky A, Sutskever I, Hinton G (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
69. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1(4):541–551
70. LeCun Y, Bengio Y (1995) Convolutional networks for images, speech, and time series. In: The handbook of brain theory and neural networks, vol 3361, no 10
71. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
72. LeCun Y, Ranzato M (2013) Deep learning tutorial. In: International conference on machine learning
73. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444
74. Lee CY, Gallagher PW, Tu Z (2016) Generalizing pooling functions in convolutional neural networks: mixed, gated, and tree. In: Artificial intelligence and statistics, pp 464–472
75. Li C (2022) Special character recognition using deep learning. Master's thesis, Auckland University of Technology, New Zealand
76. Li R (2017) Computer input of morse codes using finger gesture recognition. Master's thesis, Auckland University of Technology, New Zealand
77. Li S (2009) Markov random field modeling in image analysis. Springer, Berlin
78. Li X (2018) Preconditioned stochastic gradient descent. IEEE Trans Neural Netw Learn Syst 29(5):1454–1466
79. Liang C, Lu J, Yan W (2022) Human action recognition from digital videos based on deep learning. In: ICCCV 2022. China
80. Liang S (2021) Multi-language datasets for speech recognition based on the end-to-end framework. Master's thesis, Auckland University, New Zealand
81. Liang S, Yan W (2022) A hybrid CTC+Attention model based on end-to-end framework for multilingual speech recognition. Multimed Tools Appl
82. Lidl R, Niederreiter H (1997) Finite fields, 2nd edn. Cambridge University Press (ISBN 0-521-39231-4)
83. Lin TY, Dollar P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: CVPR, vol 1, no 2, p 4
84. Littman M (2015) Reinforcement learning improves behavior from evaluative feedback. Nature 521:445–451
85. Liu C, Yan W (2020) Gait recognition using deep learning. In: Handbook of research on multimedia cyber security. IGI Global, pp 214–226
86. Liu J (2022) Crime prediction from digital videos using deep learning. Master's thesis, Auckland University of Technology, New Zealand
87. Liu J, Pan C, Yan W (2023) Litter detection from digital images using deep learning. SN Comput Sci 4(134)
88. Liu M, Yan W Masked face recognition using MobileNetV2. In: ACM ICCCV
89. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) SSD: single shot multibox detector. In: European conference on computer vision, pp 21–37
90. Liu X (2019) Vehicle-related scene understanding using deep learning. Master's thesis, Auckland University of Technology, New Zealand
91. Liu X, Yan W, Kasabov N (2020) Vehicle-related scene segmentation using CapsNets. In: IEEE IVCNZ, pp 1–6
92. Liu X, Yan W (2021) Traffic-light sign recognition using capsule network. Multimed Tools Appl 80(10):15161–15171
93. Liu Y (2022) Sign language recognition from digital videos using feature pyramid network with detection transformer. Master's thesis, Auckland University of Technology

94. Liu Z, Yan WQ, Yang ML (2018) Image denoising based on a CNN model. In: International conference on control, automation and robotics, pp 389–393

95. Liu Z (2018) Comparative evaluations of image encryption algorithms. Master's thesis, Auckland University of Technology

96. Liu Z et al (2021) Swin transformer: hierarchical vision transformer using shifted windows. In: IEEE ICCV

97. Lu J (2016) Empirical approaches for human behavior analytics. Master's thesis, Auckland University of Technology, New Zealand

98. Lu J (2021) Deep learning methods for human behavior recognition. PhD thesis, Auckland University, New Zealand

99. Lu J, Nguyen M, Yan W (2020) Deep learning methods for human behavior recognition. In: IVCNZ

100. Lu J, Nguyen M, Yan W (2021) Sign language recognition from digital videos using deep learning methods. In: ISGV, pp 108–118

101. Luo Z, Nguyen M, Yan W (2021) Sailboat detection based on automated search attention mechanism and deep learning models. In: IEEE IVCNZ

102. Luo Z, Nguyen M, Yan W (2022) Kayak and sailboat detection based on the improved YOLO with transformer. In: ACM ICCCV

103. Luo Z (2022) Sailboat and kayak detection using deep learning methods. Master's thesis, Auckland University of Technology, New Zealand

104. Ma X (2020) Banknote serial number recognition using deep learning. Master's thesis, Auckland University of Technology, New Zealand

105. Ma X, Yan W (2021) Banknote serial number recognition using deep learning. Multimed Tools Appl 80(12):18445–18459

106. Mack E, Shoemaker T (2006) Distribution transformers. In: The lineman's and Cableman's handbook. McGraw-Hill, pp 1–22

107. Manning C, Raghavan P, Schutze H (2008) Introduction to information retrieval. Cambridge University Press

108. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5(4):115–133

109. Mehtab S (2022) Deep neural networks for road scene perception in autonomous vehicles using LiDARs and vision sensors. PhD thesis, Auckland University of Technology, New Zealand

110. Mi M (2021) Monitoring safe following distances for vehicles from digital images using deep learning. Research report, Auckland University of Technology, New Zealand

111. Mignan A, Broccardo M (2019) One neuron versus deep learning in aftershock prediction. Nature 574(7776):E1–E3

112. Mnih V et al (2015) Human-level control through deep reinforcement learning. Nature 518:529–533

113. Molchanov VV, Vishnyakov BV, Vizilter YV, Vishnyakova OV, Knyaz VA (2017) Pedestrian detection in video surveillance using fully convolutional YOLO neural network. In: Automated visual inspection and machine vision II, vol 10334

114. Muscat J (2014) Functional analysis. Springer, Berlin

115. Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Networks 1(1):4–27

116. Niculescu-Mizil A, Caruana R (2007) Inductive transfer for Bayesian network structure learning. In: International conference on artificial intelligence and statistics

117. Nie GH, Zhang P, Niu X, Dou Y, Xia F (2017). Ship detection using transfer learned single shot multi box detector. In: ITM Web of Conferences, vol 12, p 01006

118. Norvig P, Russell S (2016) Artificial intelligence: a modern approach, 3rd edn. Prentice Hall

119. Pan S, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359

120. Qi J, Nguyen M, Yan W (2022) Waste classification from digital images using ConvNeXt. PSIVT, Springer LNCS, p 13836
121. Qi J, Nguyen M, Yan W (2022) Small visual object detection in smart waste classification using Transformers with deep learning. IVCNZ, Springer LNCS, p 13763
122. Qin Z, Yan W (2021) Traffic-sign recognition using deep learning. In: International symposium on geometry and vision, pp 13–25
123. Rahlf T (2017) Data visualisation with R. Springer International Publishing
124. Reddy G et al (2018) Glider soaring via reinforcement learning in the field. Nature 562:236–239
125. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: IEEE CVPR, pp 779–788
126. Reichstein M et al (2019) Deep learning and process understanding for data-driven Earth system science. Nature 566:195–204
127. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
128. Ren Y (2017) Banknote recognition in real time using ANN. Master's thesis, Auckland University of Technology, New Zealand
129. Ronneberger O, Fischer P, Brox T (2015) U-Net: convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention. Springer, Berlin, pp 234–241
130. Rriedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting. Ann Stat 38(2):337–374
131. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by backpropagating errors. Nature 323(6088):533–536
132. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Berg AC (2015) ImageNet large scale visual recognition challenge. Int J Comput Vision 115(3):211–252
133. Sarikaya R, Hinton GE, Deoras A (2014) Application of deep belief networks for natural language understanding. IEEE/ACM Trans Audio, Speech, Lang Process 22(4):778–784
134. Scarselli F, Gori M, Tsoi A, Hagenbuchner M, Monfardini G (2009) The graph neural network model. IEEE Trans Neural Networks 20(1):61–80
135. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117
136. Shen Y, Yan W (2018) Blindspot monitoring using deep learning. In: IEEE IVCNZ
137. Shen D, Chen X, Nguyen M, Yan WQ (2018) Flame detection using deep learning. In: International conference on control, automation and robotics, pp 416–420
138. Shrivastava A et al (2017) Learning from simulated and unsupervised images through adversarial training. In: IEEE CVPR
139. Song C, He L, Yan W, Nand P (2019) An improved selective facial extraction model for age estimation. In: IEEE IVCNZ
140. Sun S (2020) Empirical analysis for earlier diagnosis of Alzheimer's disease using deep learning. Master's thesis, Auckland University of Technology, New Zealand
141. Sutton R, Barto A (2018) Reinforcement learning: an introduction, 2nd edn. MIT Press
142. Stoer J, Bulirsch R (1991) Introduction to numerical analysis, 2nd edn. Springer, Berlin
143. Sutton R, Barto A (2018) Reinforcement learning: an introduction. MIT Press
144. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA (2016) Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: AAAI, vol 4, p 12
145. Tang A, Lu K, Wang Y, Huang J, Li H (2015) A real-time hand posture recognition system using deep neural networks. ACM Trans Intell Syst Technol 6(2):21
146. Tong D, Yan W (2022) Visual watermark identification from the transparent window of currency by using deep learning. In: Applications of encryption and watermarking for information security. IGI Global
147. Van Hasselt H (2011) Double Q-learning. In: Advances in neural information processing systems, pp 2613–2622

148. Vaswani A et al (2017) Attention is all you need. In: The conference on neural information processing systems (NIPS), USA

149. Vedaldi A, Lenc K (2015) MatConvNet: convolutional neural networks for matlab. In: ACM international conference on multimedia, pp 689–692

150. Veit A, Wilber MJ, Belongie S (2016) Residual networks behave like ensembles of relatively shallow networks. In: Advances in neural information processing systems, pp 550–558

151. Vinyals O et al (2019) Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575:350–354

152. Wan L, Zeiler M, Zhang S, Le Cun Y, Fergus R (2013) Regularization of neural networks using DropConnect. In: International conference on machine learning, pp 1058–1066

153. Wang C, Bochkovskiy A, Liao H (2022) YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv:2207.02696

154. Wang H (2018) Real-time face detection and recognition based on deep learning. Master's thesis, Auckland University of Technology

155. Wang H, Yan W (2012) Face detection and recognition from distance based on deep learning. In: Aiding forensic investigation through deep learning and machine learning. IGI Global

156. Wang L, Yan W (2021) Tree leaves detection based on deep learning. In: ISGV. Springer, Berlin, pp 26–38

157. Wang X, Yan W (2019) Gait recognition using multichannel convolution neural networks. Springer Neural Comput Appl

158. Wang X, Yan W (2020) Multi-perspective gait recognition based on ensemble learning. Springer Neural Comput Appl 32:7275–7287

159. Wang X, Yan W (2020) Human gait recognition based on frame-by-frame gait energy images and convolutional long short term memory. Int J Neural Syst 30(1):1950027

160. Wang X, Yan W (2021) Non-local gait feature extraction and human identification. Multimed Tools Appl 80(4):6065–6078

161. Wang X, Yan W (2021) Human gait recognition based on SAHMM. IEEE/ACM Trans Biol Bioinform 18(3):963–972

162. Wang Y (2021) Colorizing grayscale CT images of human lung using deep learning. Master's thesis, Auckland University, New Zealand

163. Webb S (2018) Deep learning for biology. Nature 554:555–557

164. Wei J, He J, Chen K, Zhou Y, Tang Z (2017) Collaborative filtering and deep learning based recommendation system for cold start items. Expert Syst Appl 69:29–39

165. Welling M, Kingma D (2019) An introduction to variational autoencoders. Found Trends Mach Learn 12(4):307–392

166. Wu B, Iandola F, Jin PH, Keutzer K (2017) SqueezeNet: unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: IEEE conference on computer vision and pattern recognition workshops, pp 129–137

167. Xiang Y, Yan W (2021) Fast-moving coin recognition using deep learning. Multimed Tools Appl 80(16):24111–24120

168. Xiao B, Nguyen M, Yan W (2021) Apple ripeness identification using deep learning. In: International symposium on geometry and vision, pp 53–67

169. Xiao B, Nguyen M, Yan W (2021) Fruit ripeness identification using Transformer models. Available at SSRN: https://ssrn.com/abstract=4129908 or https://doi.org/10.2139/ssrn.4129908

170. Xin C (2018) Detection and recognition for multiple flames using deep learning. Master's thesis, Auckland University of Technology, New Zealand

171. Xing C, Ma L, Yang X (2016) Stacked denoise autoencoder based feature extraction and classification for hyperspectral images. J Sens

172. Xing J, Yan W Traffic sign recognition using guided image filtering. Springer ISGV, pp 85–99

173. Xing J (2021) Traffic sign recognition from digital images using deep learning. Master's thesis, Auckland University of Technology, New Zealand

174. Yan W, Kankanhalli M, Wang J (2005) Analogies based video editing. Multimedia Syst 11:3–18

175. Yao W, Zeng Z, Lian C, Tang H (2018) Pixel-wise regression using U-Net and its application on pansharpening. Neurocomputing 312:364–371

176. Yeh CY, Su WP, Lee SJ (2011) Employing multiple-kernel support vector machines for counterfeit banknote recognition. Appl Soft Comput 11(1):1439–1447

177. Yu Z (2021) deep learning methods for human action recognition. Master's thesis, Auckland University, New Zealand

178. Zamir A et al (2018) Taskonomy: disentangling task transfer learning. In: IEEE CVPR

179. Zanaty EA (2012) Support vector machines (SVMs) versus multilayer perception (MLP) in data classification. Egypt Inform J 13(3):177–183

180. Zarchan P, Musoff H (2000) Fundamentals of Kalman filtering: a practical approach. American Institute of Aeronautics and Astronautics, Incorporated

181. Zeng K, Yu J, Wang R, Li C, Tao D (2017) Coupled deep autoencoder for single image super-resolution. IEEE Trans Cybern 47(1):27–37

182. Zhang L (2020) Virus identification from digital images using deep learning. Master's thesis, Auckland University, New Zealand

183. Zhang Q (2018) Currency recognition using deep learning. Master's thesis, Auckland University of Technology, New Zealand

184. Zhang Q, Yan W, Kankanhalli M (2019) Overview of currency recognition using deep learning. J Bank Financ Technol 3(1):59–69

185. Zhang Y (2016) A virtual keyboard implementation based on finger recognition. Master's thesis, Auckland University of Technology, New Zealand

186. Zhao K (2021) Fruit detection using CenterNet. Master's thesis, Auckland University, New Zealand

187. Zhao K, Yan W (2021) Fruit detection from digital images using CenterNet. In: International symposium on geometry and vision, pp 313–326

188. Zheng K, Yan WQ, Nand P (2018) Video dynamics detection using deep neural networks. IEEE Trans Emerg Top Comput Intell 2(3):224–234

189. Zhu B et al (2018) Image reconstruction by domain-transform manifold learning. Nature 555:487–492

190. Zhu Y, Yan W (2022) Image-based storytelling using deep learning. ACM ICCCV, China

191. Zhu Y, Yan W (2022) Traffic sign recognition based on deep learning. Multimed Tools Appl 81:17779–17791

Deep Learning Platforms

2

There are a plethora of deep learning platforms available at present. The famous one is MATLAB deep learning toolbox developed by MathWorks which simplifies deep learning computations and reduces the workload of debugging in programming. Other platforms include Python-based platforms such as Google Colaboratory "Colab" and TensorFlow. The platforms such as software R and software WEKA also could be applied to the deep learning computations. In this chapter, we will introduce these platforms one by one.

2.1 Introduction

There are a pretty assortment of deep learning platforms available such as Caffe, TensorFlow, MXNet, Torch, and Theano. The platform Caffe (i.e., Convolutional Architecture for Fast Feature Embedding) is a deep learning framework, which originally was developed at the University of California, Berkeley. As shown in Fig. 2.1, Caffe supports visual object detection and classification as well as image segmentation by using CNN, R-CNN, LSTM, and fully connected neural networks (FCNN). GPU-based and CPU-based acceleration have been taken into consideration in the original version. The platform Caffe2 includes new features such as recurrent neural networks (RNN). At the end of March 2018, Caffe2 was merged into software PyTorch.

MXNet was adopted by Amazon Web Services which supports a number of programming languages, such as C++, Python, and R. MXNet is a flexible and ultra-scalable deep learning framework that contains the state-of-the-art methods in deep learning, including convolutional neural networks (CNNs) and long short-term memory networks (LSTMs). MXNet also was applied to both imperative and symbolic programming.
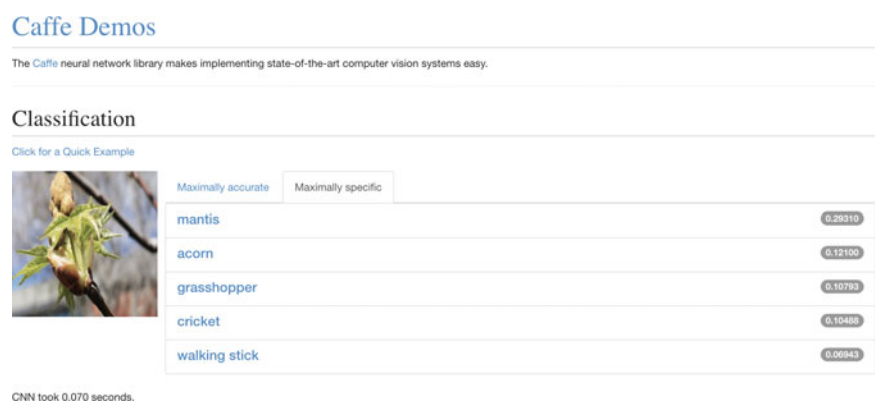
Fig. 2.1 Visual object classification by using the platform Caffe

Fig. 2.2 An example of NumPy with OpenCV and Python source code

Usually, Python includes the libraries such as NumPy (i.e., N-dimensional array package), SciPy (i.e., fundamental library for scientific computing), Matplotlib (i.e., comprehensive 2D plotting), and Scikit-learn (i.e., machine learning library).

NumPy is the fundamental package for scientific computing with Python. Besides its obvious scientific usages, NumPy is also utilized as an efficient multidimensional container of generic data. Arbitrary data types are defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases for big data processing. An example of Python source code with NumPy and OpenCV is shown in Fig. 2.2.

Matplotlib is a plotting library for Python programming and its mathematical extension, which is a platform for data visualization. The platform is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python, but it is free and open-source as shown in Fig. 2.3.

Scikit-learn (i.e., scikits.learn or sklearn) is a free machine learning library for Python programming which features various classification, regression, and clustering algorithms including SVM, random forests, gradient boosting, k-means, etc. It also
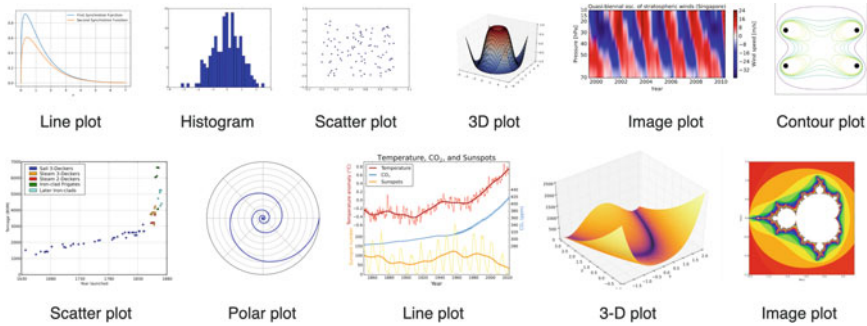
Fig. 2.3 Various plots of Matplotlib

provides various tools for data fitting, preprocessing, model selection, and evaluation, which was designed to interoperate with NumPy, SciPy, Matplotlib, etc.

All these Python libraries could be installed and updated by using the command line window of an operating system (OS) such as Microsoft Windows, Apple Mac Operating System (macOS), and Linux.

Flask is a micro web framework written in Python. The microframework does not require particular tools or libraries, which build a web application quickly by using only a single Python file. The Python-based web framework provides useful features that make creating web applications in Python easier.

Theano is a Python-based optimizing compiler for mathematical expressions, especially matrix-valued ones, which was developed by a Montreal Institute for Learning Algorithms (MILA). Theano is one of the most stable libraries, which allows automatic function gradient computations with its Python interface.

PyTorch is an open-source machine learning library for the applications such as computer vision and natural language processing (NLP). It was primarily developed by Facebook's AI Research Lab (FAIR). The platform PyTorch defines a class called Tensor to store and operate on homogeneous multidimensional rectangular arrays of numbers.

Transformers are the current state-of-the-art type of model for dealing with sequences, e.g., in text processing and machine translation. Transformers were introduced in 2017 by Google Brain for NLP problems, replacing RNN models (LSTM). Python Transformers is a library dedicated to supporting Transformer-based architectures and facilitating the distribution of pre-trained models. Transformers was designed to be extensible and simple for practitioners, and fast and robust in industrial deployments. Transformers is domain-specific which allows the system to provide automatic support for model analysis, usage, deployment, benchmarking, and easy replicability.

2.2 MATLAB for Deep Learning

MATLAB is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks®. MATLAB allows matrix manipulations, plotting functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

MathWorks® logo is a surface plot of a variant of an eigenfunction of the L-shaped membrane. If $t \in (0, \infty)$ is time, $x \geq 0$ and $y \geq 0$ are spatial coordinates with the units chosen so that the wave propagation speed is equal to one, then the amplitude of a wave satisfies the partial differential equation as shown in Eq. (2.1):

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}. \tag{2.1}$$

Periodic time behavior gives solutions of the form

$$u(t, x, y) = sin(y\sqrt{t}))v(x, y), \tag{2.2}$$

where

$$\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \lambda v = 0, \tag{2.3}$$

where λ is the eigenvalue and the corresponding function $v(x, y)$, $x, y \in \mathcal{R}$ is the eigenfunction.

In MATLAB, the membrane function as shown in Fig. 2.4 is applied to generate the MATLAB logo. $L = membrane(k)$, $k = 1, 2, \ldots, 12$ is the kth eigenfunction of the L-shaped membrane.
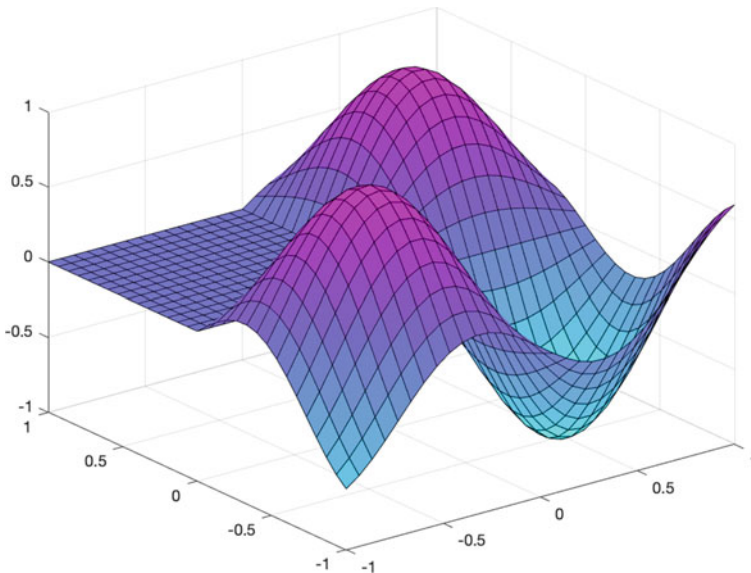
Fig. 2.4 The MATLAB membrane function with the parameter $k = 3$
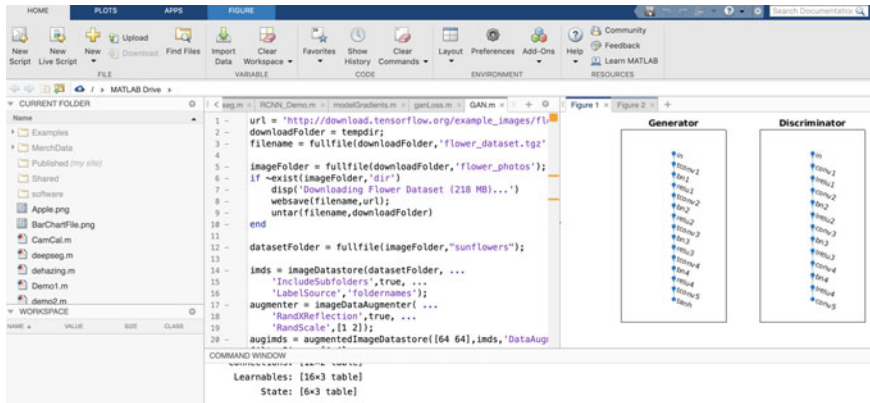
Fig. 2.5 The interface of MATLAB online

MATLAB utilizes toolboxes and the command line window to run programs. Especially, deep learning toolbox has been added into MATLAB in 2017. In 2019, MATLAB introduced generative adversarial networks (GANs), Siamese networks, variational autoencoders, and attention networks. MATLAB deep learning toolboxes also combine CNN and LSTM layers of neural networks that include 3D CNN layers [65].

MATLAB has its online version as shown in Fig. 2.5. The interfaces of MATLAB online version and offline version are almost the same. If the license is authorized and available, it is very convenient to access the software system and generate results. MATLAB provides demonstrations, documentations, and source codes for further development [65].

MATLAB provided ANN (i.e., artificial neural network) toolbox in the early days which told us how to deal with real applications, usually including function fitting (e.g., nftool), pattern classification (e.g., nprtool), clustering (e.g., nctool), time series prediction and modeling (e.g., ntstool), etc.

In the methodology, generally, we need to collect training data as the first step, configure a network, initialize weights, and train the network. We need to minimize the differences, and validate the classification. The confusion matrix is used to evaluate the results. ROC (i.e., receiver operating characteristic curve) and AUC (i.e., area under the curve) are calculated based on the classification. The ROC curve is plotted with TPR against FPR where TPR is on the y-axis and FPR is on the x-axis, which depicts relative trade-offs between benefits (i.e., true positive) and costs (i.e., false positive):

$$TPR = \frac{TP}{TP + FN}, \tag{2.4}$$

where *TP* is true positive (i.e., hit), and *FN* is false negative (i.e., correct rejection):

$$FPR = \frac{FP}{TN + FP}, \tag{2.5}$$

where *FP* is false positive (i.e., false alarm), and *TN* is true negative (i.e., miss).

An excellent model has AUC near 1.0 which means it has a good measure of separability. A poor model has AUC near 0.0 which means it has the worst measure of separability.

Deep learning toolbox provides a framework for designing and implementing deep neural networks, pre-trained models, and APPs. MATLAB has the toolbox since 2017 that includes transfer learning, LSTM network for time series analysis, etc. The latest version encapsulates AlexNet, GoogLeNet, VGG-16/VGG-19, ResNet-101, Inception v2, Generative Adversarial Network (GAN), reinforcement learning, YOLOv4, etc. MATLAB has the applications associated with multiple GPUs, parallel computing, cluster computing, cloud computing, etc., for accelerating the deep learning process.

Deep learning has been applied to time series analysis and forecast. Time series analysis comprises the methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed data. Typical methods for time series analysis include spectral analysis, wavelet analysis, autocorrelation, and cross-correlation analysis. MATLAB provides autoregressive(AR) and autoregressive integrated moving average models (ARIMA), and state-space models. In MATLAB deep learning toolbox, LSTM network as a kind of RNNs (i.e., Recurrent neural networks) has been applied to time series analysis and natural language processing (NLP) as well as text analytics, which benefits us to write a good essay or correct our writings.

We can generate new letters or words using deep learning algorithms such as LSTM. MATLAB LSTM network is able to generate text word-by-word. The first word of the text is generated by sampling a word from a probability distribution according to the first words of the text in the training data. The remaining words are produced by using the trained LSTM network to predict the next time step with the current sequence of generated text. We keep generating words one-by-one until the network predicts the end.

MATLAB provides the function of transfer learning. That means, a network, e.g., AlexNet, has been well trained; we are able to utilize the well-trained parameters and apply them to a new network. We need to load the pre-trained network, replace the final layers, and train the network again. After this transfer, if we train the new network again, we are able to get a better result. This will reduce the computing time. MATLAB also could make a neural network run faster after the model optimization.

MATLAB has embedded Fast R-CNN and Faster R-CNN algorithms (i.e., regions with convolutional neural networks) already; an example for stop sign detection has been provided; the famous 11 lines source code is shown in Fig. 2.6. The simplest deep learning network is able to accomplish the specified task effectively and efficiently [29, 55].

```
camera = webcam;                 % Connect to camera
picture = snapshot(camera);      % Take picture
imshow(picture)                  % Show picture

% Run live video capture and classification in loop
for n = 1:100
    picture = snapshot(camera);              % Take picture
    picture = imresize(picture,[227,227]);   % Resize picture
    label = classify(nnet, picture);         % Classify picture
    imshow(picture);                         % Show picture
    title(char(label))                       % Show label
    drawnow
end
```

Fig. 2.6 The famous 11 lines source code of MATLAB

MATLAB at present is able to run most of the deep learning algorithms by using both desktop version and online version as shown in Fig. 2.5. If an account is registered, it is easily to login. MATLAB users are able to develop their own toolboxes. MATLAB provides GUI for users to easy interactions. MATLAB shows experimental results or outcomes visually. We may use the GUI interface to develop our applications.

MATLAB provides computer vision toolbox, especially for autonomous vehicles, visual object detection, semantic segmentation, digital image processing, etc. MATLAB deep learning is employed for biometrics, e.g., detection and recognition of human face [3,16,66,74], fingerprint, voice, aging, gait [39,45], etc. The reason is that deep learning is able to find the latent patterns behind the given datasets.

MATLAB provides cloud computing and parallel computing. This support is employed for face detection, visual object detection, vehicle detection, lane detection, pedestrian detection, etc.

In the latest version, MATLAB supports artificial intelligence, event-based modeling, etc. In the latest version of deep learning toolbox, MATLAB users are now able to create generative adversarial networks (GANs), Siamese networks, reinforcement learning, variational autoencoders, and attention networks.

MATLAB deep learning examples can implement the feature What you see is what you get, which does not need complicated system configuration and debugging; it is very convenient for source code transplanting from one computer to another computer, especially through MATLAB Online, a cloud-based system. MATLAB deep learning examples could be applied to teaching and education as well as simulations in scientific computations, the framework greatly reduces our working time and uplifts our working efficiency.

2.3 TensorFlow for Deep Learning

TensorFlow (https://www.tensorflow.org/) is a platform developed by the Google
Brain team and has been applied to deep learning. TensorFlow is run on a desktop
(Microsoft Windows, macOS, Linux, etc.) or online Colaboratory (Colab) as shown
in Fig. 2.7.

Google Colab allows anybody to write and execute arbitrary Python code through
the browser, and is especially well suited to machine learning, data analysis, and
education. Colab is a cloud-based software and greatly reduces the debugging prob-
lems in software development, successfully provides GPU services online, and runs
entirely in the cloud. Through Colab, we write and execute codes, save and share
our experience, develop web, and access powerful computing resources all through
web browsers, without complicated configuration and software version matching
problems from the open-source developers.
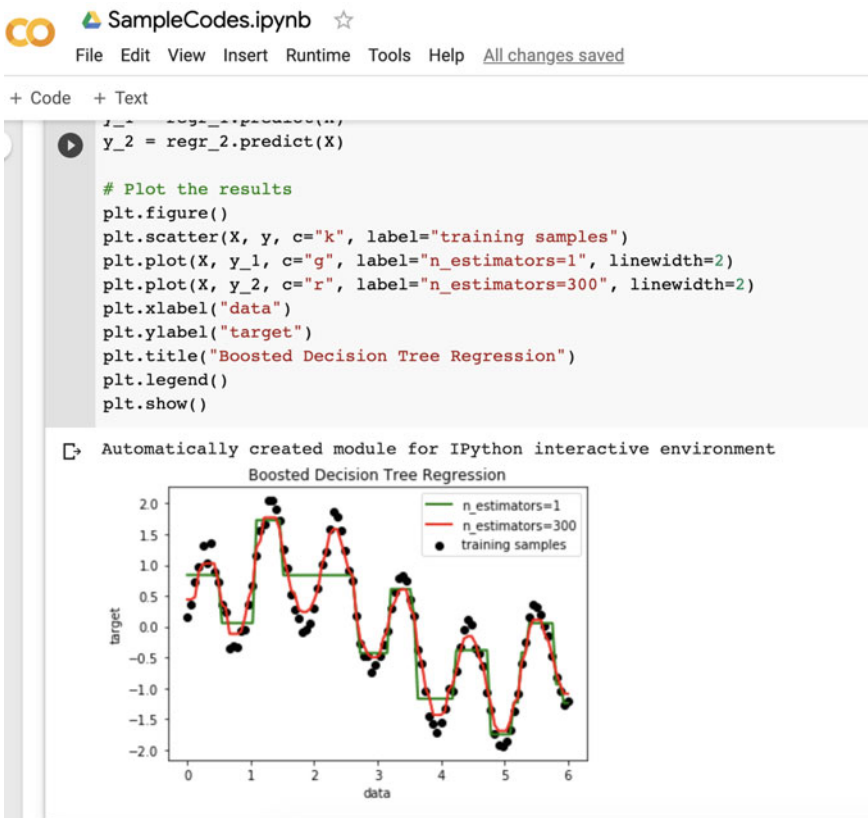
Fig. 2.7 Google Colaboratory

Tensor is a generalization of vectors and matrices to potentially higher dimensions. Generally speaking, in a tensor, the elements of a vector or matrix are still scalars, vectors, or matrices. TensorFlow is a framework to define and run computations involving tensors and vectors.

TensorFlow was especially developed for big data processing and visualization with TensorBoard [1] using graphs; numerical methods are found from TensorFlow. TensorFlow presents tensors as *n*-dimensional arrays using base data types. These types reveal the relationships between different datasets.

TensorFlow not only has the normal data types but also includes special types, such as shape, variable, constant, and placeholder. The concept rank refers to a mathematical entity such as scalar, vector, and matrix.

The installation of TensorFlow is based on macOS/Unix, Microsoft Windows, Ubuntu/Linux, etc. After Python 3.0, pip3 is used to install Python-based applications. The command is

C:> pip3 install—upgrade tensorflow

TensorFlow needs a session to show the output, usually working along with the print command together to show the output of variables; for example, the famous "Hello, TensorFlow!" program is shown in Fig. 2.8.

A TensorFlow session encapsulates the state of runtime and operations. A session represents a connection between the client program, which accesses hardware devices from local machine and remote devices using the distributed TensorFlow runtime library.

We have the exemplar source code to show the instance "add", "multiply", "dot product", "zero", etc. The "optimizer" assists us to quickly find a proper gradient

Fig. 2.8 Hello, TensorFlow!

using weights or variables from SGD (i.e., Stochastic Gradient Descent) algorithms. For example, if $z(x, y) = x^2 + xy, x, y, z \in \mathscr{R}$, then the gradients are

$$
\begin{cases}
\frac{\partial z(x,y)}{\partial x} = 2x + y \\
\frac{\partial z(x,y)}{\partial y} = x.
\end{cases}
\tag{2.6}
$$

In order to minimize the function $z(x, y)$, a standard gradient descent method would perform the following iterations or batches:

$$
\begin{cases}
x' = x - \eta \cdot \frac{\partial z(x,y)}{\partial x} \\
y' = y - \eta \cdot \frac{\partial z(x,y)}{\partial y}
\end{cases}
\tag{2.7}
$$

where η is a step size or the learning rate in machine learning. Provided $\eta = 0.1$, we randomly select $x = 5.0$, $y = 3.0$, using Eq. (2.7), then $x' = 3.7$ and $y' = 2.5$. We repeat this procedure; let $(x, y) \leftarrow (x', y')$; since $\eta < 1.0$, (x, y) will be converged and approximate to the local extrema point, namely

$$
\begin{cases}
x_{n+1} = x_n - \eta \cdot \frac{\partial z}{\partial x_n} \\
y_{n+1} = y_n - \eta \cdot \frac{\partial z}{\partial y_n}
\end{cases},
\tag{2.8}
$$

where $n = 1, 2, \ldots$, $\lim_{n\to\infty}(x_{n+1} - x_n) = 0$, $\lim_{n\to\infty}(y_{n+1} - y_n) = 0$, $\lim_{n\to\infty}(x_n, y_n) = (x_p, y_p)$, $\mathbf{P}(x_p, y_p)$ is the local extreme point, and $n, p \in \mathscr{Z}^+$.

TensorFlow graph is to show the computational network construction. The nodes (operations) and edges (tensors) indicate how individual operations are composed together. TensorFlow collections are stored by using metadata.

TensorBoard provides the visualization and tooling needed for machine learning. TensorBoard is to render a computational graph through browsers like Microsoft IE or Edge, Google Chrome, Apple Safari, etc. TensorBoard can visualize the model graph, display images, text, and audio data, etc. TensorBoard is launched by using the following command line:

c:> tensorboard—logdir = "$\cdots$ \tensorflow\graph"

Before that, we need to save the computational graph to a summary file using the function "tf.summary.FileWriter($\cdot$)". TensorBoard visualizes the structure of a graph in a browser under the support of a http server. The visualized result could be downloaded from the website:

http://localhost:6006/#graphs.

We show the graph of a neural network structure of a TensorFlow application as in Fig. 2.9; the training accuracy of the TensorFlow application is shown in Fig. 2.10. We list two figures from TensorBoard as shown in Fig. 2.11 which reveal the dataset visualization using TensorFlow and TensorBoard.

The MNIST database is a large database of handwritten digits that is used for training various image processing systems. The MNIST database contains 60,000
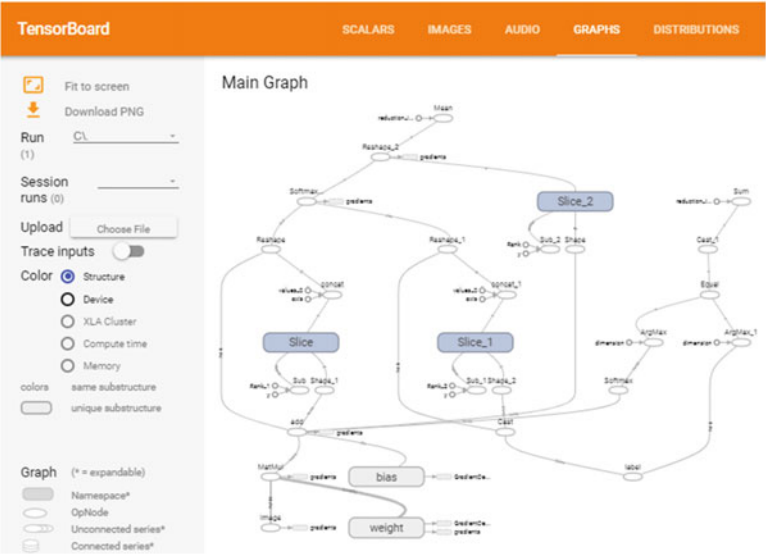
Fig. 2.9 A TensorFlow graph of a network structure
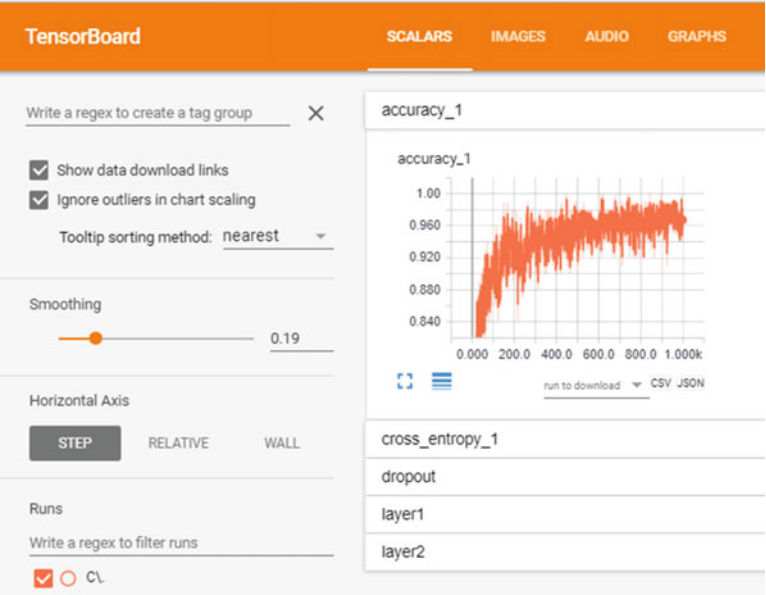
Fig. 2.10 Accuracy graph of a TensorFlow training

Fig. 2.11   TensorFlow graph rendering **a** data samples rendering and **b** data labels rendering

training images and 10,000 test images. An extended dataset similar to MNIST called
EMNIST has been published in 2017, which contains 240,000 training images, and
40,000 test images of handwritten digits and characters.

For example, the steps of CNN estimator for MNIST (modified NIST) dataset are
listed as

- **Step 1**. Load training and validation data.
- **Step 2**. Create the estimator/call the CNN model function.
- **Step 3**. CNN model function: Convolutional layers, pooling layers, and fully con-
  nected layers.
- **Step 4**. Set up logging for predictions.
- **Step 5**. Train the model.
- **Step 6**. Evaluate the model and print results.

The steps of RNN routine using the MNIST dataset are listed as

- **Step 1**. Set hyperparameters.
- **Step 2**. TensorFlow graph input.
- **Step 3**. Define weights.
- **Step 4**. Run RNN model function.
- **Step 5**. Hidden layer for output as the final results.

2.4   Data Augmentation and Labeling

Image augmentation is a set of processing operations of digital images, such as image
cropping, resizing, rotating, shearing, flipping, and reflection as well as the artifacts
such as lens distortions, adding noises, and blurs [37, 77].

In our projects, two distinct forms of data augmentation have been applied to
human face detection so as to generate image translations and horizontal reflec-
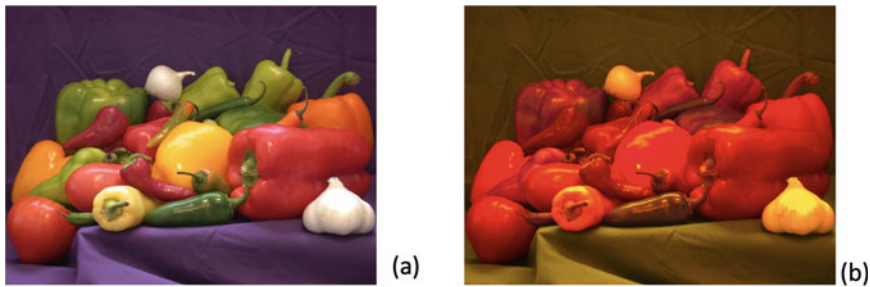
Fig. 2.12 A result of color jittering using PCA algorithm, **a** original image and **b** image after color jittering

tions, and alter the intensities of the RGB channels in training images [66]. In the project, currency recognition [10,46,56,78,79] and flame detection [35,60,69], and image processing operations were taken into account such as scaling to the uniform size, clipping or expanding, cropping, randomly rotating, and color adjusting. In the project of banknotes, serial number recognition [46], the image augmentation approaches include image rotating, translating, color jittering, and adding Gaussian noises.

The color jitter enables us to alter the colors of an image by applying a random color variation. For example, we specify the range of hue, saturation, and gain value (HSV) for the random colors. We also calculate principal components by using PCA algorithm of each color matrix of an image and generate new variations by adding offset to the principal components. An example of color jittering is shown in Fig. 2.12.

The method of data augmentation [66] is to enhance the training data by using mathematical transformations such as Affine transformation, in order to achieve the purpose of improving the accuracy of object recognition. The two data augmentations respectively are to generate image translations and horizontal reflections and to alter the intensities of the RGB channels in training images. Both methods are able to convey multiple images from the original one with very few computations.

In the project for anomaly detection and object tracking [4,5], the data augmentation includes geometric transformation, Affine transformation, noise injection, random erasing, and so on. In the project vehicle-related scene understanding [42–44], offline augmentation and online augmentation are regarded as the two categories of data enrichment. The online augmentation includes rotations, translations, flipping, etc., and the offline augmentation is typically explored for small datasets, which increases the size of the dataset by using a factor which equals the number of conversions performed. In most of the cases, multiple transformation methods are amalgamated together to achieve a much comprehensive expansion.

In the project for traffic-sign recognition [70], an algorithm based on dark channel prior, namely a guided image filtering algorithm was proposed for image defogging. By using visual features of the guided image filtering, haze image preprocessing for traffic signs achieved the results of image denoising, image smoothing, and fog removal. Furthermore, there are generally two types of image defogging algorithms:

one is histogram equalization, which simply enhances the contrast of the image. The other is an image restoration-based defogging algorithm [80], which makes use of original images to compare with the foggy images so as to reconstruct the new image [71].

In the project for fruit freshness grading [19,20], the image augmentation includes image scaling, rotating, cropping, and adding random noises based on observations. For adding random noises, the sequential order is random brightness adjustment, random contrast, and random erosion for digital images.

In the project related to virus identification from digital images [75,76], there are 6,000 electron microscopy images evenly (approximately) for the four classes: SARS, MESR, HIV, and COVID-19. The data preprocessing methods are based on electron microscopy image augmentation as well as image quality enhancement. The image augmentation includes image rotation, image random region removal, image Jaccard Index Crop, and image resizing. The data augmentation aims to improve the predictive capability of a model by adding disturbances to the raw visual features presented in source images.

In the project related to special symbol recognition [38] using deep learning, three types of image augmentations have been employed to the input images, including Mosaic data augmentation, self-adaptive anchor calculation, and self-adaptive image re-scaling. The mosaic augmentation takes four images and combines them into one image.

In the project for character-based Braille classification [40], data augmentation for enhancing the robustness of the model includes rotating each image in random angles, combining the rotated images with the original dataset. OpenCV was applied to import the images, and resize them into 52×52, then normalize the pixel intensity between [0, 1.0]. The processed data will be randomly shuffled and separated into training set, test set, and validation set.

In the project related to pose estimation of swimmers from digital images using deep learning [9], the amount of data in the dataset was increased through data augmentation, including three ways: Randomly rotating the images, randomly scaling the image, and flipping the image horizontally randomly. The methods substantially improve object detection speed.

In the project related to tree leaf recognition using deep learning [67], data augmentation operations include flipping, zooming in, zooming out, clipping, and combining. The final test was conducted by using augmented samples against a wood floor background.

MATLAB provides image augmentations by using the image processing toolbox: Random image warping transformations, cropping transformations, color transformations, synthetic noise, and synthetic blur. The details could be found on the website: https://au.mathworks.com/help/deeplearning/ug/preprocess-data-for-domain-specific-applications.html.

An image data augmenter has been designed for a set of preprocessing options of image augmentation, such as resizing, rotating, and reflecting. All of these operations could be written in a mathematical way. For example, after the rotating and scaling of an image $I(x, y)$, we will obtain image $I'(x', y')$, where $(x, y) \in \Omega = [1, W] \times [1, H]$,
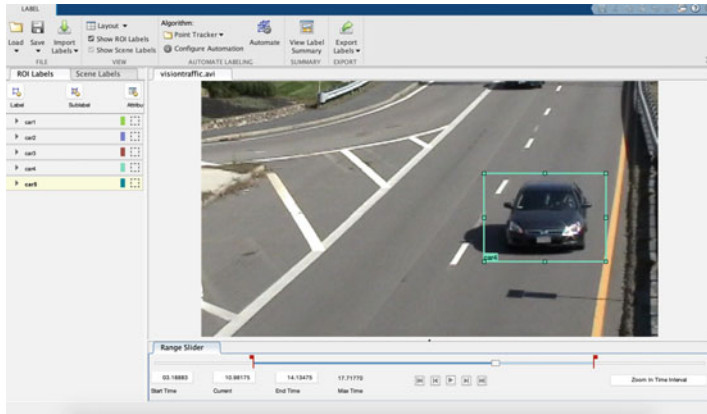
Fig. 2.13 MATLAB video labeler

and W and H are the width and height of image I, respectively. $(x', y') \in \Omega' = [1, W'] \times [1, H']$, and W' and H' are the width and height of image I', respectively:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x \cdot cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & s_y \cdot cos(\alpha) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{2.9}$$

where $\alpha \in [0, 360°)$ is the rotation angle, and (s_x, s_y), $s_x, s_y \in \mathcal{R}$ are scale factors along the x-axis and y-axis, respectively.

MATLAB also has a software Image Labeler for training data that could reduce our human labor. The Image Labeler and Video Labeler provide an easy way to mark the rectangular region of interest (ROI) labels, polyline ROI labels, pixel ROI labels, and scene labels in a video or image sequence. The video labeler automatically labels across image frames using an automation algorithm, e.g., the Kanade–Lucas–Tomasi (KLT) algorithm based on point tracking [37] as shown in Fig. 2.13. Following the steps (loading images, ROIs, labeling, data augmentation, exporting results, etc.), we annotate all sampled images. The region of interest (ROI) will be marked and output for model training and object classification. This will be applied to train computer algorithms on what the visual objects look like in a visual scene [42–44].

Correspondingly, Python has a graphical image annotation tool Labelme as shown in Fig. 2.14, which exports .json file format for image segmentation. The standard command "pip install labelme" or "pip3 install labelme" can install the software in a command line window. In Fig. 2.14, we label the regions of hair, hat, face, and shoulder of the standard test image Lenna using polygons. The manually annotated polygons and the labels could be applied for visual object detection, pattern classification, and segmentation.
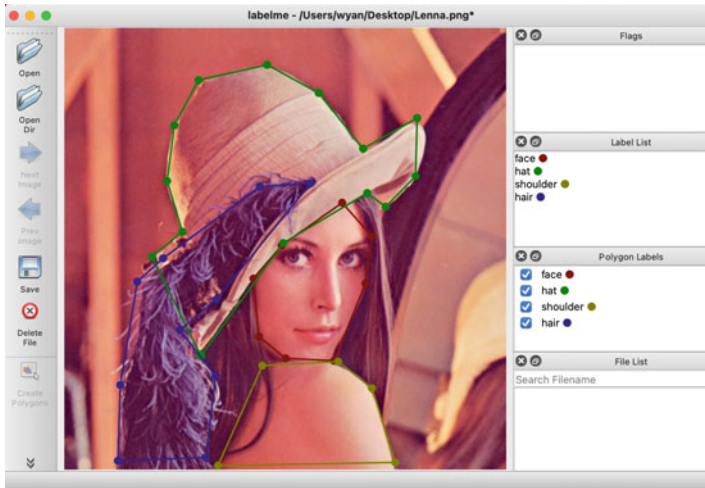
Fig. 2.14 Python image labeler: LabelMe

2.5   R for Deep Learning

R is a programming language and free software environment for statistical comput-
ing and graphics supported by the R foundation for statistical computing [7,14,51].
R was named partly after the first names of the first two 'R' authors. The R language
is widely used among statisticians and data miners for developing statistical soft-
ware and data analysis. R and its libraries are implemented with various statistical
and graphical methods, including linear and nonlinear modeling, classical statistical
tests, spatial and time series analysis, classification, clustering, and others. R is an
interpreted language; R users typically access it through a command line interpreter.
R is available as free software with general public license. R compiles source code
and runs them on a wide variety of UNIX platforms as well as Microsoft Windows,
macOS, and others.

In 1991, **R**oss Ihaka and **R**obert Gentleman at the University of Auckland, New
Zealand, began an alternative implementation of the basic S language. R is an inter-
preted language. R supports matrix arithmetic. R's data structures include vectors,
matrices, arrays, data frames (similar to tables in a relational database), and lists.

R is a well developed, simple and effective programming language which includes
conditionals, loops, user-defined recursive functions, and input and output facilities.

R is case-sensitive as most UNIX-based packages. Elementary commands consist
of either expressions or assignments. The commands are separated by a semi-colon
(';') or a newline. Elementary commands can be grouped together into one compound
expression by using braces ('{' and '}'). The comments can be put almost anywhere,
starting with a hashmark ('#'). If a command is not complete at the end of a line, R
will give a different prompt, by default '+'. Command lines entered at the console
are limited to about 4,095 bytes.

An external command file may be executed at any time in an R session with the command:

$$> source(\text{“}commands.R\text{”}).$$

The elementary arithmetic operators are the usual: '+', '−', '*', '/', and '^'. Two statistical functions are $mean(\cdot)$ and $var(\cdot)$. $sort(x)$ returns a vector of the same size as $x \in \mathscr{R}$ with the elements arranged in increasing order. $max(\cdot)$ and $min(\cdot)$ select the largest and smallest values in the arguments. $seq(\cdot)$ is for generating sequences. $rep(\cdot)$ is used for replicating an object in various ways. $is.na(x)$ gives a logical vector of the same size as x with value 'TRUE' IFF the corresponding element in x is 'NA' (not available). We now list more functions as follows:

- $paste(\cdot)$ takes a number of arguments and concatenates them into character strings.
- $matrix(\cdot)$ and $array(\cdot)$ are simpler and natural looking assignments.
- $outer(\cdot)$ generates outer product array whose data vector is got by forming all possible products of elements of the data vector.
- $aperm(a, perm)$ is used to permute an array a.
- $nrow(A)$ and $ncol(A)$ give the number of rows and columns in the matrix $\mathbf{A}$.
- $crossprod(X, y)$ is the same as $t(X)\% * \%y$.
- $diag(M)$ gives the vector of the main diagonal entries of $\mathbf{M}$.
- $solve(\mathbf{A}, \mathbf{b})$ solves the system $\mathbf{A}\mathbf{x} = \mathbf{b}$, returning $\mathbf{x}$.
- $eigen(\mathbf{S}_m)$ calculates the eigenvalues and eigenvectors of a symmetric matrix $\mathbf{S}_m$.
- $svd(\mathbf{M})$ calculates the singular value decomposition of $\mathbf{M}$.
- $lsfit(\cdot)$ returns a list giving results of a least squares fitting procedure.
- $lm(\cdot)$ for regression modeling.
- $cbind(\cdot)$ forms matrices by binding together matrices horizontally, or column-wise, and $rbind(\cdot)$ vertically, or row-wise.

R function is defined by an assignment of the form:

$$name < -function(arg\_1, arg\_2, \ldots)\{expression\}.$$

A call to R function usually takes the form: $name(expr\_1, expr\_2, \ldots)$. The basic function for fitting ordinary multiple models is $lm(\cdot)$:

$$fitted.model < -lm(formula, data = data.frame).$$

The R function to fit a generalized linear model is $glm(\cdot)$:

$$fitted.model < -glm(formula, family = family.generator, data = data.frame).$$

High-level plotting functions create a new plot on the graphics device, possibly with axes, labels, titles, and so on. Low-level plotting functions add more information to an existing plot, such as extra points, lines, and labels. Interactive graphics functions allow to interactively add information, or extract information, from an existing plot, using a pointing device such as a mouse. An example of R plotting is shown in Fig. 2.15.

```
install.packages("caTools")  # install external package
library(caTools)             # external package providing write.gif function
jet.colors <- colorRampPalette(c("red", "blue", "#007FFF", "cyan", "#7FFF7F",
                                 "yellow", "#FF7F00", "red", "#7F0000"))
dx <- 1500                   # define width
dy <- 1400                   # define height
C  <- complex(real = rep(seq(-2.2, 1.0, length.out = dx), each = dy),
              imag = rep(seq(-1.2, 1.2, length.out = dy), dx))
C <- matrix(C, dy, dx)       # reshape as square matrix of complex numbers
Z <- 0                       # initialize Z to zero
X <- array(0, c(dy, dx, 20)) # initialize output 3D array
for (k in 1:20) {            # loop with 20 iterations
  Z <- Z^2 + C               # the central difference equation
  X[, , k] <- exp(-abs(Z))   # capture results
}
write.gif(X, "Mandelbrot.gif", col = jet.colors, delay = 100)
```
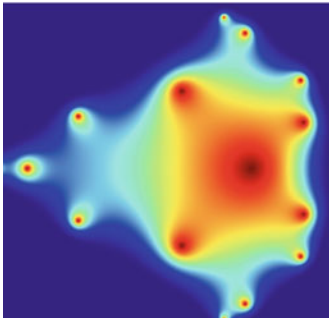

Fig. 2.15 An example of R plotting

All R functions and datasets are stored in packages. R users connected to the Internet are able to utilize the *install.packages*() and *update.packages*(). In order to see which packages are currently loaded, function *search*() takes this role. With regard to a list of all available help topics in an installed package, please refer to *help.start*().

2.6 Fundamental Mathematics

MATLAB was designed primarily for numerical analysis, especially all variables in MATLAB are arrays or vectors. TensorFlow derives from the operations on multi-dimensional data arrays, which are referred to as tensors. For better understanding the implementations of deep learning algorithms in MATLAB, we introduce the fundamental knowledge of mathematics related to MATLAB.

For any real numbers $x, y, z \in \mathcal{R}$, we have the rules such as associative and commutative relationships with regard to operations '$+$', '$-$', '$\cdot$', and '$\div$', i.e., $(x + y) + z = x + (y + z)$, $x + y = y + x$. In real analysis, we have infinity (i.e., positive infinity: $+\infty$; negative infinity: $-\infty$); we also define the operations: $\infty \pm \infty$, $\frac{\infty}{\infty}$, etc.

In real analysis, we talk about the concept set. Based on the sets of real numbers, we construct function mapping from one set onto another. A function $f(x)$, $x \in [a, b]$ is continuous over an interval $[a, b]$, $a, b, x \in \mathcal{R}$, namely $f(x) \in \mathbf{C}[a, b]$:

$$\lim_{x \to x_0} f(x) = f(x_0) = \lim_{x \to x_0^+} f(x) = f(x_0^+) = \lim_{x \to x_0^-} f(x) = f(x_0^-), \qquad (2.10)$$

where $x, x_0, x_0^+, x_0^- \in [a, b]$, $f(x) \in \mathbf{C}[a, b]$. If $f(x), g(x) \in \mathbf{C}[a, b]$, then $f(x) \pm g(x) \in \mathbf{C}[a, b]$, $f(x) \cdot g(x) \in \mathbf{C}[a, b]$, $\frac{f(x)}{g(x)} \in \mathbf{C}[a, b]$, $g(x) \neq 0$.

The differentiable means

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} = \lim_{x \to x_0^+} \frac{f(x) - f(x_0^+)}{x - x_0^+} = f'(x_0^+)$$

$$= \lim_{x \to x_0^-} \frac{f(x) - f(x_0^-)}{x - x_0^-} = f'(x_0^-). \qquad (2.11)$$

If $f'(x)$ and $g'(x) \in \mathbf{C}[a, b]$, then $f'(x) \pm g'(x) \in \mathbf{C}[a, b]$, $f'(x) \cdot g'(x) \in \mathbf{C}[a, b]$, $\frac{f'(x)}{g'(x)} \in \mathbf{C}[a, b]$, $g'(x) \neq 0$.

In the chain rule, if $f(x) = g(y)$, $y = h(x)$, then $f(x) = g(h(x))$,

$$\frac{\partial f(x)}{\partial x} = \frac{\partial g(y)}{\partial y} \cdot \frac{\partial y}{\partial x} = \frac{\partial g(y)}{\partial y} \cdot \frac{\partial h(x)}{\partial x}. \qquad (2.12)$$

Regarding Taylor expansion, given $f(x) \in \mathbf{C}[a, b]$, we have

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2!}f^{(2)}(x - x_0)^2 + \cdots + \frac{1}{k!}f^{(k)}(x - x_0)^k + R_n(\xi), \qquad (2.13)$$

where $R_n(\xi)$ is the remainder of Taylor expansion or Taylor remainder

$$R_n(\xi) = \frac{1}{n!}f^{(n)}(\xi - x_0)^n, \xi \in [a, b]. \qquad (2.14)$$

Taylor expansion shows us all continuous functions defined over $[a, b]$ that are converted to polynomials. Typically, $\lim_{x \to 0} \frac{\sin x}{x} = 1$, $\sin x \approx x$.

We interpolate a curve by using the given support points. The typical polynomials are quadratic curves, cubic polynomials, Bezier functions, B-spline functions, etc. [17].

Pertaining to Lagrange interpolation functions, we have a polynomial with the degree n,

$$f(x) = \sum_{i=0}^{n} \prod_{i=0; i \neq j}^{n} \frac{(x - x_i)}{(x_j - x_i)} \cdot y_i \qquad (2.15)$$

where (x_i, y_i), $y_i = f(x_i)$, $i = 0, 1, \ldots, n$.

A vector space is a set $\mathbf{x}, \mathbf{y} \in \mathbf{V}$ satisfying the following axioms:

- $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$ (addition is commutative).
- $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$ (addition is associative).
- $\exists$ a unique vector zero 0, such that $0 + \mathbf{x} = \mathbf{x}, \forall \mathbf{x} \in \mathbf{V}$.
- $\forall \mathbf{x} \in \mathbf{V}, \exists$ a unique vector $-\mathbf{x}$ such that $\mathbf{x} + (-\mathbf{x}) = 0$.

Regarding every pair $\alpha, \beta \in \mathscr{R}$ (real number) and $\mathbf{x} \in \mathscr{R}^{|\mathbf{V}|}$ (vector), there $\exists$ a vector $\alpha \mathbf{x}$, called scalar product of α and $\mathbf{x}$, such that

- $\alpha(\beta \mathbf{x}) = (\alpha \beta)\mathbf{x}$ (multiplication by scalars is associative).
- $1\mathbf{x} = \mathbf{x}$.
- $\alpha(\mathbf{x} + \mathbf{y}) = \alpha \mathbf{x} + \alpha \mathbf{y}$ (distributive with respect to vector addition).
- $(\alpha + \beta)\mathbf{x} = \alpha \mathbf{x} + \beta \mathbf{x}, \alpha, \beta \in \mathscr{R}$ and $\mathbf{x} \in \mathbf{V}$ (distributive with respect to scalar addition).

A vector space has the properties:

- A basis in a vector space $\mathbf{V}$ is a set $\mathbf{G} = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n\} \subset \mathbf{V}$ of linearly independent vectors such that every vector in $\mathbf{V}$ is a linear combination of elements of $\mathbf{G}$.
- A vector space $\mathbf{V}$ is finite-dimensional $|\mathbf{V}| < \infty$ if it has a finite basis $\mathbf{G} = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n\}, n < \infty$.
- The dimension of a finite-dimensional vector space $\mathbf{V}$ is the number of elements in a basis $\{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n\}$ of $\mathbf{V}$, namely $|\mathbf{V}| = n$.
- Let $\mathbf{G} = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n\}$ be a basis of an n-dimensional vector space $\mathbf{V}$. Then, $\mathbf{x} = \sum_{i=1}^{n} x^i \mathbf{g}_i$ is Einstein's summation convention, $\mathbf{x} \in \mathbf{V}$.

The scalar (inner) product is a real-valued function $\mathbf{x} \cdot \mathbf{y}$ of two vectors $\mathbf{x}$ and $\mathbf{y}$ in a vector space $\mathbf{V}$.

- $\mathbf{x} \cdot \mathbf{y} = \mathbf{y} \cdot \mathbf{x}$ (commutative rule).
- $\mathbf{x} \cdot (\mathbf{y} + \mathbf{z}) = \mathbf{x} \cdot \mathbf{y} + \mathbf{x} \cdot \mathbf{z}$ (distributive rule).
- $\alpha(\mathbf{x} \cdot \mathbf{y}) = (\alpha \mathbf{x}) \cdot \mathbf{y} = \mathbf{x} \cdot (\alpha \mathbf{y}), \forall \alpha \in \mathscr{R}, \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{V}$ (associative rule for the multiplication by a scalar).
- $\mathbf{x} \cdot \mathbf{x} \geq 0, \forall \mathbf{x} \in \mathbf{V}, \mathbf{x} \cdot \mathbf{x} = 0$ if and only if $\mathbf{x} = 0$.
- Euclidean length (also called norm) of a vector $\mathbf{x}$, $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x} \cdot \mathbf{x}}$.
- Two non-zero vectors $\mathbf{x}$ and $\mathbf{y}$ are called orthogonal $\mathbf{x} \perp \mathbf{y}$, if $\mathbf{x} \cdot \mathbf{y} = 0$.
- A basis $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ of an n-dimensional Euclidean space $\mathscr{E}^n$ is orthonormal if $\mathbf{e}_i \cdot \mathbf{e}_j = \delta_{ij}, i, j = 1, 2, \ldots n$.
  $$\delta_{ij} = \delta^{ij} = \delta^i_j = \delta^j_i = \begin{cases} 1 & \mathbf{x} = \mathbf{y} \\ 0 & \mathbf{x} \neq \mathbf{y} \end{cases} \text{ (Kronecker delta).}$$
- $\mathbf{e}_1 = \frac{\mathbf{x}_1}{\|\mathbf{x}_1\|}, \ldots, \mathbf{e}_n = \frac{\mathbf{e}'_n}{\|\mathbf{e}'_n\|}$ is the Gram–Schmidt process, where $\mathbf{e}'_n = \mathbf{x}_n - (\mathbf{x}_n, \mathbf{e}_{n-1})\mathbf{e}_{n-1} \cdots - (\mathbf{x}_n, \mathbf{e}_1)\mathbf{e}_1$.

Let $\mathbf{G} = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n\}$ be a basis in n-dimensional Euclidean space $\mathscr{E}^n$; a basis $\mathbf{G}' = \{\mathbf{g}'_1, \mathbf{g}'_2, \ldots, \mathbf{g}'_n\}$ is dual to basis $\mathbf{G}$ if $\mathbf{g}_i \cdot \mathbf{g}'_j = \delta_{ij}$, $i, j = 1, 2, \ldots, n$. $\mathbf{g}_i$ are linearly independent, if $\sum \alpha_i \mathbf{g}_i = 0$, then $\alpha_i = 0$.

The length of vector $\mathbf{x}$ thus is written by

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}_i \mathbf{g}_i \cdot \mathbf{x}'_j \mathbf{g}'_i} = \sqrt{\mathbf{x}_i \cdot \mathbf{x}'_j \cdot \delta_{ij}} = \sqrt{\mathbf{x}_i \cdot \mathbf{x}'_j}. \tag{2.16}$$

For example,

$$\mathbf{G} = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\} = \{\mathbf{e}_2 \times \mathbf{e}_3, \mathbf{e}_3 \times \mathbf{e}_1, \mathbf{e}_1 \times \mathbf{e}_2\}. \tag{2.17}$$

In complex analysis, also known as the theory of functions of a complex variable, for any complex function, the values $\mathbf{z} \in \mathscr{C}$ from the domain and their images $f(\mathbf{z})$ in the range may be separated into real and imaginary parts:

$$\mathbf{z} = a + b\mathbf{i}, \mathbf{i} = \sqrt{-1}, a, b \in \mathscr{R}, \tag{2.18}$$

and

$$f(\mathbf{z}) = f(x + \mathbf{i}y) = u(x, y) + \mathbf{i}v(x, y), \tag{2.19}$$

where x and y are real variables, and $u(x, y)$ and $v(x, y)$ are all real-valued functions.

In the complex analysis, the continuous function $f(\mathbf{z})$ at $\mathbf{z}_0, \mathbf{z} \in \mathscr{C}$ is defined as

$$f(\mathbf{z}_0) = \lim_{\mathbf{z} \to \mathbf{z}_0} f(\mathbf{z}) = \lim_{\mathbf{z} \to \mathbf{z}_0^+} f(\mathbf{z}) = f(\mathbf{z}_0^+) = f(\mathbf{z}_0^-) = \lim_{\mathbf{z} \to \mathbf{z}_0^-} f(\mathbf{z}). \tag{2.20}$$

In the complex analysis, the derivative of holomorphic function $f(\mathbf{z})$ at $\mathbf{z}_0, \mathbf{z} \in \mathscr{C}$ is defined as

$$f'(\mathbf{z}_0) = \lim_{\mathbf{z} \to \mathbf{z}_0} \frac{f(\mathbf{z}) - f(\mathbf{z}_0)}{\mathbf{z} - \mathbf{z}_0} = f'(\mathbf{z}_0^+) = f'(\mathbf{z}_0^-). \tag{2.21}$$

For vector space in complex analysis, if $< \mathbf{x}, \mathbf{y} >= \mathbf{x} + \mathbf{i}\mathbf{y}, \mathbf{i} = \sqrt{-1}$, then

$$< \alpha + \mathbf{i}\beta >< \mathbf{x}, \mathbf{y} >=< \alpha\mathbf{x} - \beta\mathbf{y})(\beta\,\mathbf{x} + \alpha\mathbf{y} >, \tag{2.22}$$

where $\mathbf{x}, \mathbf{y} \in \mathscr{E}^n, \mathbf{z} = \mathbf{x} + \mathbf{i}\mathbf{y} \in \mathscr{C}^n$,

$$\mathbf{A}(\mathbf{x} + \mathbf{i}\mathbf{y}) = \mathbf{A}\mathbf{x} + \mathbf{i}(\mathbf{A}\mathbf{y}), \tag{2.23}$$

where $\mathbf{A} \in \mathscr{T}^n$.

For a tensor space, let $\mathbf{L}^n$ be a set of all linear mappings of one vector into another within $\mathscr{E}^n, \mathbf{y} = \mathbf{A}\mathbf{x}, \mathbf{x}, \mathbf{y} \in \mathscr{E}^n, \mathbf{A} \in \mathscr{T}^n$:

- Tensor linearity: $\mathbf{A}(\mathbf{x} + \mathbf{y}) = \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{y}, \forall \mathbf{x}, \mathbf{y} \in \mathscr{E}^n, \forall \mathbf{A} \in \mathscr{T}^n$.
- $\mathbf{A}(\alpha\mathbf{x}) = \alpha(\mathbf{A}\mathbf{x}), \forall \mathbf{x} \in \mathscr{E}^n, \forall \alpha \in \mathscr{R}, \forall \mathbf{A} \in \mathscr{T}^n$.
- Product of a tensor by a scalar number: $(\alpha\mathbf{A})\mathbf{x} = \alpha(\mathbf{A}\mathbf{x}) = \mathbf{A}(\alpha\mathbf{x}), \forall \mathbf{x} \in \mathscr{E}^n$.
- Sum of tensors: $(\mathbf{A} + \mathbf{B})\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{x}$.
- Negative tensor: $-\mathbf{A} = (-1)\mathbf{A}$.
- Zero tensor: $\mathbf{0}\mathbf{x} = 0, \forall \mathbf{x} \in \mathscr{E}^n$.

- Addition commutative: $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$.
- Addition associative: $\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$.
- Element $\mathbf{0}$: $\mathbf{0} + \mathbf{A} = \mathbf{A}$, $\mathbf{A} + (\text{-}\mathbf{A}) = \mathbf{0}$.
- Multiplication by scalars is associative: $\alpha(\beta\mathbf{A}) = (\alpha\beta)\mathbf{A}$.
- Element $\mathbf{1}$: $\mathbf{1A} = \mathbf{A}$.
- Multiplication by scalars is distributive with respect to tensor addition: $\alpha(\mathbf{A}+\mathbf{B}) = \alpha\mathbf{A} + \alpha\mathbf{B}$.
- Multiplication by scalars is distributive with respect to scalar addition: $(\alpha+\beta)\mathbf{A} = \alpha\mathbf{A} + \beta\mathbf{A}$, $\alpha, \beta \in \mathscr{R}$, $\mathbf{A}, \mathbf{B} \in \mathscr{L}^n$.
- Vector product in $\mathscr{E}^3$, i.e., $\mathbf{z} = \mathbf{x} \times \mathbf{y}$, $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathscr{E}^3$.
- Rotation tensor: $\mathscr{R}(a)$, $a \in \mathbf{E}^3$ and $\mathscr{R} \in \mathscr{L}^3$.

For tensor functions, we have

- Function continuity:
$$\lim_{t \to t_0} \mathbf{x}(t) = \mathbf{x}(t_0), \tag{2.24}$$

and
$$\lim_{t \to t_0} \mathbf{A}(t) = \mathbf{A}(t_0). \tag{2.25}$$

- Differentiable:
$$\frac{d\mathbf{x}(t)}{dt} = \lim_{s \to 0} \frac{\mathbf{x}(t+s) - \mathbf{x}(t)}{s}, \tag{2.26}$$

and
$$\frac{d\mathbf{A}(t)}{dt} = \lim_{s \to 0} \frac{\mathbf{A}(t+s) - \mathbf{A}(t)}{s}. \tag{2.27}$$

- The product of a scalar function with a vector- or tensor-valued function:
$$\frac{d}{dt}[u(t)\mathbf{x}(t)] = \frac{du}{dt}\mathbf{x}(t) + \frac{d\mathbf{x}}{dt}u(t), \tag{2.28}$$

and
$$\frac{d}{dt}[u(t)\mathbf{A}(t)] = \frac{du}{dt}\mathbf{A}(t) + \frac{d\mathbf{A}}{dt}u(t). \tag{2.29}$$

- The scalar product of two vector- or tensor-valued functions:
$$\frac{d}{dt}[\mathbf{x}(t) \cdot \mathbf{y}(t)] = \frac{d\mathbf{x}}{dt} \cdot \mathbf{y}(t) + \mathbf{x}(t) \cdot \frac{d\mathbf{y}}{dt}, \tag{2.30}$$

and
$$\frac{d}{dt}[\mathbf{A}(t) : \mathbf{B}(t)] = \frac{d\mathbf{A}}{dt} : \mathbf{B}(t) + \mathbf{A}(t) : \frac{d\mathbf{B}}{dt}. \tag{2.31}$$

- The composition of two tensor-valued functions
$$\frac{d}{dt}[\mathbf{A}(t)\mathbf{B}(t)] = \frac{d\mathbf{A}}{dt}\mathbf{B}(t) + \mathbf{A}(t)\frac{d\mathbf{B}}{dt}. \tag{2.32}$$

- **Aa** $= \lambda \, \mathbf{a}, \mathbf{a} \neq 0, \mathbf{bA} = \lambda \mathbf{b}, \mathbf{b} \neq 0, \mathbf{a}, \mathbf{b} \in \mathscr{C}^n, \lambda \in \mathscr{C}$, and $\mathbf{A} \in \mathscr{L}^n$. λ is an eigenvalue of tensor **A**, $g(\mathbf{A}) = \sum_{k=0}^{m} a_k \mathbf{A}^k$, then

$$g(\lambda) = \sum_{k=0}^{m} a_k \lambda^k. \tag{2.33}$$

- Chain rule:

$$\frac{d}{dt} \mathbf{x}[u(t)] = \frac{d\mathbf{x}}{du} \frac{du}{dt}, \tag{2.34}$$

$$\frac{d}{dt} \mathbf{A}[u(t)] = \frac{d\mathbf{A}}{du} \frac{du}{dt}. \tag{2.35}$$

- The chain rule for functions with multiple arguments:

$$\frac{d}{dt} \mathbf{x}[u(t), v(t)] = \frac{d\mathbf{x}}{du} \frac{du}{dt} + \frac{d\mathbf{x}}{dv} \frac{dv}{dt}, \tag{2.36}$$

$$\frac{d}{dt} \mathbf{A}[u(t), v(t)] = \frac{d\mathbf{A}}{du} \frac{du}{dt} + \frac{d\mathbf{A}}{dv} \frac{d\mathbf{v}}{dt}, \tag{2.37}$$

$$\frac{d}{dt} [\mathbf{A}(t)\mathbf{B}(t)] = \frac{d\mathbf{A}}{dt} \mathbf{B}(t) + \frac{d\mathbf{B}}{dt} \mathbf{A}(t). \tag{2.38}$$

- Directional derivatives:

$$\mathbf{r} = (\theta_1, \ldots, \theta_n), \theta_i \in \mathscr{R}. \tag{2.39}$$

For scalar field:

$$\frac{d}{ds} \Phi(\mathbf{r} + s\mathbf{a}) = \text{grad}\Phi \cdot \mathbf{a}, \forall \mathbf{a} \in \mathscr{E}^n, s \in \mathscr{R}. \tag{2.40}$$

For vector field, we have

$$\frac{d}{ds} \mathbf{x}(\mathbf{r} + s\mathbf{a}) = \text{grad}\mathbf{x} \cdot \mathbf{a}, \forall \mathbf{a} \in \mathscr{E}^n, s \in \mathscr{R}. \tag{2.41}$$

For tensor field, we have

$$\frac{d}{ds} \mathbf{A}(\mathbf{r} + s\mathbf{a}) = \text{grad}\mathbf{A} \cdot \mathbf{a}, \forall \mathbf{a} \in \mathscr{E}^n, s \in \mathscr{R}. \tag{2.42}$$

Exercises

Question 2.1. Please list the advantages and disadvantages of MATLAB labelers and Python labeler.

Question 2.2. What are general methods for data augmentation?

Question 2.3. How to utilize the source codes and datasets from the GitHub website?

Question 2.4. What's the relationship between deep learning and machine learning? What are the differences between supervised learning and unsupervised learning?

Question 2.5. How to choose an algorithm effectively for visual object detection and recognition as well as pattern classification?

Question 2.6. What is the relationship between AI and deep learning?

Question 2.7. Why mathematics, especially computational mathematics is so important in deep learning and AI?

Question 2.8. What are the general methods for data analysis and data visualization?

Question 2.9. Please list popular software for deep learning study.

References

1. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Kudlur M (2016) TensorFlow: a system for large-scale machine learning. In: USENIX symposium on operating systems design and implementation (OSDI), USA, vol 16, pp 265–283
2. Aizenberg NN, Aizenberg IN, Krivosheev GA (1996) CNN based on universal binary neurons: learning algorithm with error-correction and application to impulsive-noise filtering on grayscale images. In: IEEE international workshop on cellular neural networks and their applications, pp 309–314
3. Albu RD (2009) Human face recognition using convolutional neural networks. J Electr Electron Eng 2:110
4. An N (2020) Anomalies detection and tracking using siamese neural networks. Master thesis, Auckland University of Technology, New Zealand
5. An N, Yan W (2021) Multitarget tracking using Siamese neural networks. ACM Trans Multimed Comput, Commun, Appl 17:1–16
6. Basu AP, Ebrahimi N (1991) Bayesian approach to life testing and reliability estimation using asymmetric loss function. J Stat Plan Inference 29(1–2):21–31
7. Bloomfield V (2014) Using R for numerical analysis in science and engineering. Chapman & Hall/CRC
8. Cao G, Xie X, Yang W, Liao Q, Shi G, Wu J (2018) Feature-fused SSD: fast detection for small objects. In: International conference on graphic and image processing (ICGIP), vol 10615
9. Cao X (2021) Pose estimation of swimmers from digital images using deep learning. Master's thesis, Auckland University of Technology, New Zealand
10. Chambers J (2013) Digital currency forensics. Master's thesis, Auckland University of Technology, New Zealand
11. Chatfield C (2004) The analysis of time series: an introduction. Chapman & Hall/CRC
12. Chatzis SP, Kosmopoulos DI (2011) A variational Bayesian methodology for hidden Markov models utilizing student's-t mixtures. Pattern Recogn 44(2):295–306
13. Chen J, Kang X, Liu Y, Wang Z (2015) Median filtering forensics based on convolutional neural networks. IEEE Signal Process Lett 22(11):1849–1853
14. Crawley M (2014) Statistics: an introduction using R, 2nd edn. Wiley

15. Dai J, Li Y, He K, Sun J (2016) R-FCN: object detection via region-based fully convolutional networks. In: Advances in neural information processing systems, pp 379–387
16. Farfade SS, Saberian MJ, Li LJ (2015) Multi-view face detection using deep convolutional neural networks. In: International conference on multimedia retrieval, pp 643–650
17. Farin G (1993) Curves and surfaces for computer-aided geometric design: a practical guide, 3rd edn. Academic Press
18. Fu R, Zhang Z, Li L (2016) Using LSTM and GRU neural network methods for traffic flow prediction. In: Youth academic annual conference of Chinese association of automation (YAC)
19. Fu Y (2020) Fruit freshness grading using deep learning. Masters thesis, Auckland University, New Zealand
20. Fu Y, Nguyen M, Yan W (2022) Grading methods for fruit freshness based on deep learning. Springer Comput Sci
21. Gers FA, Schmidhuber J (2000) Recurrent nets that time and count. In: IEEE-INNS-ENNS international joint conference on neural networks, vol 3, pp 189–194
22. Gers FA, Schmidhuber J, Cummins F (2000) Learning to forget: continual prediction with LSTM. Neural Comput 12(10):2451–2471
23. Gers FA, Schmidhuber E (2001) LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE Trans Neural Networks 12(6):1333–1340
24. Gers FA, Schraudolph NN, Schmidhuber J (2002) Learning precise timing with LSTM recurrent networks. J Mach Learn Res 3:115–143
25. Giusti A, Ciresan DC, Masci J, Gambardella LM, Schmidhuber J (2013) Fast image scanning with deep max-pooling convolutional neural networks. In: IEEE international conference on image processing, pp 4034–4038
26. Gu Q, Yang J, Yan WQ, Li Y, Klette R (2017) Local fast R-CNN flow for object-centric event recognition in complex traffic scenes. In: Pacific-rim symposium on image and video technology, pp 439–452
27. Hager GD, Dewan M, Stewart CV (2004) Multiple kernel tracking with SSD. In: CVPR 2004
28. Hassanpour H, Farahabadi PM (2009) Using hidden Markov models for paper currency recognition. Expert Syst Appl 36(6):10105–10111
29. He K, Gkioxari G, Dollar P, Girshick R (2017) Mask R-CNN. In: ICCV, pp 2980–2988
30. Heikkila M, Pietikainen M (2006) A texture-based method for modeling the background and detecting moving objects. IEEE Trans Pattern Anal Mach Intell 28(4):657–662
31. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
32. Hu X (2017) Frequency based texture feature descriptors. PhD thesis, Auckland University of Technology, New Zealand
33. Hubel DH, Wiesel TN (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J Physiol 160(1):106–154
34. Jeong J, Park H, Kwak N (2017) Enhancement of SSD by concatenating feature maps for object detection. In: BMVC
35. Jiao Y, Weir J, Yan W (2011) Flame detection in surveillance. J Multimed 6(1):22–32
36. Kivinen J, Warmuth MK (1998) Relative loss bounds for multidimensional regression problems. In: Advances in neural information processing systems, pp 287–293
37. Klette R (2014) Concise computer vision: an introduction into theory and algorithms. Springer, London, UK
38. Li C (2022) Special character recognition using deep learning. Master's thesis, Auckland University of Technology, New Zealand
39. Liu C, Yan W (2020) Gait recognition using deep learning. In: Handbook of research on multimedia cyber security (IGI global), pp 214–226
40. Li C, Yan W (2022) Braille recognition using deep learning. In: International conference on control and computer vision, pp 30–35
41. Liu W, Wen Y, Yu Z, Yang M (2016) Large-margin softmax loss for convolutional neural networks. In: ICML, pp 507–516

42. Liu X (2019) Vehicle-related scene understanding using deep learning. Master's thesis, Auckland University of Technology, New Zealand
43. Liu X, Yan W, Kasabov N (2020) Vehicle-related scene segmentation using CapsNets. In: IEEE IVCNZ, pp 1–6
44. Liu X, Yan W (2021) Traffic-light sign recognition using capsule network. Multimed Tools Appl 80(10):15161–15171
45. Lu J (2016) Empirical approaches for human behavior analytics. Master's thesis, Auckland University of Technology, New Zealand
46. Ma X (2020) Banknote serial number recognition using deep learning. Master's thesis. Auckland University of Technology, New Zealand
47. Martens J, Sutskever I (2011) Learning recurrent neural networks with Hessian-free optimization. In: International conference on machine learning, Bellevue
48. Merrienboer B, Bahdanau D, Dumoulin V, Serdyuk D, Warde-Farley Murtagh F (1991) Multilayer perceptrons for classification and regression. Neurocomputing 2(5–6):183–197
49. Mikolov T, Karafiat M, Burget L, Cernocky J, Khudanpur S (2010) Recurrent neural network based language model. In: Interspeech, vol 2, p 3
50. Rabiner L, Juang B (1986) An introduction to hidden Markov models. IEEE ASSP (Magazine) 3(1):4–16
51. Rahlf T (2017) Data visualisation with R. Springer International Publishing, New York
52. Rastegari M, Ordonez V, Redmon J, Farhadi A (2016) XNOR-Net: imageNet classification using binary convolutional neural networks. In: European conference on computer vision. Springer, Berlin, pp 525–542
53. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In: IEEE CVPR, pp 6517–6525
54. Rekeczky C, Tahy A, Vegh Z, Roska T (1999) CNN-based spatio-temporal nonlinear filtering and endocardial boundary detection in echocardiography. Int J Circuit Theory Appl 27(1):171–207
55. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
56. Ren Y (2017) Banknote recognition in real time using ANN. Master's thesis, Auckland University of Technology, New Zealand
57. Ren Y, Zhu C, Xiao S (2018) Object detection based on Fast/Faster R-CNN employing fully convolutional architectures. Math Probl Eng
58. Rriedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting. Ann Stat 38(2):337–374
59. Sahiner B, Chan HP, Petrick N, Wei D, Helvie MA, Adler DD, Goodsitt MM (1996) Classification of mass and normal breast tissue: a convolution neural network classifier with spatial domain and texture images. IEEE Trans Med Imaging 15(5):598–610
60. Shen D, Chen X, Nguyen M, Yan WQ (2018) Flame detection using deep learning. In: International conference on control, automation and robotics (ICCAR), pp 416–420
61. Takeda F, Omatu S (1995) A neuro-paper currency recognition method using optimized masks by genetic algorithm. In: IEEE international conference on systems, man and cybernetics, vol 5, pp 4367–4371
62. Taud H, Mas JF (2018) Multilayer perceptron (MLP). In: Geomatic approaches for modelling land change scenarios. Springer, Berlin, pp 451–455
63. Toselli AH, Vidal E, Romero V, Frinken V (2016) HMM word graph based keyword spotting in handwritten document images. Inf Sci 497–518
64. Vapnik VN (1995) The nature of statistical learning theory. Springer, Berlin
65. Vedaldi A, Lenc K (2015) MatConvNet: convolutional neural networks for matlab. In: ACM international conference on multimedia, pp 689–692
66. Wang H (2018) Real-time face detection and recognition based on deep learning. Master's thesis, Auckland University of Technology

67. Wang L, Yan W (2021) Tree leaves detection based on deep learning. In: ISGV. Springer, Berlin, pp 26–38
68. Wang MS, Song L, Yang XK, Luo CF (2016) A parallel-fusion RNN-LSTM architecture for image caption generation. In: International conference on image processing, pp 4448–4452
69. Xin C (2018) Detection and recognition for multiple flames using deep learning. Master's thesis, Auckland University of Technology, New Zealand
70. Xing J, Yan W (2021) Traffic sign recognition using guided image filtering. In: International symposium on geometry and vision (ISGV), pp 85–99
71. Xing J (2021) Traffic sign recognition from digital images by using deep learning (Master's thesis), Auckland University of Technology, Auckland, New Zealand
72. Xingjian SHI, Chen Z, Wang H, Yeung DY, Wong WK, Woo WC (2015) Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Advances in neural information processing systems, pp 802–810
73. Yan WQ (2019) Introduction to intelligent surveillance: surveillance data capture, transmission, and analytics. Springer, Berlin
74. Zhang K, Zhang D, Jing C, Li J, Yang L (2017) Scalable softmax loss for face verification. In: International conference on systems and informatics, pp 491–496
75. Zhang L, Yan W (2020) Deep learning methods for virus identification from digital images. In: International conference on image and vision computing New Zealand
76. Zhang L (2020) Virus identification from digital images using deep learning. Master's thesis, Auckland University, New Zealand
77. Zheng N, Xue J (2009) Statistical learning and pattern analysis for image and video processing. Springer, Berlin
78. Zhang Q (2018) Currency recognition using deep learning. Master's thesis, Auckland University of Technology, New Zealand
79. Zhang Q, Yan W, Kankanhalli M (2019) Overview of currency recognition using deep learning. J Bank Financ Technol 3(1):59–69
80. Zhou J, Leong C, Li C (2021) Multi-scale and attention residual network for single image dehazing. In: International conference on intelligent computing and signal processing (ICSP), pp 483–487

Convolutional Neural Networks and Recurrent Neural Networks

3

In this chapter, we will introduce the typical deep neural networks from the viewpoint of Convolutional Neural Network (CNN or ConvNet) family, especially Region-based CNN(R-CNN), Single Shot MultiBox Detector (SSD), and You Only Look Once (YOLO). Capsule Neural Network (CapsNet) has taken a topological structure of a scene into consideration. The output will be a vector to reflect this geometric relationship. Meanwhile, from the viewpoint of time series analysis, we depict the Recurrent Neural Network (RNN) family, namely Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), etc. In a nutshell, we expect to introduce deep learning thoroughly from spatial and temporal aspects, and deeply learn the knowledge of state-of-the-art research methods.

3.1 Multilayer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural networks (ANNs). An MLP consists of at least three layers of nodes: An input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that is use of a nonlinear activation function. MLP utilizes a supervised learning method called backpropagation for ANN model training. MLP can distinguish or discriminate data that is not linearly separable.

Feedforward networks often have one or more hidden layers of neurons followed by the output layer. Multiple neural layers with nonlinear transfer functions allow the network to learn nonlinear relationships between input and output vectors. The linear output layer is most often employed for function fitting (or nonlinear regression) problems. This neural network can approximate any functions with a finite number of discontinuities arbitrarily well, given sufficient neurons in the hidden layer. Neurons can use any differentiable transfer functions to generate their output.

While training multilayer neural networks, the data is split into training, validation, and test datasets: The training set is used for computing the gradient and updating

the network weights and biases. When the network begins to overfit the data, the error on the validation set typically begins to rise. If the error on the test set reaches a minimum at a significantly different iteration number than the validation set error, this might indicate a poor division of the dataset. The division function is accessed automatically whenever the network is trained, and is used to divide the data into training, validation, and testing subsets.

The multilayer feedforward network can be trained for function approximation (nonlinear regression) or pattern recognition. The process of training a neural network involves tuning the parameters such as values of the weights and biases of the network to optimize network performance. In incremental mode of network training, the gradient is computed and the weights are updated after each input is applied to the network. In batch mode, all the inputs in the training set are applied to the network before the weights are updated. The optimization methods use either the gradient of the network performance with respect to the network weights, or the Jacobian of the network errors with respect to the weights.

The backpropagation algorithm involves performing computations backward through the neural network. Gradient descent updates the network weights and biases in the direction in which the performance function decreases most rapidly. Backpropagation refers specifically to the gradient descent algorithm, when applied to neural network training. In a backpropagation algorithm,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \cdot \mathbf{g}_k, k = 1, 2, \ldots \tag{3.1}$$

where $\mathbf{x}_k$ is a vector of current weights, $\mathbf{g}_k$ is the current gradient, and α is learning rate.

After the network is trained and validated, the neural network can be used to calculate the response to any input. Each time when a neural network is trained, it can result in a different solution due to initial weight/bias and divisions of data into training, validation, and test sets. Different neural networks trained on the same problem can give different outputs for the same input, thus, we need to check the network performance and determine whether any changes need to be made to the training process, the network architecture, or the datasets. If the training were perfect, the network outputs and the targets would be exactly equal.

Multilayer neural networks are capable of performing any linear or nonlinear computations. Neural networks are also sensitive to the number of neurons in their hidden layers. Too few neurons can lead to underfitting; Too many neurons can contribute to overfitting, in which all training points are well fitted, but the fitting curve oscillates wildly between these points.

Horner Algorithm: $f(x) = \sum_{i=0}^{n} a_i \cdot x^i$ is written as

$$f(x) = (((a_n \cdot x + a_{n-1}) \cdot x + a_{n-2}) \cdot x) + \cdots + a_0, a_n \neq 0, a_n, x \in \mathbb{R}, i, n \in \mathbb{Z}^+. \tag{3.2}$$

Kolmogorov Theorem A MLP has the ability to represent any continuous function $g(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in [0, 1]^d = \underbrace{[0, 1] \times \cdots \times [0, 1]}_{d}$, $d \geq 2$ for properly chosen functions $\xi_j(\cdot)$ and $\psi_{ij}(\cdot)$,

$$f(\mathbf{x}) = \sum_{j=1}^{2n+1} \xi_j \left(\sum_{i=0}^{d} \psi_{ij}(x_i) \right). \tag{3.3}$$

Multilayer neural networks are trained by simple stochastic gradient descent. Gradients can be computed by using the backpropagation procedure. The backpropagation can be applied repeatedly to propagate gradients through all modules. In practice, the procedure of Stochastic Gradient Descent (SGD) consists of

- Input vector for a few of samples
- Outputs and errors
- Average gradient for those samples
- Adjustable weights
- The process is repeated until the average of the objective function stops decreasing.

The loss function of SGD is $J(\theta) = L(f_\theta(x_i), y_i)$, (x_i, y_i), $i = 1, \ldots, m$ being samples, where $\frac{\partial J(\theta)}{\partial \theta} = 0$. Here, we list typical loss functions and activation functions:

- 0–1 loss function:

$$L(Y, f(X)) = \begin{cases} 1 & Y \neq f(X) \\ 0 & Y = f(X) \end{cases} X, Y \in \mathbb{R}. \tag{3.4}$$

- Square loss function:

$$L(Y, f(X)) = (Y - f(X))^2, X, Y \in \mathbb{R}. \tag{3.5}$$

- Absolute loss function:

$$L(Y, f(X)) = |Y - f(X)|, X, Y \in \mathbb{R}. \tag{3.6}$$

- Logarithm loss function:

$$L(Y, p(Y|X)) = -\log p(Y|X), X, Y \in \mathbb{R}. \tag{3.7}$$

- Average loss function:

$$L = \frac{1}{m} \sum_{i=1}^{m} L(x_i, y_i) \tag{3.8}$$

where the set $T = \{(x_i, y_i)\}(i = 1, 2, \ldots, m)$, $x_i, y_i \in \mathbb{R}$ is the training set.
- ......

- ReLU: Rectified linear unit,

$$f(x) = \max(0, x), x \in \mathbb{R}. \tag{3.9}$$

- Tanh: Hyperbolic tangent function,

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, x \in \mathbb{R}. \tag{3.10}$$

- Sigmoid: Logistic function,

$$f(x) = \frac{1}{1 + e^{-x}}, x \in \mathbb{R}. \tag{3.11}$$

- ReLU: Rectified linear unit

$$f(x) = \max(0, x) = x^+ = \begin{cases} 0 & x \le 0 \\ x & x > 0 \end{cases}, x \in \mathbb{R} \tag{3.12}$$

$$f'(x) = \begin{cases} 0 & x \le 0 \\ 1 & x > 0 \end{cases}, x \in \mathbb{R}. \tag{3.13}$$

- Tanh: Hyperbolic tangent function,

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}; f'(x) = 1 - f(x)^2, x \in \mathbb{R}. \tag{3.14}$$

- Sigmoid: Logistic function,

$$f(x) = \frac{1}{1 + e^{-x}}; f'(x) = f(x)(1 - f(x)), x \in \mathbb{R}. \tag{3.15}$$

- ······

3.2 Convolutional Neural Network and YOLO Models

Since 2015, the focus of all researchers has been moved to deep learning, i.e., deep neural networks, especially after AlexNet [50] received an award in a contest of visual object detection and recognition using ImageNet [48,86,94]. In 2015, the world top journal Nature also published a survey paper related to deep learning [52]. Before that, most of people were interested in using SVM (support vector machine) for pattern classification [107,124].

Classical CNN (i.e., convolutional neural network or ConvNet) has been employed to digital image processing since 1995 [51]. The convolutional kernels usually are the masks with the size of 3×3, 5×5, 7×7, 9×9, etc. The convolution operations generate receptive fields which compose the feature map of convolutional neural networks [1,88,94,96]. The receptive field corresponds to a region of the given input image [40].

In mathematics, convolution is a mathematical operation on two functions that produces a third function expressing how the shape of one of them is modified or

filtered by the other. Given $\mathbf{H} = (h_{i,j}^{(k)})_{m \times m}$ at level k, and $a^{(k)}, b^{(k)}, c^{(k)}, d^{(k)} \in \mathscr{R}$, $g(\cdot)$ is a nonlinear function, a convolution operation is

$$h_{i,j}^{(k+1)} = g(a^{(k)} \cdot h_{i,j}^{(k)} + b^{(k)} \cdot h_{i+1,j}^{(k)} + c^{(k)} \cdot h_{i,j+1}^{(k)} + d^{(k)} \cdot h_{i+1,j+1}^{(k)}). \quad (3.16)$$

Average pooling includes calculating the average for each patch of the feature map. For an average pooling [53] with downsampling,

$$\bar{h}^{(k+1)} = \frac{1}{4} \left(a^{(k)} \cdot h_{i,j}^{(k)} + b^{(k)} \cdot h_{i+1,j}^{(k)} + c^{(k)} \cdot h_{i,j+1}^{(k)} + d^{(k)} \cdot h_{i+1,j+1}^{(k)} \right). \quad (3.17)$$

For a max pooling [28] with downsampling,

$$h_{\max}^{(k+1)} = \max \left(a^{(k)} \cdot h_{i,j}^{(k)}, b^{(k)} \cdot h_{i+1,j}^{(k)}, c^{(k)} \cdot h_{i,j+1}^{(k)}, d^{(k)} \cdot h_{i+1,j+1}^{(k)} \right), \quad (3.18)$$

where the max pooling is carried out by applying a max filter $\max(\cdot)$ to non-overlapping subregions of the initial representation. In deep learning, convolution operation '$\star$' is denoted as

$$s(t) = (x \star w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t - a), \quad (3.19)$$

where the function $x(a)$ is input and $w(t)$ stands for kernel, the output $s(t)$ represents feature map. For an image $I(i, j), i = 1, 2, \ldots, W, j = 1, 2, \ldots, H, W$ is the image width, and H is the image height. The convolution operation is

$$S(i, j) = (I \star K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n), \quad (3.20)$$

where $K(\cdot)$ is kernel function. Typically, Gaussian kernel in $n \in \mathscr{Z}^+$ dimensions is $G_n(\mathbf{X}, \sigma) = \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left(-\frac{\|\mathbf{X}\|^2}{2\sigma^2}\right)$, where $\mathbf{X} = (x_1, x_2, \ldots, x_n)$, and σ is variance. For example, if $n = 1, G_1(x, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$.

In convolution operation, padding is to fill up the region of an image boundary, which is to fill in the margin region with zero; this will make sure all convolution operations could be carried out at the edge region of images [53]. Meanwhile, the concept stride is the step length of convolution operations.

Convolution operation is to simulate our human visual system. Like most mammals such as cats or dogs, our human visual system (HVS) could be simulated by using the famous Gabor function. Gabor function has been applied to describe the co-occurrence of texture in texture analysis [36]:

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi \frac{x'}{\lambda} + \psi\right)\right), \quad (3.21)$$

where $i = \sqrt{-1}, \alpha, \beta_x, \beta_y, f, \phi, x_0, y_0, \tau \in \mathscr{R}$ are parameters, $x' = x\cos\theta + y\sin\theta$, and $y' = -x\sin\theta + y\cos\theta$. λ represents the wavelength of the sinusoidal factor, θ

shows the orientation of the normal to the parallel stripes of a Gabor function, ψ is the phase offset, σ is the sigma/standard deviation of the Gaussian envelope, and γ is the spatial aspect ratio.

ConvNets (i.e., CNN) also include local connections, shared weights, pooling [34, 53], and multilayer neural network (MLP) [74, 105]. The fine-tuning and pooling operations [53] feature deep neural networks.

3.2.1 Region-Based Convolutional Neural Network

The next is region-based CNN (*R-CNN*). At first, we need to understand the concept: Intersection Over Union (IOU) for visual object detection in an image, which is calculated through

$$IOU = \frac{\mathscr{A}(\mathbf{A} \bigcap \mathbf{B})}{\mathscr{A}(\mathbf{A} \bigcup \mathbf{B})}, \tag{3.22}$$

where $\mathbf{A}$ and $\mathbf{B}$ are image regions of bounding boxes, $\mathscr{A}(\cdot)$ refers to area of a region, and anchor box refers to the bounding box of visual objects with multiresolution, multiscale, and multiaspectratio, etc.

R-CNN [12, 26] is able to quickly find the object bounding box at where the features are extracted from region of interest (ROI), the classifier is still SVM (i.e., support vector machine), the regression is employed for classifying region proposals iteratively. Warp refers to anisotropically scale each object proposal to the CNN input size. R-CNN is slow because it performs a ConvNet forward pass for each object proposal without sharing computations [29]. The training is a multistage pipeline; that means, we need to work one step after another; therefore, it is expensive and time-consuming.

In MATLAB, R-CNN detector firstly generates region proposals. The proposal regions are cropped out of the image and resized. CNN classifies the cropped and resized regions. Finally, the region proposal bounding boxes are refined by using CNN features.

The training of Fast R-CNN [27, 30] is single stage by using a multitask loss. Regression has been employed for bounding-box training [47, 74, 93]. The training can update all parameters of network layers. The pooling layer makes use of max pooling to convert the features inside any valid ROI into a small feature map with a fixed spatial extent of 7×7 region.

In MATLAB, Fast R-CNN deals with the entire image and pools CNN features corresponding to each region proposal. Generally speaking, Fast R-CNN is more efficient than R-CNN, which is the design purpose of this deep learning model.

Faster R-CNN [90, 91] merged Region Proposal Network (RPN) and Fast R-CNN into a single network by sharing their convolutional features. A RPN is a fully convolutional network that predicts object boundary and objectiveness scores at each position simultaneously.

Fig. 3.1 The curve of
softmax function
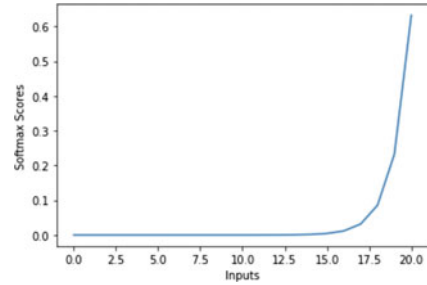
The softmax function [15] as shown in Eq. (3.23) is applied to visual object detection by using Faster R-CNN [90]. The corresponding curve generated by using Google Colaboratory ("Colab", a Jupyter Notebook) is shown in Fig. 3.1:

$$f(x) = \frac{e^{x_i}}{\sum_i e^{x_i}}, x \in (-\infty, \infty). \tag{3.23}$$

In MATLAB, Faster R-CNN adds a RPN to generate region proposals directly in the network. The RPN is use of anchor boxes for visual object detection. Generating region proposals in the network is faster.

In our project, a deep learning method was proposed for early diagnosis and screening of Alzheimer's disease (AD) [43,44,102]. We make use of an object detection network Faster R-CNN to detect the atrophy of the hippocampus region of the human brain to implement the diagnosis. The net is modified and optimized based on VGG-16 as the basic network of Faster R-CNN to extract feature maps and obtain high-precision detection of AD samples.

In our project [117], Faster R-CNN has been employed for apple ripeness analysis from digital images. The ripeness of apples in digital images will be classified by using Faster R-CNN, YOLOv3, and YOLOv4. The classifiers are able to achieve the best result, i.e., the ripeness class of an apple from a given digital image is able to be precisely predicted.

3.2.2 Mask R-CNN

Mask R-CNN is an intuitive extension of Faster R-CNN, which constructs the mask branch properly for generating ideal results [35,104].

Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks on each ROI, in parallel with the existing branch for classification and bounding-box regression.

The mask branch is a small fully convolutional network (FCN) applied to each ROI, predicting a segmentation mask in a pixel-to-pixel manner. Mask R-CNN is simple to be implemented and trained, given the Faster R-CNN framework, which facilitates a range of flexible architectures.

Mask R-CNN has been awarded as the best work in deep learning for its simple, flexible, and general framework for object segmentation. Mask R-CNN [35] from

Facebook AI Research team is led by Dr. Kaiming. He have won the Best Paper Award (Marr Prize) at the 16th International Conference on Computer vision (ICCV) 2017, held in Italy.

Instead of only performing classification and bounding-box regression, Mask R-CNN also outputs a binary mask for each ROI; the general loss in total is given as

$$L = L_{cls} + L_{box} + L_{mask}, \tag{3.24}$$

where L_{cls}, L_{box}, and L_{mask} represent the classification loss, bounding-box loss, and the average binary cross-entropy loss, respectively. Mask R-CNN combines the tasks of visual object detection and semantic segmentation together.

In our project, Faster R-CNN and Mask R-CNN have been utilized to implement facial expression recognition, which consists of a fully convolutional network and detector over a region of interest. Mask R-CNN is an extension of Faster R-CNN algorithm that carries out image segmentation. Faster R-CNN and Mask R-CNN have been employed for object detection and recognition, but for facial expression classification before [101].

3.2.3 YOLO Models

YOLO [89] (i.e., You Only Look Once) is a single pass neural network, which directly optimizes the neural network; given an image, immediately only 7×7 segmentation is employed for image segmentation. Thus, YOLO is very fast.

YOLO network has 24 convolutional layers followed by two fully connected layers, which takes advantage of alternating 1×1 convolutional layers to reduce the feature space between layers. The convolutional layers are pre-trained for classification by using the ImageNet dataset. YOLO adopts leaky rectified linear unit (ReLU) function $\phi(x) \in \mathbf{C}^0(-\infty, \infty)$, $x \in \mathscr{R}$,

$$\phi(x) = \begin{cases} x, & x > 0 \\ 0.1x, & \text{others.} \end{cases} \tag{3.25}$$

The Python code and the curve of ReLU function are shown in Fig. 3.2.

YOLOv2 predicts the location and class label using logistic activation function (*a.k.a.* sigmoid function) $\sigma(\cdot)$, namely

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}, x \in \mathbf{C}^\infty(0, 1). \tag{3.26}$$

The derivative of this monotonic function *w.r.t.* $x \in \mathscr{R}$ is

$$f'(x) = f(x)(1 - f(x)), x \in \mathbf{C}^\infty(0, 1). \tag{3.27}$$

The curve of logistic function rendered by R online software is shown in Fig. 3.3.

YOLOv2 also makes use of 448×448 images for fine-tuning the classification network based on ImageNet. Batch normalization (BN) is based on all convolutional layers in YOLOv2. Darknet-19 has 19 layers of depth which can be trained based on more than a million images from the ImageNet database. The pre-trained network

Fig. 3.2 The Python code
and the corresponding curve
of the ReLU function

```
from matplotlib import pyplot
def rectified(x):
    return max(0.0, x)
series_in = [x for x in range(-100, 100)]
series_out = [rectified(x) for x in series_in]
pyplot.plot(series_in, series_out)
pyplot.show()
```
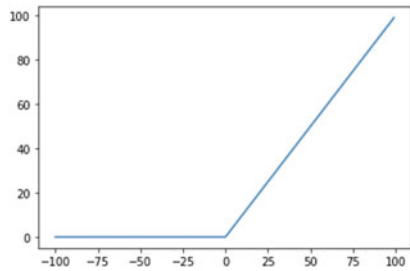

Fig. 3.3 The curve of
logistic function rendered by
R online software
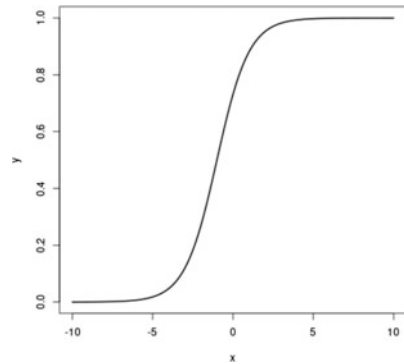
can classify images into 1,000 object categories. Darknet-19 is often used as the
foundation network for YOLO workflows.

YOLOv2 utilizes k-means clustering which leads to better IOU scores. Mathematically, k-means clustering partitions n observations into $k \leq n$ sets $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS). Namely,

$$S_K = \arg\min_{S} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \mu_i\|^2, \tag{3.28}$$

where μ_i is the mean of points in S_i.

In MATLAB, YOLOv2 object detector makes use of a single-stage object detection network and anchor boxes to detect classes of visual objects in an image. For each anchor box, YOLOv2 provides the information such as IOU, anchor box offsets, and class probability. YOLO9000 is able to detect over 9,000 visual object classes

using WordTree in real time [87]. WordTree has a hierarchical tree to link the classes and subclasses together. YOLO9000 provides a way to combine Microsoft COCO and ImageNet together.

YOLOv3 is based on the Darknet, which has a 53-layer network trained on ImageNet. YOLOv3 makes prediction at three scales, which are precisely given by downsampling the dimensions of the input image by 32, 16, and 8, respectively. YOLOv3 takes advantage of nine anchor boxes in total.

YOLOv4 is faster and more accurate than other real-time neural networks based on the Microsoft COCO dataset. Microsoft COCO dataset includes three parts: Training set (120,000 images), validation set (5,000 images), and test set (41,000 images). By using Darknet framework, YOLOv4 is able to cope with 62 frames with the resolution 608×608 per second and achieve 43.5% AP accuracy. The activation function of YOLOv4 is

$$f(x) = x \tanh(\xi(x)), x \in \mathscr{R}. \tag{3.29}$$

This function is called Mish function, a novel self-regularized non-monotonic function,

$$\xi(x) = \ln(1 + e^x) \tag{3.30}$$

where $x \in \mathscr{R}$ is a softmax function, and the derivative is

$$f'(x) = \frac{(e^x(4e^{2x} + e^{3x} + 4(1+x) + e^x(6+4x)))}{(2 + 2e^x + e^{2x})^2}, x \in [-0.308, \infty). \tag{3.31}$$

Correspondingly, the curve of the Mish function is shown in Fig. 3.4.

YOLOv4 is optimal and suitable for real-time object detection. For the resolution to detect multiple objects with various sizes and the exact location, a higher receptive field is required to keep more details of the visual objects.

As the latest net model of the YOLO family, YOLOv5 is broadly employed to various fields. YOLOv5 based on multiple datasets is highly customizable. YOLOv5 is used in GIOU as the loss of the bounding box. YOLOv5 makes use of binary cross entropy and logits loss function to calculate the loss of class probability and score.

Fig. 3.4 The curve of mish function
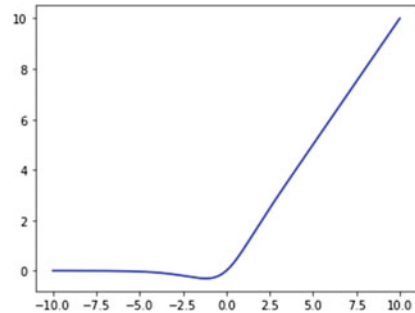
Pertaining to small visual objects, the conventional detection loss rate is very high, especially small objects are easily overlooked. An algorithm based on the improved YOLOv5 is offered to solve the problem of high loss rate of small objects and weak classifier. Firstly, the small datasets are applied to optimize the deep learning network, increase the algorithm speed to prevent the network gradient vanishing, and improve the detection result to prevent overfitting. YOLOv5 model is better than YOLOv4 model which has 1.78% missed detection rate.

In our project, YOLO and Regression CNNs are employed for fruit object locating, classification, and freshness grading [18, 19]. Fruit as visual object, its image is fed into YOLO model for segmentation and regression, then for freshness grading. The project outcome reveals that the proposed approach outperforms linear model.

In our project [110], for the purpose of comparisons, two deep learning models, namely Faster R-CNN and YOLOv5, are employed to conduct tree leaf detection. YOLOv5, as the representative of one-stage algorithm, is obviously better than Faster R-CNN, a standard two-stage algorithm, especially with its advantage in speed makes sure it has a bright prospect in deep learning applications.

In our project [81], traffic-sign recognition (TSR) is able to assist drivers in avoiding a vast number of potential hazards and improve the experience of driving. However, the TSR is a realistic task that is full of constraints, such as visual environment, physical damages, and partial occasions. In order to determine which deep learning models are the most suitable one for the TSR, we choose two models to conduct deep learning computations: Faster R-CNN and YOLOv5. We evaluate the performance of a one-stage model (YOLOv5) and a two-stage model (Faster R-CNN with VGG-16). YOLOv5 is more sufficient and important; there is a slight degradation of accuracy compared to the Faster R-CNN.

YOLOv7 structure is similar to YOLOv5 [62, 116], the main improvement being the replacement of internal components of the network structure. YOLOv7 consists of three components: Input, backbone, and head. YOLOv7 firstly resizes the input image to 640×640 and inputs it into the backbone network, then outputs a feature map with three layers of different sizes through the head layer network, and outputs the prediction results through the REP module and the conv module. YOLOv7 provides Roboflow tool, which can label the images and automatically export the custom dataset.

YOLOv8 is a state-of-the-art object detection algorithm which is an improvement over the previous version YOLOv7 and other object detection algorithms. One of the key innovations of YOLOv8 is the use of a "scale-aware training", which allows the model to better handle visual objects having different sizes in an image. This is achieved by training the model on a diverse set of images, including images with objects of different scales by using a "mosaic data augmentation", which combines multiple images to form a single training image. In addition, YOLOv8 makes use of an efficient implementation of the architecture which allows it to process images at a higher frame rate and make it suitable for real-time applications. YOLOv8 allows to detect visual objects with more accuracy and generalization, and more classes. The loss function includes four main components: Objectness loss, classification loss, localization loss, and confidence loss. YOLOv8 makes use of the Varifocal

Loss (VFL) function to calculate the loss of classification and confidence loss of the object and uses DFL (Distribution Focal Loss) + CIOU as the loss of bounding-box regression.

3.2.4   Single Shot Multibox Detector

SSD [6,65] is a single shot multibox detector (SSD). The single shot refers to the tasks of visual object localization and classification which are carried out in a single forward pass of the network. MultiBox is the term of bounding-box regression. The network is a visual object detector that also classifies those visual objects.

The architecture of SSD is based on the VGG-16 architecture, but discards the fully connected layers. A set of auxiliary convolutional layers were added and employed to extract visual features at multiple scales, which progressively decrease the size of the input to each subsequent layer. SSD makes use of the default aspect ratio and could be applied to visual object tracking in real time [31,41].

MATLAB VGG-16 is a specially designed CNN net which is trained based on ImageNet by the Visual Geometry Group at the University of Oxford. The model has 16 layers and is able to classify images into 1,000 object classes (e.g., keyboard, mouse, coffee mug, pencil, and animals).

There are a few other applications of SSD which are available online, including the original Caffe code. TensorFlow-based SSD code could be downloaded from GitHub.

In our projects related to real-time human face detection and recognition [109], a simplified SSD net was proposed which is an end-to-end model based on SSD architecture. The net includes six convolution layers and two fully connected layers. For two fully connected layers, one of the fully connected layers carries out the prediction of location of bounding box; another fully connected layer executes the classified prediction. This model is simpler than Inception V2; we adopted the dropout method at the end of convolutional layers to optimize the output results in fully connected layers.

In the project related to virus identification using deep learning [130,131], we firstly proposed a loss function which targets to reflect the viruses on the given electron micrograph. We take into account the attention mechanism for virus image classification and localization. We test five deep learning networks: R-CNN, Fast R-CNN, Faster R-CNN, YOLO, and SSD. SSD and Faster R-CNN outperform others in the virus detection from digital images.

In the project related to currency recognition using deep learning [132–134], SSD model was taken into account based on deep learning, which was employed to extract visual features of digital images of paper currency, we accurately recognize the denomination of the currency, both front and back. When a currency is tilted or moved, its denomination and front/back side can still be identified.

In the project related to flare or flame detection, we constructed deep neural network models, e.g., SSD and YOLO. We enhanced the capacity of transformation of CNNs in part of convolution net and pooling operations [97,118].

3.2.5   DenseNets and ResNets

DenseNets alleviate the vanishing gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters [39].

For each layer, the feature maps of all preceding layers are used as inputs, and its own feature maps are employed as inputs into all subsequent layers.

DenseNets introduce direct connections between any two layers with the same feature-map size, which scale naturally to hundreds of layers, while exhibiting no optimization difficulties. Thus, DenseNets require substantially fewer parameters and less computation to achieve state-of-the-art performances. DenseNets allow feature reuse throughout the networks and consequently learn more compact and more accurate models.

In deep learning, there exists the degradation problem; namely, with the network depth increasing, accuracy gets saturated. However, ResNets easily obtain accuracy gains from greatly increased depth due to the well-designed structure of this network:

$$\mathbf{y} = \mathscr{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}, \tag{3.32}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output vectors of the layers, $\mathscr{F}(\cdot)$ is the residual mapping, e.g., $\mathscr{F} = W_2 \sigma(W_1 x)$, and σ is the ReLU function.

In our projects, DenseNet has been applied to character recognition of banknote serial numbers. DenseNet was proposed as the primary classifier. The CNN net with the residual attention model is utilized for serial number recognition.

In our project, we colorize CT images using ResNet [114]. CT refers to computed tomography to sense and create detailed images of internal organs, bones, soft tissue, and blood vessels. We select appropriate reference images so as to combine the style and content of the representations to colorize the target CT lung grayscale images for the fully automated approach.

In our project [55,56], we have implemented the character-based Braille translator using ResNet models; there are three versions of ResNets for character-based Braille classifiers, including ResNet-18, ResNet-34, and ResNet-50.

3.2.6   Capsule Network

A dynamic routing mechanism for capsule networks (CapsNets) was introduced by Hinton and his team in 2017 [95]. A capsule is a set of neurons that was individually activated for various properties of a visual object. A CapsNet was employed to better model hierarchical relationships which is able to delineate the "Picasso problem", namely the images that have all the right parts are not in the correct spatial relationships. The output of a CapsNet is a vector consisting of the probability of an observation, i.e., pose (e.g., position, size, and orientation), deformation, and velocity. CapsNets replace the scalar output with vector-output capsules. Because each capsule is independent, when multiple capsules agree, the probability of correct detection or confidence is much higher.

The output of a capsule is updated by

$$b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j \tag{3.33}$$

where b_{ij} refers to the prior probability that capsule i in layer l should connect to capsule j in layer $l + 1$:

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i, \tag{3.34}$$

where $\mathbf{W}_{ij}$ is a weight matrix. The pose vector $\mathbf{u}_i$ is rotated and translated by using $\mathbf{W}_{ij}$ into a vector $\hat{\mathbf{u}}_j$ that predicts the output of the parent capsule.

In CapsNets, the squashing function is

$$\mathbf{v}_j(s_j) = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}, \tag{3.35}$$

where $\mathbf{v}_j$ is the vector output of capsule j. Capsules $\mathbf{s}_j$ in the next layer are fed from the sum of predictions from all capsules in the previous layers with a coupling coefficient c_{ij},

$$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}, \tag{3.36}$$

and

$$c_{ij} = softmax(\mathbf{b}_i) = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}, \tag{3.37}$$

where c_{ij} are coupling coefficients, and b_{ij} is the log prior probability, initially, $b_{ij} := 0$. Eventually, the network is trained by minimizing the loss function

$$L_k = T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 + \lambda(1 - T_k)\max(0, \|\mathbf{v}_k\| - m^-)^2, \tag{3.38}$$

where $m^+ = 0.9$, $m^- = 0.1$, and $\lambda = 0.5$,

$$T_k = \begin{cases} 1 & \text{digit of class } k \text{ present} \\ 0 & \text{others.} \end{cases} \tag{3.39}$$

CapsNets have multiple conceptual advantages, which learn topological relationship; the networks are organized in a hierarchical way. CapsNets have the attribute with viewpoint invariance and better generalization to new viewpoints. Moreover, CapsNets have been applied to image segmentation, which work like SegNets and U-Nets. The two deep learning networks were designed specifically for image segmentation.

U-Nets [92, 121] were also applied to pixel-wise regression, small-size object detection, and recognition. U-Net is a convolutional neural network that consists of a contracting path and an expansive path. The contracting path follows the typical architecture of a convolutional network. The network has U-shaped architecture, consisting of repeated applications of convolution, followed by a rectified linear unit (ReLU) and a max-pooling operation. The design was based on a fully convolutional network (FCN); its architecture was modified and extended with fewer

training images so as to yield more precise segmentation. U-Net is used for a pixel-wise softmax cross entropy as the loss function. The softmax function is defined as

$$p_k(\mathbf{p}) = \frac{\exp(a_k(\mathbf{p}))}{\sum \exp(a_k(\mathbf{p}))}, \qquad (3.40)$$

where $p_k(\mathbf{p})$ is the softmax function at pixel $\mathbf{p}$ within the feature channel k, $k = 1, 2, \ldots, K$, and K is the number of classes; $a_k(\mathbf{p})$ is the activation function at the pixel position $\mathbf{p} = (x, y) \in \Omega = [a, b] \times [c, d] \subset \mathscr{R}^2$, and $x \in [a, b]$ and $y \in [c, d]$ are the intervals of the image region in horizontal and vertical directions, respectively. The pixel-wise softmax cross-entropy function is

$$L(p) = \sum_{\mathbf{p} \in \Omega} w(\mathbf{p}) \log p(a_l(\mathbf{p})), \qquad (3.41)$$

where $a_l(\mathbf{p})$ is the softmax function with a channel label l, $0 < l \leq K$ at pixel $\mathbf{p}$; $w(\mathbf{p})$ is a weight map at pixel $\mathbf{p}$; the map with a pixel-wise loss weight forces the U-Net network to learn from the border pixels. The weight map is computed as

$$w(\mathbf{p}) = w_c(\mathbf{p}) + w_0(\mathbf{p}) \exp\left(-\frac{(d_1(\mathbf{p}) + d_2(\mathbf{p}))^2}{2\sigma^2}\right), \qquad (3.42)$$

where $w_c(\mathbf{p})$, $w_0(\mathbf{p})$, and $\sigma \neq 0$ are parameters, which are treated as constants. $d_1(\mathbf{p})$ and $d_2(\mathbf{p})$ are the first longest distance and the second longest distance from pixel $\mathbf{p}$ to its border pixels, respectively.

Meanwhile, SegNet [2] is use of the pre-trained convolutional layer weights from VGG [100] neural networks as pre-trained weights, which were developed by the University of Cambridge, UK. The encoder network consists of 13 convolutional layers which correspond to the first 13 convolutional layers in the VGG-16 network. Each encoder layer has a corresponding decoder layer. The final decoder output is fed to a multiclass softmax classifier to produce class probabilities for each pixel independently. The cross-entropy loss was supplied as the objective function for training SegNet, the loss is summed up over all the pixels in a mini-batch. SegNet only stores the max-pooling indices of the feature maps and makes use of them in its decoder network to achieve the ideal performance.

Moreover, SegNet [2] is a deep encoder–decoder architecture for multiclass pixel-wise segmentation. SegNet is effective for a real-time urban road scene segmentation as well as indoor scene understanding. The architecture consists of a sequence of nonlinear processing layers (encoders) and a corresponding set of decoders followed by a pixel-wise classifier. Typically, each encoder consists of one or more convolutional layers with batch normalization and a ReLU nonlinearity, followed by non-overlapping max pooling and subsampling. One key component of the SegNet is to perform upsampling (e.g., bilinear interpolation) in the decoders for low-resolution feature maps. The entire architecture is trained by using stochastic gradient descent.

Downsampling is implemented by using pooling operations such as max pooling and average pooling. Pertaining to upsampling, the nearest neighbor, bilinear, and bicubic interpolation methods [45] are employed to the operation. For a bilinear

interpolation, we have a region $\Omega = [a, b] \times [c, d]$ and a region $\Omega' = [a', b'] \times [c', d']$ in images I and I', respectively; the parameters $s \in [0, 1]$ and $t \in [0, 1]$ establish the mapping relationship between region $\Omega \subset I$ to $\Omega' \subset I'$. That means, given pixel $\mathbf{p} \in \Omega \subset I$, we will get the corresponding pixel $\mathbf{p}' \in \Omega' \subset I'$ via parameters $s_0, t_0 \in [0, 1]$. Hence, $\mathbf{p}(s, t) = \mathbf{p}'(s, t), \forall s, t \in [0, 1]$, namely

$$\mathbf{p}_A = \mathbf{p}(0, 0) = \mathbf{p}'(0, 0) = \mathbf{p}'_{A'}, \tag{3.43}$$

and

$$\mathbf{p}_B = \mathbf{p}(1, 0) = \mathbf{p}'(1, 0) = \mathbf{p}'_{B'}, \tag{3.44}$$

and

$$\mathbf{p}_C = \mathbf{p}(0, 1) = \mathbf{p}'(0, 1) = \mathbf{p}'_{C'}, \tag{3.45}$$

and

$$\mathbf{p}_D = \mathbf{p}(1, 1) = \mathbf{p}'(1, 1) = \mathbf{p}'_{D'}, \tag{3.46}$$

where pixels at the four corner points $\mathbf{p}_A, \mathbf{p}_B, \mathbf{p}_C, \mathbf{p}_D \in \Omega$ correspond to the four corner points at $\mathbf{p}_{A'}, \mathbf{p}_{B'}, \mathbf{p}_{C'}, \mathbf{p}_{D'} \in \Omega'$. Thus,

$$\mathbf{p}'(s_0, t_0) = t_0 \cdot [s_0 \cdot \mathbf{p}'(0, 0) + (1.0 - s_0) \cdot \mathbf{p}'(1, 0)] + (1.0 - t_0) \cdot [s_0 \cdot \mathbf{p}'(1, 0) + (1.0 - s_0) \cdot \mathbf{p}'(1, 1)]. \tag{3.47}$$

In the matrix form,

$$\mathbf{p}'(s_0, t_0) = (t_0, 1.0 - t_0) \mathbf{M}'(s_0, 1 - s_0)^\top, \tag{3.48}$$

where

$$\mathbf{M}' = \begin{bmatrix} \mathbf{p}'(0, 0) & \mathbf{p}'(1, 0) \\ \mathbf{p}'(0, 1) & \mathbf{p}'(1, 1) \end{bmatrix}. \tag{3.49}$$

Meanwhile,

$$\mathbf{p}(s_0, t_0) = t_0 \cdot [s_0 \cdot \mathbf{p}(0, 0) + (1.0 - s_0) \cdot \mathbf{p}(1, 0)] + (1.0 - t_0) \cdot [s_0 \cdot \mathbf{p}(1, 0) + (1.0 - s_0) \cdot \mathbf{p}(1, 1)]. \tag{3.50}$$

Similarly, in the matrix form,

$$\mathbf{p}(s_0, t_0) = (t_0, 1.0 - t_0) \mathbf{M}(s_0, 1 - s_0)^\top, \tag{3.51}$$

where

$$\mathbf{M} = \begin{bmatrix} \mathbf{p}(0, 0) & \mathbf{p}(1, 0) \\ \mathbf{p}(0, 1) & \mathbf{p}(1, 1) \end{bmatrix}. \tag{3.52}$$

In our projects, CapsNet has been successfully applied to traffic scene understanding, traffic-light sign recognition, and vehicle-related scene segmentation [59–61, 120].

In our project for sign language recognition [69], Capsule Network (CapsNet) is proposed. The CapsNet attains the accuracy of overall recognition up to 98.72% based on our own dataset.

3.3   Recurrent Neural Networks and Time Series Analysis

Recurrent neural network (RNN) is one of the deep neural networks which is now applied to a plethora of applications because its unique structure is quite helpful and beneficial while dealing with sequence data.

RNN structure has the recurrent hidden layer connected to that of the next step. Compared with other multilayer neural networks, RNN can impact time. To explain this in more detail, there is a unidirectional flow of information from the input unit to the hidden unit, while there is another unidirectional flow of information from the hidden unit to the output one. In addition, the input of the hidden layer also contains the state of previous hidden layer; the nodes of the hidden layer are self-connected or interconnected.

An LSTM [37] network is a type of recurrent neural network (RNN) that learns long-term dependencies between time steps of sequence data. The core components of an LSTM network are a sequence input layer and an LSTM layer. The input layer imports sequence or time series data into the network. LSTM networks can remember the state of the network.

In MATLAB, LSTM networks support input data with varying sequence lengths. While passing data through the network, the network pads, truncates, or splits sequences so that all the sequences in each mini-batch have the specified length.

If x is the input layer, o is the output layer, t is the number of times, s is the hidden layer, V, W, and U are all weights, and the state of the hidden layer at time t is calculated as

$$S_t = f(U \cdot \mathbf{x}_t + W \cdot S_{t-1}), \tag{3.53}$$

where $f(\cdot)$ is an activation function. If there is a sequence of inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T \in \mathscr{R}^n$, the sequence of hidden states is $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T \in \mathscr{R}^m$, the sequence of prediction is $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T \in \mathscr{R}^k$, and the following equations are employed for the iterations:

$$\mathbf{t}_i = \mathbf{W}_h^x \mathbf{x}_i + \mathbf{W}_h^h \mathbf{x}_{i-1} + \mathbf{b}_h, \tag{3.54}$$

$$\mathbf{h}_i = e(\mathbf{t}_i), \tag{3.55}$$

$$\mathbf{s}_i = \mathbf{W}_y^h \mathbf{h}_i + \mathbf{b}_y, \tag{3.56}$$

and

$$\mathbf{y}_i = g(\mathbf{s}_i), \tag{3.57}$$

where $\mathbf{W}_h^x$, $\mathbf{W}_h^h$, and $\mathbf{W}_y^h$ are the weight matrices; the sequence of $\mathbf{t}_i$ represents the inputs to the hidden units, the sequence of $\mathbf{s}_i$ stands for the inputs to the output units; $\mathbf{b}_h$ and $\mathbf{b}_y$ are bias vectors; $e(\cdot)$ and $g(\cdot)$ are the predefined vector-valued functions.

In our projects, CNN+LSTM and 3D-CNN are harnessed to identify human actions from medics who wear a PPE (i.e., personal protective equipment) in health and disability care. Our experimental results demonstrate that the CNN+LSTM method has effectively identified human actions in given videos [120, 122, 123].

3.3.1 Hidden Markov Model

We usually use FSM (i.e., finite state machine) and HMM (i.e., hidden Markov model) [9,33,82,106] to detect computable events. HMM is generally employed to predict what will happen there. A typical example is to predict whether a person is healthy or has fever in a day using probability. It will be helpful for preparing medicine in a hospital for a season based on weather changes. The stochastic automation is
(1) Initial probabilities: $\pi_i \equiv p(q_1 = S_i)$, $\sum_{i=1}^{N} \pi_i = 1$; $\Pi = (\pi_1, \pi_2, \ldots, \pi_N)$.
(2) Transition matrix: $\mathbf{A} = (a_{ij})_{N \times N}$, $a_{ij} \equiv p(q_{t+1} = S_j | q_t = S_i) \in [0, 1]$ and $\sum_{j=1}^{N} a_{ij} = 1$.
   HMM model $\lambda = (\mathbf{A}, \mathbf{B}, \Pi)$ is related to

1. State: $\mathbf{S} = \{S_1, S_2, \ldots, S_N\}$
2. Observation: $\mathbf{V} = \{v_1, v_2, \ldots, v_M\}$
3. Transition matrix: $\mathbf{A} = (a_{ij})_{N \times N}$, $a_{ij} \equiv p(q_{t+1} = S_j | q_t = S_i)$
4. Emission probabilities: $\mathbf{B} = (b_j(m))_M$, $b_j(m) \equiv p(O_t = v_m | q_t = S_j)$
5. Initial probabilities: $\Pi = (\pi_i)_N$, $\pi_i \equiv p(q_1 = S_i)$
6. Output: $\mathbf{O} = \{O_1 O_2 \cdots O_T\}$
7. Latent variables: $\mathbf{Q} = \{Q_1 Q_2 \cdots Q_T\}$.

HMM [82,83] has two very important algorithms: Viterbi algorithm and Baum–Welch (BM) algorithm. The Viterbi algorithm could assist us to quickly find the best path which has been applied to information theory [11] for coding.
   Given $\mathbf{Q} = \{q_1, \ldots, q_T\}$ and $\mathbf{O} = \{o_1, \ldots, o_T\}$,

$$\delta_t(i) = \max \ p(q_1 q_2 \cdots q_{t-1}, q_t = S_i, O_1 \cdots O_t | \lambda). \tag{3.58}$$

Computationally,

1. Initialization: $\delta_1(i) = \pi_i b_i(O_1)$, $\psi_1(i) = 0$
2. Recursion: $\delta_t(j) = \max_i (\delta_{t-1}(i) \cdot a_{ij}) b_j(O_t)$, $\psi_t(j) = \arg \max_i (\delta_{t-1}(i) \cdot a_{ij})$
3. Termination: $p^* = \max_i \delta_T(i)$, $q_T^* = \arg \max_i \delta_T(i)$
4. Path: $q_t^* = \psi_{t+1}(q_{t+1}^*)$, $t = T - 1, T - 2, \ldots, 1$.

BM algorithm is applied to predict the highest probability of parameters by using EM algorithm. Given $\lambda = (\mathbf{A}, \mathbf{B}, \Pi)$, the Baum–Welch (BW) algorithm is applied to seek $\lambda^* = \arg \max_\lambda p(\chi | \lambda)$,

E-step,

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j); \ \xi_t(i, j) \equiv p(q_t = S_i, q_{t+1} = S_j | \mathbf{O}, \lambda), \tag{3.59}$$

M-step,

$$p(\chi | \lambda) = \Pi_{k=1}^{K} p(O^k | \lambda), \tag{3.60}$$

and

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i, j)}, \hat{b}_j(m) = \frac{\sum_{t=1}^{T} \gamma_t(j)\mathbf{1}(O_t = v_m)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{3.61}$$

HMMs make use of transition probability in each step for predicting events that will happen. FSM has not probability prediction, which is employed to capture events during state transition.

HMM is not a neural network; it has no neurons and activation functions. RNNs [20,73,75] are a family of artificial neural networks for processing sequential data, which is a dynamical system driven by external $\mathbf{x}^{(t)}$,

$$h^{(t)} = f(h^{(t-1)}, \mathbf{x}^{(t)}; \theta) = g^{(t)}(\mathbf{x}^{(t)}, \mathbf{x}^{(t-1)}, \dots, \mathbf{x}^{(1)}), \tag{3.62}$$

where $t = 1, 2 \dots, \tau$, and h is the state. This tells us it is possible to use the same transition function with the same parameters at every time step during unfolding operations.

For $i = 1, 2, \dots, \tau$,

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}h^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)}, \tag{3.63}$$

where $\mathbf{b}$ and $\mathbf{c}$ are vectors, and $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ are weight matrices,

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)}), \tag{3.64}$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}, \tag{3.65}$$

and

$$\hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{o}^{(t)}). \tag{3.66}$$

The loss function L is

$$L = \sum_{t=1}^{\top} L^{(t)} = \sum_{t=1}^{\top} \log p(\mathbf{y}^{(t)} | \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)}\}). \tag{3.67}$$

Hence,

$$\frac{\partial L}{\partial L^{(t)}} = 0. \tag{3.68}$$

Thus,

$$(\nabla_{o^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}}, (\nabla_{h^{(t)}} L)_i = \frac{\partial L}{\partial h_i^{(t)}}. \tag{3.69}$$

Furthermore,

$$\nabla_c L = \frac{\partial L}{\partial c}, \nabla_b L = \frac{\partial L}{\partial b}, \nabla_V L = \frac{\partial L}{\partial V}, \nabla_W L = \frac{\partial L}{\partial W}, \nabla_U L = \frac{\partial L}{\partial U}. \tag{3.70}$$

In our project [112], we proposed a method for human gait recognition based on a self-adaptive hidden Markov model (SAHMM). We presented a feature extraction algorithm based on local gait energy image (LGEI) and constructed an observation

vector set. We optimized parameters of the SAHMM-based method for gait recognition. The proposed method is evaluated extensively based on the CASIA Dataset B for gait recognition under various conditions such as cross view, human dressing, and bag carrying.

In our project [98], we introduced a method to detect and alert a driver if cars appear in the blind spots by using videos captured from digital cameras installed in the car. In this project, two algorithms based on histogram were compared to detect the states of the blind spots. HMM was then employed to obtain the maximum probability of this state occurring. After the data was acquired, RNN was harnessed to analyze the results, and predict future sequences. The deep learning RNN model was designed to predict the number of cars that will turn up in the blind spots.

3.3.2   Recurrent Neural Networks

RNN [23,25,73,73,75] refers to recurrent neural network; most of the time, we unfold the neural network; the procedure is simulated usually by using the fixed-point theorem. We calculate the difference by using loss functions. A loss function is a part of a cost function which is a type of an objective function. Hence, the concepts loss function, cost function, and objective function have minor differences.

Loss function $L(y_i, \hat{y}_i)$ refers to a single sample in a dataset, $\hat{y}_i$ is the output of a neural network model, and y_i is the real value or ground truth. Cost function $J(\cdot)$ means the entire training dataset with all samples $J = \sum L(y_i, \hat{y}_i), i = 1, 2, \ldots, n$, for example, mini-batch in gradient descent makes use of all the samples of the training set. Objective function means a function $f(\cdot)$ will be optimized by using optimization algorithm which is subject to constraints.

In mathematics, the basic concept of loss function [3] is a distance. The popular one is Euclidean distance, but entropy $e = -\sum_{i=1}^{n} h_i \log h_i = -\mathbf{E}(\log h_i), h_i \in (0, 1]$ has been applied to calculate the distance using mathematical expectation of a logarithm function. The softmax function $f(x) = \frac{e^{x_i}}{\sum_i e^{x_i}}$ has been applied to the calculation [15,66,128]. The loss functions typically include 0∼1 loss function, square loss function, absolute loss function, average loss function, hinge loss function, etc. 0∼1 loss function is

$$\mathscr{L}(Y, f(X)) = \begin{cases} 1 & Y \neq f(X) \\ 0 & Y = f(X). \end{cases} \tag{3.71}$$

Squared error cost function or quadratic cost function is shown as

$$J = \|\mathbf{Y}, f(\mathbf{X})\|^2 = \sum_{i=1}^{n} (y_i - f(x_i))^2, \tag{3.72}$$

where $(x_i, y_i), i = 1, 2, \ldots, n$ is a group of given points, $\mathbf{X} = (x_1, x_2, \ldots, x_n)^\top$, $\mathbf{Y} = (y_1, y_2, \ldots, y_n)^\top$, and y_i is different from $f(x_i)$.

The squared cost function has an important position in linear algebra. For example, regarding a straight line $y = ax + b$, where parameters a and b are unknown, if we

have n 2D sampling points $\mathbf{p}(x_i, y_i)$, $i = 1, 2, \ldots, n$, we treat $\theta = (a, b)^\top$ as parameters, which is estimated by using linear regression. Hence,

$$J(a, b) = \sum_{i=1}^{n} (ax_i + b - y_i)^2. \tag{3.73}$$

We rewrite Eq. (3.73) as a quadratic polynomial; a quadratic polynomial is a polynomial of degree 2; a bivariate quadratic polynomial has the form

$$J(a, b) = A \cdot a^2 + B \cdot b^2 + C \cdot ab + D \cdot a + E \cdot b + F, \tag{3.74}$$

where $A, B, C, D, E,$ and F are the constants ($A \neq 0$) which are related to (x_i, y_i), $n = 1, 2, \ldots, n$. Bivariate polynomials are fundamental to the study of conic sections, which are characterized by equating the expression $L(a, b)$ to zero. Thus, we modify Eq. (3.74) and obtain

$$J(a, b) = a^2 + \frac{B}{A} \cdot b^2 + \frac{C}{A} \cdot ab + \frac{D}{A} \cdot a + \frac{E}{A} \cdot b + \frac{F}{A}, A \neq 0. \tag{3.75}$$

Moreover, we simplify Eq. (3.75) in the form of matrix

$$J(a, b) = (a, b, 1)\mathbf{M}(a, b, 1)^\top, \tag{3.76}$$

where $\mathbf{M} = (m_{i,j})_{3\times3}$, and $m_{i,j}$ was derived from the constants $A, B, C, D, E,$ and F. We see that matrices could be applied to express a quadratic polynomial, and linear algebra could be applied to the square loss function.

Absolute loss function is expressed as

$$\mathcal{L}(Y, f(X)) = |Y - f(X)|. \tag{3.77}$$

Logarithm loss (i.e., log-loss) function is

$$\mathcal{L}(Y, p(Y|X)) = -\log p(Y|X), \tag{3.78}$$

where $p(Y|X)$ is the conditional probability. The average cost function is

$$\bar{J} = \frac{1}{m} \sum_{i=1}^{m} L(x_i, y_i), \tag{3.79}$$

where the set $\mathbf{T} = \{(x_i, y_i)\}(i = 1, 2, \ldots, m)$ is the training dataset, $\mathbf{X} = (x_1, x_2, \ldots, x_m)^\top, \mathbf{Y} = (y_1, y_2, \ldots, y_m)^\top$.

In machine learning, hinge loss function is employed for training classifiers. For an output $t = \pm1$ and a classifier score $x \in \mathscr{R}$, the hinge loss of the prediction $L(x)$ is defined as

$$\mathcal{L}(x) = \max(0, 1 - t \cdot x), \tag{3.80}$$

if $t = -1$ and $x \geq 0$, then $L(x) = 1 + x > 0$; if t=-1 and $x < 0$, then $L(x) = \max(0, 1 - |x|)$; if $t = +1$, $x \geq 0$, $L(x) = \max(0, 1 - |x|)$; if $t = +1$, $x < 0$, $L(x) = 1 + x > 0$. Hence, $sgn(x) \cdot sgn(t) = -1$, $L(x) = 1 + x > 0$, where $sgn(\cdot)$ is the sign function and returns $+1$ or -1. Namely, $sgn(x) = +1$ if $x > 0$; $sgn(x) = -1$ if $x < 0$.

For $sgn(x) \cdot sgn(t) = +1$, $L(x) = \max(0, 1 - |x|)$. If $|x| > 1$, $1 - |x| < 0$, $L(x) = 0$; if $|x| < 1$, $1 - |x| > 0$, $L(x) = 1 - |x| > 0$. That means, $0 \leq x < 1$, $L(x) = 1 - x > 0$; if $-1 < x \leq 0$, $L(x) = 1 + x > 0$.

In summary, if $sgn(x) \cdot sgn(t) = -1$, then $L(x) = 1 + x > 0$; if $sgn(x) \cdot sgn(t) = +1$, $|x| < 1$, then $L(x) = 1 + x > 0$; if $sgn(x) \cdot sgn(t) = +1$, $|x| > 1$, $1 - |x| < 0$, then $L(x) = 0$.

In a loss function $y = f(x)$, we usually need to calculate the derivative $y' = \frac{df(x)}{dx}$; the chain rule is therefore applied if $x = s(t)$, then,

$$y' = \frac{df(x)}{dx} = \frac{df(x)}{dx} \cdot \frac{dx(t)}{dt}. \tag{3.81}$$

Most of the time, we ensure the continuity of a function, but we cannot guarantee that the derivative exists. LSTM [17,22,24,25,111,119] has been utilized to avoid this vanishing gradient or exploding problems. LSTM (i.e., long short-term memory) is a typical RNN net,

$$\mathbf{f}_t = \sigma_g(\mathbf{W}_f \cdot \mathbf{x}_t + \mathbf{U}_f \cdot \mathbf{h}_{t-1} + \mathbf{b}_f), \tag{3.82}$$

$$\mathbf{i}_t = \sigma_g(\mathbf{W}_i \cdot \mathbf{x}_t + \mathbf{U}_i \cdot \mathbf{h}_{t-1} + \mathbf{b}_i), \tag{3.83}$$

$$\mathbf{o}_t = \sigma_g(\mathbf{W}_o \cdot \mathbf{x}_t + \mathbf{U}_o \cdot \mathbf{h}_{t-1} + \mathbf{b}_o), \tag{3.84}$$

$$\mathbf{c}_t = \mathbf{f}_t \cdot \mathbf{c}_{t-1} + \mathbf{i}_t \circ \sigma_c(\mathbf{W}_c \cdot \mathbf{x}_t + \mathbf{U}_c \cdot \mathbf{h}_{t-1} + \mathbf{b}_c), \tag{3.85}$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \sigma_h(\mathbf{c}_t), \tag{3.86}$$

where $\mathbf{x}_t$ and $\mathbf{h}_t$ are input and output vectors; $\mathbf{c}_0 = 0$, $\mathbf{h}_0 = 0$; $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ are activation vectors of forget, input, and output gates; $\mathbf{W}, \mathbf{U}, \mathbf{b}$ are weight matrices and bias vector; $\mathbf{c}_t$ is the cell state vector; '$\circ$' is Hadamard product, i.e.,

$$\mathbf{A}_{m \times n} \cdot \mathbf{B}_{m \times n} = (a_{ij})_{m \times n} \cdot (b_{ij})_{m \times n} = (a_{ij} \cdot b_{ij})_{m \times n}, \tag{3.87}$$

where $\sigma_g(\cdot)$, $\sigma_c(\cdot)$, and $\sigma_h(\cdot)$ are activation functions.

ConvLSTM (i.e., convolutional LSTM) [119] made use of the spatiotemporal relationship,

$$\mathbf{f}_t = \sigma_g(\mathbf{W}_f \star \mathbf{x}_t + \mathbf{U}_f \star \mathbf{h}_{t-1} + \mathbf{V}_f \circ \mathbf{c}_{t-1} + \mathbf{b}_f), \tag{3.88}$$

$$\mathbf{i}_t = \sigma_g(\mathbf{W}_i \star \mathbf{x}_t + \mathbf{U}_i \star \mathbf{h}_{t-1} + \mathbf{V}_i \circ \mathbf{c}_{t-1} + b_i), \tag{3.89}$$

$$\mathbf{o}_t = \sigma_g(\mathbf{W}_o \star \mathbf{x}_t + \mathbf{U}_o \star \mathbf{h}_{t-1} + \mathbf{V}_o \circ \mathbf{c}_{t-1} + \mathbf{b}_o), \tag{3.90}$$

$$\mathbf{c}_t = \mathbf{f}_t \cdot \mathbf{c}_{t-1} + \mathbf{i}_t \circ \sigma_c(\mathbf{W}_c \star \mathbf{x}_t + \mathbf{U}_c \star \mathbf{h}_{t-1} + \mathbf{b}_c), \tag{3.91}$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \sigma_h(\mathbf{c}_t), \tag{3.92}$$

where $\mathbf{x}_t$ and $\mathbf{h}_t$ are input and output vectors, $\mathbf{c}_0 = 0$, $\mathbf{h}_0 = 0$; $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ are activation vectors of forget, input, and output gates; $\mathbf{W}$, $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{b}$ are weight matrices and bias vector; $\mathbf{c}_t$ is the cell state vector; '$\circ$' is Hadamard product, and '$\star$' is the convolution operator. $\sigma_g(\cdot)$, $\sigma_c(\cdot)$, and $\sigma_h(\cdot)$ are activation functions. Furthermore, we have the peephole LSTM,

$$\mathbf{f}_t = \sigma_g(\mathbf{W}_f \cdot \mathbf{x}_t + \mathbf{U}_f \cdot \mathbf{c}_{t-1} + \mathbf{b}_f), \tag{3.93}$$

$$\mathbf{i}_t = \sigma_g(\mathbf{W}_i \cdot \mathbf{x}_t + \mathbf{U}_i \cdot \mathbf{c}_{t-1} + \mathbf{b}_i), \tag{3.94}$$

$$\mathbf{o}_t = \sigma_g(\mathbf{W}_o \cdot \mathbf{x}_t + \mathbf{U}_o \cdot \mathbf{c}_{t-1} + \mathbf{b}_o), \tag{3.95}$$

$$\mathbf{c}_t = \mathbf{f}_t \cdot \mathbf{c}_{t-1} + \mathbf{i}_t \circ \sigma_c(\mathbf{W}_c \cdot \mathbf{x}_t + \mathbf{U}_c \cdot \mathbf{h}_{t-1} + \mathbf{b}_c), \tag{3.96}$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \sigma_h(\mathbf{c}_t), \tag{3.97}$$

where $\mathbf{x}_t$ and $\mathbf{h}_t$ are input and output vectors, $\mathbf{c}_0 = 0$, $\mathbf{h}_0 = 0$; $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ are activation vectors of forget, input, and output gates; $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are weight matrices and bias vector; $\mathbf{c}_t$ is the cell state vector; '$\circ$' is Hadamard product; $\sigma_g(\cdot)$, $\sigma_c(\cdot)$, and $\sigma_h(\cdot)$ are activation functions. Meanwhile, we have fully gated unit (FRU), which has the following steps.

Initially, $t = 0$ and $\mathbf{h}_0 = 0$, regarding update gate,

$$\mathbf{z}_t = \sigma_g(\mathbf{W}_z \cdot \mathbf{x}_t + \mathbf{U}_z \cdot \mathbf{h}_{t-1} + \mathbf{b}_z). \tag{3.98}$$

With regard to reset gate,

$$\mathbf{r}_t = \sigma_g(\mathbf{W}_r \cdot \mathbf{x}_t + \mathbf{U}_r \cdot \mathbf{h}_{t-1} + \mathbf{b}_r). \tag{3.99}$$

Pertaining to the new memory,

$$\tilde{\mathbf{h}}_t = \sigma_h(\mathbf{W}_h \cdot x_t + \mathbf{U}_h(r_t \circ \mathbf{h}_{t-1}) + \mathbf{b}_h). \tag{3.100}$$

In relevance to the hidden state,

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \circ \mathbf{h}_{t-1} + \mathbf{z}_t \circ \tilde{\mathbf{h}}_t \tag{3.101}$$

where $\mathbf{x}_t$ and $\mathbf{h}_t$ are input and output vectors, $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are weight matrices and vector; '$\circ$' is the Hadamard product. $\sigma_g(\cdot)$ and $\sigma_h(\cdot)$ are sigmoid function and tanh function, respectively. For simplifying the problem, the minimal gated unit (MGU) has the following steps.

Initially, $t = 0$, $\mathbf{h}_0 = 0$,

$$\mathbf{f}_t = \sigma_g(\mathbf{W}_f \cdot \mathbf{x}_t + \mathbf{U}_f \cdot \mathbf{h}_{t-1} + \mathbf{b}_f), \qquad (3.102)$$

$$\mathbf{h}_t = \mathbf{f}_t \circ \mathbf{h}_{t-1} + (1 - \mathbf{f}_t) \circ \sigma_h(\mathbf{W}_h \cdot x_t + \mathbf{U}_h(\mathbf{f}_t \circ \mathbf{h}_{t-1}) + \mathbf{b}_h), \qquad (3.103)$$

where $\mathbf{x}_t$ and $\mathbf{h}_t$ are input and output vectors, $\mathbf{f}_t$ is forget vector; $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are weight matrices and bias vector; '$\circ$' is Hadamard product. $\sigma_g(\cdot)$ and $\sigma_h(\cdot)$ are sigmoid functions, and $\tanh(\cdot)$ is an activation function.

In our project regarding human gait recognition [113], we implemented the gait recognition using deep learning and proposed a method based on convolutional Long Short-Term Memory (Conv-LSTM). Firstly, we present a variation of gait energy images (GEI), i.e., frame-by-frame GEI (ff-GEI), to expand the volume of available GEI data and relax the constraints of gait cycle segmentation required by using gait recognition methods. Secondly, we demonstrate the effectiveness of ff-GEI by analyzing the cross-covariance of one person's gait data. Then, making use of the temporality of our human gait, we design a gait recognition model by using Conv-LSTM. The proposed ff-GEI model using Conv-LSTM, coupled with the new gait representation, can effectively solve the problems related to cross-view gait recognition.

In our project regarding video dynamics detection [136], video dynamics detection needs to utilize the present, preceding, and next frames of a given video. In the work, video dynamics detection based on deep learning is implemented and our contributions are to effectively improve the accuracy of video dynamics detection. By combining CNNs and RNNs together, the training time is greatly reduced.

3.3.3 Transformer Models

Most natural language processing (NLP) systems relied on gated RNNs, such as LSTMs and gated recurrent units (GRUs), with added attention mechanisms. Attention mechanisms [115] are in conjunction with RNN nets. RNNs (LSTM, GRU, etc.) have been firmly established approaches in sequence modeling and transduction problems such as language modeling and machine translation [57,58]. RNN models typically factor computation along the symbol positions of the input and output sequences. This nature of RNNs precludes parallelization within training examples.

Transformers are the state-of-the-art (SOTA) model for dealing with sequences, e.g., in text processing, machine translation, computational linguistics, and computer vision. Transformer [7] is regarded as deep learning model that adopts the mechanism of attention, deferentially weighing the significance of each part of the input data. Transformers were from Google Brain for NLP problems by replacing RNN models (LSTM). Like RNNs, Transformers were designed to handle sequential input data, such as natural languages, for tasks such as *translation* and text *summarization*. Unlike RNNs, Transformers do not necessarily process the data in order. The attention mechanism provides context for any position in the input sequence.

The attention mechanism provides context for any position in the input sequence. This feature allows Transformers for more parallelization than RNNs and therefore reduces training times. The additional training parallelization allows model training based on larger datasets. Transformer models can be fine-tuned for specific tasks.

Transformer utilizes an encoder–decoder architecture. The encoder consists of encoding layers that cope with the input iteratively one layer after another, while the decoder consists of decoding layers that work as the same as the encoder. Transformers use attention mechanism and calculate attention weights between them in successive layers.

Each encoder consists of two major components: A self-attention mechanism and a feedforward neural network (FFNN). Each decoder has three major components: A self-attention mechanism, an attention mechanism over the encodings, and a feedforward neural network. Both the encoder and decoder have a feedforward neural network for additional processing of the outputs including residual connections and layer normalization.

Transformer training is based on larger datasets. This has led to the development of pre-trained systems such as BERT model (i.e., Bidirectional Encoder Representations from Transformers) [13,137] and GPT models (i.e., Generative Pre-trained Transformer).

BERT is a family of language models from Google in 2018. The datasets for BERT model training are SQuAD (i.e., Stanford Question Answering Dataset) and SWAG (i.e., Situations With Adversarial Generations). After pre-training, BERT can be fine-tuned with fewer resources on smaller datasets to optimize its performance on specific tasks such as language inference, text classification, question-answering, and conversational response generation.

GPT was trained on large datasets and is famous for its version 3 and the system ChatGPT from OpenAI Laboratory. The differences between ChatGPT system and Google Bard system (bart.google.com) are shown in Fig. 3.5, which were generated by using the Bard system.
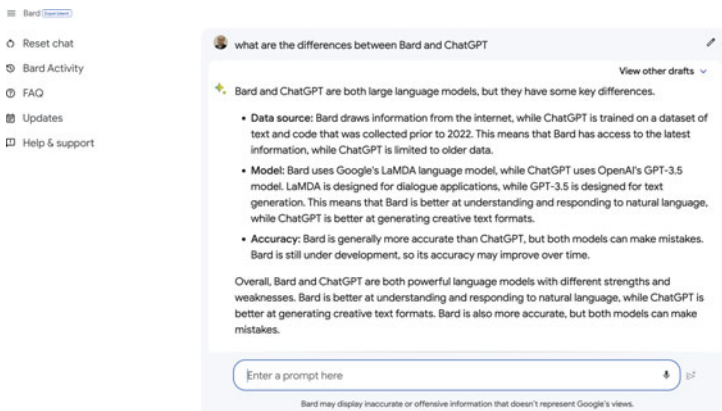
Fig. 3.5 Google Bard system

Algorithm 1: Transformer algorithm

Data: $\mathbf{x}_i$ for token i, $i = 1, 2, 3, \cdots, n$.
Result: Attention $A(\mathbf{Q}, \mathbf{K}, \mathbf{V})$
Construct matrices: $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$;
$\mathbf{q}_i \leftarrow \mathbf{x}_i \mathbf{W}_Q$;
$\mathbf{k}_i \leftarrow \mathbf{x}_i \mathbf{W}_K$;
$\mathbf{v}_i \leftarrow \mathbf{x}_i \mathbf{W}_V$;
$\mathbf{Q} = (q_1, q_2, \cdots, q_n)^\top$;
$\mathbf{K} = (k_1, k_2, \cdots, k_n)^\top$;
$\mathbf{V} = (v_1, v_2, \cdots, v_n)^\top$;
$A(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \leftarrow softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{|\mathbf{k}_i|}}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})$;
d_k is the dimension of the key vectors.

Transformer is the first sequence transduction model based on attention. Transformer can be trained significantly fast which is expected to solve problems involving input and output modalities.

Transformer has rapidly become the dominant architecture for natural language processing, surpassing alternative neural models. Transformer architecture scales with training data and model size, facilitates efficient parallel training, and captures long-range sequence features.

The attention mechanisms are expected to efficiently handle large inputs and outputs such as images, audio, and video. Self-attention is the cornerstone for Transformer models. Image Transformer [46, 80] is a model based on a self-attention mechanism which attains the state-of-the-art modeling images from the standard dataset ImageNet, as measured by using log-likelihood and stacks of self-attention and position-wise feedforward layers. The typical applications of the Image Transformer are image super-resolution and new image generation.

Image Transformer is a model based entirely on a self-attention mechanism. Image Transformer uses stacks of self-attention and position-wise feedforward layers. The dropout, merged in residual connections and layer normalization, is carried out after each application of self-attention and the position-wise feedforward networks. Image Transformer is a sequence modeling formulation of image generation with a tractable likelihood. Image Transformer significantly improves over the state-of-the-art in unconditional, probabilistic image modeling of comparatively complex images from ImageNet.

Vision Transformer (ViT) [14] is a pure Transformer which was applied directly to the sequences of image patches for pattern classification. ViT attains excellent results compared to CNN while requiring substantially fewer computational resources. ViT attains excellent results when pre-trained at sufficient scale and transferred to tasks with fewer data samples. ViT splits an image into fixed-size patches, linearly embeds each of them, adds position embeddings, and feeds the sequence of vectors to a standard Transformer encoder. Vision Transformer can handle arbitrary sequence lengths (up to memory constraints). The first layer of the ViT linearly projects the flattened patches into a lower dimensional space. After the projection, a learned

position embedding is added to the patch representations. Self-attention allows ViT to integrate information across the entire image even in the lowest layers.

Video Transformer (VidTr) [135] was proposed with separable-attention for video classification, which is able to aggregate spatiotemporal information via stacked attentions for human action recognition.

Moreover, a proposed framework is called DEtection TRansformer (DETR) [7,64]. Given a fixed small set of learned object queries, DETR reasons the relationships of the visual objects and the global image context to directly output the final set of predictions in parallel.

Swin Transformer builds hierarchical feature maps by merging image patches in deeper layers which has linear computational complexity to input image size due to the computations of self-attention only within each local window. A key design element of Swin Transformer is its shift of window partition between consecutive self-attention layers.

Swin Transformer constructs a hierarchical representation by starting from small-sized patches and gradually merging neighboring patches in deeper Transformer layers. This hierarchical architecture has the flexibility to model at various scales and has linear computational complexity with respect to image size. With these hierarchical feature maps, Swin Transformer model can conveniently leverage advanced techniques for dense prediction. Swin Transformer architecture is able to achieve the best speed accuracy trade-off among these methods on image classification.

Swin Transformer V2 model is a large dense vision model that makes it capable of training with images of up to $1,536 \times 1,536$ resolution. In Swin Transformer, a local vision Transformer architecture, the window size can be either fixed or changed during fine-tuning. A log-spaced continuous position bias approach (Log-CPB) weights at low resolution to deal with higher resolution windows. The Log-CPB approach is inspired by these efforts while solving a different problem of transferring relative position biases in vision Transformers.

Log-CPB approach adopts a small meta network on the relative coordinates,

$$B(\Delta x, \Delta y) = \mathscr{G}(\Delta x, \Delta y) \tag{3.104}$$

where $\mathscr{G}(\cdot)$ is a small network, which generates bias values for arbitrary relative coordinates. Δx and Δy are the linear-scaled coordinates. Meanwhile, the log-spaced coordinates are

$$\begin{cases} \Delta x' = sign(x) \cdot \log(1 + \Delta x) \\ \Delta y' = sign(y) \cdot \log(1 + \Delta y). \end{cases} \tag{3.105}$$

In our project with regard to early diagnosis of Alzheimer's diseases based on selective kernel network, spatial attention mechanism is added to the bottom of blocks to emphasize important features and suppress unnecessary ones for more accurate representation of the network [44].

In the project about sailboat detection, deep learning models based on attention mechanisms are able to further improve the ability to detect the regions of interest (ROI); we proposed an automated design scheme based on neural architecture

search (NAS) to migrate the attention mechanism for visual object detection and obtain better results of sailboat detection by using both public datasets and our own collected datasets. We verify the effectiveness of our proffered method and evaluate the performance compared with other algorithms [70,71].

Sign language recognition is one of the fundamental ways to assist deaf people to communicate with others. An accurate visual-based sign language recognition system using deep learning is a long-term research goal. In the project sign language recognition [63,64], Vision Transformer related to Detection Transformer (DETR) was proposed to improve the current state-of-the-art sign language recognition accuracy. The proposed method is able to recognize sign language from digital videos with high accuracy.

3.3.4   Generative Pre-trained Transformer Models

3.3.4.1   GTP-1

OpenAI founded in 2015 is an AI research laboratory whose goal is to promote and develop friendly AI (i.e., Artificial Intelligence) that can benefit humanity. GPT [84] was realized by generative pre-training of a large language model on a diverse corpus of unlabeled text, followed by discriminative fine-tuning on each specific task. GPT makes use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. GPT outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state-of-the-art tasks.

The ability to learn effectively from raw text is crucial to alleviate the dependence on supervised learning in natural language processing (NLP). Learning good representations in an unsupervised fashion can provide a significant performance boost, e.g., extensive use of pre-trained word embeddings. Leveraging more than word-level information from unlabeled text is challenging; typically, a semi-supervised approach was explored for language understanding tasks using a combination of pre-training and fine-tuning operations.

Transformers have shown to perform strongly on various tasks. There is a two-stage training procedure: (1) The unlabeled data to learn the initial parameters of a neural network model; (2) The parameters are adapted to a target task using the corresponding supervised objective. The task-specific input adaptations were derived from traversal-style approaches, which treat structured text input as a single contiguous sequence of tokens; the adaptations enable us to fine-tune effectively with minimal changes to the architecture of the pre-trained model. The approach is evaluated based on four types of language understanding tasks—natural language inference, question- answering, semantic similarity, and text classification. The general task-agnostic model outperforms discriminatively trained models that employ architectures specifically crafted for each task.

In large language models, the training procedure consists of two stages: (1) Learning a high-capacity language model on a large corpus of text; (2) A fine-tuning stage to adapt the model to a discriminative task with labeled data, namely pre-training,

fine-tuning, and task-specific input transformations. The training procedure consists of two stages: (1) Training a high-capacity language model based on a large corpus of text; (2) Fine-tuning to adapt the model to a discriminative task with labeled data.

A multilayer Transformer decoder was employed for large language models, which is a variant of the classical Transformers. This model applies a multi-headed self-attention operation over the input context tokens (i.e., after tokenization in natural language processing) followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$h_0 = U W_e + W_p \tag{3.106}$$

$$h_i = T(h_{i-1}) \tag{3.107}$$

$$P(u) = softmax(h_n \cdot W_e) \tag{3.108}$$

where $U = (u_i | u_{i-k}, \ldots, u_{i-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix. After training the model, we adapt the parameters to the supervised target task. Each instance consists of a sequence of input tokens, x_i, $i = 1, \ldots, m$ with a label y. The inputs are passed through the pre-trained model with the final Transformer block's activation h_l^m, h_l^m is later fed into an added linear output layer with parameter W_y to predict y, $P(y | x_1, \ldots, x_m) = softmax(h_l^m \cdot W_y)$, where $softmax(\cdot)$ is the softmax function. This gives us the objective to maximize

$$L_2 = \log \sum P(y | x_1, \ldots, x_m). \tag{3.109}$$

Finally,

$$L_3 = L_2 + \lambda L_1 \tag{3.110}$$

where $L_1 = \log P(u_i | u_{i-k}, \ldots, u_1)$, and $\mathbf{U} = \{u_i\}$ is a set of the given tokens.

A traversal-style approach converts structured inputs into an ordered sequence that the pre-trained model can process. All transformations include adding randomly initialized start and end tokens, textual entailment, similarity, and question-answering and commonsense reasoning:

- In *textual entailment*, we concatenate the premise p and hypothesis h token sequences, with a delimiter token in between.
- In *semantic similarity*, we modify the input sequence to contain both possible sentence orderings (with a delimiter in between) and process each independently to produce two sequence representations h_l^m which are added element-wise before being fed into the linear output layer.

- In *question-answering* and *commonsense reasoning*, we concatenate the document context and question with each possible answer, adding a delimiter token in between. Each of these sequences is processed independently with our model and then normalized via a softmax layer to produce an output distribution over possible answers.

In a framework for achieving natural language understanding with a single task-agnostic model through generative pre-training and discriminative fine-tuning, by pre-training on a diverse corpus with long stretches of contiguous text, the model acquires significant knowledge to process long-range dependencies which are then successfully transferred to solve discriminative problems. GPT offers hints as to what Transformer models and the datasets work best with this approach.

3.3.4.2 GPT-2

Machine learning (ML) models are trained by using a combination of large datasets, high-capacity models, and supervised learning. The dominant approach to create machine learning models is to collect a dataset of training samples, train a model to imitate these behaviors, and test its performance. The current best performance on language tasks is based on a combination of pre-training and supervised fine-tuning. The language models can perform downstream tasks in a zero-shot setting—without any parameters or architecture modifications; hence, zero-shot learning has zero demonstration.

The training dataset WebText contains the text subset of 45 million links. The input representation is Byte Pair Encoding (BPE) between byte level and word level. The model is based on a Transformer from OpenAI GPT having few modifications (OpenAI GPT-2). The models were evaluated by computing the log-probability of sample distribution in a dataset dividing by the number of canonical units. The experiments were conducted based on

- *Children's Book Test*: GPT-2 achieves new state-of-the-art results of 93.3% on common nouns and 89.1% on named.
- *Test the Ability of Systems*: Using LAmbDA (i.e., Label Ambiguous Domain Adaptation Dataset), GPT-2 increases the accuracy from 19 to 52.66%.
- *Winograd Schema Challenge*: GPT-2 improves the state-of-the-art accuracy by 7%, achieving 70.70%.
- *Reading Comprehension*: Using the Conversation Question Answering dataset (CoQA), GPT-2 achieves 55 F1 on the development set.
- *Summarization*: Using CNN and Daily Mail dataset, GPT-2's performance drops by 6.4 points.
- *Translation*: Using WMT-14 English-French test dataset, GPT-2 gets 5 BLEU.
- *Question-Answering*: Using Natural Questions dataset, GPT-2 answers 4.1% of questions correctly.

BLEU (i.e., Bilingual Evaluation Understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another.

Given a candidate corpus, $\hat{S} = (\hat{y}^{(1)}, \ldots, \hat{y}^{(M)})$, and reference candidate corpus $S = (S_1, \ldots, S_M)$, BLEU score with regard to weight $\mathbf{w}$ is

$$BLEU_{\mathbf{w}}(\hat{S}, S) = BP(\hat{S}, S) \cdot \exp\left(\sum_{n=1}^{\infty} w_n \ln p_n(\hat{S}, S)\right), \qquad (3.111)$$

where $\mathbf{w} = (w_1, w_2, \ldots)$, $\sum_{i=1}^{\infty} w_i = 1$, $\forall i \in \{1, 2, 3, \ldots\}$, $w_i \in [0, 1]$:

$$BP(\hat{S}, S) = e^{-(r/c-1)^+} \qquad (3.112)$$

where $(r/c - 1)^+ = \max(0, r/c - 1)$, c is the length of the candidate corpus, and r is the effective reference corpus length. Furthermore,

$$p_n(\hat{S}, S) = \frac{\sum_{i=1}^{M} \sum_{s \in G_n(\hat{y}^{(i)})} \min(C(s, \hat{y}^{(i)}), \max_{y \in S_i} C(s, y))}{\sum_{i=1}^{M} \sum_{s \in G_n(\hat{y}^{(i)})} C(s, \hat{y}^{(i)})}, \qquad (3.113)$$

where $C(\cdot)$ is the substring count and G_n is a set of unique elements. Given any two strings s and y, $C(s, y)$ is the number of appearances of s as a substring of y. $\sum_{s \in G_n(\hat{y})} C(s, y)$ is a number of n-substrings in $\hat{y}$ that appear in y.

The data overlapping between WebText training data and specific evaluation datasets provides a small but consistent benefit to report results. The use of n-gram overlap-based de-duplication is an important verification step and sanity check during the creation of training and test splits for new NLP datasets. The performances based on both the training and test sets of WebText are similar that can be improved as the model size is increased.

The performance of GPT-2 is competitive with the baselines in a zero-shot setting. GPT-2 outperforms on trivial baselines when it has sufficient capacity. The zero-shot performance becomes a baseline of the potential performance of GPT-2.

GPT-2 [85] zero-shot was the state-of-the-art (SOTA) performance on 7 out of 8 tested language modeling datasets. The diversity of this model is able to perform in a zero-shot setting. The high-capacity model was trained to maximize the likelihood of a sufficiently varied text corpus to learn how to accomplish a surprising amount of tasks without the need for explicit supervision.

3.3.4.3   GPT-3

GPT-3 [5] is a computational model designed to generate sequences of words, code, or other data, starting from a source input, called "prompt". The language model was trained based on an unlabeled dataset that is made up of texts, such as Wikipedia and many other sites, primarily in English, but also in other languages. GPT-3 generates automatically and autonomously texts with excellent quality.

GPT-3 works in terms of statistical patterns, e.g., $x + 4 = 10$, $x = 6$. GPT-3 answers questions and solves problems better than many people in mathematics,

physics, and chemistry in semantics. People whose jobs still are related to writing will be supported by GPT-3, increasingly and significantly.

GPT-3 greatly improves task-agnostic and few-shot performance, and even reached the competitiveness with the prior fine-tuning approaches, which is an autoregressive language model with 175 billion parameters, $10\times$ more than any previous non-sparse language model. GPT-3 achieved strong performance on NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation. GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. GPT-3 and its in-context learning abilities were evaluated in various ways. Here are the differences between few-shot learning, one-shot learning, and zero-shot learning in GPT-3:

- Few-shot learning, or in-context learning (i.e., meta learning), has as many demonstrations as it will fit into the model's context window;
- One-shot learning has only one demonstration;
- Zero-shot learning has no demonstrations, only an instruction in natural language is given to the model.

Meta learning [54] is a machine learning algorithm based on metadata; the main goal is to explore how to use metadata to improve the performance of the existing learning algorithms or to learn (induce) model training algorithm itself; it is also called learning to learn.

Larger models make increasingly efficient use of in-context (meta) information. The meta learning means the model develops a broad set of skills and pattern recognition abilities at training time, and then makes use of those abilities at inference time to rapidly adapt to or recognize the desired task.

3.3.4.4   WebGPT

WebGPT [78] is fine-tuned to answer long-form questions using a text-based web-browsing environment, which allows the model to search information from web. WebGPT was designed based on imitation learning, and optimizes answer quality with human feedback. Imitation learning follows essentially a Markov Decision Process (MDP). The simplest form of imitation learning is behavior cloning (BC), which learns from the expert's policy. The agent learns the optimal policy by following and imitating the expert's decisions.

WebGPT was fine-tuned by using behavior cloning, and rejection sampling against a reward model to predict human preferences. Human feedback was employed to directly optimize answer quality to achieve competitive performance.

WebGPT was evaluated based on ELI5, a dataset of questions asked by Reddit users. WebGPT collected examples of humans using a browser to answer questions, which is called *demonstrations*. Human demonstrations usually include

- *Fact-checking*: Human demonstrators provide answers with references.
- *Objectivity*: The more detailed instructions enable more interpretable and consistent comparisons.
- *Blinding*: WebGPT composes answers that are different in style to Reddit answers, making the comparisons less blinded.
- *Answer intent*: With human demonstrations, it is easier to ensure that the desired intent and level of effort are used consistently.

WebGPT collected pairs of model-generated answers to the same question, and asked humans which one they preferred, which is called comparisons. A vast majority of questions were taken from ELI5, a dataset of long-form questions. WebGPT also mixed a small number of questions from other sources, such as TriviaQA [42]. WebGPT designed a similar interface, allowing auxiliary annotations as well as comparison ratings to be provided, though only the final comparison ratings (i.e., better, worse, or equally good overall) were adopted in training. For both demonstrations and comparisons, the answers should be relevant, coherent, and supported by trustworthy references.

WebGPT model is evaluated based on the ELI5 dataset in two different ways: (1) Demonstrators using a web-browsing environment; (2) The reference answers using the ELI5 dataset. The use of human feedback is essential, since one would not expect to exceed 50% preference by imitating demonstrations alone. The evaluations against human demonstrations are much meaningful.

WebGPT was trained primarily to answer questions from ELI5 dataset. The data was from the demonstrations of humans using the web-browsing environment. The data was from the comparisons between two model-generated answers to the same question. The generated answers are evaluated based on factual accuracy, coherence, and overall usefulness.

WebGPT makes use of a combination of behavior cloning and rejection sampling. The collected data was applied in four ways: (1) Behavior cloning (i.e., supervised fine-tuning) using the demonstrations; (2) Reward modeling using comparisons; (3) Reinforcement learning against reward model; (4) Rejection sampling against reward model.

The WebGPT model training includes the following: (1) WebGPT is fine-tuned based on demonstrations using supervised learning, with the commands issued by the human demonstrators as labels. (2) WebGPT trained a model to take in a question and an answer with references, and output a scalar reward. (3) WebGPT took the reward model score at the end of each episode, and added this to a KL penalty from the behavior cloning (BC) model at each token to mitigate overoptimization of the reward model. (4) WebGPT sampled a fixed number of answers from either the BC model or the reinforcement model and selected the one that was ranked as the highest by the reward model.

The best model is assessed in three ways: (1) Comparing the generated answers from WebGPT model to the answers written by human demonstrators on a held-out set of questions. (2) Comparing WebGPT answers to the highest-voted answer

provided by the ELI5 dataset. (3) Comparing the WebGPT answers based on Truth-fulQA, an adversarial dataset of short-form questions.

Let Q be the distribution of questions; given a question q, $A(q)$ is the distribution of answers produced by the model, and a is an answer with references; let $R^{train}(a|q)$ be the original reward model score, and let $R^{val}(a|q)$ be the validation reward model score. Let n be the number of answers sampled while conducting rejection sampling (i.e., n in the best-of-n). To predict Elo score,

$$R_n^{pred} = \mathbb{E}_{A_i \sim A(q), i=1,\ldots,n}[R^{val}(\arg \max_{a \in \{Ai\}} R^{train}(a|q))|q]. \qquad (3.114)$$

The general Elo score is

$$S_{Elo} = \mathbb{E}_Q(R_n^{pred}(Q)). \qquad (3.115)$$

Elo ratings [103] are comparative only, and are valid only within the rating pool in which they were calculated, rather than being an absolute measure. The simplest way to estimate R_n^{pred} for a given question q is with a Monte Carlo estimator:

$$\mathbb{E}_{A_i \sim A(q), i=1,\ldots,n}[R^{val}(\arg \max_{a \in \{Ai\}} R^{train}(a|q))|q]$$

$$= 1/C_n^i \cdot \sum [R^{val}(\arg \max_{a \in \{Ai\}} R^{train}(a|q))|q] = \sum \frac{C_{n-1}^{i-1}}{C_n^i} \cdot R^{val}(S_i|q)$$

$$= \sum \frac{C_{n-1}^{i-1}}{C_n^i} \cdot R^{val}(S_i|q). \qquad (3.116)$$

Rejection sampling [21] outperforms reinforcement learning. In numerical analysis and computational statistics, rejection sampling is to generate observations from a distribution. It is also called the acceptance–rejection method or accept–reject algorithm.

The rejection sampling method generates sampling values from a target distribution X with arbitrary probability density function $f(x)$ by using a proposal distribution Y with probability density $g(x)$. One can generate and accept a sample x from X by using $u < f(x)/(M \cdot g(x))$, $u \sim \mathbf{U}(0, 1)$, $\mathbf{U}(0, 1)$ is the uniform distribution over the unit interval $(0, 1)$, if not, the sample x will be rejected and the sampling step will be restarted, repeating the process until a value is accepted, where M is a constant, the finite bound is based on the likelihood ratio $f(x)/g(x)$, satisfying $1 < M < \infty$ and $f(x) \leq M \cdot g(x)$, $g(x) > 0$, $f(x) > 0$.

With rejection sampling, the model can visit many websites, and then evaluate the information with the benefit of hindsight. The reward model was trained primarily on data collected from rejection sampling policies, which may have made it more robust to overoptimization by rejection sampling.

Reinforcement learning requires hyperparameter tuning, whereas rejection sampling does not. The combination of reinforcement learning and rejection sampling also fails to offer much benefit over rejection sampling alone. Reinforcement learning reduces the entropy of policy, which hurts exploration.

WebGPT synthesizes information from the existing sources giving it the potential to reinforce and entrench the existing beliefs and norms. WebGPT usually accepts the implicit assumptions made by questions. WebGPT is used, both by limiting access and by tailoring the design and documentation of applications.

3.3.4.5 InstructGPT

InstructGPT model is a turning point in the development of OpenAI GPT models. InstructGPT model shows great improvements in truthfulness and reductions in output textual generation while having minimal regressions on public datasets. InstructGPT is an aligning language model with user intent on a wide range of tasks by fine-tuning with human feedback. InstructGPT collects a dataset of ranking model outputs, which was employed to further fine-tune this supervised model using reinforcement learning from human feedback.

InstructGPT is helpful, honest, and harmless. InstructGPT is use of reinforcement learning from human feedback to fine-tune GPT-3 model to follow a broad class of instructions. Human preferences were employed as a reward signal. A team of 40 contractors were employed to label the training data, based on the performance on a screening test. A dataset of human-written demonstrations of the desired output behavior on (mostly in English) prompts is submitted to OpenAI API-3 model for the supervised learning baselines. A reward model (RM) was trained based on this dataset to predict which model outputs the labels. Correspondingly, a reward function is adopted to fine-tune the supervised learning and maximize this reward using the proximal policy optimization (PPO) algorithm. Policy gradient methods compute the policy gradient by using a stochastic gradient ascent (SGA) algorithm. The gradient estimator is

$$g = \mathbf{E}_t[\nabla_\theta \log \pi_\theta(a_t|s_t) A_t] \tag{3.117}$$

where π_θ is a stochastic policy, A_t is an estimator of the advantage function at time step t, and $\mathbf{E}_t$ indicates the empirical average over a finite batch of samples. In implementations,

$$L(\theta) = \mathbf{E}_t[\log \pi_\theta(a_t|s_t) A_t]. \tag{3.118}$$

An objective function or surrogate objective function is maximized which is subject to a constraint based on the policy update:

$$\max_\theta \mathbf{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t] \tag{3.119}$$

s.t.

$$\mathbf{E}_t[KL(\pi_\theta(\cdot|s_t)\|\pi_{\theta_{old}}(\cdot|s_t)] \leq \delta \tag{3.120}$$

where $\pi_{\theta_{old}}(a_t|s_t)$ is the vector of policy parameters before the update, $\delta > 0$. Using a penalty β,

$$L(\theta) = \max_\theta \mathbf{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t - \beta[KL(\pi_\theta(\cdot|s_t)\|\pi_{\theta_{old}}(\cdot|s_t)]. \tag{3.121}$$

The main objective is

$$L(\theta) = \mathbf{E}_t[\min[r_t(\theta)A_t, clip(r_t(\theta), 1 + \varepsilon, 1 - \varepsilon)A_t]] \qquad (3.122)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, and $clip(\cdot)A_t$ modifies the surrogate objective by clipping the probability ratio.

InstructGPT models show significant improvements in truthfulness over GPT-3. InstructGPT demonstrates promising generalization to instructions outside of the fine-tuning distribution. InstructGPT generalizes to the preferences of "held-out" human labelers that did not produce any training data.

The cost of increasing model alignment is modest with regard to pre-training. InstructGPT generalizes instructions to the settings that are not supervised. Instruct-GPT mitigates most of the performance degradation by using fine-tuning operation.

InstructGPT aligns with demonstrations and preferences provided by using human labelers. OpenAI API customers are the right source of human preferences. Instruct-GPT implicitly aligns with what the end-users think is valuable.

InstructGPT aligns to an specific human reference group for a specific application. The question is how to design an alignment process that is transparent; people's values are synthesized in a way that achieves broad consensus. The question of who these models are aligned with is extremely important, and will significantly affect whether the impact of these models is positive or negative.

3.3.4.6   GPT-3.5

As well known, OpenAI GPT-3.5 is named as ChatGPT. Human labels are employed to train a model of reward, and then optimize that model. Reward is defined by human's judgment; a reward is given by asking human's questions. Reward learning for natural language processing is a key to make reinforcement learning practical and safe for real-world tasks. Reward learning in GPT is successfully employed to natural language tasks.

GPT-3.5 [126] is to pre-train a large generative model based on a corpus of unlabeled data, then fine-tune the model for supervised NLP tasks. GPT often substantially outperforms other models in training based on the labeled datasets from scratch. A single pre-trained model often can be fine-tuned for the performance based on a variety of datasets. The generatively trained models show reasonable performance for NLP tasks with no additional training (zero-shot).

In GPT-3.5, labeled datasets are unavailable or insufficient, where programmatic reward functions are poor for the true goals. The pre-training in natural language processing is combined with human preferences. The pre-trained models are fine-tuned with reinforcement learning rather than supervised learning, using a reward model trained from human preferences based on text continuations. KL divergence as a constraint has been harnessed to prevent the fine-tuned model from drifting too far from the pre-trained model. The learning from human feedback was explored more

generally at a larger computational scale. Starting with a vocabulary set, a language model is defined from a probability distribution p over a sequence of tokens via

$$p(x_0, v, x_{n-1}) = \prod_{0 \leq k < n} p(x_k | x_0, \ldots, x_{k-1}). \tag{3.123}$$

A policy π is initialized if the task was defined by a reward function r; reinforcement learning is employed to directly optimize the expected reward,

$$\mathbf{E}_\pi[r] = \mathbf{E}_{x \in X, y \in Y}[r(x, y)] \tag{3.124}$$

where X and Y are output space and input space, respectively.

The tasks defined by human judgments were accomplished owing to what we can learn about the rewards by asking humans. Human labels as typical annotations were employed to train a reward model and further optimize the reward model. The human labelers were required to pick which value of y_i is the best response to a given input x. A reward model r was fitted by using loss function,

$$loss(r) = \mathbf{E}_{(x, \{y_i\}, b)} \log[\frac{e^{r(x, y_b)}}{\sum e^{r(x, y_i)}}] \tag{3.125}$$

where b is the best option.

The policy π was fine-tuned to optimize the reward model r; a penalty was added with the mathematical expectation; the reinforcement learning was created based on the modified reward,

$$R(x, y) = r(x, y) - \beta \log \frac{\pi(x|y)}{\rho(x|y)} \tag{3.126}$$

where we either choose a constant β or vary it dynamically to achieve a particular value. Additionally, (1) this plays the role of an entropy bonus; (2) This prevents the policy from moving too far from the range; (3) This also is an important part of the task definition; (4) Based on this, the coherence and topicality of reinforcement learning models are required to be developed further.

ChatGPT [126] gathered data samples and asked humans to pick the best y_i, furthermore to initialize r and ρ using random initialization for the final linear layer. ChatGTP-3.5 trains π via Proximal Policy Optimization (PPO) algorithm with reward R. In the online data collection, it continues collecting additional data samples, and periodically retraining the reward model r.

With the pre-trained language model, the reward model was trained by using Adam optimizer. Adam optimizer is an optimization algorithm for stochastic gradient descent for training deep learning models. For training the policy π, we make use of the PPO2 algorithm, namely version 2 of Proximal Policy Optimization algorithm. The large language model was trained with different seeds and the same KL divergence as penalty; the model with the best validation loss is chosen. The human data samples were collected throughout the fine-tuning process while continuously gathering new data by sampling and retraining the reward model. A function $l(n)$

was selected for describing how many labels we want before beginning the n-th reinforcement learning episodes. To estimate overall progress, we gather validation samples.

PPO is a family of model-free reinforcement learning algorithms which have been developed at OpenAI since 2017. PPO algorithms are policy gradient methods, which alternate between sampling data through interaction with the environment, and optimize an objective function using stochastic gradient ascent (SGA). The standard policy gradient methods accomplish one gradient update using per data sample, and the new objective function that enables multiple epochs of mini-batch updates. PPO algorithms are simpler to implement, more general, and have better sample complexity. It is fulfilled by using a different objective function.

Scale API (i.e., Application Programming Interface) from OpenAI Laboratory was employed to collect data samples and labels. The Scale API accepts requests of the form and returns the selection; the task is described through a combination of instructions. The models are incentivized to exploit idiosyncrasies of the labeling process.

Online data collection is hard (e.g., complexities and quality control issues). The right middle ground between offline and online data collection is batched in data collection, which is a well-studied setting for active learning. While sharing parameters between reward models, ambiguous tasks make the labeling work hard.

The previous reward learning was extended with pre-trained models and KL-divergence-based regularization to prevent the policy from diverging too far from natural languages. The superior results were achieved by using the zero-shot baseline which is evaluated by humans with very few samples. The interactive communication between humans and machine learning models is a requirement for scalable reward learning methods. Using direct human preferences for language tasks is a necessary step in the direction of scalable reward learning for natural languages, and further steps are possible.

3.3.4.7   GPT-4

GPT-4 [127] is a large-scale and multimodal model that can accept image and text inputs and produce text outputs. GPT-4 exhibits human-level performance on various professional and academic benchmarks. GPT-4 is a Transformer-based model which was pre-trained to predict the next token in a document.

GPT-4 is a large multimodal model capable of processing image and text inputs and producing text outputs. GPT-4 is to improve the ability to understand and generate natural language text, particularly in more complex and nuanced scenarios. GPT-4 was evaluated based on a variety of exams originally designed for humans. GPT-4 surpasses the English-language state-of-the-art in 24 of 26 languages. GPT-4 has a limited context window, which does not learn from experience like others.

GPT-4 is a Transformer-style model pre-trained to predict the next token in a document using both publicly available data and data licensed from third-party providers. GPT-4 model was fine-tuned using Reinforcement Learning from Human Feedback (RLHF). A focus of GPT-4 project was on building a deep learning stack that scales

predictably. The loss of properly trained large language models is thought to be well approximated by power laws in the amount of computations which were used to train the deep learning model.

In GPT-4, a methodology was developed to predict more interpretable metrics of capability: (1) The pass rate based on the HumanEval dataset from OpenAI measures the ability to synthesize Python functions of varying complexity. (2) The metric final loss is based on a dataset derived from the internal codebase.

GPT-4 was tested based on a diversity of benchmarks, including simulating exams that were originally designed for humans. The exams were sourced from publicly available materials. The evaluation was designed based on performance on a validation set of exams and held-out test exams. Overall, the scores were determined by combining multiple-choice and free-response question scores using publicly available methodologies for each exam. GPT-4 exhibits human-level understanding on the majority of these professional and academic exams. GPT-4 considerably outperforms the existing models, as well as previous systems from OpenAI.

GPT-4 has similar limitations as the earlier GPT models. GPT-4 makes progress on public benchmarks, which tests the ability to separate the fact from an adversarially selected set of incorrect statements. GPT-4 generally lacks knowledge of events that have occurred after September 2021. GPT-4 can be confidently wrong in its predictions which has various biases in its outputs.

In general, GPT-4 is a large multimodal model with human-level performance on professional and academic benchmarks. GPT-4 outperforms the existing large language models based on a collection of NLP tasks and exceeds a vast majority of the state-of-the-art systems. GPT-4 shows a significant step toward broadly useful and safely deployed AI systems.

We have a project related to Objective-Oriented Transformer (OOT) for document abstractive summarization [129]. In this project, a pre-trained Transformer was treated as the starting point for a majority of NLP tasks such as document summarization. A Transformer model forms the basis for any type of models in which it is trained to produce a sequence of text given an input. The produced sequence emulates the embedded semantics between the input and output sequences in the training time. Hence, a Transformer can be trained to emulate generic semantics. In this project, we present a document summarization model to generate three different outputs. These outputs are then merged to generate the final summary.

3.3.5 Time Series Analysis

With time changes, the states will be altered. Time series analysis [32,72] is applied to deal with these state changing issues. For example, time series analysis has been applied to water quality control or air quality assessment with time changes. We need to find out the patterns behind time series analysis after our observations [8].

Observation is one of the steps of artificial intelligence (AI) besides learning, presentation, planning, and inference or reasoning. AI can predict what will happen from what have already happened; we also call it forecasting or prediction [16,77].

There are two main goals of time series analysis: Identifying the nature of phenomenon represented by using a sequence of observations; forecasting is to predict future values of the time series variable.

Most of time series patterns are described in terms of a basic class of components: Trend analysis (i.e., smoothing and fitting a function), analysis of seasonality (i.e., autocorrelation correlogram, examining correlograms, partial autocorrelations, and removing serial dependency).

In time series analysis, we use the concept of seasonality that means the patterns usually follow the seasons annually. We make use of the changes of data in time series to analyze the patterns.

In time analysis, for a sequence $x(1), x(2), \ldots, x(t), \ldots, x(i) \in \mathcal{R}, i = 1, 2, \ldots,$ we have

- Mean
$$\mu_t = \mathbf{E}(X(t)). \tag{3.127}$$

- Variance
$$\sigma^2 = Var(X(t)). \tag{3.128}$$

- Autocovariance
$$\gamma(t_1, t_2) = \mathbf{E}\{[X(t_1) - \mu(t_1)][X(t_2) - \mu(t_2)]\}. \tag{3.129}$$

- Autocovariance lag
$$\gamma(\tau) = \mathbf{E}\{[X(t) - \mu][X(t + \tau) - \mu]\}. \tag{3.130}$$

- Autocovariance function
$$\rho(\tau) = \gamma(\tau)/\gamma(0), \tag{3.131}$$

$$\rho(\tau) = \rho(-\tau), |\rho(\tau)| < 1. \tag{3.132}$$

- Random walk
$$X_t = X_{t-1} + Z_t. \tag{3.133}$$

- MA(q) process
$$X_t = \beta_0 Z_t + \beta_1 Z_{t-1} + \cdots + \beta_q Z_{t-q}. \tag{3.134}$$

- AR(p) process
$$X_t = \alpha_1 X_{t-1} + \cdots + \alpha_p X_{t-p} + Z_t. \tag{3.135}$$

- Mixed autoregressive moving average model (ARMA)
$$X_t = \alpha_1 X_{t-1} + \cdots + \alpha_p X_{t-p} + Z_t + \beta_0 Z_t + \cdots + \beta_q Z_{t-q}. \tag{3.136}$$

- Exponential smoothing
$$S(x_t) = \sum_{j=0}^{\infty} \alpha(1 - \alpha)^j x_{t-j}. \tag{3.137}$$

- Residual

$$R(x_t) = x_t - S(x_t), \tag{3.138}$$

$$a_j = \alpha(1 - \alpha)^j. \tag{3.139}$$

- Convolution

$$\{c_k\} = \{a_r\} \star \{b_j\} \Leftrightarrow c_k = \sum_r a_r \cdot b_{k-r}, \tag{3.140}$$

$$z_t = \sum_k c_k \cdot x_{t+k} = \sum_j b_j \cdot y_{t+j}. \tag{3.141}$$

- Additive seasonality

$$X_t = m_t + S_t + \varepsilon_t. \tag{3.142}$$

- Multiplicative seasonality

$$X_t = m_t \cdot S_t + \varepsilon_t. \tag{3.143}$$

In time series analysis, we need to remove the noises from the collected data. Usually, a kernel for convolution operation is needed [120]. The convolution operation will reduce the noises and make the signals smooth, especially for digital images [67].

In time analysis, we usually need to use spectrum analysis. Fourier series [120] has been applied to signal decomposition and multiresolution analysis based on trigonometric function because sine and cosine functions construct an orthogonal function system:

$$f(t) \approx \frac{a_0}{2} + \sum_{r=1}^{k}(a_r \cos rt + b_r \sin rt), \tag{3.144}$$

where

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t)dt, \tag{3.145}$$

$$a_r = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \cos rt dt, \tag{3.146}$$

and

$$b_r = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \sin rt dt. \tag{3.147}$$

Kalman filter [4,49,125] is a linear model for object tracking or signal filtering. Kalman filter has an updating function and a predicting function. Dynamical updates have been applied to improve the parameters for the purpose of prediction and correction.

- Prediction equations:

$$\theta_{t|t-1} = G_t \theta_{t-1}, \tag{3.148}$$

  and

$$P_{t|t-1} = G_t P_{t-1} G_t^\top + w_t. \tag{3.149}$$

- Updating equations:

$$\theta_t = \theta_{t|t-1} + K_t e_t, \tag{3.150}$$

  and

$$P_t = P_{t|t-1} - K_t h_t^\top P_{t|t-1}, \tag{3.151}$$

  where

$$K_t = P_{t|t-1} h_t / [h_t^\top P_{t|t-1} h_t + \sigma_n^2]. \tag{3.152}$$

For a nonlinear system in time series analysis, we have the following:

- Nonlinear autoregressive model (NLAR) is

$$X_t = f(X_{t-1}, X_{t-2}, \ldots, X_{t-p}) + Z_t. \tag{3.153}$$

- Threshold autoregressive model is

$$X_t = \begin{cases} \alpha_1 X_{t-1} + Z_t & \text{if } X_{t-1} < r \\ \alpha_2 X_{t-1} + Z_t & \text{if } X_{t-1} \geq r. \end{cases} \tag{3.154}$$

- Artificial neural networks:

$$y = \phi_0 \left[\sum_j w'_j \phi_h \left(\sum_i w_{ij} x_i \right) + w'_0 \right], \tag{3.155}$$

where y is the output, $v_j = \sum_i w_{ij} x_i$, and ϕ_h is the activation function.

LSTM has been applied to time series analysis for forecasting, especially MAT-LAB has provided a program for predicting or forecasting; meanwhile, LSTM is able to provide RMS errors for the prediction. The predictions are much accurate while updating the network state with the observed values instead of the predicted values.

In order to forecast the values of future a time step of a sequence, we train a regression LSTM network, where the responses are the training sequences with values shifted by one time step. That is, at each time step of the input sequence, the LSTM network has the ability to predict the state of the next time step.

If we access to the actual values of time steps between predictions, then we can update the network state with the observed values. For each prediction, we predict the next time step by using the observed value of the previous time step. We calculate the root-mean-square error (RMSE). Abnormal detection or anomaly detection is one of the typical applications of time series analysis [10].

3.4 Functional Analysis

In deep learning, we need to calculate the loss function; actually, it is a kind of distance. In this section, we will introduce how to measure the distances in functional spaces.

3.4.1 Metric Space

Metric spaces [76] are thought as very basic ones where the ideas of convergence and continuity exist. Distance or metric is a measure of how close two elements are to each other [77].

A distance (or metric) on a metric space $\mathbf{X}$ is a function: $d : \mathbf{X}^2 \to \mathbf{Y} \subset \mathscr{R}^+$; $(x, y) \mapsto d(x, y) = \|x - y\|$, $x, y, z \in \mathbf{X}$.

- Triangle inequality is $d(x, y) \leq d(x, z) + d(z, y)$.
- Symmetry is $d(x, y) = d(y, x)$.
- Equality is $d(x, y) = 0 \Leftrightarrow x = y$.

Therefore, we have

- $d(x, y) \geq |d(x, z) - d(z, y)|$.
- $x_1, x_2, \ldots, x_n \in \mathbf{X}$, $d(x_1, x_n) \leq \sum_{i=1}^{n-1} d(x_i, x_{i+1})$.

A sequence (x_n) in a metric space $\mathbf{X}$ converges to a limit x, denoted as $\lim_{n\to\infty} x_n = x$, $n \in \mathscr{Z}^+$ if $\forall \varepsilon > 0$, $\exists N, n \gg N \Rightarrow x_n \in B_\varepsilon(x)$.

A function $f : \mathbf{X} \to \mathbf{Y}$ between metric spaces is continuous when it preserves convergence, $x_n \to x \in \mathbf{X} \Rightarrow f(x_n) \to f(x) \in \mathbf{Y}$. If a function f is continuous, it is invertible and its inverse $f^{-1}(x)$ is continuous.

A Cauchy sequence is such a one that $d(x_n, x_m) \to 0$ as $n, m \to \infty$; namely $\forall \varepsilon > 0$, $\exists N$, if $n, m \geq N$, then $d(x_n, x_m) < \varepsilon$.

A sequence $x_1, x_2, \ldots, x_n, \ldots$ is Cauchy sequence, if and only if every subsequence is asymptotic to this sequence.

A uniformly continuous function maps any Cauchy sequence to a Cauchy sequence. A function is *Uniformly continuous*, which refers to $\delta > 0$ being independent of x_i.

A function $f : \mathbf{X} \to \mathbf{Y}$ is a *Lipschitz map* when $\exists c > 0$, $\forall x, x' \in \mathbf{X}$, $d_{\mathbf{Y}}(f(x), f(x')) \leq c \cdot d_{\mathbf{X}}(x, x')$.

A metric space is complete if every Cauchy sequence converges, e.g., the real space is complete.

A metric space is separable if it contains a countable dense subset, $\exists \mathbf{A} \subseteq \mathbf{X}$, $\mathbf{A}$ is countable, and $\bar{\mathbf{A}} = \mathbf{X}$.

A set $\mathbf{B}$ is bounded if the distance between any two points in the set has an upper bound, i.e., $\exists r > 0$, $\forall x, y \in \mathbf{B}$, $d(x, y) \leq r$.

The least upper bound is called the diameter of the set: $diam\ \mathbf{B} := \sup_{x,y\in\mathbf{B}} d(x, y)$.

A set $\mathbf{K}$ is compact, given any cover of balls, there is a finite subcollection of them that still cover the set (a subcover) $K \subseteq \bigcup_i B_{\varepsilon_i}(a_i) \Rightarrow K \subseteq \bigcup_{n=1}^{N} B_{\varepsilon_{i_n}}(a_{i_n})$.

3.4.2   Vector Space

Distance is a scalar value. Given two n-dimensional vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)^\top$, Euclidean distance is

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}. \tag{3.156}$$

Manhattan distance is

$$d = \sum_{i=1}^{n}(|x_i - y_i|). \tag{3.157}$$

Manhattan is a region in New York city, New York, USA. An example of distance is shown in Fig. 3.6 for the distance from the Times Square to the United Nations Headquarters in Manhattan downtown area.
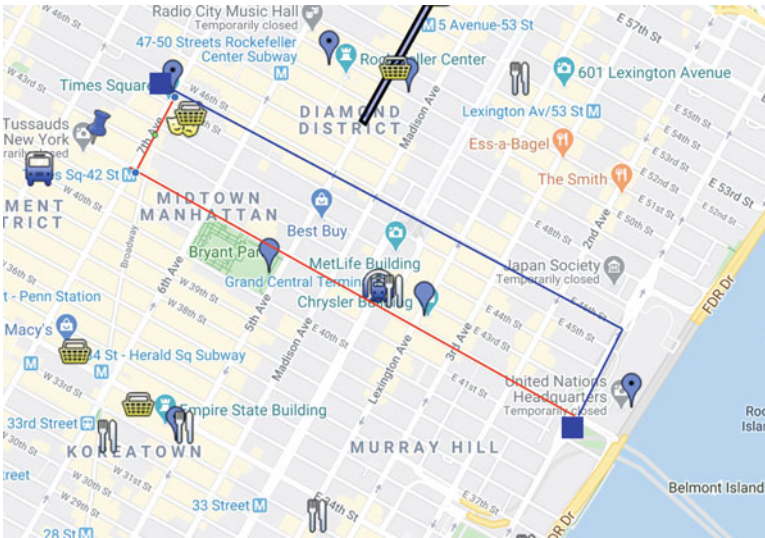
Fig. 3.6 The Google map at Manhattan downtown in New York city. The distance between United Nations Headquarters and Times Square is roughly same no matter which way will be taken in the city (red and blue)

Chebyshev distance is

$$d = \max_{i=1}^{n}(|x_i - y_i|). \tag{3.158}$$

Minkowski distance is a metric which is a generalization of both the Euclidean distance and the Manhattan distance:

$$d = \lim_{n \to +\infty} \left(\sum_{i=1}^{n} (x_i - y_i)^p \right)^{\frac{1}{p}} = \max_{i=1}^{n}(|x_i - y_i|), \tag{3.159}$$

and

$$d = \lim_{n \to -\infty} \left(\sum_{i=1}^{n} (x_i - y_i)^p \right)^{\frac{1}{p}} = \min_{i=1}^{n}(|x_i - y_i|). \tag{3.160}$$

Given sets $\mathbf{A}$ and $\mathbf{B}$, Jaccard index, also known as Intersection over Union (IOU), or the Jaccard similarity coefficient, is

$$J(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cap \mathbf{B}}{\mathbf{A} \cup \mathbf{B}}. \tag{3.161}$$

Jaccard distance measures dissimilarity between sample sets

$$J_d(\mathbf{A}, \mathbf{B}) = 1 - J(\mathbf{A}, \mathbf{B}) = 1 - \frac{\mathbf{A} \cap \mathbf{B}}{\mathbf{A} \cup \mathbf{B}}. \tag{3.162}$$

Mahalanobis distance is defined as a dissimilarity measure between two n-dimensional vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)^\top$ of the same distribution with the covariance matrix Σ

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top \Sigma^{-1} (\mathbf{x} - \mathbf{y})} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - y_i)^2}{\sigma_i^2}}, \tag{3.163}$$

where σ_i is the standard deviation of x_i and y_i over the sample set.

In statistics, Pearson correlation coefficient or the bivariate correlation measures linear correlation between two n-dimensional variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$:

$$d(\mathbf{x}, \mathbf{y}) = \frac{cov(\mathbf{x}, \mathbf{y})}{\sigma_{\mathbf{x}} \sigma_{\mathbf{y}}} = \frac{\mathscr{E}(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{y} - \bar{\mathbf{y}})}{\sigma_{\mathbf{x}} \sigma_{\mathbf{y}}} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{3.164}$$

where $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$ and $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$ are means, $cov(\mathbf{x}, \mathbf{y})$ is the covariance, $\sigma_{\mathbf{x}}$ is the standard deviation of $\mathbf{x}$, $\sigma_{\mathbf{y}}$ is the standard deviation of $\mathbf{y}$, and $\mathscr{E}$ is the expectation.

A vector space $\mathbf{V}$ [76] over a field $\mathbf{F}$ is a set on which an operation of vector addition is defined $+ : \mathbf{V}^2 \to \mathbf{V}^2$ satisfying associativity, commutativity, zero and inverse axioms:

- For every $x, y, z \in \mathbf{V}$,
  $x + (y + z) = (x + y) + z, x + y = y + x,$
  $0 + x = x, x + (-x) = 0.$
- An operation of scalar multiplication $\mathbf{F} \times \mathbf{V} \to \mathbf{V}$ satisfies the respective distributive laws.
- For every $\lambda, \mu \in \mathbf{F}$,
  $\lambda(x + y) = \lambda x + \lambda y, (\lambda + \mu)x = \lambda x + \mu x,$
  $(\lambda\mu)x = \lambda(\mu x), 1x = x.$

Every vector has a base. For an n-dimensional vector $\mathbf{V} = (v_1, v_2, \ldots, v_n)^\top$, we have the corresponding p-norm $\|\mathbf{V}\|_p = (\sum_{i=1}^{n} v_i^p)^{\frac{1}{p}}$, $p = 0, 1, 2, \ldots, \infty$. If $p = 1$, then it is 1-norm $\|\mathbf{V}\|_1 = (|\sum_{i=1}^{n} v_i|)$; if $p = 0$, then it is zero norm $\|\cdot\|_0 = min_i(|v_i|)$; if $p = \infty$, then it is infinity norm or maximum norm $\|\cdot\|_\infty = max_i(|v_i|)$.

3.4.3 Normed Space

A norm on a real vector space $\mathbf{X}$ is a function: $\mathbf{X} \to \mathscr{R}, \mathbf{u} \mapsto \|\mathbf{u}\|$:

- $\forall\, \mathbf{u} \in \mathbf{X}, \|\mathbf{u}\| \geq 0.$
- $\forall\, \mathbf{u} \in \mathbf{X}, \alpha \in \mathbf{R}, \|\alpha\mathbf{u}\| = |\alpha|\|\mathbf{u}\|.$
- $\forall\, \mathbf{u}, \mathbf{v} \in \mathbf{X}, \|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ (Minkowski's inequality).

A normed space $\mathbf{X}$ is a vector space over $\mathbf{F} = \mathbf{R}$ or $\mathbf{C}$ with a function called the norm $\|\cdot\| : \mathbf{X} \to \mathscr{R}$ such that for any $x, y \in \mathbf{X}, \lambda \in \mathbf{F}, \|x + y\| \leq \|x\| + \|y\|$, $\|\lambda x\| = \lambda\|x\|, \|x\| = 0 \Leftrightarrow x = 0$. Thus,

$$\|x - y\| \geq \|x\| - \|y\|, \tag{3.165}$$

and

$$\|x_1 + x_2 + \cdots + x_n\| \leq \|x_1\| + \|x_2\| + \cdots + \|x_n\|. \tag{3.166}$$

Given

$$\|(a_n)\|_2 = \sqrt{\sum_{n=0}^{\infty} \|a_n\|^2}, \tag{3.167}$$

and

$$\|(b_n)\|_2 = \sqrt{\sum_{n=0}^{\infty} \|b_n\|^2}, \tag{3.168}$$

Cauchy's inequality is

$$|\sum_{n=0}^{\infty} a_n b_n| \leq \|(a_n)\|_2 \|(b_n)\|_2, \tag{3.169}$$

and

$$\sqrt{\sum_{n=0}^{\infty} \|a_n + b_n\|^2} \leq \|(a_n)\|_2 + \|(b_n)\|_2. \tag{3.170}$$

- Vector addition, scalar multiplication, and the norm are continuous.
- If (x_n) and (y_n) converge, $(x_n + y_n)$, (λx_n), and $(\|(x_n)\|)$ converge, namely

$$\lim_{n\to\infty} (x_n + y_n) = \lim_{n\to\infty} x_n + \lim_{n\to\infty} y_n, \tag{3.171}$$

$$\lim_{n\to\infty} (\lambda x_n) = \lambda \lim_{n\to\infty} x_n, \tag{3.172}$$

$$\lim_{n\to\infty} \|x_n\| = \| \lim_{n\to\infty} x_n \|. \tag{3.173}$$

3.4.4 Hilbert Space

Hilbert spaces are Banach spaces with a norm derived from a scalar product [76]. A scalar product on the (real) vector space $\mathbf{X}$ is a function: $(u, v) \in \mathbf{X} \times \mathbf{X} \to \mathscr{R}$, $(u, v) \mapsto (u|v)$:

- $\forall \mathbf{u} \in \mathbf{X}$, $(\mathbf{u}|\mathbf{u}) \geq 0$.
- $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbf{X}$, $\alpha, \beta \in \mathscr{R}$, $(\alpha\mathbf{u} + \beta\mathbf{v}|\mathbf{w}) = \alpha(\mathbf{u}|\mathbf{w}) + \beta(\mathbf{v}|\mathbf{w})$. $\|\mathbf{u}\| = \sqrt{(u|u)}$.

A Hilbert space is an inner product space which is complete as a metric space. An inner product on a vector space is a positive-definite sesquilinear form: $<, >$: $\mathbf{X} \times \mathbf{X} :\to \mathbf{F}$. For $x, y, z \in \mathbf{X}$, $\lambda \in \mathbf{F}$:

- $< x, y + z > = < x, y > + < x, z >$, $< x, \lambda y > = \lambda < x, y >$, $< x, y > = \overline{< y, x >}$, $< x, x > \geq 0$, $< x, x > = 0 \Rightarrow x = 0$.
- Cauchy–Schwarz inequality:

$$| < x, y > | \leq \|x\|\|y\|. \tag{3.174}$$

- Pythagoras' theorem:

$$\|x + y\| \leq \|x\| + \|y\|. \tag{3.175}$$

- The inner product is continuous if

$$\lim_{n\to\infty} < x_n, y_n > = < \lim_{n\to\infty} x_n, \lim_{n\to\infty} y_n > . \tag{3.176}$$

- A norm is induced from an inner product, if and only if it satisfies, for all vectors $x, y \in \mathscr{R}$,

$$\|x + y\|^2 + \|x - y\|^2 = 2(\|x\|^2 + \|y\|^2). \tag{3.177}$$

The orthogonal spaces of subsets $\mathbf{A} \subseteq \mathbf{X}$,

$$\mathbf{A}^{\perp} := \{\forall x \in X, <x, a> = 0, \forall a \in \mathbf{A}\}, \tag{3.178}$$

satisfy $\mathbf{A} \cap \mathbf{A}^{\perp} \subseteq 0$, $\mathbf{A} \subseteq \mathbf{B} \Leftrightarrow \mathbf{B}^{\perp} \subseteq \mathbf{A}^{\perp}$, $\mathbf{A} \subseteq \mathbf{A}^{\perp\perp}$, and $\mathbf{A}^{\perp}$ is a closed subspace of $\mathbf{X}$.

If $\mathbf{M}$ is a closed convex subset of a Hilbert space $\mathbf{H}$, then any point in $\mathbf{H}$ has a unique point in $\mathbf{M}$ which is closest to it by using the least squares approximation.

An orthonormal basis of a Hilbert space $\mathbf{H}$ is a set of orthonormal vectors $\mathbf{E}$ whose span is dense: $\forall e_i, e_j \in \mathbf{E}, <e_i, e_j> = \delta_{ij}$.

Parseval's identity (Fourier Series): If $x = \sum_{i=1}^n \alpha_i e_i$, $y = \sum \beta_i e_i$, $\{e_i\}$ is orthonormal, then $x, y \in \mathbf{H}$, $<x, y> = \sum <x, e_i><e_i, y>$, $\sum |<x, e_i>|^2 = \|x\|^2$.

Bessel's inequality: $x = \sum \alpha_i e_i$, $\sum |<x, e_i>|^2 \leq \|x\|^2$.

An instance in Hilbert space is Fourier transform, which refers to both the frequency domain representation and the mathematical operation that associates the frequency domain representation to a function of time [38]. The Fourier transform is an extension of Fourier series. If we increase the length of the interval in the Fourier series, then the Fourier coefficients begin to resemble the Fourier transform. The Fourier transform of a function of time is a complex-valued function of frequency, whose magnitude (i.e., absolute value) represents the amount of that frequency present in the original function, and whose argument is the phase offset of the basic sinusoid in that frequency.

Fourier transform of a function $f(\cdot)$ is traditionally denoted by $\hat{f}(\cdot)$; the Fourier transform of an integrable function $f : \mathscr{R} \to \mathscr{C}$,

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} dx, \tag{3.179}$$

for any real number ξ.

If the independent variable x represents time, the transform variable ξ represents frequency, and $f(\cdot)$ is determined by $\hat{f}(\cdot)$ via the inverse transform

$$f(\xi) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i x \xi} d\xi, \tag{3.180}$$

for any real number x.

2D discrete Fourier transform (DFT) maps a scalar image I from spatial domain into a complex-valued Fourier transform $\mathbf{I}$ in frequency domain:

$$\mathbf{I}(u, v) = \frac{1}{W \cdot H} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} I(x, y) \exp\left[-i2\pi\left(\frac{x \cdot u}{W} + \frac{y \cdot v}{H}\right)\right], \tag{3.181}$$

where $u = 0, 1, \ldots, W-1$ and $v = 0, 1, \ldots, H-1$, $i = \sqrt{-1}$ as the imaginary unit of complex numbers, and W and H are the width and height of the image, respectively.
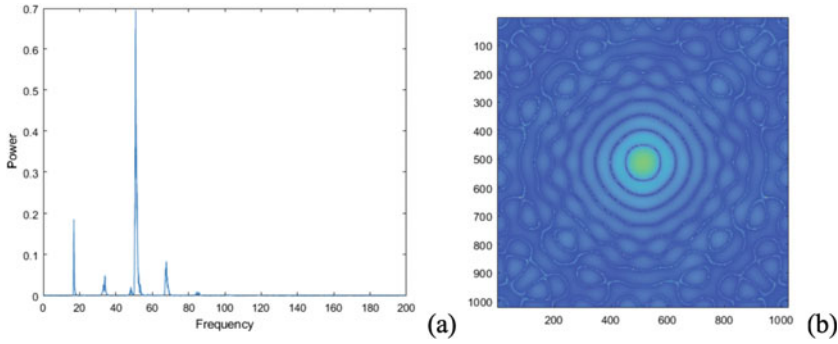
Fig. 3.7 An example of Fourier transform by using MATLAB: **a** 1D and **b** 2D

The inverse 2D DFT maps a Fourier transform **I** in frequency domain back into the spatial domain,

$$I(x, y) = \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} \mathbf{I}(u, v) \exp \left[i2\pi \left(\frac{x \cdot u}{W} + \frac{y \cdot v}{H} \right) \right]. \tag{3.182}$$

The discrete Fourier transform (DFT) for an image I satisfies Parseval's theorem

$$\frac{1}{|\Omega|} \sum_{\Omega} |I(x, y)|^2 = \sum_{\Omega} |\mathbf{I}(u, v)|^2, \tag{3.183}$$

where $\Omega = [1, W] \times [1, H]$.

An example of 1D Fourier transform is available at https://au.mathworks.com/help/matlab/math/fourier-transforms.html?s_tid=srchtitle.

An example of 2D Fourier transform is at https://au.mathworks.com/help/matlab/math/two-dimensional-fft.html.

The screenshots are shown in Fig. 3.7. Figure 3.7a displays 1D Fourier transform, and Fig. 3.7b indicates 2D Fourier transform.

Exercises

Question 3.1. Please explain the relationships between deep learning concepts: CNN, R-CNN, Fast R-CNN, Faster R-CNN, YOLO, and SSD.

Question 3.2. Can YOLOs cope with small object detection and classification?

Question 3.3. Please explain the relationships between deep learning concepts: RNN, LSTM, GRU, and FGU.

Question 3.4. How to merge or fuse different deep learning networks such as U-Net, SSD, and YOLOs?

Question 3.5. What are the differences between CNNs and CapsNets? What can CapsNets bring to us?

Question 3.6. What is the loss function of CapsNets? What is the squashing function of CapsNets?

Question 3.7. What are the differences between DenseNets and ResNets?

Question 3.8. In deep learning, how to select a proper algorithm for visual object detection? What balance should we take into account?

Question 3.9. What is the difference between FSM, HMM, and RNN?

Question 3.10. How to select a loss function in deep learning?

Question 3.11. Regarding time series analysis, what are the advantages of deep learning methods?

Question 3.12. How to understand cost functions of artificial neural networks (ANN)?

Question 3.13. How to understand various Transformers? what are the differences between RNN and Transformers?

Question 3.14. What are OpenAI GPT models? Why are GPT-X models so amazing?

Question 3.15. What are the relationships between OpenAI GPT models and reinforcement learning?

Question 3.16. How to learn policies in OpenAI GPT models?

Question 3.17. What is the relationship between norm and regularization in deep learning from the viewpoint of functional analysis?

Question 3.18. How to understand the relationship between Fourier transform and Hilbert space? What machine learning method is closely related to Hilbert space?

References

1. Aizenberg NN, Aizenberg IN, Krivosheev GA (1996) CNN based on universal binary neurons: learning algorithm with error-correction and application to impulsive-noise filtering on grayscale images. In: IEEE international workshop on cellular neural networks and their applications, pp 309–314

2. Badrinarayanan V, Handa A, Cipolla R (2017) SegNet: a deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. IEEE Trans Pattern Anal Mach Intell 39(12):2481–2495

3. Basu AP, Ebrahimi N (1991) Bayesian approach to life testing and reliability estimation using asymmetric loss function. J Stat Plan Inference 29(1–2):21–31

4. Bania P, Baranowski J (2016) Field Kalman filter and its approximation. In: IEEE conference on decision and control (CDC), pp 2875–2880

5. Brown T et al (2020) Language models are few-shot learners. NeurIPS

6. Cao G, Xie X, Yang W, Liao Q, Shi G, Wu J (2018) Feature-fused SSD: fast detection for small objects. In: International conference on graphic and image processing (ICGIP), vol 10615

7. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S (2020) End-to-end object detection with transformers. In: European conference on computer vision, pp 213–229

8. Chatfield C (2004) The analysis of time series: an introduction. Chapman & Hall/CRC

9. Chatzis SP, Kosmopoulos DI (2011) A variational Bayesian methodology for hidden Markov models utilizing Student's-t mixtures. Pattern Recogn 44(2):295–306

10. Chen J, Kang X, Liu Y, Wang Z (2015) Median filtering forensics based on convolutional neural networks. IEEE Signal Process Lett 22(11):1849–1853

11. Cover T, Thomas J (1991) Elements of information theory. Wiley

12. Dai J, Li Y, He K, Sun J (2016) R-FCN: object detection via region-based fully convolutional networks. In: Advances in neural information processing systems, pp 379–387

13. Devlin J, Chang M, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. NAACL

14. Dosovitskiy A et al (2021) An image is worth 16×16 words: transformers for image recognition at scale. ICLR

15. Dunne RA, Campbell NA (1997) On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In: Australian conference on neural networks, vol 181. Melbourne, p 185

16. Ertel W (2017) Introduction to artificial intelligence. Springer, Berlin

17. Fu R, Zhang Z, Li L (2016) Using LSTM and GRU neural network methods for traffic flow prediction. In: Youth academic annual conference of Chinese association of automation (YAC)

18. Fu Y (2020) Fruit freshness grading using deep learning. Master's thesis, Auckland University of Technology, Auckland, New Zealand

19. Fu Y, Nguyen M, Yan W (2021) Grading methods for fruit freshness based on deep learning. Springer Nat Comput Sci

20. Gal Y, Ghahramani Z (2016) A theoretically grounded application of dropout in recurrent neural networks. In: Advances in neural information processing systems, pp 1019–1027

21. Gilks WR, Wild P (1992) Adaptive rejection sampling for Gibbs sampling. J R Stat Soc Ser C (Appl Stat) 41(2):337–348

22. Gers FA, Schmidhuber J, Cummins F (2000) Learning to forget: continual prediction with LSTM. Neural Comput 12(10):2451–2471

23. Gers FA, Schmidhuber J (2000) Recurrent nets that time and count. In: IEEE-INNS-ENNS international joint conference on neural networks, vol 3, pp 189–194

24. Gers FA, Schmidhuber E (2001) LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE Trans Neural Netw 12(6):1333–1340

25. Gers FA, Schraudolph NN, Schmidhuber J (2002) Learning precise timing with LSTM recurrent networks. J Mach Learn Res 3:115–143

26. Girshick R, Donahue J, Darrell T, Malik J (2016) Region-based convolutional networks for accurate object detection and segmentation. IEEE Trans Pattern Anal Mach Intell 38(1):142–158

27. Girshick R (2015) Fast R-CNN. In: IEEE international conference on computer vision, pp 1440–1448

28. Giusti A, Ciresan DC, Masci J, Gambardella LM, Schmidhuber J (2013) Fast image scanning with deep max-pooling convolutional neural networks. In: IEEE international conference on image processing, pp 4034–4038

29. Gkioxari G, Girshick R, Malik J (2015) Contextual action recognition with R-CNN. In: IEEE ICCV, pp 1080–1088

30. Gu Q, Yang J, Yan WQ, Li Y, Klette R (2017) Local Fast R-CNN flow for object-centric event recognition in complex traffic scenes. In: Pacific-rim symposium on image and video technology, pp 439–452

31. Hager GD, Dewan M, Stewart CV (2004) Multiple kernel tracking with SSD. IEEE CVPR

32. Hamilton J (1994) Time series analysis. Princeton University Press, Princeton

33. Hassanpour H, Farahabadi PM (2009) Using hidden Markov models for paper currency recognition. Expert Syst Appl 36(6):10105–10111

34. He K, Zhang X, Ren S, Sun J (2014) Spatial pyramid pooling in deep convolutional networks for visual recognition. In: European conference on computer vision, pp 346–361

35. He K, Gkioxari G, Dollar P, Girshick R (2017) Mask R-CNN. In: ICCV, pp 2980–2988

36. Heikkila M, Pietikainen M (2006) A texture-based method for modeling the background and detecting moving objects. IEEE Trans Pattern Anal Mach Intell 28(4):657–662

37. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

38. Hu X (2017) Frequency based texture feature descriptors. PhD thesis, Auckland University of Technology, New Zealand

39. Huang G, Liu Z, Weinberger KQ, van der Maaten L (2017) Densely connected convolutional networks. In: IEEE CVPR, p 3

40. Hubel DH, Wiesel TN (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J Physiol 160(1):106–154

41. Jeong J, Park H, Kwak N (2017) Enhancement of SSD by concatenating feature maps for object detection. BMVC

42. Joshi M, Choi E, Weld D, Zettlemoyer L (2017) TriviaQA: a large scale distantly supervised challenge dataset for reading comprehension. In: Annual meeting of the association for computational linguistics

43. Ji H, Liu Z, Yan W, Klette R (2019) Early diagnosis of Alzheimer's disease using deep learning. In: ICCCV'19, pp 87–91

44. Ji H, Liu Z, Yan W, Klette R (2019) Early diagnosis of Alzheimer's disease based on selective kernel network with spatial attention. In: ACPR'19, pp 503–515

45. Keys R (1981) Cubic convolution interpolation for digital image processing. IEEE Trans Acoust Speech Signal Process 29(6):1153–1160

46. Kim K et al (2021) Rethinking the self-attention in vision transformers. In: IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW), pp 3065–3069

47. Kivinen J, Warmuth MK (1998) Relative loss bounds for multidimensional regression problems. In: Advances in neural information processing systems, pp 287–293

48. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105

49. Koller D, Friedman N (2009) Probabilistic graphical models. MIT Press, Massachusetts

50. Krizhevsky A, Sutskever I, Hinton G (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90

51. LeCun Y, Bengio Y (1995) Convolutional networks for images, speech, and time series. In: The handbook of brain theory and neural networks, vol 3361. issue 10

52. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444
53. Lee CY, Gallagher PW, Tu Z (2016) Generalizing pooling functions in convolutional neural networks: mixed, gated, and tree. In: Artificial intelligence and statistics, pp 464–472
54. Lemke C, Budka M, Gabrys B (2013) Metalearning: a survey of trends and technologies. Artif Intell Rev 44(1):117–130
55. Li C (2021) Special character recognition using deep learning. Master's thesis, Auckland University of Technology, New Zealand
56. Li C, Yan W (2021) Braille recognition using deep learning. ACM, ICCCV
57. Liang S, Yan W (2022) A hybrid CTC+Attention model based on end-to-end framework for multilingual speech recognition. Multimed Tools Appl
58. Liang S (2022) Multi-language datasets for speech recognition based on the end-to-end framework. Master's thesis, Auckland University of Technology, New Zealand
59. Liu X (2019) Vehicle-related scene understanding using deep learning. Master's thesis, Auckland University of Technology, New Zealand
60. Liu X, Yan W, Kasabov N (2020) Vehicle-related scene segmentation using CapsNets. In: IEEE IVCNZ, pp 1–6
61. Liu X, Yan W (2019) Traffic-light sign recognition using capsule network. Multimed Tools Appl 80(10):15161–15171
62. Liu X, Yan W (2022) Vehicle-related distance estimation using customized YOLOv7. In: IEEE IVCNZ
63. Liu Y (2023) Sign language recognition from digital videos using feature pyramid network with detection transformer. Master's thesis, Auckland University of Technology, New Zealand
64. Liu Y, Nand P, Hossain M, Nguyen M, Yan W (2023) Sign language recognition from digital videos using feature pyramid network with detection transformer. Multimed Tools Appl
65. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) SSD: single shot multibox detector. In: European conference on computer vision, pp 21–37
66. Liu W, Wen Y, Yu Z, Yang M (2016) Large-margin softmax loss for convolutional neural networks. In: ICML, pp 507–516
67. Liu Z, Yan W, Yang B (2018) Image denoising based on a CNN model. In: IEEE ICCAR, pp 389–393
68. Liu Z et al (2021) Swin transformer: hierarchical vision transformer using shifted windows. In: ICCV 2021
69. Lu J, Nguyen M, Yan W Yang B (2021) Sign language recognition from digital videos using deep learning methods. In: ISGV. Springer, Berlin, pp 108–118
70. Luo Z, Nguyen M, Yan Q (2021) Sailboat detection based on automated search attention mechanism and deep learning models. In: IEEE IVCNZ
71. Luo Z, Nguyen M, Yan Q (2022) Kayak and sailboat detection based on the improved YOLO with transformer. In: ACM ICCCV
72. Lutkepohl Helmut (1991) Introduction to multiple time series analysis. Springer, Berlin, Heidelberg
73. Martens J, Sutskever I (2011) Learning recurrent neural networks with Hessian-free optimization. In: International conference on machine learning
74. Merrienboer B, Bahdanau D, Dumoulin V, Serdyuk D, Warde-Farley Murtagh F (1991) Multilayer perceptrons for classification and regression. Neurocomputing 2(5–6):183–197
75. Mikolov T, Karafiat M, Burget L, Cernocky J, Khudanpur S (2010) Recurrent neural network based language model. In: Interspeech, p 3
76. Muscat J (2014) Functional analysis. Springer, Berlin
77. Norvig P, Russell S (2016) Artificial intelligence: a modern approach, 3rd edn. Prentice Hall
78. Nakano R, Hilton J (2021) WebGPT: browser-assisted question-answering with human feedback. In: OpenAI
79. Ouyang L, Wu J (2022) Training language models to follow instructions with human feedback. In: OpenAI

80. Parmar N, Vaswani A, Uszkoreit J, Kaiser L, Shazeer N, Ku A, Tran S (2018) Image trans-
    former. In: ICML, pp 4052–4061
81. Qin Z, Yan W (2021) Traffic-sign recognition using deep learning. In: ISGV. Springer, Berlin,
    pp 13–25
82. Rabiner L, Juang B (1986) An introduction to hidden Markov models. IEEE ASSP (Magazine)
    3(1):4–16
83. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech
    recognition. Proc IEEE 77(2):257–286
84. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding
    by generative pre-training. In: OpenAI
85. Radford A et al (2019) Language models are unsupervised multitask learners. In: OpenAI
86. Rastegari M, Ordonez V, Redmon J, Farhadi A (2016) XNOR-Net: imageNet classification
    using binary convolutional neural networks. In: European conference on computer vision, pp
    525–542
87. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In: IEEE CVPR, pp 6517–
    6525
88. Rekeczky C, Tahy A, Vegh Z, Roska T (1999) CNN-based spatio-temporal nonlinear filtering
    and endocardial boundary detection in echocardiography. Int J Circuit Theory Appl 27(1):171–
    207
89. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time
    object detection. In: IEEE CVPR, pp 779–788
90. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with
    region proposal networks. In: Advances in neural information processing systems, pp 91–99
91. Ren Y, Zhu C, Xiao S (2018) Object detection based on Fast/Faster R-CNN employing fully
    convolutional architectures. Math Probl Eng
92. Ronneberger O, Fischer P, Brox T (2015) U-Net: convolutional networks for biomedical
    image segmentation. In: International conference on medical image computing and computer-
    assisted intervention. Springer, Berlin, pp 234–241
93. Rriedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of
    boosting. Ann Stat 38(2):337–374
94. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Berg AC (2015) ImageNet large
    scale visual recognition challenge. Int J Comput Vision 115(3):211–252
95. Sabour S, Frosst N, Geoffrey E (2017) Hinton dynamic routing between capsules. In: The
    conference on neural information processing systems (NIPS), USA
96. Sahiner B, Chan HP, Petrick N, Wei D, Helvie MA, Adler DD, Goodsitt MM (1996) Classi-
    fication of mass and normal breast tissue: a convolution neural network classifier with spatial
    domain and texture images. IEEE Trans Med Imaging 15(5):598–610
97. Shen D, Chen X, Nguyen M, Yan WQ (2018) Flame detection using deep learning. In:
    International conference on control, automation and robotics (ICCAR), pp 416–420
98. Shen Y, Yan W (2018) Blindspot monitoring using deep learning. In: IEEE IVCNZ'18
99. Schulman J et al (2017) Proximal policy optimization algorithms. In: OpenAI
100. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image
    recognition. In: International conference on learning representations
101. Sun P (2019) Facial expression classification using R-CNN based methods. Master's thesis,
    Auckland University of Technology, Auckland, New Zealand
102. Sun S (2020) Empirical analysis for earlier diagnosis of Alzheimer's disease using deep
    learning. Master's thesis, Auckland University of Technology, Auckland, New Zealand
103. Szczecinski L, Djebbi A (2020) Understanding draws in Elo rating algorithm. J Quant Anal
    Sport 16(3):211–220
104. Takeda F, Omatu S (1995) A neuro-paper currency recognition method using optimized masks
    by genetic algorithm. In: IEEE international conference on systems, man and cybernetics, vol
    5, pp 4367–4371

105. Taud H, Mas JF (2018) Multilayer perceptron (MLP). In: Geomatic approaches for modelling land change scenarios. Springer, Berlin, pp 451–455

106. Toselli AH, Vidal E, Romero V, Frinken V (2016) HMM word graph based keyword spotting in handwritten document images. Inf Sci 497–518

107. Vapnik VN (1995) The nature of statistical learning theory. Springer, Berlin

108. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, Kaiser L, Polosukhin I (2017) Attention is all you need. In: NIPS 2017, pp. 5998–6008

109. Wang H (2018) Real-time face detection and recognition based on deep learning. Master's thesis, Auckland University of Technology, Auckland, New Zealand

110. Wang L, Yan W (2021) Tree leaves detection based on deep learning. In: ISGV. Springer, Berlin, pp 26–38

111. Wang MS, Song L, Yang XK, Luo CF (2016). A parallel-fusion RNN-LSTM architecture for image caption generation. In: International conference on image processing, pp 4448–4452

112. Wang X, Yan W (2021) Human gait recognition based on self-adaptive hidden Markov model. IEEE/ACM Trans Biol Bioinform

113. Wang X, Yan W (2020) Human gait recognition based on frame-by-frame gait energy images and convolutional long short term memory. Int J Neural Syst 30(1):1950027:1–1950027:12

114. Wang Y (2021) Colorizing grayscale CT images of human lung using deep learning. Master's thesis, Auckland University of Technology, Auckland, New Zealand

115. Woo S, Park J, Lee J, Kweon I (2018) CBAM: convolutional block attention module. In: European conference on computer vision

116. Xia Y, Nguyen M, Yan W (2022) A real-time Kiwifruit detection based on improved YOLOv7. In: IVCNZ

117. Xiao B, Nguyen M, Yan W (2021) Apple ripeness identification using deep learning. In: ISGV. Springer, Berlin, pp 53–67

118. Xin C (2018) Detection and recognition for multiple flames using deep learning. Master's thesis, Auckland University of Technology, New Zealand

119. Xingjian SHI, Chen Z, Wang H, Yeung DY, Wong WK, Woo WC (2015) Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Advances in neural information processing systems, pp 802–810

120. Yan WQ (2017) Introduction to intelligent surveillance: surveillance data capture, transmission, and analytics. Springer, Berlin

121. Yao W, Zeng Z, Lian C, Tang H (2018) Pixel-wise regression using U-Net and its application on pansharpening. Neurocomputing 312:364–371

122. Yu Z (2021) Deep learning methods for human action recognition. Masters thesis, Auckland University of Technology, Auckland, New Zealand

123. Yu Z, Yan W (2021) Human action recognition using deep learning methods. In: IEEE IVCNZ

124. Zanaty EA (2012) Support vector machines (SVMs) versus multilayer perception (MLP) in data classification. Egypt Inform J 13(3):177–183

125. Zarchan P, Musoff H (2000) Fundamentals of Kalman filtering: a practical approach. American Institute of Aeronautics and Astronautics, Incorporated

126. Ziegler D et al (2020) Fine-tuning language models from human preferences. In: OpenAI

127. OpenAI (2023) GPT-4 technical report

128. Zhang K, Zhang D, Jing C, Li J, Yang L (2017) Scalable softmax loss for face verification. In: International conference on systems and informatics, pp 491–496

129. Zhang C (2022) A novel transformer pre-training objective and a novel fine-tuning method for abstractive summarization. Master's thesis, Auckland University of Technology, Auckland, New Zealand

130. Zhang L (2020) Virus identification from digital images using deep learning. Master's thesis, Auckland University of Technology, Auckland, New Zealand

131. Zhang L, Yan W (2020) Deep learning methods for virus identification from digital images. In: IEEE IVCNZ

132. Zhang Q (2018) Currency recognition using deep learning. Master's thesis, Auckland University of Technology, Auckland, New Zealand
133. Zhang Q, Yan W, Kankanhalli M (2018) Overview of currency recognition using deep learning. J Bank Financ Technol 3(1):59–69
134. Zhang Q, Yan W (2018) Currency recognition using deep learning. In: IEEE AVSS
135. Zhang Y et al (2021) VidTr: video transformer without convolutions. In: IEEE ICCV
136. Zheng K, Yan WQ, Nand P (2018) Video dynamics detection using deep neural networks. IEEE Trans Emerg Top Comput Intell 2(3):224–234
137. Zhou Y, Tao C (2020) Multitask BERT for problem difficulty prediction. In: International conference on communications, information system and computer engineering (CISCE), pp 213–216

Generative Adversarial Networks and Siamese Nets

4

In this chapter, we will emphasize computational iterations in GANs (i.e., generative adversarial networks) [46] and Siamese nets [3,6,15]. In deep learning, these models are named as contrastive networks [3]. Additionally, we will deeply learn deep learning and interpret how information theory has been applied to the implementations of deep learning models.

4.1 Generative Adversarial Networks

Contrastive learning [3,15] provides a way to discover a better model. The idea is to train a feedforward neural network (FFNN) that produces very similar output vectors. After measurement and iterations, we expect the input and output vectors will be much similar and approach to each other.

Generative adversarial network (GAN) produces new data with the same statistics as the given training set [14]. The generative network generates candidates, while the discriminative network evaluates them. The generator is typically a deconvolutional neural network, and the discriminator is a convolutional neural network [34,38]. GAN is applied to digital forensics and find the real one from fake ones.

Mathematically, given data samples $\{x_1, x_2, \cdots, x_m\} \sim P_{data}(x)$, $P_{data}(x) \approx P_{G(x,\Theta)}$, the maximum likelihood estimation of x_i in $P_{G(x,\Theta)}$ is

$$L = \prod_{i=1}^{m} P_G(x_i, \Theta). \tag{4.1}$$

For the parametric optimization,

$$\Theta^* = \arg \max_{\Theta} \prod_{i=1}^{m} P_G(x_i, \Theta). \tag{4.2}$$

Hence,

$$\Theta^* = \arg\max_{\Theta} \sum_{i=1}^{m} \log P_G(x_i, \Theta). \tag{4.3}$$

Furthermore,

$$\Theta^* = \arg\max_{\Theta} KL(P_{data}(x)||P_G(x, \Theta)). \tag{4.4}$$

Mathematically, we define

- **Generator** G: Generate x from z.
- **Discriminator** D: Evaluate the difference between $P_{data}(x)$ and $P_G(x, \Theta)$ through function

$$V(G, D) \overset{\Delta}{=} \mathbf{E}_{x \sim P_{data}}(\log D(x)) + \mathbf{E}_{x \sim P_G}(\log(1 - D(x))), \tag{4.5}$$

and

$$G^* = \arg\min_{G}\max_{D} V(G, D). \tag{4.6}$$

Given G, if

$$D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}, \tag{4.7}$$

because of

$$V(G, D^*) = \max V(G, D) = -2\log 2 + 2JS(P_{data}(x)||P_G(x)), \tag{4.8}$$

Jensen–Shannon (JS) divergence is

$$JS(P||Q) = \frac{1}{2}KL(P||M) + \frac{1}{2}KL(Q||M), \tag{4.9}$$

where $M = \frac{P+Q}{2}$ and $JS(P||Q) = JS(Q||P)$, however $KL(P||Q) \neq KL(Q||P)$.

Thus,

$$G^* = \arg\min_{G}\max_{D^*} V(G, D^*), \tag{4.10}$$

and

$$L(G) = \max_{D^*} V(G, D^*), \tag{4.11}$$

therefore,

$$G^* = \arg\min_{G} L(G). \tag{4.12}$$

Hence,

$$\Theta_G := \Theta_G - \beta \cdot \frac{\partial L(G)}{\partial \Theta_G}, \tag{4.13}$$

where $\beta \geq 0$ is the learning rate. We solve this problem by using the method as follows:

- Given G_0,

$$D_1 = \arg\max_{D} V(G_0, D). \tag{4.14}$$

- Given D_1,

$$G_1 = \arg\max_G V(G, D_1). \tag{4.15}$$

- $\cdots\cdots$
- $G_i \Rightarrow D_{i+1}; D_{i+1} \Rightarrow G_{i+1}.$
- $\cdots\cdots$

$$G^* = \arg\min_G \max_D V(G, D), \tag{4.16}$$

where

$$V = \mathbf{E}_{x \sim P_{data}}[\log D(x)] + \mathbf{E}_{x \sim P_G}[\log(1 - D(x))]. \tag{4.17}$$

Discretely,

$$V = \frac{1}{m} \cdot \sum_{i=1}^{m} \log D(x_i) + \frac{1}{m} \cdot \sum_{i=1}^{m} \log[1 - D(\hat{x}_i)], \tag{4.18}$$

where $x_i \sim P_{data}$ and $\hat{x}_i \sim P_G$. We thus have

$$\Theta_D := \Theta_D - \beta \cdot \frac{\partial V}{\partial \Theta_D}. \tag{4.19}$$

If $z_i \sim N(0, 1)$, $\hat{x}_i = G(z_i)$, then,

$$V = \frac{1}{m} \cdot \sum_{i=1}^{m} \log[1 - D(G(z_i))]. \tag{4.20}$$

Thus, we have

$$\Theta_G := \Theta_G - \beta \cdot \frac{\partial V}{\partial \Theta_G}. \tag{4.21}$$

Therefore, we make use of Eqs. (4.20) and (4.21) to implement GAN. If we set image processing as an example, GAN can make a picture clear using existing details like super-resolution; GAN also can remove artifacts of digital images, etc. MATLAB has an example on how to train GAN models.

SimGAN [38] refines the output of the simulator of a neural network. We need to minimize the image difference between the synthetic one and the refined images, and finally update the discriminator alternately.

SimGAN uses unlabeled real data to refine the synthetic images, trains a refined network to add random numbers to synthetic images, further stabilizes GAN training, and prevents the refiner network from producing artifacts as well as generates the results without human annotation by training deep neural networks on the refined output images. The overall loss function is

$$L_R(\theta) = \sum_i X_i l_{real}(\theta; \mathbf{x}_i, L) + \lambda l_{reg}(\theta; \mathbf{x}_i), \tag{4.22}$$

where

$$l_{real}(\theta; \mathbf{x}_i, L) = -\log(1 - D_\phi(R_\theta(\mathbf{x}_i))), \tag{4.23}$$

and

$$l_{reg}(\theta; \mathbf{x}_i) = \|\psi(\tilde{\mathbf{x}}) - \mathbf{x}\|, \tag{4.24}$$

where $y_i \in \mathbf{y}$ is an unlabeled real image, and $x_i \in \mathbf{x}$ is a synthetic training image.

The discriminator updates its parameters by minimizing the loss function

$$L_D(\phi) = -\sum_i \log(D_\phi(x_i)) - \sum_j \log(1 - D_\phi(x_i)), \qquad (4.25)$$

where x_i is a synthetic image.

GANs have been applied to our project regarding human face image inpainting [5,11–13,20,21]. In general, the method is composed of generators and discriminators. The loss function combines the losses from Mean Square Error (MSE) and Generative Adversarial Networks (GANs). In the completion net, we added convolution operations with a 3×3 kernel. The global discriminator is designed as a 5-layer convolutional network; the local discriminator is designed as a 4-layer convolutional network.

4.2 Siamese Neural Networks

Siamese networks or twin networks [8] are typically employed for the tasks that involve finding the relationship between two comparable and similar things, such as signature verification [4], handwritten check, face recognition [41], object tracking [1,9,10,24,25,28], and similar document matching [2,32]. Similarity measures [37] are utilized based on a pair of twin networks. The objective of the Siamese network is to output a feature vector for each image so that the feature vectors are similar for similar images and different for dissimilar images [32]. In this way, the network can discriminate the two input images.

Siamese networks are particularly useful in cases where there are a large number of classes with a small number of observations. Siamese networks are applied to dimensionality reduction.

A Siamese network is a type of deep learning network which makes use of two or more identical subnetworks that have the same architecture, and share the same parameters and weights. In that vein, two different n-dimensional input vectors $\mathbf{x}_i \in \mathscr{R}^n$ and $\mathbf{x}_j \in \mathscr{R}^n$ will be computed for comparing output vectors; one of the output vectors is treated as the baseline, and the other will be compared by computing the distance. The loss function is defined with squared Euclidean distance. The goal of Siamese network is to minimize a distance metric for similar objects and maximize for distinct ones:

$$\mathscr{L}(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \min(\|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2) & \mathbf{x}_i = \mathbf{x}_j \\ \max(\|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2) & \mathbf{x}_i \neq \mathbf{x}_j. \end{cases} \qquad (4.26)$$

For a half-twin network,

$$\mathscr{L}(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \min(\|f(\mathbf{x}_i) - g(\mathbf{x}_j)\|_2) & \mathbf{x}_i = \mathbf{x}_j \\ \max(\|f(\mathbf{x}_i) - g(\mathbf{x}_j)\|_2) & \mathbf{x}_i \neq \mathbf{x}_j \end{cases}, \qquad (4.27)$$

where i and j are indexes of two input vectors from the same dataset, and $f(\cdot)$ and $g(\cdot)$ are the scoring functions implemented by using the twin network and half-twin

network, respectively. More generally, the loss function is often approximated as a Mahalanobis distance for a linear space as

$$\mathscr{L}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \cdot \mathbf{M} \cdot (\mathbf{x}_i - \mathbf{x}_j), \tag{4.28}$$

where $\mathbf{M}$ is a matrix from the Siamese network.

Contrastive loss has been employed in twin networks. The contrastive loss for a pair is given by

$$\mathscr{L} = \frac{1}{2} y d^2 + \frac{1}{2}(1 - y) \max(m - d, 0), \tag{4.29}$$

where y is the value of pair label, and $y = 1$ for similar images; $y = 0$ for dissimilar images, and d is the Euclidean distance between two feature vectors $\mathbf{v}_1$ and $\mathbf{v}_2$:

$$d = \|\mathbf{v}_1 - \mathbf{v}_2\|_2, \mathbf{v}_1, \mathbf{v}_2 \in \mathscr{R}^n. \tag{4.30}$$

The margin parameter m is used for constraint: If two images in a pair are dissimilar, then their distance should be at least as the same as m, or a loss will be incurred.

The dimension-reduced features allow the network to discriminate images that are similar and dissimilar. MATLAB provides a Siamese network for a demonstration to compare images and an example for dimensionality reduction at https://au.mathworks.com/help/deeplearning/ug/train-a-siamese-network-to-compare-images.html and another demonstration for data dimensionality reduction at https://au.mathworks.com/help/deeplearning/ug/train-a-siamese-network-for-dimensionality-reduction.html.

The Siamese network reduces the dimension of input images and outputs the dimension-reduced images with the same label. The network is able to discriminate images that are similar and dissimilar. Because of two inputs and similarity measurement, Siamese networks have been applied to visual object tracking in computer vision [1,2]. By measuring the similarity between exemplar and each part of the search image, a map of similarity score can be generated by using the twin network.

Siamese network has been applied to anomaly detection and object tracking [2]. We detect and track anomalies on the sidewalk using deep learning. The proposed network consists of two parts: The first part is to detect and classify objects, then we get the abnormal targets. The second one is to find data association of objects. Tracking objects is regarded as learning similarity problems. If the Siamese network has been chosen as the tracking network [1,40], the algorithm is only working for single-object tracking, however it can be combined with other deep learning algorithms such as fully connected neural network (FCNN), region proposal network (RPN), and LSTM. SiamRPN + LSTM algorithm has the highest MOTA (i.e., multiple object tracking accuracy).

4.3　Autoencoder

Basic autoencoder [31,39,44] is a feedforward and non-recurrent neural network which is unsupervised learning. That means, our computers can learn by themselves.

Given a group of training data samples, how to encode the data and remove noises among them, these are typical applications of an autoencoder. Our objective of deep autoencoder is to reduce the dimensionality of data [19] and minimize the differences between the encoded data and the decoded data. Thus, autoencoder is a generative network; one of its advantages is to treat the output as input and reduce the dimensionality of raw data [19].

For $\mathbf{x} \in \mathscr{R}^d$, $\mathbf{z} \in \mathscr{R}^p$,

$$\mathbf{z} = \sigma(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}), \tag{4.31}$$

and

$$\mathbf{x}' = \sigma'(\mathbf{W}' \cdot \mathbf{z} + \mathbf{b}'). \tag{4.32}$$

To minimize the reconstruction errors, we have

$$L(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'(W' \cdot \sigma(W \cdot \mathbf{x} + \mathbf{b}) + \mathbf{b}')\|^2. \tag{4.33}$$

Thus, the global loss function [22] is

$$J_{AE}(\Theta) = \sum_{\mathbf{x}} L(\mathbf{x}, \mathbf{x}'), \tag{4.34}$$

where $\Theta = (\mathbf{W}, \mathbf{b}, \mathbf{W}', \mathbf{b}')^\top$. The decay process is

$$\Theta_{i+1} := \Theta_i - \alpha \cdot \frac{\partial L_{AE}(\Theta_i)}{\partial \Theta_i}, \tag{4.35}$$

where $\alpha \geq 0$ is the learning rate.

4.4　Regularizations

For $\mathbf{L}^2$ regularization [43], if $\lambda \in \mathscr{R}$ is the parameter of weight decay (wd), then

$$J_{AE+wd}(\Theta) = J_{AE}(\Theta) + \lambda \cdot \sum_{w_{i,j} \in \mathbf{W}} w_{i,j}^2. \tag{4.36}$$

For sparse regularization (sp) [36] using KL divergence [7], if $\beta \in \mathscr{R}$ is the parameter of sparse weight, then

$$J_{AE+sp}(\Theta) = J_{AE}(\Theta) + \beta \cdot \sum_{j=1}^{m} KL(\rho||\hat{\rho}_j), \tag{4.37}$$

where

$$KL(\rho||\hat{\rho}_j) \overset{\Delta}{=} \rho \cdot \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \cdot \log \frac{1 - \rho}{1 - \hat{\rho}_j}, \tag{4.38}$$

and

$$\hat{\rho}_j = \frac{1}{N} \sum_{i=1}^{N} h_j(x^{(i)}) \tag{4.39}$$

$\hat{\rho}_j = \rho_j, j = 1, 2, \cdots, m; \mathbf{x} = \{x^{(i)}\}_{i=1}^{N}$.

From the definition of Eq. (4.38), we know that $KL(\rho||\hat{\rho}_j) \neq KL(\hat{\rho}_j||\rho)$. Furthermore,

$$J_{AE+wd+sp}(\Theta) = J_{AE}(\Theta) + \lambda \cdot \sum_{w_{ij} \in \mathbf{W}} w_{ij}^2 + \beta \cdot \sum_{j=1}^{m} KL(\rho||\hat{\rho}_j). \tag{4.40}$$

Thus,

$$J_{AE+wd+sp}(\Theta) = J_{AE+wd}(\Theta) + \beta \cdot \sum_{j=1}^{m} KL(\rho||\hat{\rho}_j), \tag{4.41}$$

and

$$J_{AE+wd+sp}(\Theta) = J_{AE+sp}(\Theta) + \lambda \cdot \sum_{w_{ij} \in \mathbf{W}} w_{ij}^2. \tag{4.42}$$

In denoising by using autoencoder [26,47], data corruption means

$$\mathbf{x}' = \mathbf{x} + \varepsilon, \tag{4.43}$$

where $\varepsilon \sim \mathbf{N}(\mu, \delta) \rightarrow \mathbf{N}(0, \delta^2\mathbf{I})$, and $\mathbf{N}(0, \delta^2\mathbf{I})$ is the additive isotropic Gaussian noise.

In contractive autoencoder (*CAE*),

$$J_{CAE}(\Theta) = J_{AE}(\Theta) + \lambda \cdot \|J_f\|^2, \tag{4.44}$$

where $\|J_f\|_F$ is Frobenius norm,

$$J_f = (a_{ij})_{m \times n} \triangleq (\frac{\partial h_i}{x_j})_{m \times n}, \tag{4.45}$$

and

$$\|J_f\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} (\frac{\partial h_i}{x_j})^2, \tag{4.46}$$

where

$$h_i = \sigma(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}). \tag{4.47}$$

Hence,

$$\|J_f\|_F^2 = \sum_{i=1}^{m} h_i \cdot (1 - h_i) \cdot \sum_{j=1}^{n} w_{ij}^2. \tag{4.48}$$

For variational autoencoder (VAE) [30,45], we define Kullback–Leibler (KL) divergence [7] as

$$KL(Q||P) \triangleq \sum_{x \in \mathbf{x}} Q(x) \cdot \log \frac{Q(x)}{P(x)} \triangleq \int_{x \in \mathbf{x}} Q(x) \frac{Q(x)}{P(x)} dx. \tag{4.49}$$

Bayes' Theorem [23] is

$$P(x|z) = \frac{P(z|x) \cdot P(x)}{P(z)} = \frac{P(z|x) \cdot P(x)}{\sum_{x \in \mathbf{x}} P(z|x) \cdot P(x)}, \quad (4.50)$$

where the probabilities $P(x|z)$, $P(z|x)$, $P(x)$, and $P(z)$ refer to posterior, likelihood, prior, and evidence, respectively. Therefore, we have variational inference [23]. If $Q(z) = P(x|z)$, then

$$KL(Q(z)||P(z|x)) = KL(Q(z)||P(z)) - \sum_{z \in \mathbf{z}} Q(z) \log P(x|z) + \log P(x). \quad (4.51)$$

Now, we define variational autoencoder (VAE). If $x \sim \mathbf{N}(\mu, \delta)$, $z = x + \varepsilon = g(x, \varepsilon)$, then $z \sim \mathbf{N}(\mu + \varepsilon, \delta)$, $\mathbf{N}(\mu, \delta)$ is the normal or Gaussian distribution, μ is mean, and δ is variance,

$$Q(z) = P(x|z) = P(\varepsilon). \quad (4.52)$$

Furthermore,

$$L_{VAE}(Q, P) = KL(Q(z)||P(z|x)), \quad (4.53)$$

$$= KL(Q(z)||P(z)) - \sum_{\mathbf{z} \sim N(\mu + \varepsilon, \delta)} Q(z) \log P(x|z) + \log P(x), \quad (4.54)$$

$$= KL(Q(z)||P(z)) - \sum P(\varepsilon) \log P(x|g(x, \varepsilon)) + \log P(x), \quad (4.55)$$

where $g(x, \varepsilon)$ is an encoder model, and $P(x|z)$ is a decoder model. Hence, the cost function is

$\min[KL(Q(z)||P(z|x))] \Leftrightarrow$

$$\underbrace{\min KL(Q(z)||P(z))}_{Encoder:KL\ Divergence} - \max \underbrace{\sum_{\mathbf{z} \sim N(\mu + \varepsilon, \delta)} Q(z) \log P(x|z)}_{Decoder:MaxLikelihood}, \quad (4.56)$$

where $\log P(x)$ $w.r.t.$ $x \in \mathbf{x}$ is independent on $z \in \mathbf{z}$ [23].

In a nutshell, a VAE consists of an encoder, a decoder, and a loss function. The term "variational" comes from the close relation between the regularization and the variational inference method in statistics. VAE outputs a Gaussian probability distribution with mean and standard deviation for every dimension. For a given set of possible encoders and decoders, VAEs look for the pair that keeps the maximum of information if the encoding has the minimum of reconstruction error while decoding. VAE is trained by using gradient descent to optimize the loss with respect to the parameters of the encoder and decoder.

Autoencoders are a type of self-supervised learning models that learn a representation through input data. An LSTM autoencoder [29] is an implementation of an autoencoder for sequence data using an encoder–decoder LSTM architecture. LSTM autoencoders learn a representation of sequence data. For a given dataset of sequences, an encoder–decoder LSTM is configured to read the input sequence, encode it, decode it, and recreate it. The performance of LSTM autoencoder is evaluated based on the model's ability to recreate the input sequence.

Autoencoders have been applied to remove image noises, such as haze removal; it eventually is employed to implement image inpainting [18,20] such as TV logo removal [48,49] because the output could be used as the input, iteratively.

In MATLAB, an autoencoder is a neural network which replicates its input as its output. If the number of neurons in the hidden layer is less than the size of the input, autoencoder learns a compressed representation of the input. This autoencoder makes use of regularizers to learn a sparse representation in the first layer. The regularizers are controlled by setting various parameters.

A MATLAB example for training variational autoencoder (VAE) with regard to generate images is available from https://au.mathworks.com/help/deeplearning/ug/train-a-variational-autoencoder-vae-to-generate-images.html. GANs, Siamese nets, and autoencoders as well as Transformers [27,35,42] all are employed for generating new images, but their mechanisms are different; thus the final results have a minor difference.

An example for training an autoencoder to generate text is available from https://au.mathworks.com/help/deeplearning/ug/generate-text-using-autoencoders.html. The encoder utilizes an LSTM operation to map the input text into latent vectors. The decoder uses an LSTM operation and the same embedding to reconstruct the text from the latent vectors.

4.5   Information Theory

Text processing is employed to natural language processing. Text information has entropy; the information capacity is measured by using entropy; even the text short message (SMS) only has 144 letters that could be measured by entropy:

$$H = -\sum_{i=1}^{m} p_i \ln p_i = \mathbf{E} \ln \frac{1}{p_i}, \qquad (4.57)$$

where H is entropy, and $p_i \in [0, 1]$ is the probability, it may be the histogram of 256 letters (ASCII code) or pixels with 256 grayscale intensities after the normalization, and m is the bin number, i.e., $m = 256$. ASCII refers to American Standard Code for Information Interchange, which is a character encoding standard for electronic communication.

Entropy normalization refers to the normalized entropy that falls in the interval [0, 1], namely

$$\eta = \frac{H}{H_{max}} = -\sum_{i=1}^{m} p_i \frac{\ln p_i}{\ln m}, \qquad (4.58)$$

where m is the number of p_i, and $H_{max} = \ln m$ is the maximum entropy.

Probability is usually between 0 and 1, $viz., p \in [0, 1]$. Entropy could be written in the way of mathematical expectation $\mathbf{E}(\cdot)$; correspondingly, we define joint entropy, conditional entropy, and relative entropy.

We denote conditional probability as $p(x|y)$. Given x, the entropy $h(x|y)$ is not the same as the entropy given y. Therefore, we define the joint entropy and mutual entropy.

Joint probability is $h(x, y)$. Information capacity is defined as $c = \max I(x, y)$. On the Internet, information theory and entropy have their broad applications.

Relative entropy is also called KL divergence between p and q, which reflects the information distance between p and q,

$$KL(p||q) = -\sum_{i=1}^{m} p_i \ln \frac{p_i}{q_i}. \tag{4.59}$$

Again, $KL(p||q) \neq KL(q||p)$.

Mutual information is defined based on joint probability. KL divergence has been used in deep learning for entropy-based loss functions and distance computing.

In graphical models, we make use of relative entropy, joint entropy, and mutual information. The mutual information has multiple definitions; they are equal to each other and could be written in the product form if the probability of each element is independent.

Joint entropy and mutual entropy are shown in Venn diagram (also called primary diagram, set diagram, or logic diagram), which shows all possible logical relations between a finite collection of different sets. Regarding these concepts of entropy, we have the chain rule for the case of infinity. Correspondingly, we take conditional entropy into account.

According to Bayes' theorem, we have the relationship between joint entropy, conditional entropy, relative entropy, and mutual information.

Jensen's inequality tells us what a convex or concave function is. If the points on a curve are always located at one side of the straight line, we say the curve is convex. Mathematically, a convex function is shown as in Eq. (4.60):

$$f(\alpha \cdot x_1 + (1 - \alpha)x_2) \leq \alpha \cdot f(x_1) + (1 - \alpha) \cdot f(x_2), \alpha \in [0, 1]. \tag{4.60}$$

If a function is convex, the second derivative could be applied to decide whether it is such a function. $f'(x) \geq 0$ and $f''(x) \geq 0, x \in [a, b], f(x) \in \mathbf{C}(a, b)$.

Suppose we have $p(x)$ and $q(x)$; for any two functions, we have the relative entropy

$$H(p||q) = -p \ln \frac{p}{q}. \tag{4.61}$$

Another concept is entropy rate

$$H(p) = -\frac{1}{n} \sum_{i=1}^{m} \ln p_i. \tag{4.62}$$

Generally, the entropy $H(X)$ of a discrete random variable X is defined by using

$$H(X) = -\sum_{x \in \mathbf{X}} p(x) \log p(x) = -\mathbf{E} \log p(X) = \mathbf{E} \log \frac{1}{p(X)}. \tag{4.63}$$

The joint entropy $H(X, Y)$ of a pair of discrete random variables (X, Y) with a joint distribution $p(x, y)$ is defined as

$$H(X, Y) = -\sum_{x \in \mathbf{X}, y \in \mathbf{Y}} p(x, y) \log p(x, y) = \mathbf{E} \log \frac{1}{p(X, Y)}. \tag{4.64}$$

If $(X, Y) \sim p(x, y)$, then conditional entropy [7] $H(Y|X)$ is

$$H(Y|X) = \sum_{x \in X} p(x) H(Y|X = x), \tag{4.65}$$

and

$$-\sum_{x \in \mathbf{X}} p(x) \sum_{y \in \mathbf{Y}} p(y|x) \log p(y|x) = \mathbf{E}_{p(x,y)} \log \frac{1}{p(Y|X)}. \tag{4.66}$$

Corollary,

$$H(X|Y) \neq H(Y|X). \tag{4.67}$$

Equivalently, we denote

$$\log p(X, Y) = \log p(X) + \log p(Y|X). \tag{4.68}$$

For the chain rule of entropy,

$$H(X, Y) = H(X) + H(Y|X), \tag{4.69}$$

and

$$H(X, Y|Z) = H(X|Z) + H(Y|X, Z). \tag{4.70}$$

Mutual information [7] $I(X; Y)$ is a measure of the dependence between two random variables, which is symmetric and always nonnegative,

$$I(X; Y) = H(X) - H(X|Y), \tag{4.71}$$

and

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X). \tag{4.72}$$

For a communication channel with input X and output Y, the capacity C is defined as

$$C = \max_{p(x)} I(X; Y). \tag{4.73}$$

The capacity is the maximum rate at which we send information over the channel and recover the information at the output with a low probability of error.

Relative entropy (KL Divergence) [7] is a measure of the "distance" between two probability mass functions p and q:

$$D(p||q) = \sum_{x \in \mathbf{X}} p(x) \log \frac{p(x)}{q(x)} = \mathbf{E}_{p(X)} \log \frac{p(X)}{q(X)}, \tag{4.74}$$

where $D(p||q) \neq D(q||p)$. Mutual information [7] $I(X; Y)$ is the relative entropy between the joint distribution and the product distribution $p(x)q(x)$:

$$I(X; Y) = \sum_{x \in \mathbf{X}} \sum_{y \in \mathbf{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}, \tag{4.75}$$

and

$$D(p(x, y)\|p(x)p(y)) = \mathbf{E}_{p(x,y)} \log \frac{p(X, Y)}{p(X)p(Y)}. \tag{4.76}$$

Meanwhile,

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X), \tag{4.77}$$

and

$$I(X; Y) = H(X) + H(Y) - H(X, Y). \tag{4.78}$$

According to Bayes' theorem [23],

$$p(x_1, x_2) = p(x_2)p(x_1|x_2) = p(x_1)p(x_2|x_1), \tag{4.79}$$

$$H(X_1, X_2) = H(X_1) + H(X_2|X_1), \tag{4.80}$$

and

$$H(X_1, X_2, X_3) = H(X_1) + H(X_2, X_3|X_1) = H(X_1) + H(X_2|X_1) + H(X_3|X_2, X_1). \tag{4.81}$$

$$\cdots\cdots$$

Therefore,

$$H(X_1, X_2, X_3, \cdots, X_n) = \sum_{i=1}^{N} H(X_i|X_{i-1}, \cdots X_2, X_1). \tag{4.82}$$

Chain rule for relative entropy is

$$D(p(x, y)\|q(x, y)) = D(p(x)\|q(x)) + D(p(y|x)\|q(y|x)), \tag{4.83}$$

where

$$D(p(y|x)\|q(y|x)) = \sum_{\mathbf{x}} p(x) \sum_{\mathbf{y}} p(y|x) \log \frac{p(y|x)}{q(y|x)} = \mathbf{E} \log \frac{p(y|x)}{q(y|x)}. \tag{4.84}$$

If $f(\cdot)$ is a convex function and X is a random variable, then

$$\mathbf{E}f(X) \geq f(\mathbf{E}(X)). \tag{4.85}$$

A function $f(x)$ is convex over an interval (a, b) if for every $x_1, x_2 \in (a, b)$ and $0 \leq \lambda \leq 1$,

$$f[\lambda x_1 + (1 - \lambda)x_2] \leq \lambda f(x_1) + (1 - \lambda)f(x_2). \tag{4.86}$$

If $f(\cdot)$ is a convex function and X is a random variable, then

$$\mathbf{E}f(X) \geq f(\mathbf{E}(x)). \tag{4.87}$$

If the function $f(\cdot)$ has a second derivative $f''(x) \geq 0$ everywhere, then the function is convex (strictly convex). Let $p(x), q(x), x \in \mathbf{X}$ be two probability mass functions, then $D(p\|q) \geq 0, D(p(x|y)\|q(x|y)) \geq 0$.

For any two random variables X and Y, $I(X; Y) \geq 0$ and $I(X; Y|Z) \geq 0$; furthermore, because $I(X; Y) = H(X) - H(X|Y) \geq 0$, $H(X) \geq H(X|Y)$, therefore,

$$H(X_1, X_2, \cdots, X_n) = \sum_{i=1}^{N} H(X_i|X_{i-1}, \cdots, X_1) \leq \sum_{i=1}^{N} H(X_i). \tag{4.88}$$

For $a_i, b_i \geq 0$, $i = 1, 2, \cdots, n$,

$$\sum_{i=1}^{N} a_i \log \frac{a_i}{b_i} \geq \sum_{i=1}^{N} a_i \log \frac{\sum_{i=1}^{N} a_i}{\sum_{i=1}^{N} b_i}, \qquad (4.89)$$

and

$$D(p||q) = \sum_{i=1}^{N} p(x) \log \frac{p(x)}{q(x)} \geq \sum p(x) \log \frac{\sum p(x)}{\sum q(x)}. \qquad (4.90)$$

Hence,

$$\lambda D(p_1||q_1) + (1 - \lambda)D(p_2||q_2) \geq D(\lambda p_1 + (1 - \lambda)p_2||\lambda q_1 + (1 - \lambda)q_2). \quad (4.91)$$

Therefore, $H(X)$ is a convex function and $I(X; Y)$ is a concave function [33]. The entropy rate of a stochastic process X_i is defined by

$$H(\mathbf{X}) = \lim_{n \to \infty} \frac{1}{n} H(X_1, X_2, \cdots, X_n). \qquad (4.92)$$

A related quantity for entropy rate [7] is

$$H'(\mathbf{X}) = \lim_{n \to \infty} \frac{1}{n} H(X_n | X_{n-1}, \cdots, X_1). \qquad (4.93)$$

For a stationary stochastic process,

$$H'(\mathbf{X}) = H(\mathbf{X}) \Rightarrow \qquad (4.94)$$

$$\lim_{n \to \infty} H(X_n | X_{n-1}, \cdots, X_1) = \lim_{n \to \infty} H(X_n | X_{n-1}) = H(X_2 | X_1). \qquad (4.95)$$

Let X_i, $i = 1, 2, \cdots$ be a stationary Markov chain with stationary distribution μ and transition matrix $\mathbf{P} = (P_{ij})$. Then, the entropy rate is

$$H(\mathbf{X}) = -\sum_{ij} \mu_i P_{ij} \log P_{ij}. \qquad (4.96)$$

The entropy rate of the two states Markov chain is

$$H(\mathbf{X}) = H(X_2 | X_1) = \frac{\alpha}{\alpha + \beta} \cdot H(\beta) + \frac{\beta}{\alpha + \beta} \cdot H(\alpha). \qquad (4.97)$$

Entropy is defined not only in the discrete way but also in a continuous way. Previously, it was based on the sum function; now it is based on integral operation. Previously, the entropy is H, and now it is h. If $f(\cdot)$ is a continuous function, the entropy function will be continuity. The continuous entropy is

$$h = -\int p(x) \ln p(x) dx = -\mathbf{E} \ln \frac{1}{p(x)}. \qquad (4.98)$$

The continuous conditional entropy is

$$h = -\int p(x|y) \ln p(x|y) dx = -\mathbf{E} \ln \frac{1}{p(x|y)}. \qquad (4.99)$$

The continuous joint entropy is

$$h = -\int p(x, y) \ln p(x, y) dx = -\mathbf{E} \ln \frac{1}{p(x, y)}. \qquad (4.100)$$

The continuous entropy rate is

$$h = -\frac{1}{L} \int p(x, y) \ln p(x, y) dx = -\frac{1}{L} \mathbf{E} \ln \frac{1}{p(x, y)}. \qquad (4.101)$$

In our project [16, 17], we proposed an Entropy-Based Convolutional Layer Estimation (EBCLE) heuristic which is robust, simple, and effective in resolving the problem of over-parameterization with regards to net depth of CNN model. A priori knowledge of the entropic data distribution of input datasets was applied to determine an upper bound for convolutional network depth, and identity transformations which offer significant contributions for enhancing model performance.

Exercises

Question 4.1. How to measure the performance of the generator and discriminator in GANs?

Question 4.2. What are the relationships between Siamese nets and GANs?

Question 4.3. How does the autoencoder generate digital images which look similar to the original image? What are the differences if you apply GANs, Transformers, and autoencoders to generate a new image?

Question 4.4. What are the relationships between autoencoders and GANs?

Question 4.5. What's the chain rule for relative entropy?

Question 4.6. What are the advantages and disadvantages of using relative entropy (i.e., KL Divergence) as a measure between two probability mass functions? How to solve this problem?

References

1. An N, Yan W (2021) Multitarget tracking using Siamese neural networks. ACM Trans Multimed Comput Commun Appl 17(2s):1–16
2. An N (2020) Anomalies detection and tracking using Siamese neural networks. Master's thesis, Auckland University of Technology, New Zealand
3. Bengio Y, Lecun Y, Hinton G (2021) Deep learning for AI. Commun ACM 64(7):58–65

4. Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R (1994) Signature verification using a Siamese time delay neural network. In: Advances in neural information processing systems, pp 737–744

5. Chen Y, Zhang H, Liu L, Chen X, Zhang Q, Yang K, Xia R, Xie J (2021) Research on image inpainting algorithm of improved GAN based on two-discrimination networks. Appl Intell 51(6):3460–3474

6. Chen X, He K (2021) Exploring simple Siamese representation learning. In: IEEE CVPR, pp 15750–15758

7. Cover T, Thomas J (1991) Elements of information theory. John Wiley & Sons Inc

8. Chicco D (2020) Siamese neural networks: an overview, artificial neural networks. Methods in molecular biology, 3rd edn. Springer Protocols, USA, Humana Press, pp 73–94

9. Cui S, Tian S, Yin X (2019) Combined correlation filters with Siamese region proposal network for visual tracking. In: International conference on neural information processing, pp 128–138

10. Dong X, Shen J (2018) Triplet loss in Siamese network for object tracking. In: European conference on computer vision (ECCV), pp 459–474

11. Gao X, Nguyen M, Yan W (2022) A face image inpainting method based on autoencoder and adversarial generative networks. In: PSIVT

12. Gao X A method for face image inpainting based on generative adversarial networks. Master's thesis, Auckland University of Technology, New Zealand

13. Gao X, Nguyen M, Yan W (2021) Face image inpainting based on generative adversarial network. In: IEEE IVCNZ

14. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. In: International conference on neural information processing systems (NIPS), pp 2672–2680

15. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press

16. Gowdra N, Sinha R, MacDonell S, Yan W (2021) Mitigating severe over-parameterization in deep convolutional neural networks through forced feature abstraction and compression with an entropy-based heuristic. Pattern Recognit 119

17. Gowdra N, Sinha R, MacDonell S, Yan W (2021) Entropy-based optimization strategies for convolutional neural networks. PhD thesis, Auckland University of Technology, New Zealand

18. Guillemot C, Le Meur O (2013) Image inpainting: overview and recent advances. IEEE Signal Process (Mag) 31(1):127–144

19. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Sci 313(5786):504–507

20. Jam J, Kendrick C, Walker K, Drouard V, Hsu J, Yap M (2020) A comprehensive review of past and present image inpainting methods. Comput Vis Image Underst :103–147

21. Jiang Y, Xu J, Yang B, Xu J, Zhu J (2020) Image inpainting based on generative adversarial networks. IEEE Access 8(22):884–892

22. Ko YH, Kim KJ, Jun CH (2005) A new loss function-based method for multiresponse optimization. J Qual Technol 37(1):50–59

23. Koller D, Friedman N (2009) Probabilistic graphical models. MIT Press, Massachusetts

24. Li B, Yan J, Wu W, Zhu Z, Hu X (2018) High performance visual tracking with Siamese region proposal network. In: IEEE conference on computer vision and pattern recognition, pp 8971–8980

25. Li B, Wu W, Wang Q, Zhang F, Xing J, Yan J (2019) SiamRPN++: Evolution of Siamese visual tracking with very deep networks. In: IEEE conference on computer vision and pattern recognition, pp 4282–4291

26. Li CP, Qin PY, Zhang JJ (2017) Research on image denoising based on deep convolutional neural network. Comput Eng 43(3)

27. Liu Z et al (2021) Swin transformer: Hierarchical vision transformer using shifted windows. ICCV 2021

28. Liu M, Lei Q, Yu L, Gao Y, Zhang X (2019) Visual object tracking via an improved lightweight Siamese network. In: Chinese conference on pattern recognition and computer vision (PRCV), pp 284–294

29. Marchi E, Vesperini F, Squartini S, Schuller B (2017) Deep recurrent neural network-based autoencoders for acoustic novelty detection. Comput Intell Neurosci. Hindawi

30. Marreiros AC, Daunizeau J, Kiebel SJ, Friston KJ (2008) Population dynamics: variance and the sigmoid activation function. Neuroimage 42(1):147–157

31. Masci J, Meier U, Cirean D, Schmidhuber J (2011) Stacked convolutional autoencoders for hierarchical feature extraction. In: International conference on artificial neural networks. Springer, pp 52–59

32. Melekhov I, Kannala J, Rahtu E (2016) Siamese network features for image matching. In: International conference on pattern recognition (ICPR), pp 378–383

33. Muscat J (2014) Functional analysis. Springer

34. Ng AY, Jordan MI (2002) On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In: Advances in neural information processing systems, pp 841–848

35. Parmar N, Vaswani A, Uszkoreit J, Kaiser L, Shazeer N, Ku A, Tran S (2018) Image transformer. In: ICML, pp 4052–4061

36. Poultney C, Chopra S, Cun YL (2007) Efficient learning of sparse representations with an energy-based model. In: Advances in neural information processing systems, pp 1137–1144

37. Shen C, Jin Z, Zhao Y, Fu Z, Jiang R, Chen Y, Hua XS (2017) Deep Siamese network with multi-level similarity perception for person re-identification. In: ACM international conference on multimedia, pp 1942–1950

38. Shrivastava A et al (2017) Learning from simulated and unsupervised images through adversarial training. In: IEEE CVPR

39. Xing C, Ma L, Yang X (2016) Stacked denoise autoencoder based feature extraction and classification for hyperspectral images. J Sens 2016

40. Valmadre J, Bertinetto L, Henriques J, Vedaldi A, Torr P (2017) End-to-end representation learning for correlation filter based tracking. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 2805–2813

41. Varior RR, Shuai B, Lu J, Xu D, Wang G (2016) A Siamese long short-term memory architecture for human re-identification. In: European conference on computer vision, pp 135–153

42. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, Kaiser L, Polosukhin I (2017) Attention is all you need. In: NIPS 2017, pp 5998–6008

43. Wan L, Zeiler M, Zhang S, Le Cun Y, Fergus R (2013) Regularization of neural networks using DropConnect. In: International conference on machine learning, pp 1058–1066

44. Wang J, Zhang C (2018) Software reliability prediction using a deep learning model based on the RNN encoder—decoder. Reliab Eng Syst Saf 170:73–82

45. Welling M, Kingma D (2019) An introduction to variational autoencoders. Found Trends Mach Learn 12(4):307–392

46. Wu E, Wu K, Cox D, Lotter W (2018) Conditional infilling GANs for data augmentation in mammogram classification. In: Image analysis for moving organ, breast, and thoracic images, pp 98–106

47. Xie J, Xu L, Chen E (2012) Image denoising and inpainting with deep neural networks. In: Advances in neural information processing systems 34, p 341

48. Yan W, Kankanhalli M (2002) Erasing video logos based on image inpainting. In: IEEE ICME, pp 521–524

49. Yan W, Kankanhalli M (2005) Automatic video logo detection and removal. Multimed Syst 10(5):379–391

Reinforcement Learning

5

In this chapter, we introduce fundamental concepts of reinforcement learning [21] such as Bellman equation, Q-learning, deep Q-learning, and double Q-learning. We detail why reinforcement learning is thought as a method of deep learning. Mathematically, we narrate mathematical control theory, dynamic programming, optimization and data fitting, understand how these subjects are applied to deep learning, especially deep reinforcement learning. The recent publications of the journal Nature related to reinforcement learning uplift the relevant research work.

5.1 Introduction

Learning from interaction like OpenAI ChatGPT is a foundational idea underlying nearly all theories of machine learning and artificial intelligence [34]. Reinforcement learning [36] as a third machine learning approach is to answer the question how to map states to actions. Actions affect not only the immediate reward, but also the next states and all subsequent rewards. Reinforcement learning [13] is related to dynamical systems, specifically, optimal control and Markov decision process (MDP). Reinforcement learning explicitly resolves the problem of a goal-directed agent interacting with an uncertain environment. Reinforcement learning seeks the trade-off between exploration and exploitation (EE).

Reinforcement learning systems include a policy, a reward, a value function, and a model of the environment. A policy defines the rule of agent's behaviors at a given time. A reward signal defines the goal in a reinforcement learning problem. A value function specifies what is good in the long run. A value of state is the total amount of reward that an agent is expected to accumulate over the time span, starting from that state. Rewards determine the immediate, intrinsic desirability of environmental states. The final element of reinforcement learning is a model of the environment.

Google street view could navigate us in an outdoor environment, but within a building without GPS information [39, 40], how could Google street view assist us?

Fig. 5.1 An example of puzzle garden based on Hilbert curve
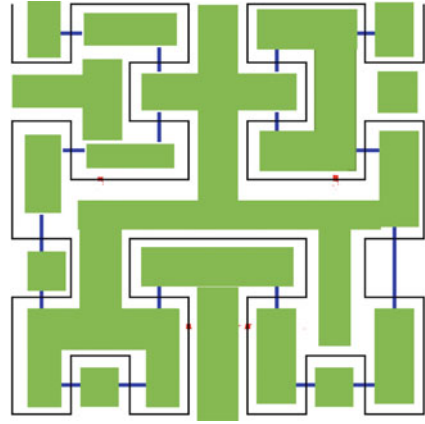
Reinforcement learning [27, 30] could provide relative information and navigate us to get out of this environment. Reinforcement learning is one of the three basic machine learning paradigms, alongside supervised learning and unsupervised learning [17]. Assume we have a building map, how could a robot lead us to leave this building or find a room in this building? Successfully solving this problem will assist us to find the shortest path in shopping mall, subway, or underground without GPS information. In a university, it also could quickly aid students to find their meeting rooms or classrooms, and rapidly guide a robot to get the destination in an indoor environment. An example of puzzle garden is shown in Fig. 5.1.

Basic reinforcement is modeled as a Markov decision process (MDP):

- A set of environment and agent states $\mathbf{S}$;
- A set of actions $\mathbf{A}$ of the agent;
- $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability of transition (at time t) from state s to state s' under action a;
- $R_a(s, s')$ is the immediate reward after transition from s to s' with action a.

The basic reinforcement learning agent interacts with its environment [37] using time steps. At each time t, the agent receives current state s_t and reward r_t. It then chooses an action a_t from the set of available actions, which is subsequently sent to the environment. The environment related to a new state of the agent s_{t+1} and the reward r_{t+1} associated with the transition (s_t, a_t, s_{t+1}) are determined. The goal of a reinforcement learning agent is to learn a policy: $\pi : \mathbf{A} \times \mathbf{S} \to [0, 1], \pi(a, s) = \Pr(a_t = a \mid s_t = s)$, which maximizes the expected cumulative reward [25].

Reinforcement learning relies heavily on states and policy, which is a computational approach to understand and automate goal-directed learning for decision-making. Reinforcement learning makes use of Markov decision process (MDP) [5] to define the interaction between an agent and its environment in terms of states, actions, and rewards. The actions are the decision made by the agent; the states are

the basis for making the decision. The rewards are the basis for evaluating the decision. A policy is a stochastic rule by which the agent selects actions as a function of states. The agent's objective is to maximize the amount of reward it receives over time. The return is the function of future rewards that the agent seeks to maximize the expected value.

5.2 Bellman Equation

The research problems in reinforcement learning include the bandit problems [15], finite Markov decision problem, Bellman equation [19], and value functions. The Bellman equation is special consistency conditions, from which an optimal policy is determined with relative ease. A Bellman equation is a necessary condition for optimality associated with the mathematical optimization method known as dynamic programming. The Bellman equation was firstly applied to engineering control theory and other topics in applied mathematics, subsequently became an important tool in economic theory.

In Markov decision process, a Bellman equation is a recursion for the expected rewards. In the finite Markov decision process, the research methods such as dynamic programming , Monte Carlo methods [12], and temporal-difference learning methods (i.e., TD methods) are taken into consideration. Monte Carlo is a way of solving reinforcement learning problems based on averaging sample returned.

We call a software robot as the agent, which has intelligent capability to make decision by itself, the agent is living in an environment. We need an environment with a policy, actions, related rewards, and states. States are thought as the input of a policy and actions. The best policy and the best reward [1] are obtained by using optimization.

Assume we have an agent and environment, we denote the action of an agent a, a reward r, a policy π, a state s, an action is defined by policy and state, i.e., $a \stackrel{\Delta}{=} \pi(s)$. We denote the samples from our observations as $(s_1, a_1, r_1, \cdots, s_t)$, $t = 1, 2, \cdots$; therefore, reinforcement learning is to find $\max(r)$, s.t. $(s_1, a_1, r_1, \cdots, s_t) \to \pi$.

A finite MDP is the MDP with a finite number of states, actions, and reward sets [8]. The return is the function of future rewards that the agent seeks to maximize in expected value. Markov decision process only affects the next time, which does not influence too much of the sequence at present [24], that means, a state s_t is Markov if

$$P(s_{t+1}|s_t) = P(s_{t+1}|s_1, \cdots, s_t). \tag{5.1}$$

The value function is defined as

$$v(s) \stackrel{\Delta}{=} \mathbf{E}(G_t|s_t). \tag{5.2}$$

The return function is defined as

$$G_t \stackrel{\Delta}{=} \sum_{k=0}^{\infty} \lambda^k \cdot r_{t+k+1}, \tag{5.3}$$

where λ is the discount factor. Because the value function is

$$v(s) = \mathbf{E}(G_t|s_t) = \mathbf{E}(\sum_{k=0}^{\infty} \lambda^k \cdot r_{t+k+1}|s_t), \tag{5.4}$$

we have

$$v(s) = \mathbf{E}(r_{t+1} + \lambda \cdot G_{t+1}|s_t) \tag{5.5}$$

and

$$v(s) = \mathbf{E}(r_{t+1} + \lambda \cdot v(s_{t+1})|s_t). \tag{5.6}$$

Therefore, the action-value function is

$$Q^{\pi}(s, a) \triangleq \mathbf{E}_{s'}(r + \lambda \cdot Q^{\pi}(s', a')|s, a). \tag{5.7}$$

The optimal action-value function therefore is

$$Q^*(s, a) = \mathbf{E}_{s'}(r + \lambda \cdot \max_{a'} Q^*(s', a')|s, a), \tag{5.8}$$

where $\mathbf{E}(\cdot)$ is probability expectation, we call Eq. (5.8) as Bellman equation.

Iteratively, we have the solution of Bellman equation as

$$Q_{i+1}(s, a) = \mathbf{E}_{s'}(r + \lambda \cdot \max_{a'} Q_i(s', a')|s, a) \to Q^*(i \to \infty). \tag{5.9}$$

Rewards are decided by actions. Action-value function $Q(a, s)$ is defined by actions and states. The best Q is dependent on both action a and state s. The process to find the best Q is called Q-learning algorithm. We use Q-learning to find the optimized policy and maximize the reward. Q-learning is a simplified Bellman equation, if $\alpha \in [0, 1]$,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \lambda \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)). \tag{5.10}$$

The best policy, state, and reward are associated with each other. For a deep network *w.r.t.* Q and weight w,

$$Q(s, a, w) = Q^{\pi}(s, a). \tag{5.11}$$

Thus, the loss or objective function is

$$L(w) = \mathbf{E}([r + \gamma \cdot \max_{a'} Q(s', a', w) - Q(s, a, w)]^2). \tag{5.12}$$

The gradient is

$$\frac{\partial L(w)}{\partial w} = -\mathbf{E}([r + \gamma \cdot \max_{a'} Q(s', a', w) - Q(s, a, w)]) \cdot \frac{\partial Q(s, a, w)}{\partial w}. \tag{5.13}$$

Reinforcement learning is the use of value functions to organize and structure the search for good policies. Each occurrence of state in an episode is called a visit. The first time visited in an episode is called the first visit. The first-visit Monte Carlo (MC) method estimates the average of the returns following the first visit.

The policy iteration of Monte Carlo methods is natural to alternate between evaluation and improvement on an episode-by-episode basis. After each episode, the

observed returns are employed for policy evaluation, the policy is improved at all the states visited in the episode.

$$q_\pi(s, a) \doteq \mathbf{E}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = a] \tag{5.14}$$

and

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]. \tag{5.15}$$

The policy improvement theorem: Let π and π' be any pair of deterministic policies such that, for all $s \in \mathscr{S}$, $q_{\pi'}(s, \pi'(s)) \geq v_\pi(s)$, then π' is as good as or better than π, namely, $v_{\pi'}(s) \geq v_\pi(s)$.

$$\pi'(s) \doteq \arg\max_\pi q_\pi(s, a) = \arg\max_\pi \sum_{s',r} p(s', r|s, a)[r + \gamma \cdot v_\pi(s')] \tag{5.16}$$

and

$$v'_\pi(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v'_\pi(s')]. \tag{5.17}$$

In Monte Carlo methods, we make use of first-visit MC methods to estimate the action-value function for the current policy. With the Monte Carlo methods, one must wait until the end of an episode, because only till then, the return is known, whereas with temporal-difference (TD) methods, one need wait only one time step. TD methods can learn directly from raw experience without a model of the environmental dynamics, which update estimates based on other learned estimates, without waiting for the final outcome. The simplest TD method makes the update immediately on transition to S_{t+1} and receives R_{t+1}.

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma \cdot V(S_{t+1}) - V(S_t)]. \tag{5.18}$$

TD error measures the difference between the estimated value S_t and the better estimate $\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$.

In gradient-descent methods [41], $\mathbf{w} = (w_1, \cdots, w_d)^\top$,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha[v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t)]\nabla\hat{v}(S_t, \mathbf{w}_t), \tag{5.19}$$

where α is a positive step-size parameter, $\nabla\hat{v}(S_t, \mathbf{w}_t)$ is the gradient with respect to $\mathbf{w}$. This yields the following general SGD method for state-value prediction:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha[U_t - \hat{v}(S_t, \mathbf{w}_t)]\nabla\hat{v}(S_t, \mathbf{w}_t). \tag{5.20}$$

Because the true value of a state is the expected value of the return, the Monte Carlo target is $U_t \doteq G_t$.

Linear methods approximate the state-value function by using inner product

$$\hat{v}(s, \mathbf{w}) = \mathbf{w}^\top \mathbf{x}(s) = \sum_{i=1}^d w_i x_i(s), \tag{5.21}$$

where $\hat{v}(\cdot, \mathbf{w})$ is a linear function of the weight vector $\mathbf{w}$, $\mathbf{x}(s) = (x_1(s), x_2(s), \cdots, x_d(s))^\top$ is a real-valued vector, $\mathbf{x}(s)$ is called a vector representing state s.

The gradient of the approximate value function [41] is

$$\nabla \hat{v}(s, \mathbf{w}) = \mathbf{x}(s). \tag{5.22}$$

The general SGD update is

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha[U_t - \hat{v}(S_t, \mathbf{w})]\mathbf{x}(s_t). \tag{5.23}$$

The gradient of the approximate value function [41] is $\nabla \hat{v}(s, \mathbf{w}) = \mathbf{x}(s)$. The general SGD update is

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha[U_t - \hat{v}(S_t, \mathbf{w})]\mathbf{x}(s_t). \tag{5.24}$$

For example,

$$\mathbf{w}_{t+n} = \mathbf{w}_{t+n-1} + \alpha[G_{t:t+n} - \hat{v}(S_t, \mathbf{w}_{t+n-1})]\nabla\hat{v}(S_t, \mathbf{w}_{t+n-1}), \tag{5.25}$$

$$G_{t:t+n} = R_{t+1} + \gamma \cdot R_{t+2} + \cdots + \gamma^{n-1} \cdot R_{t+n} + \gamma^n \cdot \hat{v}(S_{t+n}, w_{t+n-1}), \tag{5.26}$$

where $0 \le t \le T - n$.

To produce a policy parameterization, the policy is defined as the normal probability density over a real-valued scalar action with mean and standard deviation given by parametric function approximators that depend on the state.

$$\pi(a|s, \theta) = \frac{1}{\sigma(s, \theta)\sqrt{2\pi}} \exp\left(-\frac{(a - \mu(s, \theta))^2}{2\sigma(s, \theta)^2}\right) \tag{5.27}$$

where $\theta = (\theta_\mu, \theta_\sigma)^\top$, $\mu(s, \theta) = \theta_\mu^\top \mathbf{x}_\mu(s)$, $\sigma(s, \theta) = \exp(\theta_\sigma^\top \mathbf{x}_\sigma(s))$, $x_\mu(s)$ and $x_\sigma(s)$ are state feature vectors.

5.3 Deep Q-Learning

Reinforcement learning is provided to seek the best policy and maximize the total reward. The key of reinforcement learning is to maximize the rewards, the question is how to get the best action so as to achieve the best reward. The sequence of actions thus will have the maximum cumulative reward. For each policy π, there is a reward $v^\pi(s_t)$, we hope to find the optimal policy

$$v^*(s_t) = \max_\pi(v^\pi(s_t)), \forall s_t. \tag{5.28}$$

In a simple case, action is defined as

$$a(t) \stackrel{\Delta}{=} \pi(s_t), \tag{5.29}$$

where $Q(a_t) = r(a_t) > 0$. If $r(a)$ is the reward function

$$Q(a_{t+1}) \leftarrow Q_t(a_t) + \eta \cdot [r(a_{t+1}) - Q(a_t)], \tag{5.30}$$

where η is learning rate.

On the other hand, in a full reinforcement learning, a policy π defines the action to be taken in any state

$$a_t \overset{\Delta}{=} \pi(s_t). \tag{5.31}$$

The value of state s_t satisfies

$$v(s_t) = \max_{a_t} Q(s_t, a_t), \tag{5.32}$$

$$a_t^* = \arg\max_{a_t} Q(s_t, a_t), \tag{5.33}$$

and

$$\pi^*(s_t^*) = a_t^*. \tag{5.34}$$

The value iteration is

$$|v^{(l+1)}(s) - v^{(l)}(s)| < \delta, \tag{5.35}$$

where $\delta > 0, l = 1, 2, \cdots$ and

$$v(s_t) \leftarrow v(s_t) + \eta \cdot [r_{t+1} + \gamma \cdot v(s_{t+1}) - v(s_t)]. \tag{5.36}$$

The policy iteration is

$$\pi' = \arg\max_{\pi} (v^\pi(s')); \tag{5.37}$$

and

$$v^\pi(s) \leftarrow v^\pi(s'). \tag{5.38}$$

The rewards and actions are

$$Q(a_t, s_t) = r_{t+1} + \gamma \cdot \max_{a_{t+1}} Q(a_{t+1}, s_{t+1}). \tag{5.39}$$

This iteration has been employed to approximate the best value. Hence, we develop the iteration further:

- **Episode**: $\exists T, (s_1, a_1, r_2, \cdots, s_T) \to \pi$
- **Monte Carlo Method**: Using empirical mean to replace Bellman equation instead of expected return, i.e.,

$$v_\pi(s) = \frac{1}{T} \sum_{t=1}^{T} (G_t | s_t = s), \tag{5.40}$$

where $G_t = \sum_{k=1}^{T-t} \lambda^{k-1} r_{t+k}$. Hence,

$$\pi(s) \leftarrow \arg\max_{a} Q(s, a), \tag{5.41}$$

and

$$v(s_t) \leftarrow v(s_t) - \alpha \cdot (G_t - v(s_t)). \tag{5.42}$$

- Temporal Difference (TD):

$$v(s_t) \leftarrow v(s_t) - \alpha \cdot (r_{t+1} + \gamma \cdot v(s_{t+1}) - v(s_t)), \tag{5.43}$$

where $\delta = r_{t+1} + \gamma \cdot v(s_{t+1}) - v(s_t)$ is the TD error and $\sigma = r_{t+1} + \gamma \cdot v(s_{t+1})$ is the TD target.

For the best convergence, we adopt Q-learning and double Q-learning for finding the best policies and actions:

- **Episode**: $\exists T, (s_1, a_1, r_2, \cdots, s_T) \to \pi$.
- **SARSA (State-Action-Reward-State-Action) Algorithm**:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \cdot [r + \gamma \cdot Q(s', a') - Q(s, a)], \qquad (5.44)$$

  where $s \leftarrow s'$ and $a \leftarrow a'$.
- **Q-Learning**: An off-policy TD control algorithm:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \cdot [r + \gamma \cdot \max_a Q(s', a) - Q(s, a)], \qquad (5.45)$$

  where $s \leftarrow s'$.
- **Double Q-Learning**:

$$Q_1(s, a) \leftarrow Q_1(s, a) + \alpha \cdot [r + \gamma \cdot Q_2(s', \arg\max_a Q_1(s', a)) - Q_1(s, a)], \quad (5.46)$$

  and

$$Q_2(s, a) \leftarrow Q_2(s, a) + \alpha \cdot [r + \gamma \cdot Q_1(s', \arg\max_a Q_2(s', a)) - Q_2(s, a)], \quad (5.47)$$

  where $s \leftarrow s'$.

The control is very similar to Kalman filtering [7,28], but Kalman filtering is a linear dynamical system for signal filtering [3,20]. In control theory, Kalman filtering, i.e., Linear Quadratic Estimation (LQE), is an algorithm that produces estimates of unknown variables by estimating a joint probability distribution over the variables for each timeframe [18,43].

Reinforcement learning enables a computer to make a series of decisions and maximize the cumulative reward for the task without human intervention and without being explicitly programmed to achieve the task. MATLAB lists all examples of reinforcement learning at https://www.mathworks.com/help/reinforcement-learning/examples.html.

An example to swing up and balance pendulum with image observation could be found from https://au.mathworks.com/help/deeplearning/ug/train-ddpg-agent-to-swing-up-and-sbalance-pendulum-with-image-observation.html.

The screenshots are shown in Fig. 5.2, Fig. 5.2a,b and c display the pendulum in various positions amid the swing.

In recent years, the journal Nature has published a slew of papers related to reinforcement learning [9,30,33,38]. The high quality work has highlighted the state-of-the-art research work and greatly uplifted the popularity and depth of reinforcement learning.
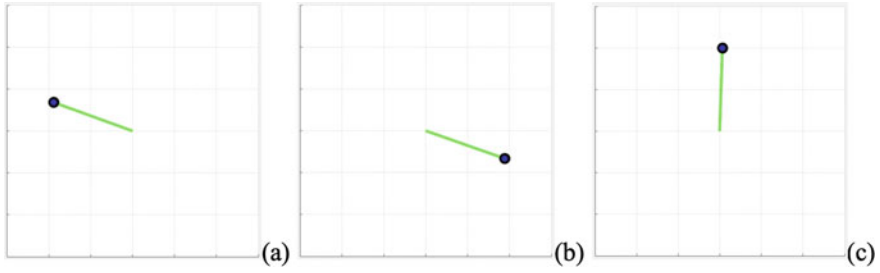
Fig. 5.2 An example shows to swing up and balance a pendulum with an image observation modeled in MATLAB

5.4   Control Theory

In deep learning, mathematical control theory could support our model training process, from the continuous observations of loss and accuracy, we make decisions whether our training process should go on or not. In robotics, we can control our algorithms using the mathematical control theory as well.

5.4.1   Mathematical Control Theory

A starting point of control theory [42] is the differential equation

$$\dot{y} = f(y, u), \ y(0) = x \in \mathscr{R}^n, \ u \in \mathbf{U} \in \mathscr{R}^m, \ m, n, \in \mathscr{Z}^+ \qquad (5.48)$$

where $u(\cdot)$ is control function, $\mathbf{U}$ is the set of control parameters. In control theory, we have

- Controllability: If an arbitrary state z is reachable from an arbitrary state x in a time $t = T$, $y(T) = z$; $t = 0$, $y(0) = x$.
- Stabilizability: $k(\bar{x}) = \bar{u}$ is called a stabilizing feedback if $f(\bar{x}, \bar{u}) = 0$ and $\dot{y}(t) = f(y(t)), k(y(t)))$.
- Observability: $\dot{y}(t) = f(y, u)$, $y(0) = x$, $w = h(y)$ where $h(\cdot)$ is observation function.
- Optimality: Looking for a control function $u(\cdot)$ which minimizes the integral $I = \int_0^T g(y(t), u(t))dt + G(y(T))$, $g(\cdot)$ and $G(\cdot)$ are given functions, the time $t = T > 0$ is fixed.

Classical control theory is a linear system described by a differential equation

$$\dot{y} = Ay(t) + Bu(t), \ y(0) = x \in \mathscr{R}^n \qquad (5.49)$$

where $w(t) = Cy(t)$ is an observation, $t \geq 0$.

The unique solution of this equation is $\dot{q} = A(t)q(t) + a(t)$, $q(t_0) = q_0 \in \mathcal{R}^n$, $t_0 \in [0, T]$ is

$$q(t) = S(t)S^{-1}(t_0)q_0 + \int_0^T S(t)S^{-1}(s)a(s)ds$$

where $t \in [0, T]$, $\dot{S} = A(t)S(t)$, $S(0) = I$. The function $S(t)$, $t \in [0, t]$ is called fundamental solution.

Let $A \in \mathbf{M}(n, n)$ and consider linear systems,

$$\dot{z} = Az, z(0) = x \in \mathcal{R}^n. \tag{5.50}$$

The solution is

$$z^x(t) = S(t)x = e^{eA}x, t \geq 0. \tag{5.51}$$

The system is stable, if $z^x(t) \to 0$, $x \in \mathcal{R}^n$ as $t \to +\infty$.

If $\mathbf{P}$ is a nonsingular matrix and $\mathbf{A}$ is a stable matrix, then matrix $\mathbf{PAP}^{-1}$ is stable, $\mathbf{PAP}^{-1} = \mathbf{diag}(\mathbf{J}_1, \cdots, \mathbf{J}_r)$, $\mathbf{J}_i$ is Jordan blocks, $w(\mathbf{A}) = \sup\{Re(\lambda) : \lambda \in \sigma(\mathbf{A})\} < 0$.

If polynomial $p(\lambda) = \lambda^n + a_1\lambda^{n-1} + \cdots + a_{n-1}\lambda_n$, $\lambda \in \mathcal{C}$ is stable, then all its coefficients $a_1, \cdots, a_{n-1}$ are positive.

The system $\dot{f}(t) = Ay + Bu$, $y(0) = x \in \mathcal{R}^n$ is stabilizable or the pair (A, B) is stabilizable if there exists a matrix $K \in \mathbf{M}(m, n)$ such that the matrix $A + BK$ is stable.

The system $\dot{f}(t) = Ay + Bu$, $y(0) = x \in \mathcal{R}^n$ is completely stabilizable if and only if for arbitrary $w > 0$ there exist a matrix $\mathbf{K}$ and a constant $M > 0$ such that for an arbitrary solution $y^x(t)$, $t > 0$ of $\dot{y}(t) = (A + B\mathbf{K})y(t)$, $y(0) = x$, then $|y^x(t)| < Me^{-wt}|x|$, $t \geq 0$.

For a nonlinear system,

$$\dot{y} = f(y, u), y(0) = x \in \mathcal{R}^n, w = h(y) \tag{5.52}$$

where $f(\cdot) \in \mathcal{R}^n, h(\cdot) \in \mathcal{R}^k$.

If $f(\cdot)$ is continuous, for a number $c > 0$, then

$$|f(x, u)| \leq c(1 + |u| + |x|) \tag{5.53}$$

and

$$|f(x, u) - f(y, u)| \leq c|x - y|. \tag{5.54}$$

For an arbitrary control $u(\cdot)$, there exists exactly one solution. Assume that the pair (A, B) where $A \in \mathbf{M}(n, n)$, $B \in \mathbf{M}(n, m)$, is controllable, and shows that

$$\dot{y} = Ay + Bu, y(0) = x \tag{5.55}$$

is locally controllable at $0 \in \mathcal{R}^n$ in arbitrary time $T > 0$.

Assume that the mapping $f(\cdot)$ is differentiable at $f(\bar{u}, \bar{v})$, with

$$A = f_x(\bar{u}, \bar{v}), B = f_u(\bar{u}, \bar{v}). \tag{5.56}$$

The nonlinear system is called linearization at $(\bar{x}, \bar{u})$.

Assume that the pair (A, B) where $A \in \mathbf{M}(n, n)$, $B \in \mathbf{M}(n, m)$, is controllable, which shows that

$$\dot{y} = Ay + Bu, \, y(0) = x \tag{5.57}$$

is locally controllable at $0 \in \mathcal{R}^n$ in arbitrary time $T > 0$.

Assume that the mapping $f(\cdot)$ is continuously differentiable in a neighborhood of a point $(\bar{x}, \bar{u})$ for which $f(\bar{x}, \bar{u}) = 0$. The linearization is controllable, the nonlinear system is locally controllable at the point $\bar{x}$ in arbitrary time $T > 0$.

The nonlinear system is locally controllable at $\bar{x} \in \mathcal{R}^n$, if and only if (i.e., iff) the pair (A, B) is controllable,

$$A\bar{x} + B\bar{u} = 0, \, \forall \bar{u} \in \mathcal{R}^m. \tag{5.58}$$

Let's assume that the state equation is independent on the control parameter.

$$\dot{z} = f(z), \, z(0) = x \in \mathcal{R}^n \tag{5.59}$$

and the observation is $w(t) = h(z(t)), \, t \geq 0$.

This system is observable at a point z if there exists a neighborhood D of z such that for arbitrary $x_1 \in D$, $x_1 \neq x$, there exists $t > 0$ for $h(z(t, x)) \neq h(z(t, x_0))$.

Bellman equations are a derivation of dynamic programming equations which solve linear regulator problem based on a finite time interval.

Regarding control system, $\dot{y} = f(y, u)$, $y(0) = x$, if the control interval is $[0, \infty]$, the cost functional is

$$J(x, u) = \int_0^\infty g(y(t), u(t))dt. \tag{5.60}$$

Our aim is to find $\hat{u}(\cdot)$ for all $u(\cdot)$

$$J(x, \hat{u}) \leq J(x, u), \tag{5.61}$$

has exactly one nonnegative, continuous, and bounded solution $v = \hat{v}$.

5.4.2 Stochastic Control Theory

Let $(\Omega, \mathcal{F}, P)$ be a complete probability space, $(\Omega, \mathcal{F})$ is a measurable space, P is a probability measure on $\mathcal{F}$. Let (U, d) be a Polish space [31], i.e., U is a separable complete metric space with a metric d, let $B(U)$ be a field generated by the open subsets of U. $X : \Omega \mapsto U$ is called a U-valued random variable if X is U-measurable. If $U \in \mathcal{R}^d$, we call X as a d-dimensional random variable.

A stochastic process $X = (X(t); t \in [0, T])$ is a family of d-dimensional random variables, defined on $(\Omega, \mathcal{F}, P)$. Based on a continuous Markov process with transition probability function $p(\cdot)$, if the condition $X(s) = s$ is given, then the probability distribution of the process $(X(t); t > s)$ does not depend on its past $(X(t); t < s)$, the initial condition is $X(s) = x$.

Let $\theta \in (0, T]$, consider d-dimensional stochastic control theory (SDE),

$$dX(t) = b(t, X(t), \Omega)dt + \alpha(t, X(t), \Omega)dW(t), \, t > \theta. \tag{5.62}$$

With the initial condition $X_\theta = X(\theta)$, $b(\cdot)$ and $\alpha(\cdot)$ are the drift coefficient and the diffusion coefficient, respectively, we see $X = (X(t), t \in (\theta, T])$ is a solution if X is continuous and satisfies

$$X(t) = X_\theta + \int_\theta^t b(t, X(t), \Omega)dt + \int_\theta^t \alpha(t, X(t), \Omega)dW(t), t \in (\theta, T]. \quad (5.63)$$

The solution X is unique if

$$p(X(t) = \hat{X}(t), t \in [\theta, T]) = 1. \quad (5.64)$$

5.4.3 Fuzzy Control Theory

Let $x \in \mathcal{F}(X)$ be a fuzzy set over the universe of discourse X, $0 \le \alpha \le 1$. The set is called α-level set or α-cut of the fuzzy set μ [29].

$$[x]_\alpha = \{x \in X, \mu(x) > \alpha\}. \quad (5.65)$$

The operations on fuzzy sets include

- The intersection of two fuzzy sets μ_1 and μ_2, $\mu_1 \cap_t \mu_2(x) = t(\mu_1(x), \mu_2(x))$
- The union of two fuzzy sets μ_1 and μ_2, $\mu_1 \cup_t \mu_2(x) = s(\mu_1(x), \mu_2(x))$
- The complement of the (ordinary) set, $\bar{\mu}(x) = 1 - \mu(x)$, $\bar{\bar{\mu}} = \mu$.

Fuzzy relations refer to model dependencies, correlations or connections between variables, quantities, or attributes.

$$x R y = (x, y) \in X \times Y = R \quad (5.66)$$

where relation R over the universes of discourse X and Y is a subset of the Cartesian product $X \times Y$ of X and Y.

A fuzzy set $\mathcal{G} \in \mathcal{F}(X \times Y)$ is called (binary) fuzzy relation between the universes of discourse X and Y. For every fuzzy set $\mu \in \mathcal{F}(X)$, we have $(\mathcal{G}_1 \circ \mathcal{G}_2)[\mu] = \mathcal{G}_2[\mathcal{G}_1[\mu]]$.

Let $\mathcal{G}_1 \in \mathcal{F}(X \times Y)$ and $\mathcal{G}_2 \in \mathcal{F}(Y \times Z)$ be fuzzy relations, the composition of two fuzzy relations is

$$\mathcal{G}_1 \circ \mathcal{G}_2 = \sup \min\{\mathcal{G}_1(x, y), \mathcal{G}_2(y, z)|y \in Y\} \quad (5.67)$$

between the universes of discourse X and Z. Equivalently,

$$(x, z) \in R_1 \circ R_2 \Leftrightarrow \exists y \in Y((x, y) \in R_1 \wedge (y, z) \in R_2) \quad (5.68)$$

and

$$(\mathcal{G}_1 \circ \mathcal{G}_2)(x, z) = \sup(\min(\mathcal{G}_1(x, y), \mathcal{G}_2(y, z))|y \in Y). \quad (5.69)$$

For step function $x(s) = \frac{1}{s}$, using the general equation $y(s) = G(s)x(s)$, we obtain the step response

$$y(s) = G(s)\frac{1}{s}. \quad (5.70)$$

A system is called stable if, for $t \to \infty$, its step response converges toward a finite value. Otherwise, it is said to be unstable.

A linear system is called stable if for an input signal with limited amplitude, its output signal will also show a limited amplitude. This is the BIBO-Stability (i.e., bounded input–bounded output).

The overall output quantity $\mathbf{u}$ of a controller can be computed as an overlapping of these vectors by using $\mathbf{u} = \sum k_i(\mathbf{z}(t))\mathbf{u}_i$, where $k_i(\mathbf{z}(t))$ are the truth functions. The vector $\mathbf{z}(t)$, for that the truth function k_i takes the value '1' and all other truth functions the value '0'. The vector $\mathbf{u}_i$ is the output vector of the controller at the i-th supporting point. The weighting factor $k_i(\mathbf{z}(t))$ depends on actual value of the input variables.

For a fuzzy controller $u = \sum k(\mathbf{x})_i u_i = \mathbf{u}^\top \mathbf{k}(\mathbf{x})$, fuzzy clustering refers to $F_{\min}(\mathbf{U}, \mathbf{W}, \mathbf{X}) = \sum \sum u_{ij} d(\mathbf{w}_i, \mathbf{x}_j)$, $d(\mathbf{w}_i, \mathbf{x}_j)$ is the distance of the data object $\mathbf{x}_j$ to the cluster $\mathbf{w}_i$, $\mathbf{U} = (u_{ij})$, $\sum u_{ij} = 1$. Fuzzy c-means algorithm (FCM) takes into account of $d(\mathbf{w}_i, \mathbf{x}_j) = \|\mathbf{w}_i - \mathbf{x}_j\|^2$. Fuzzy c-varieties algorithm (FCV) is based on $d((\mathbf{v}_i, \mathbf{e}_i), \mathbf{x}_j) = ((\mathbf{x}_j - \mathbf{v}_i)^\top \mathbf{e}_i)^2$.

Fuzzy systems provide the possibility of interpreting the controller and introducing prior knowledge. Artificial neural networks contribute its capability of learning and the possibility of automatic optimization or automatic generation of the whole controller. With neural fuzzy systems, the possibility to learn a controller and then interpret its control strategy by analyzing the learned fuzzy rules and fuzzy sets, if necessary, revise them regarding the stability criteria. With the mapping onto a neural network structure, we are able to provide that there are adequate training data to optimize the parameters of a fuzzy set.

Neural fuzzy controllers are grouped in cooperative and hybrid models. In cooperative models, the neural net and the fuzzy controller operate separately. A hybrid fuzzy controller is interpreted as neural net which can be implemented with the help of a neural net.

Compared to classical control, fuzzy controller is a nonlinear characteristic surface without internal dynamics, which can be interpreted as nonlinear state space controller. The design of a fuzzy controller is usually based on heuristics. Fuzzy controllers have a high degree of robustness.

5.5 Optimization

Optimization is the core method in deep neural networks. The optimization includes linear programming-based [10], nonlinear-based [2,4], dynamic-based [6], or neural network-based one [22], etc. Local optimization and global optimization are the main problems that the optimization aims to solve. The local minimum and global minimum are always sought in the optimization algorithms.

There are two categories of optimizations: Unconstrained optimization and constrained optimization. The unconstrained optimization refers to the optimization without constraint conditions. Meanwhile, the constrained optimization refers to the

optimization having constraint conditions. Most of optimization problems with constraint conditions, therefore it is constrained optimization. The constraints are usually with regard to (i.e., *w.r.t.*) or subject to (i.e., *s.t.*) constraint conditions.

Linear programming problem is

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x}) \tag{5.71}$$

it is subject to (i.e., *s.t.*) a condition $\mathbf{A}\mathbf{x} = \mathbf{b}$.

In linear programming, if we modify the parameters such as $\mathbf{A}$, $\mathbf{b}$ will be changed to $\mathbf{A} + \Delta\mathbf{A}$ and $\mathbf{b} + \Delta\mathbf{b}$, where $\Delta\mathbf{A}$ and $\Delta\mathbf{b}$ are the small changes, we need to check how they will affect our optimization, find out whether the solution of this optimization problem is under control or not.

In optimization, we have multiple objective programming problem. How to find the best solution of this multiple objective programming problem is a key issue in mathematical optimization. Usually, we need to seek the derivatives. Sometimes, if we could not find the derivatives of a function for seeking the local optimization solution, we may extend the problem by using mathematical regularization [16].

For dynamic optimization problem, we also need to calculate the derivatives. If the derivatives could not be found, one of the solutions is to utilize genetic algorithm (GA). Modern optimization refers to nature-inspired computing. Usually, the modern optimization algorithms include genetic algorithm (GA), simulated annealing, particle swarm optimization, ant colony optimization, etc. [32].

5.6 Data Fitting

If $y = f(z, x_1, \cdots, x_n)$, $y_k = f(z_k, x_1, \cdots, x_n)$, $k = 1, \cdots, m$, $m > n$, the best solution is to minimize

$$\varepsilon(x_1, \cdots, x_n) = \sum_{i=1}^{m} (y_i - f(z_i, x_1, \cdots, x_n))^2 \tag{5.72}$$

or

$$\varepsilon(x_1, \cdots, x_n) = \sum_{i=1}^{m} (y_i - f_i(x_1, \cdots, x_n))^2. \tag{5.73}$$

Therefore,

$$\frac{\partial\varepsilon(x_1, \cdots, x_n)}{\partial x_i} = \frac{\partial}{\partial x_i} \sum_{i=1}^{m} (y_i - f_i(x_1, \cdots, x_n))^2. \tag{5.74}$$

Linear least squares problem: If the functions $f_k(x_1, \cdots, x_n)$, $k = 1, \cdots, m$ are linear, let

$$\|\mathbf{y} - A\mathbf{x}\|^2 = (\mathbf{y} - A\mathbf{x})^\top (\mathbf{y} - A\mathbf{x})$$

be minimized as $\mathbf{x} = (x_1, \cdots, x_n)^\top$, namely,

$$\min_{\mathbf{x} \in \mathbf{R}^n} \|\mathbf{y} - A\mathbf{x}\|$$

$$\Rightarrow \nabla_{\mathbf{x}}[(A\mathbf{x} - \mathbf{y})^\top (A\mathbf{x} - \mathbf{y})] = 2A^\top A\mathbf{x} - 2A^\top \mathbf{y} = 0$$

$$\Rightarrow A^\top A\mathbf{x} - A^\top \mathbf{y} = 0$$

$$\Rightarrow \mathbf{x} = (A^\top A)^{-1} A^\top \mathbf{y}. \tag{5.75}$$

If the function $f(\mathbf{x}) = (f_1, \cdots, f_m)^\top$ is nonlinear, $\mathbf{y} = (y_1, \cdots, y_m)^\top$, let $\|\mathbf{y} - f(\mathbf{x})\|^2$ be minimized as $\mathbf{x} = (x_1, \cdots, x_n)^\top$, the Jacobian matrix is

$$\frac{\partial \mathbf{J}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = 0. \tag{5.76}$$

The solution $\bar{\mathbf{x}}$ of the nonlinear least squares problem satisfies

$$\|\mathbf{y} - f(\bar{\mathbf{x}})\|^2 \le \|\mathbf{y} - f(\mathbf{x})\|^2. \tag{5.77}$$

The solution is given by using Gauss–Newton method

$$\mathbf{x}^{(i+1)} := \mathbf{x}^{(i)} - \nabla^{-1} f(\mathbf{x}^{(i)}) f(\mathbf{x})^{(i)}. \tag{5.78}$$

For nonlinear function, if

$$f(\xi) = f(\mathbf{x}_0) + f'(\mathbf{x}_0)(\xi - \mathbf{x}_0) = 0 \tag{5.79}$$

then,

$$\xi = \mathbf{x}_0 - \frac{f(\mathbf{x}_0)}{f'(\mathbf{x}_0)}. \tag{5.80}$$

The generalized Newton method for solving systems of equations is given by

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \frac{f(\mathbf{x}_i)}{f'(\mathbf{x}_i)} \tag{5.81}$$

where $i = 0, 1, 2, \cdots$.

A sequence $\mathbf{x}_i \in \mathcal{R}^n$ is convergent if and only if for every $\varepsilon > 0$, there exists an $N(\varepsilon)$, such that $|x_l - x_m| < \varepsilon, \forall l, m \ge N(\varepsilon)$.

Theorem 5.1 *General convergence theorem: Let function $\mathbf{y} = \Phi(\mathbf{x})$, $\mathbf{x}, \mathbf{y} \in \mathcal{R}^n$ have a point $\xi = \Phi(\xi)$ and $S_r(\xi) = \{\mathbf{x} : \|\mathbf{x} - \xi\| < r\}$ be a neighborhood of ξ such as $\Phi(\cdot)$ is a contractive mapping in $S_r(\xi)$, namely,*

$$\|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\| \le K \|\mathbf{x} - \mathbf{y}\| \tag{5.82}$$

where $K \in [0, 1]$, $\mathbf{x}, \mathbf{y} \in S_r(\xi)$.

For the generated sequence $\mathbf{x}_i = \Phi(\mathbf{x}_i)$, $i = 0, 1, 2, \cdots$, $\mathbf{x}_i \in S_r(\xi)$,

$$\|\mathbf{x}_{i+1} - \xi\| \le K \|\mathbf{x}_i - \xi\|. \tag{5.83}$$

If function $\mathbf{y} = f(\mathbf{x})$, $\mathbf{x} \in S_r(\mathbf{x}_0) = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_0\| < r\}$ has the properties:

- $\|f'(\mathbf{x}) - f'(\mathbf{y})\| < \gamma \|\mathbf{x} - \mathbf{y}\|$, $\forall \mathbf{x}, \mathbf{y} \in S_r(\mathbf{x}_0)$, $\gamma \in [0, 1]$.
- $f'(\mathbf{x})^{-1}$ exists, $\|f'(\mathbf{x})^{-1}\| < \beta$, $\forall \mathbf{x} \in S_r(\mathbf{x}_0)$, $\beta \in [0, 1]$.
- $\|f'(\mathbf{x}_0)^{-1} f(\mathbf{x}_0)\| < \alpha$, $\alpha \in [0, 1]$,

then,

-
$$\mathbf{x}_{i+1} := \mathbf{x}_i - f'(\mathbf{x}_i)^{-1} f(\mathbf{x}_i) \tag{5.84}$$

  where $\mathbf{x}_i \in S_r(\mathbf{x}_0)$, $i = 0, 1, \cdots$
-
$$\lim_{k \to \infty} x_k = \xi \tag{5.85}$$

  where $\xi \in S_r(\mathbf{x}_0), f(\xi) = 0$.
- $\forall k \geq 0$,

$$\|\mathbf{x}_k - \xi\| < \eta \cdot \frac{h^{2k-1}}{1 - h^{2k}} \tag{5.86}$$

  where $\eta \in [0, 1]$.

Given a matrix $\mathbf{A} = (a_{ij})_{n \times n}$, find $\lambda \in \mathbf{C}$, such as the linear system of equations has a nontrivial solution $\mathbf{x} \neq 0$.

$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{x} = 0 \tag{5.87}$$

where λ is an eigenvalue of the matrix $\mathbf{A}$, $\mathbf{x}$ is an eigenvector of matrix $\mathbf{A}$ associated with eigenvalue λ, the set of all eigenvalues are called the spectrum of $\mathbf{A}$.

$$\phi(\mu) = det(\mathbf{A} - \mu \mathbf{I}) \tag{5.88}$$

is denoted as the characteristic polynomial

$$\phi(\mu) = (\mu - \lambda_1)^{\sigma_1} (\mu - \lambda_2)^{\sigma_2} \cdots (\mu - \lambda_k)^{\sigma_k} \tag{5.89}$$

where $\sigma_i = \sigma(\lambda_i)$, $\sigma_1 + \sigma_2 + \cdots + \sigma_k = n$. Especially,

$$\phi(\mathbf{A}) = 0. \tag{5.90}$$

Given matrices $\mathbf{A}$ and $\mathbf{B}$, there exists a vector $\mathbf{x} \neq 0$,

$$\mathbf{A}\mathbf{x} = \mathbf{B}\lambda\mathbf{x}. \tag{5.91}$$

If $|\mathbf{B}| \neq 0$,

$$\mathbf{B}^{-1}\mathbf{A}\mathbf{x} = \lambda\mathbf{x}. \tag{5.92}$$

and $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, then

$$|\lambda| \leq \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \tag{5.93}$$

where $\rho = \max_{1 \leq i \leq n}(|\lambda_i|)$ is the spectrum radius of $\mathbf{A}$.

5.7 Polynomials

Generally, a polynomial over $\mathscr{R}$ is an expression of the form

$$f(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{5.94}$$

where n is a nonnegative integer, the coefficients $a_i \in \mathscr{R}$, $0 \leq i \leq n$. Regarding polynomials, we have the properties:

- $f(x) = \sum_{i=0}^{n} a_i x^i$ and $g(x) = \sum_{i=0}^{n} b_i x^i$ are equal, iff $a_i = b_i \in \mathscr{R}$.
- $f(x) + g(x) = \sum_{i=0}^{n} (a_i + b_i) x^i$ (i.e., *polynomial sum*).
- $f(x)g(x) = \sum_{k=0}^{m+n} (c_k) x^k$, $c_k = \sum_{i+j=k} a_i b_j$ (i.e., *polynomial product*).

The degree of polynomials $n \in \mathscr{Z}^+$ satisfies

- $n = \deg(f(x)) = \deg(f)$.
- $\deg(f + g) \leq \max(\deg(f), \deg(g))$.
- $\deg(fg) \leq \deg(f) + \deg(g)$.

The ring [26], formed by the polynomials over $\mathscr{R}$, is called polynomial ring over $\mathscr{R}$ and denoted as $\mathscr{R}[x]$, 0 stands for the zero polynomial ring.

Polynomial division is defined as follows: Let $g \in \mathscr{R}[x]$, $\forall f \in \mathscr{R}[x]$, there exists $q, r \in \mathscr{R}[x]$ such as $f = q \cdot g + r$, where $\deg(r) < \deg(g)$.

The greatest common divisor is defined: $g = gcd(f_1, f_2, \cdots, f_n)$, $f_i \in \mathscr{R}[x]$ is a polynomial. If $gcd(f_1, f_2, \cdots, f_n) = 1$, then $f_1, f_2, \cdots, f_n$ are relatively prime.

The least common multiple is defined as $m = lcm(f_1, f_2, \cdots, f_n)$, $f_i \in \mathscr{R}[x]$, $i = 1, 2 \cdots, n$.

An element $b \in \mathscr{R}$ is called a root of the polynomial $f \in \mathscr{R}[x]$ if $f(b) = 0$ and $x - b$ divides $f(x)$.

Let $b \in \mathscr{R}$ is a root of the polynomial $f \in \mathscr{R}[x]$, if k is a positive integer $\mathscr{Z}^+$, such as $f(x)$ is divisible by $(x - b)^k$, but not $(x - b)^{k+1}$, then k is the multiplicity of b.

If $f(x) = \sum_{i=0}^{n} a_i x^i \in \mathscr{R}[x]$, then the derivative f' of f is defined by $f'(x) = \sum_{i=1}^{n} i a_i x^{i-1} \in \mathscr{R}[x]$.

If we have interpolating points $f(a_i) = b_i$, $i = 0, 1, \cdots, n$, the polynomial is defined as Lagrange interpolation polynomial, if

$$f(x) = \sum_{i=0}^{n} b_i \prod_{k=0, k \neq n}^{n} (a_i - a_k)^{-1} (x - a_k) \in \mathscr{R}[x]. \tag{5.95}$$

Generally, Lagrange interpolation [23] $L(x) \in \mathscr{R}[x]$ with degree $k - 1$ is

$$L_{k-1}(x) = \sum_{i=1}^{k} l_i(x) = \sum_{i=1}^{k} \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{x - x_j}{x_i - x_j} \cdot f(x_i) \tag{5.96}$$

where

$$l_i(x) = \prod_{j=1}^{k} \frac{x - x_j}{x_i - x_j} \tag{5.97}$$

where $x_0 \leq \xi \leq x_k$ is the base function. Lagrange interpolation has the properties:

- Error bound is

$$|R(x)| \leq \frac{(x_k - x_0)^k}{k!} \max_{x_0 \leq \xi \leq x_k} |f^{(k)}(\xi)|. \tag{5.98}$$

  The remainder item is

$$R(x) = \frac{f^{(k)}(\xi)}{k!} \prod_{i=1}^{k} (x - x_i) \tag{5.99}$$

  where $x_0 \leq \xi \leq x_k$.
- There exist a table of Chebyshev nodes to ensure the convergence of Lagrange interpolation:

$$\lim_{k \to \infty} L_k(x) = f(x) \tag{5.100}$$

  where $f(x) \in C[a, b]$, $x_i \in [a, b]$.
- Runge's phenomenon exists in polynomials, like Lagrange interpolations,

$$\lim_{k \to \infty} (\max_{x \in [a,b]} |f(x) - L_k(x)|) = +\infty. \tag{5.101}$$

- Lagrange interpolation is a special case of Chinese remainder theorem (CRT) for polynomials [11,35].

Furthermore, let $f \in \mathscr{R}[x]$ be a nonzero polynomial, the least positive integer e for which $f(x)$ divides $x^e - 1$, e is called the order of f, namely, $ord(f) = ord(f(x)) = e$. For the order of a polynomial, we have

- Let c be a positive integer, the polynomial $f \in \mathscr{R}[x]$ divides $x^c - 1$ iff $ord(f)$ divides c.
- If e_1 and e_2 are positive integers, the greatest common divisor of $x^{e_1} - 1$ and $x^{e_2} - 1$ is $x^d - 1$, d is the greatest common divisor of e_1 and e_2.
- $ord(f)$ is equal to the least common multiple of $ord(g_1)ord(g_2)\cdots ord(g_n)$ if $f = g_1, g_2 \cdots, g_n$; $g_i \in \mathscr{R}[x]$ is pairwise relatively prime polynomials.
- A polynomial is symmetric $f \in \mathscr{R}[x_1, x_2, \cdots, x_n]$ if $f(x_{i_1}, x_{i_2}, \cdots, x_{i_n}) = f(x_1, x_2, \cdots, x_n)$ for any permutation.

Exercises

Question 5.1. How to understand the differences between supervised learning, unsupervised learning, and reinforcement learning?

Question 5.2. What's Markov decision process? Why is reinforcement learning related to Markov decision process?

Question 5.3. Please explain what exploration and exploitation (EE) are. How does reinforcement learning seek the trade-off between exploration and exploitation ?

Question 5.4. What's Bellman equation? What's dynamic programming? What are the elementary concepts in Bellman equation?

Question 5.5. What's an episode in reinforcement learning?

Question 5.6. How is reinforcement learning related to gradient-descent methods?

Question 5.7. What are Q-learning and double Q-learning?

Question 5.8. What's linear programming? What's nonlinear programming? What's dynamic programming?

Question 5.9. What are modern optimization algorithms? Please list three of them.

Question 5.10. How to understand controllability, stabilizability, observability, optimality in modern mathematical control theory?

Question 5.11. What's the generalized Newton method for solving systems of equations?

References

1. Alpaydin E (2009) Introduction to machine learning. MIT Press
2. Avriel M (2003) Nonlinear programming: analysis and methods. Dover Publishing
3. Bania P, Baranowski J (2016) Kalman filter and its approximation. In: IEEE conference on decision and control (CDC), pp 2875–2880
4. Bazaraa M, Shetty C (1979) Nonlinear programming. John Wiley & Sons, Theory and Algorithms
5. Bellman R (1957) A Markovian decision process. J Math Mech 6(5):679–684
6. Busoniu L, Babuska R, De Schutter B, Ernst D (2010) Reinforcement learning and dynamic programming using function approximators. Taylor & Francis CRC Press

7. Burkhart M, Brandman D, Franco B, Hochberg L, Harrison M (2020) The discriminative Kalman filter for Bayesian filtering with nonlinear and non-Gaussian observation models. Neural Comput 32(5):969–1017
8. Burnetas A, Katehakis M (1997) Optimal adaptive policies for Markov decision processes. Math Oper Res 22:222–255
9. Dabney W et al (2020) A distributional code for value in dopamine-based reinforcement learning. Nat 577:671–675
10. Dantzig G, Thapa M (1997) Linear programming. Springer, New York
11. Ding C, Pei D, Salomaa A (1996) Chinese remainder theorem: applications in computing, coding, cryptography. World Scientific Publishing
12. Fishman GS (1995) Monte Carlo: concepts, algorithms, and applications. Springer, New York
13. Francois-Lavet V et al (2018) An introduction to deep reinforcement learning. Found Trends Mach Learn 11(3–4):219–354
14. George K, Bouffanais R (2019) Self-organizing maps for storage and transfer of knowledge in reinforcement learning. Adapt Behav 27(2):111–126
15. Gittins J (1979) Bandit processes and dynamic allocation indices. J R Stat Society Ser B (Methodol) 41(2):148–177
16. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press
17. Hu J, Niu H, Carrasco J, Lennox B, Arvin F (2020) Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. IEEE Trans Veh Technology 69(12):14413–14423
18. Humpherys J (2012) A fresh look at the Kalman filter. SIAM Review 54(4):801–823
19. Jones M, Peet M (2021) A generalization of Bellman's equation with application to path planning, obstacle avoidance and invariant set estimation. Automatica 127:109510
20. Julier SJ, Uhlmann J (1997) New extension of the Kalman filter to nonlinear systems. In: Signal processing, sensor fusion, and target recognition VI. Proceedings of SPIE, vol 3, pp 182–193
21. Kaelbling L, Littman M, Moore A (1996) Reinforcement learning: a survey. J Artif Intell Research 4:237–285
22. Kasabov N (1996) Foundations of neural networks, fuzzy systems, and knowledge engineering. MIT Press
23. Kline M (1986) Mathematics and the search for knowledge. Oxford University Press
24. Koller D, Friedman N (2009) Probabilistic graphical models. MIT Press, Massachusetts
25. Lee D, Seo H, Jung M (2012) Neural basis of reinforcement learning and decision making. Annu Rev Neurosci 35(1):287–308
26. Lidl R, Niederreiter H (1994) Introduction to finite fields and their applications. Cambridge University Press
27. Littman M (2015) Reinforcement learning improves behavior from evaluative feedback. Nat 521:445–451
28. Menegaz H, Ishihara J, Borges G, Vargas A (2015) A systematization of the unscented Kalman filter theory. IEEE Trans Autom Control 60(10):2583–2598
29. Michels K, Klawonn F, Kruse R, Nurnberger A (2006) Fuzzy control: fundamentals, stability and design of fuzzy controllers. Springer, Berlin
30. Mnih V et al (2015) Human-level control through deep reinforcement learning. Nat 518:529–533
31. Nisio M (2015) Stochastic control theory: dynamic programming principle. Springer
32. Rao S (2009) Engineering optimization: theory and practice, 4th edn. ISBN: 978-0-470-18352-6
33. Reddy G et al (2018) Glider soaring via reinforcement learning in the field. Nat 562:236–239
34. Russell S, Norvig P (2020) Artificial intelligence: a modern approach, 4th edn. Pearson Education
35. Sengupta A (2012) Representing finite groups: a semisimple introduction. Springer
36. Sutton R, Barto A (2018) Reinforcement learning: an introduction, 2nd edn. MIT Press

37. van Otterlo M, Wiering M (2012) Reinforcement learning and Markov decision processes. In: Reinforcement learning. Adaptation, learning, and optimization, vol 12, pp 3–42
38. Vinyals Q et al (2019) Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nat 575:350–354
39. Wang L (2012) iNavigation: an image based indoor navigation system. Master's thesis, Auckland University of Technology, New Zealand
40. Wang E, Yan W (2014) iNavigation: an image based indoor navigation system. Multimed Tools Appl 73:1597–1615
41. Williams R (1987) A class of gradient-estimating algorithms for reinforcement learning in neural networks
42. Zabczyk J (1995) Mathematical control theory: an introduction. Birkhauser, Berlin
43. Zarchan P, Musoff H (2000) Fundamentals of Kalman filtering: a practical approach. American Institute of Aeronautics and Astronautics, Incorporated

Manifold Learning and Graph Neural Network

6

In this chapter, we will introduce manifold learning and graph neural networks. We hope to introduce graphical probability models as the starting point of basestone. We need to introduce our readers why we should study graphs, what we can benefit from the graphs. Furthermore, we will introduce graph neural networks (GNN) and how to combine GNN with manifold learning together.

6.1 Manifold Learning

Manifold alignment [36] is a class of machine learning algorithms that produce projections between sets of data, given the original datasets on a manifold. Manifold learning, which is emphasized on infinity continuity and was originated from differential geometry, has been applied to nonlinear dimensionality reduction in machine learning.

Manifold [60,68,70] is a generalization of curve and surface, a line is 1D manifold, a curve is 2D manifold, a surface is 3D manifold, n-dimensional manifold is n-manifold. In manifold, *chart* is an important concept in Euclidean space, which is related to neighborhood. *Atlas* is a local Euclidean space, a collection of topology with continuity of infinity. If a manifold is smooth, we call it as smooth manifold. Manifolds include analytic manifolds, complex manifolds, Euclidean manifolds, topological manifolds, etc.

In manifold, if the left continuity of function $f(x) \in [a, b]$, $x \in \mathcal{R}$ is equal to the right continuity, namely,

$$\lim_{x \to x_0} f(x) = \lim_{x \to x_0^+} f(x) = f(x_0^+) = f(x_0^-) = \lim_{x \to x_0^-} f(x) = f(x_0), \quad (6.1)$$

where C^k is continuous if $f^k(x)$ with derivatives of order $k \in \mathcal{Z}^+$. C^∞ is continuous, that means, k tends to infinity, i.e., $k \to \infty$.

A real-valued function $f : \mathbf{U} \to \mathscr{R}^n$ is $C^k (k \to \infty)$ at $p \in \mathbf{U}$ if its partial derivatives $\frac{\partial^j f}{\partial x_1 \cdots \partial x_j}$ of all orders $j \leq k$ exist and are continuous at $p \in \mathbf{U}$, $\mathbf{U}$ is an open set or neighborhood [60], typically, it is a tangent vector field.

Topological manifold [36] is defined in Hausdorff space with Euclidean distance. Topological space is countable if it has a base, n-manifold has the base with a dimension n, which is countable. Thus, the topological manifold is countable.

We call the pair $(\mathbf{U}, \phi : \mathbf{U} \to \mathscr{R}^n)$ is a chart, $\mathbf{U}$ is a coordinate neighborhood or a coordinate open set, ϕ is a coordinate map or a coordinate system on $\mathbf{U}$.

Two charts have a compatible relationship, if we have invertible functions $\Psi^{-1}(\cdot)$ and $\Phi^{-1}(\cdot)$, $x \in C_1$ and $y \in C_2$, $y = \Phi(x)$, $\Phi^{-1}\Phi(x) = x$, $\Psi^{-1}\Psi(x) = x$, $\Psi^{-1}\Phi\Psi(x) = \Phi(x)$ and $\Phi^{-1}\Psi\Phi(x) = \Psi(x)$.

Homomorphism is a mapping which keeps the relationship

$$\Phi(u \cdot v) = \Phi(u) \circ \Phi(v), \forall u, v \in \mathbf{A}, \tag{6.2}$$

where '$\cdot$' is the operation on X and '$\circ$' is the operation on Y, $\forall u, v \in X$ and $\Phi(u), \Phi(v) \in Y$. This equation indicates that before and after the homomorphism mapping, the defined operations are reserved.

Manifold is homomorphism that means the manifold is defined on a continuous domain. Riemann manifold is a smooth manifold based on derivatives or tangent vectors that is a differentiable manifold in which each tangent space is equipped with an inner product. The center of manifold is defined as a set $\mathbf{C} = \{x : ax = xa = 0, x \in \mathbf{K}, a \in \mathbf{A}\}$. A manifold is a homomorphism if there is an open neighborhood $\mathbf{N}(x)$, $\forall x \in \mathscr{R}^d$, then $f : \mathbf{N}(x) \to \mathscr{R}^d$.

Let manifolds $\mathbf{N} \subset \mathscr{R}^n$ and $\mathbf{M} \subset \mathscr{R}^n$ be manifolds of dimension $n \in \mathscr{Z}^+$ and $m \in \mathscr{Z}^+$, the continuous mapping $f : \mathbf{N} \to \mathbf{M}$ is C^∞ if it is C^∞ at every point of $\mathbf{N}$. If $f : \mathbf{N} \to \mathbf{M}$ and $g : \mathbf{M} \to \mathbf{P}$ are C^∞ maps of manifolds, then the composite $g \circ f : \mathbf{N} \to \mathbf{P}$ is C^∞. Moreover, let $f : \mathbf{N} \to \mathbf{M}$ be a C^∞ map between two manifolds of the same dimension, $p \in \mathbf{N}$. Then, $f(\cdot)$ is locally invertible at p if and only if (i.e., iff) its Jacobian determinant $\det[\frac{\partial f^j(p)}{\partial x_1 \cdots \partial x_j}]$ is nonzero.

In most manifold learning algorithms, we assume that the input data resides on or is close to a low-dimensional manifold embedded in the ambient space. Manifold learning has been applied to medical image processing [70], data compression, data dimensionality reduction, noise removal, etc. PCA (i.e., Principle Component Analysis) is a linear dimensionality reduction method, meanwhile manifold learning is for nonlinear dimensionality reduction which could be applied to noise removal. The data dimensionality reduction has been applied to resolve the "curse of dimensionality" problem.

PCA refers to an orthogonal linear transformation that transforms the given data to a new coordinate system so that the greatest variance by using scalar projection of the given data replies on the principal components. The principal component decomposition of a raw vector $\mathbf{x}$ is given as

$$\mathbf{b} = \mathbf{x}\mathbf{A}, \tag{6.3}$$

where $\mathbf{A} = (a_{ij})_{p \times p}$ is a weight matrix, $p \in \mathscr{Z}$, $a_{ij} \in \mathscr{R}$. The transformation maps a data vector $\mathbf{x} = (x_{ij})_{1 \times p}$ from an original space to a vector $\mathbf{b} = (b_{ij})_{1 \times p}$ in new space, $x_{ij}, b_{ij} \in \mathscr{R}$. If $0 < l < p, l, p \in \mathscr{Z}$, we have another mapping,

$$\mathbf{b}_l = \mathbf{x} \mathbf{A}_l, \tag{6.4}$$

where $\mathbf{A}_l = (a_{ij})_{p \times l}$, $\mathbf{b}_l = (b_{ij})_{1 \times l}$, $a_{ij}, x_{ij}, b_{ij} \in \mathscr{R}$. We expect to minimize the squared reconstruction error,

$$\varepsilon = \|\mathbf{x} - \mathbf{x}_l\|_2^2 > 0, \tag{6.5}$$

where $\mathbf{x}_l$ has a lower dimension than the vector $\mathbf{x}$, namely, $l < p \in \mathscr{Z}^+$. Hence, the principal components of the original data are preserved, and the dimension of the raw data is reduced.

In order to implement dimensionality reduction by using PCA through the covariance method, we seek the eigenvectors and eigenvalues of the covariance matrix

$$\mathbf{B} = \mathbf{X} - \mathbf{h}\mathbf{u}^\top, \tag{6.6}$$

where $\mathbf{X} = (x_{ij})_{n \times p}$ is a matrix consisting of n given vectors $\mathbf{x}_i = (x_{ij})_{1 \times p}$, $\mathbf{h} = (h_{ij})_{1 \times n}$, $h_{ij} = 1$ and $\mathbf{u} = (u_{ij})_{1 \times p}$,

$$u_{1,j} = \frac{1}{n} \sum_{i=1}^{n} x_{i,j}, \tag{6.7}$$

where $j = 1, 2, \ldots, p, i = 1, 2, \ldots, n, x_{ij}, u_{i,j} \in \mathscr{R}$. Thus, we have the covariance matrix $\mathbf{C}$ from matrix $\mathbf{B}$ and,

$$\mathbf{C} = \frac{1}{n-1} \mathbf{B}^* \mathbf{B}, \tag{6.8}$$

where '*' is the conjugate transpose operator. Thus,

$$\mathbf{V}^{-1} \mathbf{C} \mathbf{V} = \Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_n), \tag{6.9}$$

where $\mathbf{V}$ is the matrix consisting of eignvectors, Λ is the diagonal matrix of eigenvalues of $\mathbf{C}$, $\lambda_i \neq 0, i = 1, 2, \ldots, n$ are eignvalues. Namely, the eigenvalues satisfy the characteristic polynomial

$$f(\lambda) = |\mathbf{C} - \lambda\mathbf{I}| = 0, \tag{6.10}$$

where $\mathbf{I}$ is the identity matrix. We sort the eigenvalues in decreasing order, thus the order of corresponding eigenvectors is swapped. We thus select the principal components using Eq. (6.11)

$$\delta = \frac{\sum_{i=1}^{l} |\lambda_i|}{\sum_{i=1}^{p} |\lambda_i|} > 0, (p \geq l > 0). \tag{6.11}$$

If $\delta > 0.90$ for example, $\lambda_i, i = 1, 2, \ldots, l$ $(p \geq l > 0)$ are the main components of matrix $\mathbf{C}$.

$$\mathbf{V}_l^{-1} \mathbf{C}_l \mathbf{V}_l = \Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_l), \tag{6.12}$$

where $\mathbf{V}_l$ is the matrix consisting of the eignvectors corresponding to eignvalues $\lambda_1, \lambda_2, \ldots, \lambda_l$. Correspondingly, we have found $l \in \mathscr{Z}^+$, which satisfies $\mathbf{b}_i = \mathbf{x}\mathbf{C}_l$, $\forall \mathbf{x}$.

In manifold learning, we assume that we always are able to find the lower dimensional data using dimensionality reduction. We assume the lowest dimensional data is hidden or embedded in noisy data. This is our base point or hypnosis of using manifolds to machine learning, or directly we call it as manifold learning.

In manifold learning, we construct relationship matrix between nodes or vertices x_i and x_j, $i, j = 1, 2, \ldots, n$ of a given graph G, $\mathbf{W} = \{w_{ij}\}$. Given a dataset with training samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, $\mathbf{x}_i \in \mathscr{R}^d$, $G = < \mathbf{X}, \mathbf{W} >$ is an undirected graph, $\mathbf{W} = (w_{ij})_{n \times n}$ is the similarity or affinity matrix, $w_{ij} \in [0, 1]$

$$w_{ij} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\gamma_i \gamma_j}), \tag{6.13}$$

where $\gamma_i = \|\mathbf{x}_i - \mathbf{x}_j\|$, $i \neq j$ is the local scale of data samples in the neighborhood of $\mathbf{x}_i$. $\mathbf{x}_i$ is k-nearest neighbor of $\mathbf{x}_j$ $(i \neq j)$. Graph Laplacian is

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{6.14}$$

where $\mathbf{D} = (d_{ij})_{n \times n}$, $d_{ii} = \sum_j w_{ij}$, $\forall i = 1, 2, \ldots$.

For dimensionality reduction of data samples $\{\mathbf{y}_i\}_{i=1}^n$, $\mathbf{y}_i \in \mathscr{R}^D$, $D \gg d$, the eigens or spectrum-based method is

$$\mathbf{L}\mathbf{y} = \lambda \mathbf{B}\mathbf{y}, \tag{6.15}$$

where $\mathbf{y}\mathbf{B}\mathbf{y}^\top = \mathbf{I}$, $\mathbf{I}$ is the identify matrix. Thus, we get $\mathbf{y}^*$ by using

$$\mathbf{y}^* = \arg\min_{\mathbf{y}\mathbf{B}\mathbf{y}^\top = \mathbf{I}} \mathbf{y}\mathbf{L}\mathbf{y}^\top. \tag{6.16}$$

Given a graph, a matrix of edge weights $\mathbf{x}_i \in \mathscr{R}^d$, $\mathbf{W} = (w_{ij})_{n \times n}$,

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{6.17}$$

where $\mathbf{D} = (d_{ii})_{n \times n}$, $d_{ii} = \sum_j w_{ij}$,

$$w_{ij} = \begin{cases} \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}) & x_j \in \mathbf{N}(i) \\ 0 & \text{Others.} \end{cases}, \tag{6.18}$$

where w_{ij} is Gaussian kernel, $\mathbf{N}(i)$ is the neighborhood of x_i.

$$\mathbf{L}\mathbf{y} = \lambda \mathbf{D}\mathbf{y}, \tag{6.19}$$

where $\mathbf{Y} = (\mathbf{y}_i)_n$ is the output.

An example of manifold learning is available from the site: https://scikit-learn.org. Python has been applied to implement the manifold methods, the examples show how to reduce the data dimension of a Swiss roll using manifold learning (Fig. 6.1).

In contrast, a MATLAB example is available to demonstrate the Laplacian eigenmap in manifold learning at: https://www.mathworks.com/matlabcentral/fileexchange/36141-laplacian-eigenmap-diffusion-map-manifold-learning. The result is shown in Fig. 6.2 for the purpose of recovering low-dimensional geometries.
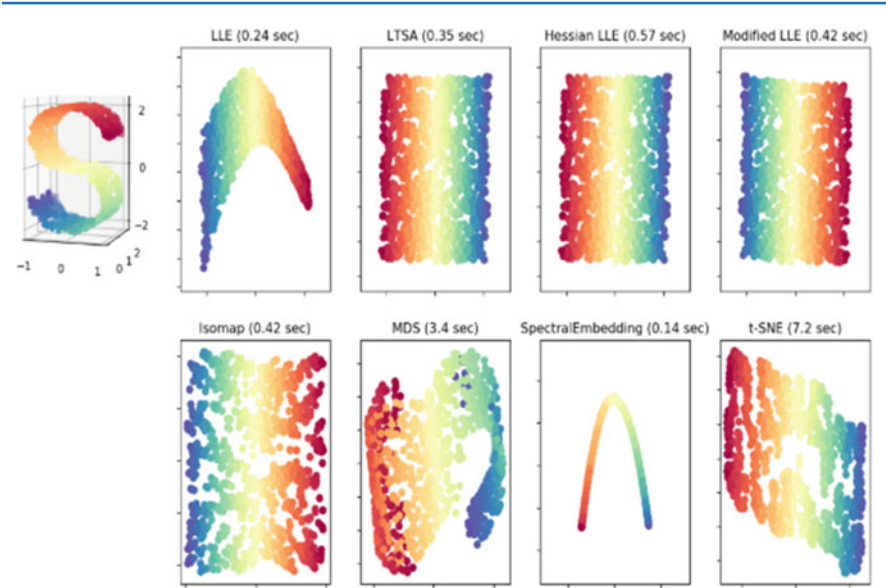
Fig. 6.1 Python examples for dimensionality reduction of Swiss roll by using manifold learning
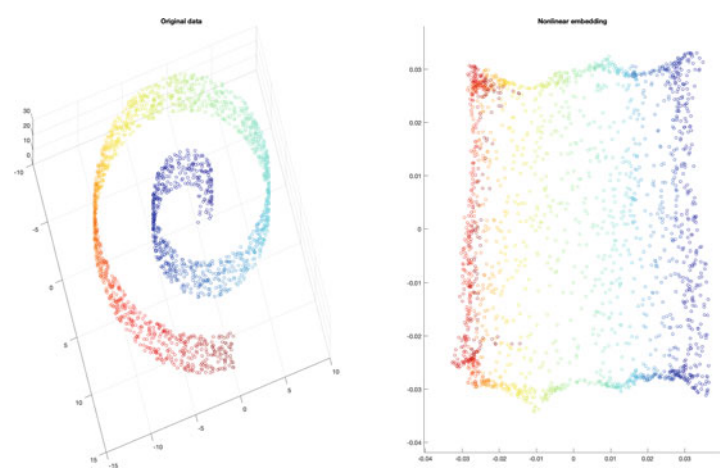
Fig. 6.2 MATLAB examples for the Laplacian eigenmap method of Swiss roll by using manifold learning

The aim of manifold learning for dimensionality reduction is to find a mapping from high-dimensional manifold to a low-dimensional space so as to represent the data points in high-dimensional space and achieve the purpose of data dimensionality reduction. By introducing metric to learn the inter-sample distance in each gait manifold space, we have implemented a supervised LLE dimensionality reduction method to generate a low-dimensional gait manifold. A pedestrian identification network is proposed and trained, which identifies a query person by comprehensively considering the similarity between gait and its gait manifold [64].

6.2   Probabilistic Graphical Models

Graphical model exists everywhere, it is not deep learning, but it is much wider than deep learning models. There are two types of graphs: Directed graphs and undirected graphs. Bayesian networks are directed networks, Markov random field (MRF) is a typical undirected model. In fact, we have template-based graphical models, we need create and fill up our content there. The template-based models are general, not specific and concrete. Meanwhile, generative model is to create a new model while discriminative model is to adjust or modify the model to suit our requirements. Hybrid models are to combine all of these models together.

Our variables in a graph include target variables (output), observable variables (input), and latent variables (hidden). Inferences include exact inferences and approximate inferences. Uncertainty is the typical concept in machine learning. The graphical inferences can help us find from what we know to infer what we do not know. Inferences could assist us to test models and find the sensitivities and errors.

Why do we study graphical models? Graphical model is a simple way to visualize the structure of a probabilistic model which can be applied to design new models [29]. In graphical model, we have various presentations, we need to find which model is explainable and which is the best one for solving our given problem.

In graphical model, we have parameter learning, feature learning, and knowledge learning. These are related to knowledge discovery. Latency refers to the hidden variables that we do not know in a graph.

Graphical model insights into the properties of a model, including conditional independence properties, which can be obtained by inspection of the graph. Complex computations, required to perform inference and learning in sophisticated models, are expressed in terms of graphical manipulations, in which underlying mathematical expressions are carried out implicitly.

Bayes' theorem is the base stone of pattern classification [18]. From the prior probability, likelihood, and evidence, we infer the posterior probability for a class. Bayes' theorem is shown as Eq. (6.20).

$$P(c_i|O) = \frac{P(O|c_i)P(c_i)}{P(O)} \tag{6.20}$$

where O is observation, c_i refers to class $i \in \mathscr{Z}$, $P(O) \in [0, 1]$ is the normalization factor or probability of observations, $P(O|c_i) \in \mathscr{R}$ is the probability of observation

Fig. 6.3 An example of influence diagram
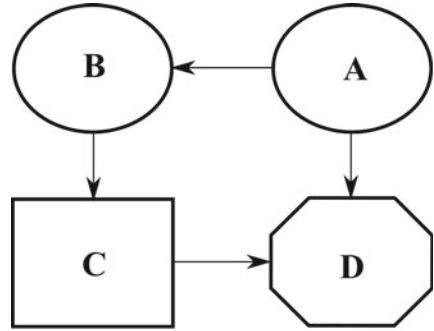
given class i or likelihood probability, $P(c_i)$ is priori probability of class i, $P(c_i|O)$ is the probability of class i given observation or posterior probability. The joint probability is $P(c_i, O)$ as shown in Eq. (6.21).

$$P(c_i, O) = P(c_i|O)P(O) = P(O|c_i)P(c_i). \qquad (6.21)$$

Bayesian model is a simple yet highly effective method for pattern classification in machine learning. The joint probability of multiple variables is

$$P(G, S, R) = P(G|S, R) \cdot P(S|R) \cdot P(R). \qquad (6.22)$$

Meanwhile, the conditional probability of multiple variables is

$$P(G|R) = \frac{P(R, G)}{P(R)} = \frac{P(G, S, R)}{P(S, R)}, \qquad (6.23)$$

where S is a latent variable, given the joint probabilities $P(S, R) \in [0, 1]$ and $P(G, S, R) \in [0, 1]$.

Naïve Bayesian model is a family of simple probabilistic classifiers based on applied Bayes' theorem with strong (Naïve) assumption between the features, typically the classification only has two classes labeled as "Yes" (e.g., +1/True) or "No" (e.g., -1/False). Naïve Bayesian model has been applied to automatically classify spam emails using the special words appeared in the emails.

Influence diagram [13,45] as shown in Fig. 6.3 is a decision-theoretic diagram for making decision under uncertainty. Influence diagram is also called relevance diagram, decision diagram or decision network [23]. It is a generalization of a Bayesian network, in which not only probabilistic inference problems but also decision-making problems will be solved. Usually, uncertainty node is shown as the shape oval, e.g., A and B in Fig. 6.3, decision node is represented by using rectangles, e.g., C in Fig. 6.3, and value node is marked as octagons, e.g., D in Fig. 6.3. Influence diagram as a Bayesian network is a directed acyclic graph (DAG).

Markov random field (MRF) [31] is undirected graphs with conditional probability distributions. Factor graphs are encompassed in both Bayesian networks and Markov networks. Factorization is a product of factors over cliques in the graph [29].

$$P(x) = \frac{1}{Z} \exp \sum_k \sum_{i=1}^{N_k} w_{ki} f_{ki}(x_{\{k\}}), \, x \in X \qquad (6.24)$$

and

$$Z = \sum_{x \in \mathscr{X}} \exp \sum_{k} \sum_{i=1}^{N_k} w_{ki} f_{ki}(x_{\{k\}}), \tag{6.25}$$

where $Z(\cdot)$ is called partition function, w_{ki} is the weights between cliques, f_{ki} is the link between clique $k \in \mathscr{Z}^+$ and clique $i \in \mathscr{Z}^+$.

A distribution P_Φ is a Gibbs distribution parameterized by using a set of factors $\Phi = (\phi_1(D_1), \ldots, \phi_k(D_k))$, then

$$\Phi = \phi_1(D_1) \times \cdots \times \phi_m(D_m) = \prod_{i=1}^{m} \phi_i(D_i). \tag{6.26}$$

Therefore,

$$P_\Phi = \frac{1}{Z}\phi_1(D_1) \times \cdots \times \phi_m(D_m) = \frac{1}{Z}\Phi \tag{6.27}$$

where $Z = \sum_{X_1, X_2, \ldots, X_n} \phi_1(D_1) \times \cdots \times \phi_m(D_m)$ is a normalizing constant.

A conditional random field (CRF) is an undirected graph [10, 69], the network is annotated with a set of factors $\phi_1(D_1), \ldots, \phi_m(D_m)$, a conditional distribution is

$$P(Y|X) = \frac{1}{Z} \prod_{i=1}^{m} \phi_i(Y_i, Y_{i+1}), \tag{6.28}$$

and

$$Z = \sum_{Y} \prod_{i=1}^{m} \phi_i(X_i, Y_i). \tag{6.29}$$

A CRF over $X = \{X_1, X_2, \ldots, X_n\}$, $Y = \{0, 1\}$,

$$\phi_i(X_i, Y) = \exp(w_i \mathbf{I}(X_i = 1, Y = 1)), \tag{6.30}$$

and

$$P(Y = 1|x_1, \ldots, x_k) = \sigma(w_0 + \sum_{i=1}^{k} w_i x_i), \tag{6.31}$$

where $\sigma(\cdot)$ is a sigmoid function.

Logistic CPD is

$$P(Y = 1|X_1, \ldots, X_n) = \sigma(w_0 + \sum_{i=1}^{N} w_i), \tag{6.32}$$

where $\sigma(\cdot)$ is a sigmoid function. Logistic distributions only have two labels: "-1" and "+1".

Linear or multivariate Gaussian distribution is

$$p(Y|\mathbf{x}) = \mathbf{N}(\beta_0 + \beta \mathbf{x}; \sigma^2), \tag{6.33}$$

where $\mathbf{N}(\cdot)$ is Gaussian distribution.

Conditional Bayesian network is

$$P(Y|X) = \sum_{Z} P(Y, Z|X) = \prod_{X \in Y \bigcup Z} P(X|P_X). \tag{6.34}$$

Multivariate Gaussian distribution is

$$P(X) = \frac{1}{(2\pi)^{(n/2)}|\Sigma|^{1/2}} e^{(X-\mu)^\top \Sigma^{-1}(X-\mu)}. \tag{6.35}$$

A joint normal distribution over $\{X, Y\}$ is $P(X, Y) \sim \mathbf{N}(\mu, \Sigma)$,

$$\mu_{(n+m)\times 1} = \begin{pmatrix} (\mu_X)_{n\times 1} \\ (\mu_Y)_{m\times 1} \end{pmatrix} \tag{6.36}$$

and

$$\Sigma_{(n+m)\times(n+m)} = \begin{pmatrix} (\Sigma_{XX})_{n\times n} & (\Sigma_{XY})_{n\times m} \\ (\Sigma_{YX})_{m\times n} & (\Sigma_{YY})_{m\times m} \end{pmatrix}. \tag{6.37}$$

For Gaussian Bayesian networks, if

$$p(Y|x) \sim \mathbf{N}(\beta_0 + \beta^\top x; \sigma^2), \tag{6.38}$$

then

$$p(Y) \sim \mathbf{N}(\mu_Y; \sigma_Y^2), \tag{6.39}$$

$$\mu_Y = \beta_0 + \beta^\top x, \tag{6.40}$$

and

$$\sigma_Y^2 = \sigma^2 + \beta^\top \Sigma \beta. \tag{6.41}$$

The conditional density is

$$P(Y|X) \sim \mathbf{N}(\beta_0 + \beta^\top X; \sigma^2), \tag{6.42}$$

where

$$\beta_0 = \mu_Y \Sigma_{YX} \Sigma_{XX}^{-1} \mu_X, \tag{6.43}$$

and

$$\beta = \Sigma_{XX}^{-1} \Sigma_{YX}, \tag{6.44}$$

and

$$\sigma^2 = \Sigma_{YY} - \Sigma_{YX} \Sigma_{XX} \Sigma_{XY}. \tag{6.45}$$

Gaussian distribution is

$$P(x) = \frac{1}{\sqrt{2\pi}} \exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}, \tag{6.46}$$

we generalize it as

$$P(x) = \frac{1}{Z(\mu, \sigma^2)} \exp(< t(\theta), \tau(x) >) \tag{6.47}$$

where $\tau(x) =< x, x^2 >$ is inner product,

$$t(\mu, \sigma^2) =< \frac{\mu}{\sigma}, -\frac{1}{2\sigma} >, \tag{6.48}$$

and

$$Z(\mu, \sigma^2) = \sqrt{2\pi}\sigma \exp(\frac{\mu^2}{2\sigma^2}). \tag{6.49}$$

Linear exponential family is

$$P_\theta(x) = \frac{1}{Z(\theta)} \exp(< t(x), \theta >), \tag{6.50}$$

where

$$\Theta = \{\theta \in \mathbf{R}^k, \int \exp(< t(x), \theta >) dx < \infty\}. \tag{6.51}$$

The exponential factor family is

$$\Phi_\theta(x) = A(x) \exp(< t(\theta), \tau(x) >), \tag{6.52}$$

and

$$P_\theta(x) \propto \prod_i \phi_{\theta_i}(x) = \prod_i A_i(x) \exp(\sum_i < t_i(\theta_i), \tau_i(x) >). \tag{6.53}$$

Bayesian networks are denoted in the corresponding way,

$$P(x|u) = \exp(t_{P(\mathbf{X}|\mathbf{U})}(\theta), \tau_{P(\mathbf{X}|\mathbf{U})}(x, u)). \tag{6.54}$$

Entropy is

$$\mathrm{H}(X) = \ln Z(\theta) - < \mathrm{E}(\tau(X)), t(\theta) > . \tag{6.55}$$

Relative entropy is

$$\mathrm{D}(P_{\theta_1}||P_{\theta_2}) = \mathrm{E}_{P(\theta_1)}[\ln(\frac{P_{\theta_1}(\mathscr{X})}{P_{\theta_2}(\mathscr{X})})] = -\ln \frac{Z(\theta_1)}{Z(\theta_2)} + < \mathrm{E}_{P(\theta_1)}(\tau(\mathscr{X})), t(\theta_1) - t(\theta_2) > . \tag{6.56}$$

Information projection is

$$Q^I = \arg\min_{Q \in \mathscr{Q}} D(Q||P). \tag{6.57}$$

Moment projection is

$$Q^M = \arg\min_{Q \in \mathscr{Q}} D(P||Q). \tag{6.58}$$

If G_ϕ is an empty graph, $Q^M = \arg\max_{Q \in G_\phi} D(P|Q)$, then,

$$Q^M(X_1, X_2, \ldots, X_n) = P(X_1) P(X_2) \cdots P(X_n). \tag{6.59}$$

Hence,

$$D(P||Q) = -\mathrm{H}_P(\mathscr{X}) + \mathrm{E}_P[\ln Q(\mathscr{X})] \geq D(P|Q^M). \tag{6.60}$$

Furthermore, if and only if $Q_i(X) = P_i(X)$, then,

$$D(P|Q) = D(P|Q^M), \tag{6.61}$$

i.e., $Q = Q^M$, additionally,

$$D(P|Q_\theta) - D(P|Q_{\theta'}) = D(Q_{\theta'}||Q_\theta) \geq 0. \tag{6.62}$$

6.3 Boltzmann Machine

Hopfield networks [22,37,49] are a form of recurrent artificial neural network and a type of spin glass system. With regard to this network, on a equidistance circle, we link all the nodes together. Boltzmann machines [2] are seen as the stochastic and generative counterpart of Hopfield nets (Fig. 6.4).

Boltzmann machine is a general "connectionist" approach to learn arbitrary probability distributions over binary vectors. For a d-dimensional binary random vector $\mathbf{x} \in \{0, 1\}^d$, an energy-based model is

$$P(\mathbf{x}) = \frac{\exp(-E(\mathbf{x}))}{Z}, \tag{6.63}$$

and

$$E(\mathbf{x}) = -\mathbf{x}\mathbf{U}^\top\mathbf{x} - \mathbf{b}^\top\mathbf{x}, \tag{6.64}$$

where $E(\mathbf{x})$ is the energy function, $Z(\cdot)$ is the partition function, $\sum_{\mathbf{x}} P(\mathbf{x}) = 1$, $\mathbf{U}$ is the weight parameters, $\mathbf{x}$ is bias parameters. Units $\mathbf{x}$ could be decomposed into visible $\mathbf{v}$(visible) and hidden (latent) $\mathbf{h}$,

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^T\mathbf{R}\mathbf{v} - \mathbf{v}^T\mathbf{W}\mathbf{h} - \mathbf{h}^T\mathbf{S}\mathbf{h} - \mathbf{b}^T\mathbf{v} - \mathbf{c}^T\mathbf{h}. \tag{6.65}$$

Restricted Boltzmann machines (RBM) [14] further restrict Boltzmann machines to those without visible-visible and hidden-hidden connections. A deep Boltzmann machine (DBM) is a type of binary pairwise MRFs (i.e., undirected probabilistic graphical model) with multiple layers of hidden random variables [7,31]. The global energy E in a Boltzmann machine is identical in the form to that of a Hopfield network. Mathematically,

$$E \overset{\Delta}{=} -\left(\sum_{i<j} w_{ij}s_i \cdot s_j + \sum_i \theta_i \cdot s_j\right), \tag{6.66}$$

Fig. 6.4 An example of the fully-connected modern Hopfield network, the weight matrix is symmetric
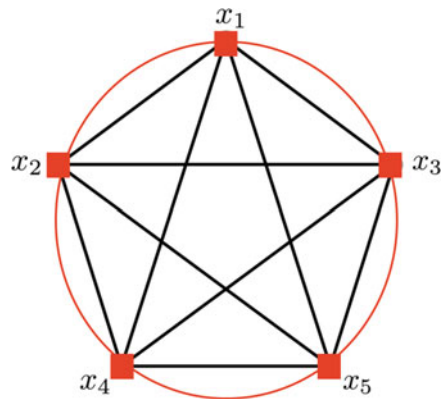
where

- w_{ij} is the connection strength between unit j and unit i.
- s_i is the state of unit i, $s_i \in \{0, 1\}$.
- θ_i is the bias of unit i in the global energy function.
- w_{ij} is represented as a symmetric matrix $\mathbf{W} = (w_{ij})_{N \times N}$, with zeros along the diagonal.
- The probability of the i-th unit is

$$p_i \overset{\Delta}{=} \frac{1}{1 + \exp(-\frac{E_i}{T})}. \tag{6.67}$$

RBM [14] is intractable and bipartite graph, its energy function is

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^\top \mathbf{W} \mathbf{h} - \mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h}. \tag{6.68}$$

Therefore,

$$P(\mathbf{v} = v, \mathbf{h} = h) = \frac{\exp(-E(\mathbf{v}, \mathbf{h}))}{Z}, \tag{6.69}$$

and

$$Z = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})). \tag{6.70}$$

A restricted Boltzmann machine (RBM) [14] is a generative neural network that is able to learn a probability distribution from its set of inputs. The energy function from product of expert (POE) [29] is

$$E(v, h) = -\sum_{ij} w_{ij} h_i v_j - \sum_j b_j v_j + \sum_i c_i h_i. \tag{6.71}$$

The probabilities are

$$p(v) \overset{\Delta}{=} \frac{\sum_h e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}}, \tag{6.72}$$

$$p(h) \overset{\Delta}{=} \frac{\sum_v e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}}, \tag{6.73}$$

and

$$p(v, h) \overset{\Delta}{=} \frac{e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}}, \tag{6.74}$$

hence,

$$p(v|h) = \frac{e^{-E(v,h)}}{\sum_v e^{-E(v,h)}}. \tag{6.75}$$

The loss function is

$$L(\theta) = \prod_v L(\theta|v) = \prod_v p(v), \theta = (\mathbf{W}, \mathbf{b}, \mathbf{c}). \tag{6.76}$$

The derivatives are

$$\frac{\partial L(\theta)}{\partial \theta} = \sum_v \frac{\partial \ln L(\theta|v)}{\partial \theta} = \sum_v \frac{\partial \ln p(v)}{\partial \theta}, \tag{6.77}$$

$$\ln p(v) = \ln(\sum_h e^{-E(v,h)}) - \ln(\sum_{v,h} e^{-E(v,h)}), \tag{6.78}$$

and

$$\frac{\partial L(\theta)}{\partial \theta} = E_{p(h|v)}(-\frac{\partial E(v,h)}{\partial \theta}) - E_{p(h,v)}(-\frac{\partial E(v,h)}{\partial \theta}). \tag{6.79}$$

The energy function from the product of expert (POE) [29] is

$$E(v,h) = -\sum_{ij} w_{ij} h_i v_j - \sum_j b_j v_j + \sum_i c_i h_i, \tag{6.80}$$

$$\frac{\partial \ln p(v)}{\partial w_{ij}} = p(h_i = 1|v)v_j - \sum_v p(v)p(h_i = 1|v)v_j, \tag{6.81}$$

$$\frac{\partial \ln p(v)}{\partial b_j} = v_j - \sum_v p(v)v_j, \tag{6.82}$$

and

$$\frac{\partial \ln p(v)}{\partial c_i} = p(h_i = 1|v) - \sum_v p(v)p(h_i = 1|v). \tag{6.83}$$

A DBM is an energy-based model

$$P(v,h) = \frac{1}{Z(\theta)} \exp(-E(v, h^{(1)}, h^{(2)}, h^{(3)}, h^{(4)}; \theta)), \tag{6.84}$$

where $E(\cdot)$ is the energy function, v is visible layer, $h^{(i)}$ are hidden layers, $i = 1, 2, 3$,

$$E(v, h; \theta) = -v^\top W^{(1)} h^{(1)} - h^{(1)\top} W^{(2)} h^{(2)} - h^{(2)\top} W^{(3)} h^{(3)}. \tag{6.85}$$

In the case with two hidden layers,

$$P(v_i = 1|h^{(1)}) = \sigma(W^{(1)}_{i,:} h^{(1)}), \tag{6.86}$$

$$P(h^{(1)}_i = 1|v, h^{(2)}) = \sigma(v^\top W^{(1)}_{:,i} + W^{(2)}_{i,:} h^{(2)}), \tag{6.87}$$

and

$$P(h^{(2)}_k = 1|h^{(1)}) = \sigma(h^{(1)} W^{(2)}_{:,k}). \tag{6.88}$$

Restricted Boltzmann machines (RBMs) are created for unsupervised learning. The aim of RBMs is to find patterns in data by reconstructing the inputs using only two layers (i.e., visible layer and hidden layer). MATLAB and Python source codes for implementing RBMS all are available. An example from the site: https://scikit-learn.org/ shows how to improve the classification accuracy using a RBM.

A deep Boltzmann machine (DBM) is a type of binary pairwise MRFs with multiple layers of hidden random variables, which is a network of symmetrically coupled stochastic binary units. The probability assigned to vector **v** is

$$P(\mathbf{v}) = \frac{1}{Z} \sum_h e^{[\sum_{ij}(W_{ij}^{(1)} v_i h_j^{(1)}) + \sum_{jl}(W_{jl}^{(2)} h_j^{(1)} h_l^{(2)}) + \sum_{lm}(W_{lm}^{(3)} h_l^{(2)} h_m^{(3)})]}, \qquad (6.89)$$

where $\mathbf{h} = \{h^{(1)}, h^{(2)}, h^{(3)}\}$ is the set of hidden units, $\mathbf{W} = \{W^{(1)}, W^{(2)}, W^{(3)}\}$ is the model parameters, $p(h_i|v)$ and $p(v_i|h)$ is independent,

$$p(h|v) = \sigma\left(\sum_j w_{ij} \cdot v_j + c_i\right), \qquad (6.90)$$

where $\sigma(x) = 1/(1 + e^{-x})$ and

$$p(h|v) = \prod_i p(h_i|v), \; p(v|h) = \prod_i p(v_i|h). \qquad (6.91)$$

With regard to DBM, if $p(h_i|v)$ and $p(v_i|h)$ are independent, then $p(h|v) = \prod_i p(h_i|v)$ or $p(v|h) = \prod_i p(v_i|h)$.

DBMs are understood as multilayer perceptron with restricted Boltzmann machines. Deep belief nets are thought as Bayesian belief nets [20,55] with deep Boltzmann machines. In machine learning,

$$\mathbf{Y} = F(\mathbf{X}, \theta), \qquad (6.92)$$

where θ is the parameter vector, $\mathbf{X}$ is the input, $\mathbf{Y}$ is the labeled dataset.

$$E_A(\theta) = \frac{1}{N} \sum_{i=1}^N E(x_i, d_i, \theta), \qquad (6.93)$$

where $\{(x_i, d_i)\}$ is the training set, $x_i \in \mathbf{X}$, $d_i \in \mathbf{D}$.

$$\theta_{k+1} := \theta_k - \varepsilon \cdot \frac{\partial E(\theta)}{\partial \theta}, k = 1, 2, \dots \qquad (6.94)$$

6.4   Graph Neural Networks

6.4.1   Machine Learning on Graphs

Graphs are a ubiquitous data structure and a universal language for describing complex systems [29]. In the most general view, a graph is a collection of objects (i.e., nodes), along with a set of interactions (i.e., edges) between pairs of these objects. The power of graph formalism lies both in its focus on relationships between points, as well as in its generality, rather than the properties of individual points. Historically, the famous four color problem is thought as one of the most stimulating problems in graph theory [51].

A graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is defined by a set of nodes $\mathbf{V}$ and a set of edges $\mathbf{E}$ between these [6,8]. We denote an edge from node $u \in \mathbf{U}$ to node $v \in \mathbf{V}$ as $(u, v) \in \mathbf{E}$. In

order to represent a graph with an adjacency matrix $\mathbf{A} = (a_{uv})_{|\mathbf{V}| \times |\mathbf{V}|} \in \mathscr{R}^{|\mathbf{V}| \times |\mathbf{V}|}$, $u, v = 1, 2, \ldots, |\mathbf{V}|$, we order the nodes in the graph so that every node indexes a particular row and column in the adjacency matrix. $\mathbf{A}(u, v) = \{a_{u,v}\}$, $a_{u,v} = 1$ if $(u, v) \in \mathbf{E}$, otherwise $a_{u,v} = 0$. For weighted edges, $a_{u,v} \in [0, 1]$, pertaining to directed graph, $a_{u,v} \neq a_{v,u}$, generally, $A(u, v) \neq A(v, u)$.

The goal of node classification or node attribute inference [28] in machine learning on graphs is to predict the label, if we are only given the true labels on a training set of nodes [15]. The key insight behind most successful node classification approaches is to explicitly leverage the connections between nodes. Node classification is useful for inferring information based on its relationship with other nodes in the graph. The advantage over other machine learning methods is that node attribute inference gives us the ability to bring in context and neighborhood information into our predictions.

MATLAB provides an example of node classification using graph convolutional network, which shows how to classify nodes in a graph using a graph convolutional network (GCN). The example is based on the given the molecular structure, see:

https://au.mathworks.com/help/deeplearning/ug/node-classification-using-graph-convolutional-network.html

Relation prediction in machine learning on graphs includes link prediction, graph completion, and relational inference [46]. It is the problem of inferring missing or finding hidden relationships between entities. The goal is to apply this partial information to infer the missing edges. The complexity of this task is highly dependent on the type of graph data. Relation prediction requires inductive biases that are specific to the graph domain.

Graph clustering in machine learning on graphs is to learn an unsupervised measure of similarity between pairs of graphs [56]. Meanwhile, graph regression is to make use of a labeled set of training data to train a mapping from data points (i.e., graphs) with regard to labels [40]. In graph kernel methods, node degree is defined as $d_u = \sum_{v \in \mathbf{V}} \mathbf{A}(u, v)$, eigenvector centrality refers to

$$e_u = \frac{1}{\lambda} \sum_{v \in \mathbf{V}} \mathbf{A}(u, v) e_v \tag{6.95}$$

or

$$\lambda \mathbf{e} = \mathbf{A} \mathbf{e} \tag{6.96}$$

where λ is a constant. Clustering coefficient method means

$$c_u = \frac{|\{(v_1, v_2) \in \mathbf{E}, v_1, v_2 \in \mathbf{N}(u)\}|}{C_{d_u}^2} \tag{6.97}$$

where $\mathbf{N}(u) = \{v \in \mathbf{V}, (u, v) \in \mathbf{E}\}$.

The Weisfeiler–Lehman (WL) kernel [58] has the steps:

- Assign an initial label $l^{(0)}(v)$, $v \in \mathbf{V}$ to each node.
- Iteratively assign a new label to each node by hashing the multiset of the current labels within the node's neighborhood.
- After running k iterations of re-labeling, we now have a label $l^{(k)}(v)$ for each node.

In graph Laplacians, the basic Laplacian matrix is

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{6.98}$$

where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}$ is the degree matrix. It is symmetric ($\mathbf{L}^\top = \mathbf{L}$) and positive semi-definite

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} \geq 0 \tag{6.99}$$

where

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{(u,v) \in \mathbf{E}} (\mathbf{x}(u) - \mathbf{x}(v))^2 \tag{6.100}$$

where $\mathbf{L}$ has non-negative eigenvalues $\lambda_i > 0$, $i = 1, \ldots, |\mathbf{V}|$. The normalized Laplacians is

$$\mathbf{L}_{sys} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}}. \tag{6.101}$$

The random walk Laplacian is $\mathbf{L}_{RW} = \mathbf{D}^{-1} \mathbf{L}$ [35].

Given a partitioning of the graph [39,48] into k non-overlapping subsets $\mathbf{A}_1, \ldots, \mathbf{A}_k, \mathbf{A}_i \cap \mathbf{A}_j = \emptyset (i \neq j)$, the graph cut [38] is

$$Cut(\mathbf{A}_1, \ldots, \mathbf{A}_k) = \frac{1}{2} \sum_{i=1}^{k} |\{(u,v) \in \mathbf{E}, u \in \mathbf{A}_i, v \in \bar{\mathbf{A}}_i, \mathbf{A} \cap \bar{\mathbf{A}} = \emptyset\}|. \tag{6.102}$$

There are two kinds of graph cuts [38], i.e., ratio cut and normalized cut:

- Ratio cut:

$$RCut(\mathbf{A}_1, \ldots, \mathbf{A}_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{|\{(u,v) \in \mathbf{E}, u \in \mathbf{A}_i, v \in \bar{\mathbf{A}}_i, \mathbf{A} \cap \bar{\mathbf{A}} = \emptyset\}|}{|\mathbf{A}_i|}. \tag{6.103}$$

- Normalized cut:

$$NCut(\mathbf{A}_1, \ldots, \mathbf{A}_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{|\{(u,v) \in \mathbf{E}, u \in \mathbf{A}_i, v \in \bar{\mathbf{A}}_i, \mathbf{A} \cap \bar{\mathbf{A}} = \emptyset\}|}{vol(\mathbf{A})} \tag{6.104}$$

where $vol(\mathbf{A}) = \sum_{u \in \mathbf{A}} d_u$.

The generalized spectral clustering [43] has steps as follows:

- Find the smallest eigenvectors of $\mathbf{L}$ (excluding the smallest).
- Form the matrix $\mathbf{U}$ with the eigenvectors from the first step.
- Represent each node by using its corresponding row in the matrix $\mathbf{z}_u = \mathbf{U}(u)$, $u \in \mathbf{V}$.
- Run k-means clustering on the embeddings $\mathbf{z}_u, u \in \mathbf{V}$.

6.4.2 Node Embeddings

Node embeddings are based upon the framework of encoding and decoding graphs [1, 11]. In the encoder–decoder framework, an encoder model maps each node in the graph into a low-dimensional vector or embedding. In the encoder–decoder framework, a decoder model takes advantage of the low-dimensional node embeddings and makes use of them to reconstruct information about each node's neighborhood in the original graph.

Node embedding algorithms compute low-dimensional vector representations of nodes in a graph. In shallow embedding, the encoder is the function that maps node $v \in \mathbf{V}$ to vector embedding $\mathbf{z}_v \in \mathscr{R}^d$, where $\mathbf{z}_v$ corresponds to the embedding for node $v \in \mathbf{V}$.

$$\mathbf{ENC}(v) = \mathbf{Z}(v) \tag{6.105}$$

where $\mathbf{Z}$ denotes the row of matrix $\mathbf{Z}$ corresponding to node v.

The role of decoder is to reconstruct graph statistics from the node embeddings. The goal is to minimize the reconstruction loss. $\mathbf{S}(u, v)$ is a graph-based similarity measure between nodes.

$$\mathbf{DEC}(\mathbf{ENC}(u), \mathbf{ENC}(v)) = \mathbf{DEC}(\mathbf{z}_u, \mathbf{z}_v) \approx \mathbf{S}(u, v) \stackrel{\triangle}{=} \mathbf{A}(u, v). \tag{6.106}$$

In order to achieve the reconstruction objective, we minimize an empirical reconstruction loss $\mathscr{L}$ over a set of training node pairs.

$$\mathscr{L} = \sum_{(u,v) \in \mathbf{D}} l(\mathbf{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}(u, v)) \tag{6.107}$$

where $l(\cdot)$ is a loss function measuring the discrepancy between $\mathbf{DEC}(u, v)$ and $\mathbf{S}(u, v)$. The reconstruction objective for these approaches is written as

$$\mathscr{L} = \sum_{(u,v) \in \mathbf{D}} \|\mathbf{z}_u^\top \mathbf{z}_v - \mathbf{S}(u, v)\|_2^2 \approx \|\mathbf{Z}\mathbf{Z}^\top - \mathbf{S}\|_2^2. \tag{6.108}$$

Laplacian eigenmaps [62] are

$$\mathbf{DEC}(\mathbf{z}_u, \mathbf{z}_v) = \|\mathbf{z}_u - \mathbf{z}_v\|_2^2 \tag{6.109}$$

and

$$\mathscr{L} = \sum_{(u,v) \in \mathbf{D}} \mathbf{DEC}(\mathbf{z}_u, \mathbf{z}_v) \cdot \mathbf{S}(u, v). \tag{6.110}$$

Inner-product methods are

$$\mathbf{DEC}(\mathbf{z}_u, \mathbf{z}_v) = \mathbf{z}_u^\tau \mathbf{z}_v \tag{6.111}$$

and

$$\mathscr{L} = \sum_{(u,v) \in \mathbf{D}} \|\mathbf{DEC}(\mathbf{z}_u, \mathbf{z}_v) - \mathbf{S}(u, v)\|_2^2. \tag{6.112}$$

Random walk embeddings [35,63] include

- **DeepWalk** [47] and **node2vec** [19]:

$$\mathbf{DEC}(\mathbf{z}_u, \mathbf{z}_v) \triangleq \frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{\mathbf{z}_k \in \mathbf{V}} e^{\mathbf{z}_u^\tau \mathbf{z}_k}} \approx p_{G,T}(v|u) \tag{6.113}$$

where $p_{G,T}(v|u)$ is the probability of visiting v on a length T random walk starting at u, $T = \{2, 3, \ldots, 10\}$.

The cross-entropy loss is

$$\mathcal{L} = - \sum_{(u,v) \in \mathbf{D}} \log[\mathbf{DEC}(\mathbf{z}_u, \mathbf{z}_v)] \tag{6.114}$$

where $\mathbf{D}$ is training set of random walks, which is generated by sampling random walks starting from each node [35].

- **LINE**:

$$\mathbf{DEC}(\mathbf{z}_u, \mathbf{z}_v) = \frac{1}{1 + e^{-\mathbf{z}_u^\top \mathbf{z}_v}} \tag{6.115}$$

where LINE refers to Large-scale Information Network Embeddings.

The shallow embedding methods [16] do not share any parameters between nodes in the encoder. The lack of parameter sharing is due to both statistically and computationally inefficient. The shallow embedding approaches do not leverage node features in the encoder.

The shallow embedding methods are inherently transductive, the restriction prevents shallow embedding methods from being used on inductive applications, which involve generalizing to unseen nodes after training.

The goal of embedding multirelational graphs (i.e., knowledge graph) is to reconstruct the relationship between these nodes. Given a multirelational graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, the edges are $e = (u, \tau, v)$, the decoder is

$$\mathbf{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_v) = \mathbf{z}_u^\top \mathbf{R}_\tau \mathbf{z}_v \tag{6.116}$$

where $\mathbf{R}_\tau \in \mathscr{R}^{d \times d}$ is a matrix specified to relation $\tau \in \mathscr{R}$. A basic reconstruction loss is

$$\mathcal{L} = \sum_{u \in \mathbf{U}} \sum_{v \in \mathbf{V}} \sum_{\tau \in \mathbf{R}} \|\mathbf{DEC}(u, \tau, v) - \mathbf{A}(u, \tau, v)\|^2 = \sum_{u \in \mathbf{U}} \sum_{v \in \mathbf{V}} \sum_{\tau \in \mathbf{R}} \|\mathbf{z}_u^\top \mathbf{R}_\tau \mathbf{z}_v - \mathbf{A}(u, \tau, v)\|^2 \tag{6.117}$$

where $\mathbf{A} \in \mathscr{R}^{|\mathbf{U}| \times |\mathbf{R}| \times |\mathbf{V}|}$ is the adjacency tensor for the multirelational graph.

The cross-entropy loss with negative sampling distribution is

$$\mathcal{L} = \sum_{(u,\tau,v) \in \mathbf{E}} \{-\log \sigma(\mathbf{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_v)) - \gamma \mathscr{E}_{v_n \in p_{n,u}(\mathbf{V})}[\log \sigma(\mathbf{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_{v_n}))]\}. \tag{6.118}$$

Popularly,

$$\mathcal{L} = \sum_{(u,\tau,v) \in \mathbf{E}} \{-\log \sigma(\mathbf{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_v)) - \sum_{v_n \in \mathbf{p}_{n,u}} [\log \sigma(\mathbf{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_{v_n}))]\} \tag{6.119}$$

where $\sigma(\cdot)$ is logistic function, $p_{n,u}(\mathbf{V})$ is a negative sampling distribution, $\mathbf{p}_{n,u}$ is a subset of nodes from $p_{n,u}(\mathbf{V})$, $\gamma > 0$ is a parameter.

The margin loss or hinge loss is

$$\mathscr{L} = \sum_{(u,\tau,v)\in\mathbf{E}} \sum_{v_n\in\mathbf{p}_{n,u}} \{max(0, -\mathbf{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_v) + \mathbf{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_{v_n}) + \triangle)\} \quad (6.120)$$

where $\triangle$ is called as margin.

6.4.3   Deep Graph Neural Networks

CNNs are well-defined only over grid-structured inputs (e.g., images). RNNs are well-defined only over sequences (e.g., text). Graph Neural Networks (GNNs) are a kind of general framework for defining deep neural networks on graph-structured data. A deep neural network over graphs would be to simply harness the adjacency matrix as input to a deep neural network.

$$\mathbf{z_G} = \mathbf{MLP}(\mathbf{A}(1) \oplus \cdots \oplus \mathbf{A}(|\mathbf{V}|)) \quad (6.121)$$

where $\mathbf{A}(i), i = 1, 2, \ldots, |\mathbf{V}|$ is a row of the adjacency matrix. $\mathbf{MLP}(\cdot)$ is a multilayer perceptron, $\oplus$ is vector concatenation.

Neighborhood normalization [33] is

- Simple normalization:

$$\mathbf{m}_{\mathbf{N}(u)} = \frac{\sum_{v\in\mathbf{N}(u)} \mathbf{h}_v}{|\mathbf{N}(u)|}. \quad (6.122)$$

- Symmetric normalization:

$$\mathbf{m}_{\mathbf{N}(u)} = \sum_{v\in\mathbf{N}(u)} \frac{\mathbf{h}_v}{\sqrt{|\mathbf{N}(u)||\mathbf{N}(v)|}}. \quad (6.123)$$

In the self-loop GNN approach, we have

$$\mathbf{h}_u^{(k)} = \mathbf{AGGREGATE}(\{\mathbf{h}_v^{(k-1)}, \forall v \in \mathbf{N}(u) \cup \{u\}) \quad (6.124)$$

and

$$\mathbf{H}^{(k)} = \sigma((\mathbf{A} + \mathbf{I})\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}). \quad (6.125)$$

Graph attention network [61] makes use of attention weights to define a weighted sum of the neighbors,

$$\mathbf{m}_{\mathbf{N}(u)} = \sum_{u\in\mathbf{N}(u)} \alpha_{u,v}\mathbf{h}_v \quad (6.126)$$

where $\alpha_{u,v}$ denotes the attention on neighbor $v \in \mathbf{N}(u)$ if we are aggregating information at node u.

Popularly, bilinear attention model [26] is

$$\alpha_{u,v} = \frac{\exp(\mathbf{h}_u^\top \mathbf{W}\mathbf{h}_v)}{\sum_{v'\in\mathbf{N}(u)} \exp(\mathbf{h}_u^\top \mathbf{W}\mathbf{h}_{v'})}. \quad (6.127)$$

MATLAB provides an example which shows how to classify graphs that have multiple independent labels using graph attention networks (GATs). If the observations of given data have a graph structure with multiple independent labels, the data labels could be predicted. Using the graph structure and available information on graph nodes, a multihead self-attention mechanism is employed to aggregate features across neighboring nodes, and computes output features or embeddings for each node in the graph. The output features are used to classify the graph usually after employing a readout, or a graph pooling, operation to aggregate or summarize the output features of the nodes. The example is available at

https://au.mathworks.com/help/deeplearning/ug/multilabel-graph-classification-using-graph-attention-networks.html

Using $\mathbf{MLP}(\cdot)$, we have

$$\alpha_{u,v} = \frac{\exp(\mathbf{MLP}(\mathbf{h}_u^\top \mathbf{W} \mathbf{h}_v))}{\sum_{v' \in \mathbf{N}(u)} \exp(\mathbf{MLP}(\mathbf{h}_u^\top \mathbf{W} \mathbf{h}_{v'}))}. \tag{6.128}$$

The representations for all the nodes in the graph can become very similar to one another after several iterations of GNN message passing. Oversmoothing occurs if node-specific information becomes "washed out" or "lost" after several iterations of GNN message passing. It is quantified by examining the magnitude of the corresponding Jacobian matrix $\frac{\mathbf{h}_u^{(k)}}{\mathbf{h}_v^{(0)}}$. Jacobian matrix $\frac{\mathbf{h}_u^{(k)}}{\mathbf{h}_v^{(0)}}$ is a measure of how much the initial embedding of node u influences the final embedding of node v in the GNN.

Gating methods are employed to improve the stability and learning ability of RNNs. The gated updates are very effective at facilitating deep GNN architectures and preventing oversmoothing:

$$\mathbf{z}_u = \mathbf{h}_u^{(k)} = \mathbf{GRU}(\mathbf{h}_u^{(k-1)}), m_{\mathbf{N}(u)}^{(k)} \tag{6.129}$$

where $\mathbf{GRU}(\cdot)$ denotes the update equation of the gated recurrent unit (GRU) cell.

Relational graph neural network [24] (RGNN) has the transformation function:

$$\mathbf{m}_{\mathbf{N}(u)} = \sum_{\tau \in \mathbf{R}} \sum_{v \in \mathbf{N}(u)} \frac{\mathbf{W}_\tau \mathbf{H}_v}{f_n(\mathbf{N}(u), \mathbf{N}(v))} \tag{6.130}$$

where $f_n(\cdot)$ is a normalization function that depends on both the neighborhood of the node u as well as the neighbor v.

RGNN has a drastic increase in the number of parameters, the relations of this increase lead to overfitting and slow learning.

GNNs are employed for one of three tasks: Node classification, graph classification [67], or relation prediction:

- Node classification [9]:

$$\mathcal{L} = \sum_{u \in \mathbf{V}} \mathbf{softmax}(\mathbf{z}_u, \mathbf{y}_u) \tag{6.131}$$

$$\mathbf{softmax}(\mathbf{z}_u, \mathbf{y}_u) = \sum_{i=1}^{c} \mathbf{y}_u^{(i)} \frac{e^{\mathbf{z}_u^\top \mathbf{w}_i}}{\sum_{j=1}^{c} e^{\mathbf{z}_u^\top \mathbf{w}_j}} \tag{6.132}$$

where $\mathbf{w}_i$, $i = 1, \ldots, c$ are trainable parameters.

- Graph classification [67]:

$$\mathscr{L} = \sum_{\mathbf{G}_i} \|\mathbf{MLP}(\mathbf{z}_{\mathbf{G}_i}) - \mathbf{y}_{\mathbf{G}_i}\|_2^2 \tag{6.133}$$

where $\mathbf{MLP}(\cdot)$ is a densely connected neural network with a univariate output and $\mathbf{y}_{\mathbf{G}_i}$ is the target value for training graph $\mathbf{G}_i$.

- Graph embedding [42]: Deep graph infomax (DGI) maximizes the mutual information between node embedding $\mathbf{z}_u$.

$$\mathscr{L} = -\sum_{u \in \mathbf{V}} \mathbf{E_G} \log(D(\mathbf{z}_u, \mathbf{z_G})) + \gamma \mathbf{E_{\hat{G}}} \log(1 - D(\hat{\mathbf{z}}_u, \mathbf{z_G}). \tag{6.134}$$

In graph convolutions, let $f(\cdot)$ and $h(\cdot)$ be two functions, general continuous convolution operation '$\star$' is

$$(f \star h)(\mathbf{x}) = \int_{\mathbf{x}, \mathbf{y} \in \mathbf{R}^d} f(\mathbf{y}) h(\mathbf{x} - \mathbf{y}) d\mathbf{y}. \tag{6.135}$$

Convolution translation (or shift) is

$$f(t + \alpha) \star g(t) = f(t) \star g(t + \alpha) = \triangle(f \star g)(t + \alpha). \tag{6.136}$$

A corollary convolution regarding difference operation is

$$\triangle f(t) \star g(t) = f(t) \star \triangle g(t) = \triangle(f \star g)(t) \tag{6.137}$$

where $\triangle f(t) = f(t + 1) - f(t)$ is the Laplace (i.e., difference) operator.

For a cyclic chain graphs or chain-cyclic graphs [30],

$$f \star h(t) = \sum_{\tau=0}^{N-1} f(\tau) h(\tau - t) = \mathbf{Q}_h \mathbf{f} \tag{6.138}$$

where $\mathbf{Q}_h$ is a matrix of the convolution operation with function $\mathbf{h}$, e.g.,

$$\mathbf{Q}_h = \sum_{i=0}^{N-1} \alpha_i \mathbf{A}_c^i \tag{6.139}$$

where $\mathbf{f} = [f(t_0), \ldots, f(t_{N-1})]^\top$ and

$$\mathbf{A}_c(i, j) = \begin{cases} 1 \text{ if } i = (j + 1) \mod N; \\ 0 \text{ if } \mathbf{otherwise}. \end{cases} \tag{6.140}$$

For a cyclic chain graph, matrix $\mathbf{Q}_h$ has the properties:

- Translation equivariance (commutativity):

$$\mathbf{A}_c \mathbf{Q}_h = \mathbf{Q}_h \mathbf{A}_c. \tag{6.141}$$

- Difference operator:

$$\mathbf{L}_c \mathbf{Q}_h = \mathbf{Q}_h \mathbf{L}_c \tag{6.142}$$

where $\mathbf{L}_c = \mathbf{I} - \mathbf{A}_c$.

- These requirements are satisfied for a real matrix $\mathbf{Q}_h$ if

$$\mathbf{Q}_h = \sum_{i=0}^{N-1} \alpha_i \mathbf{A}_c^i. \tag{6.143}$$

In general graph convolutions, for an arbitrary graph with adjacency matrix $\mathbf{A}$ and a filter h, the convolution matrix is

$$\mathbf{Q}_h = \sum_{i=0}^{N} \alpha_i \mathbf{A}^i. \tag{6.144}$$

Laplace operator simply corresponds to the difference operator (i.e., the difference between consecutive time points).

$$\mathbf{Lx}(i) = \mathbf{A}(i, j)(\mathbf{x}_i - \mathbf{x}_j) \tag{6.145}$$

which measures the difference between the value of signal $x(i)$ at a node i and the signal values of all of its neighbors.

By considering the eigendecomposition of the general graph Laplacian,

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top \tag{6.146}$$

where the eigenvectors $\mathbf{U}$ is the graph Fourier modes. The matrix Λ is the corresponding eigenvalues along the diagonal.

Fourier transform of signal (or function) $\mathbf{f} \in \mathcal{R}^{|\mathbf{V}|}$ on a graph is

$$\mathbf{s} = \mathbf{U}^\top \mathbf{f}. \tag{6.147}$$

Its inverse Fourier transform is

$$\mathbf{f} = \mathbf{U}\mathbf{s}. \tag{6.148}$$

A graph convolution via elementwise products is

$$\mathbf{f} \circ_{\mathbf{G}} \mathbf{h} = \mathbf{U}(\mathbf{U}^\top \mathbf{f} \circ \mathbf{U}^\top \mathbf{h}) \tag{6.149}$$

where $\mathbf{U}$ is the matrix of eigenvectors of the Laplacian $\mathbf{L}$ and $\circ_{\mathbf{G}}$ is harnessed to denote that this convolution is specific to a graph $\mathbf{G}$.

In Hilbert space embeddings of distributions, we represent the density $p(\mathbf{x})$, $\mathbf{x} \in \mathcal{R}^m$ under the feature map $\phi : \mathcal{R}^m \to \mathcal{R}$ as

$$\mu_{\mathbf{x}} = < \phi(\mathbf{x}), p(\mathbf{x}) >= \int_{\mathcal{R}^m} \phi(\mathbf{x})p(\mathbf{x})d\mathbf{x}. \tag{6.150}$$

A well-known example of a feature map of the Gaussian radial basis function (RBF) kernel is $\phi(\mathbf{x}) = \exp(-\varepsilon\mathbf{x})^2$, etc. RBF functions are typically applied to build up function approximations of the form

$$y(\mathbf{x}) = \sum_{i=1}^{N} \omega_i \phi(\|\mathbf{x} - \mathbf{x}_i\|). \tag{6.151}$$

A graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ defines a Markov random field as

$$p(\{\mathbf{x}_v\}, \{\mathbf{z}_v\}) \propto \prod_{v \in \mathbf{V}} \Psi(\mathbf{x}_v, \mathbf{z}_v) \prod_{(u,v) \in \mathbf{E}} \Phi(\mathbf{z}_u, \mathbf{z}_v) \tag{6.152}$$

where $\Psi(\cdot)$ and $\Phi(\cdot)$ are non-negative potential functions, $\Psi(\mathbf{x}_v, \mathbf{z}_v)$ indicates the likelihood of a node feature vector $\mathbf{x}_v$, given its latent node embedding $\mathbf{z}_v$, $\Phi(\cdot)$ controls the dependency between connected nodes.

Given a Markov random field, we approximate the posterior based on the assumption

$$p(\{\mathbf{z}_v\}|\{\mathbf{x}_v\}) \approx q(\{\mathbf{z}_v\}) = \prod_{v \in \mathbf{V}} q_v(\{\mathbf{z}_v\}). \tag{6.153}$$

In the mean-field approximation using KL divergence,

$$\mathbf{KL}(q(\{\mathbf{z}_v\})|p(\{\mathbf{z}_v|\mathbf{x}_v\})) = \int q(\{\mathbf{z}_v\}) \log \frac{q(\{\mathbf{z}_v\})}{p(\{\mathbf{z}_v|\mathbf{x}_v\})} d\mathbf{z}_v. \tag{6.154}$$

The updated Hilbert space embedding for node v is based on its neighbors' embeddings [65] as well as its feature inputs.

$$\mu_v^{(t)} = \sigma(\mathbf{w}_v^{(t)}\mathbf{x}_v + \mathbf{w}_u^{(t)} \sum_{u \in \mathbf{N}(v)} \mu_u^{(t-1)}). \tag{6.155}$$

Two graphs are isomorphic if and only if (iff) there exists a permutation matrix $\mathbf{P}$ such that

$$\mathbf{P}^\top \mathbf{A}_1 \mathbf{P} = \mathbf{A}_2, \mathbf{P}\mathbf{X}_1 = \mathbf{X}_2 \tag{6.156}$$

where two graphs $\mathbf{G}_1$ and $\mathbf{G}_2$ with adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, as well as node features $\mathbf{X}_1$ and $\mathbf{X}_2$.

Two graphs being isomorphic means that they are essentially identical in terms of their underlying structure. Graph isomorphism [57] has the following optimization problem,

$$\min_{\mathbf{p} \in \mathscr{P}} \|\mathbf{A}_2 - \mathbf{P}\mathbf{A}_1\mathbf{P}^\top\| + \|\mathbf{P}\mathbf{X}_1 - \mathbf{X}_2\| \to 0. \tag{6.157}$$

Graph isomorphism network (GIN) [27] is defined as

$$\mathbf{h}_u^{(k)} = \mathbf{MLP}^{(k)}((1 + \varepsilon^{(k)})\mathbf{h}_u^{(k-1)} + \sum_{v \in \mathbf{N}(u)} \mathbf{h}_v^{(k-1)}) \tag{6.158}$$

where $\varepsilon^{(k)}$ is a trainable parameter.

6.4.4   Graph Generating

The goal of graph generating is to create models that can generate realistic graph structures. The key challenge in graph generating is to generate graphs that have desirable properties.

The Erdös-Rényi (ER) model simply assumes that the probability of an edge occurring between any pairs of nodes is equal [12]. i.e., $P(\mathbf{A}(u, v) = 1) = r \in [0, 1]$, $\forall u, v \in \mathbf{V}, u \neq v$.

Stochastic block models (SBM) [21] seek to generate graphs with community structure. i.e., $P(\mathbf{A}(u, v) = 1) = \mathbf{C}(i, i)$, $\mathbf{C}(i, j) \in [0, 1]^{|\mathbf{V}| \times |\mathbf{V}|}$ is a block-to-block probability matrix.

Preferential attachment (PA) model [53] connects new nodes to existing nodes with a probability that is proportional to the existing node degrees.

$$P(\mathbf{A}(u, v)) = \frac{d_v^{(t)}}{\sum d_{v'}^{(t)}}, \tag{6.159}$$

where $d_v^{(t)}$ denotes the degree of node v at iteration t.

We jointly train the encoder and decoder so that the decoder is able to reconstruct graphs. We assume a probabilistic encoder model q_ϕ, a probabilistic decoder model p_θ, a prior distribution $p(\mathbf{Z}), \mathbf{Z} \in \mathbf{N}(0, 1)$ over the latent space. Given a set of training graphs,

$$\mathscr{L} = \sum_{\mathbf{G}_i \in \{\mathbf{G}_1, \mathbf{G}_2, ..., \mathbf{G}_n\}} \mathbf{E}_{q_\theta}[p_\theta(\mathbf{G}_i|\mathbf{Z})] - \mathbf{KL}(q_\theta(\mathbf{Z}|\mathbf{G}_i) \| p_\theta(\mathbf{Z})). \tag{6.160}$$

We seek to maximize the reconstruction ability of the decoder, i.e., the likelihood term $\mathbf{E}_{q_\theta}$ while minimizing the KL divergence between the posterior latent distribution $q_\theta(\mathbf{Z}|\mathbf{G}_i)$ and the prior $p_\theta(\mathbf{Z})$.

The encoder generates latent representations for each node in the graph. The decoder takes advantage of embeddings as input and uses the embeddings to predict the likelihood of an edge occurring between the two nodes.

- Encoder model: Given an adjacency matrix $\mathbf{A}$ and node features $\mathbf{X}$, we generate mean $\mu_{\mathbf{z}}$ and variance $\sigma_{\mathbf{z}}$,

$$\mathbf{Z} = \varepsilon \circ \exp(\log(\sigma_{\mathbf{z}})) + \mu_{\mathbf{z}}, \tag{6.161}$$

where $\varepsilon \in \mathscr{N}(0, 1)$.
- Decoder model: The independence between edges defines the posterior over the full graph, which corresponds to a binary cross-entropy loss over the edge probabilities.

$$p_\theta(\mathbf{G}|\mathbf{Z}) = \prod_{(u,v) \in \mathbf{V}^2} p_\theta(\mathbf{A}(u, v) = 1|\mathbf{z}_u, \mathbf{z}_v) = \sigma(\mathbf{z}_u^\top \mathbf{z}_v). \tag{6.162}$$

A general GAN-based generative model [17, 44] is:

- The generator network $g_\theta(\cdot)$ is trained to generate realistic (but fake) data samples.
- The discriminator $d_\phi(\cdot)$ is to distinguish between real data samples and samples generated by the generator.
- To train a GAN, both the generator and discriminator are optimized jointly in an adversarial game:

$$\mathscr{L} = \min_\theta \max_\phi \{\mathbf{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} \log[1 - d_\phi(\mathbf{x})] + \mathbf{E}_{\mathbf{z} \sim p_{seed}(\mathbf{z})} \log[d_\phi(g_\theta(\mathbf{z}))]\} \tag{6.163}$$

where $p_{data}(\mathbf{x})$ denotes the empirical distribution of real data samples (e.g., a uniform sample over a training set) and $p_{seed}(\mathbf{z})$ is the random seed distribution.

In modeling edge dependencies,

$$p(\mathbf{G}|\mathbf{z}) = \prod_{(u,v)\in|\mathbf{V}|^2} p(\mathbf{A}(u, v)|\mathbf{z}). \tag{6.164}$$

In the autoregressive approach, we assume that edges are generated sequentially and that the likelihood of each edge can be conditioned on the edges.

$$p(\mathbf{G}|\mathbf{z}) = \prod_{i=1}^{|\mathbf{V}|} p(\mathbf{L}(v_i, :)|\mathbf{L}(v_1, :), \ldots, \mathbf{L}(v_{i-1}, :), \mathbf{z}) \tag{6.165}$$

where $\mathbf{L}$ denotes the lower-triangular portion of the adjacency matrix $\mathbf{A}$.

Recurrent models for graph generating include

- **GraphRNN**: A hierarchical RNN to model the edge dependencies.

$$\mathbf{h}_{i+1} = \mathbf{RNN}_{graph}(\mathbf{h}_i, \mathbf{L}(v_i, :)) \tag{6.166}$$

where $\mathbf{RNN}_{graph}$ denotes a generic RNN state update.
- **GRAN**: Graph Recurrent Attention Networks [32],

$$p(\mathbf{L}(v_i, :)|\mathbf{L}(v_1, :), \ldots, \mathbf{L}(v_{i-1}, :)) \approx \mathbf{GNN}(L(v_1 : v_{i-1}, :), \hat{\mathbf{X}}) \tag{6.167}$$

where $\hat{\mathbf{X}}$ is the input feature matrix.

We compute the distance between the statistical distribution on the test graph and generated graph by using the total variation distance [5],

$$d(s_{i,\mathbf{G}_{test}}, s_{i,\mathbf{G}_{gen}}) = \sup_{x\in\mathscr{R}} |s_{i,\mathbf{G}_{test}}(\mathbf{x}) - s_{i,\mathbf{G}_{gen}}(\mathbf{x})| \tag{6.168}$$

where $s_{i,\mathbf{G}_{test}}$ and $s_{i,\mathbf{G}_{gen}}$ are particular statistic s_i based on test graph and generated graph.

Exercises

Question 6.1. Please list the algorithms for data dimensionality (dimension) reduction and the differences between them.

Question 6.2. Why could manifold learning be applied to dimensionality reduction?

Question 6.3. What are the directed graphs and undirected graphs? Please give an example for each category.

Question 6.4. What are the linear exponential families? Please give an example.

Question 6.5. What are the relationships between relative entropy, information projection, and moment projection?

Question 6.6. What are the aim and feature of Restricted Boltzmann machines (RBMs)? Why are deep Boltzmann machines thought as multilayer perceptron with RBMs?

Question 6.7. What are the differences between MRF and CRF?

Question 6.8. What are the linear exponential families? Please give an example.

Question 6.9. What are the relationships between relative entropy, information projection, and moment projection?

Question 6.10. What are the differences when convolutional neural networks are applied to graphs and digital images?

References

1. Abu-El-Haija S, Perozzi B, Al-Rfou R, Alemi A (2018) Watch your step: learning node embeddings via graph attention. In: International conference on neural information processing systems (NeurIPS). Montréal, Canada
2. Ackley DH, Hinton GE, Sejnowski TJ (1987) A learning algorithm for Boltzmann machines. In: Readings in computer vision, pp 522–533
3. Alpaydin E (2009) Introduction to machine learning. MIT Press
4. Badrinarayanan V, Handa A, Cipolla R (2017) SegNet: a deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. IEEE Trans Pattern Anal Mach Intell 39(12):2481–2495
5. Barbour A, Chen L (2005) Stein's method and applications. Lecture notes series. World Scientific
6. Biggs N, Lloyd E, Wilson R (1986) Graph theory, 1736–1936. Oxford University Press
7. Blake A, Rother C, Brown M, Perez P, Torr P (2004) Interactive image segmentation using an adaptive GMMRF model. In: European conference on computer vision. Springer, pp 428–441
8. Bondy J, Murty U (2008) Graph theory. Springer
9. Chandra, N (2021) Node classification on relational graphs using deep-RGCNS. Master's thesis, California Polytechnic State University
10. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2018) DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans Pattern Anal Mach Intell 40(4):834–848
11. Domokos K, Palovics R, Benczúr A (2019) Node embeddings in dynamic graphs. Appl Netw Sci
12. Erdos P, Renyi A (1959) On random graphs. Math 6:290–297
13. Ertel W (2017) Introduction to artificial intelligence. Springer International Publishing
14. Fischer A, Igel C (2012) An introduction to restricted Boltzmann machines. In: Iberoamerican congress on pattern recognition, pp 14–36
15. Gibbons A (1985) Algorithmic graph theory. Cambridge University Press
16. Gibbons J, Nicolas Wu N (2014) Folding domain-specific languages: deep and shallow embeddings (functional pearl). In: ACM SIGPLAN, pp 339–347

17. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. In: International conference on neural information processing systems (NIPS), pp 2672–2680
18. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press
19. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: ACM SIGKDD international conference on knowledge discovery and data mining
20. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
21. Holland P, Laskey K, Leinhardt S (1983) Stochastic blockmodels: first steps. Soc Netw 5(2):109–137
22. Hopfield JJ (1988) Artificial neural networks. IEEE Circuits Devices (Mag) 4(5):3–10
23. Howard R, Matheson J (2005) Influence diagrams. Decis Anal 2(3):127–143
24. Jing Y, Wang J, Wang W, Wang L, Tan T (2020) Relational graph neural network for situation recognition. Pattern Recognit 108
25. Keys R (1981) Cubic convolution interpolation for digital image processing. IEEE Trans Acoust Speech Signal Process 29(6):1153–1160
26. Kim J, Jun J, Zhang B (2018) Bilinear attention networks. In: International conference on neural information processing systems, pp 1571–1581
27. Kim B, Ye J (2020) Understanding graph isomorphism network for rs-fMRI functional connectivity analysis. Front Neurosci
28. Kipf T, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: International conference on learning representations
29. Koller D, Friedman N (2009) Probabilistic graphical models. MIT Press, Massachusetts
30. Krepkiy I (2014) The sandpile groups of chain-cyclic graphs. J Math Sci (N Y) 200(6):698–709
31. Li S (2009) Markov random field modeling in image analysis. Springer
32. Liao R, Li Y, Song Y, Wang S, Hamilton W, Duvenaud D, Urtasun R, Zemel R (2019) Efficient graph generation with graph recurrent attention networks. In: NIPS'19, pp 4255–4265
33. Liu X, Killeen B, Sinha A, Ishii M, Hager G, Taylor R, Unberath M (2021) Neighborhood normalization for robust geometric feature learning. In: IEEE CVPR, pp 13049–13058
34. Littman M (2015) Reinforcement learning improves behavior from evaluative feedback. Nat 521:445–451
35. Lovasz L (1996) Random walks on graphs: a survey. Comb 2:353–397
36. Ma Y (2012) Manifold learning theory and applications. Taylor & Francis Group
37. MacKay D (2003) Hopfield networks. Information theory, inference and learning algorithms, p 508
38. Matula D, Shahrokhi F (1990) Sparsest cuts and bottlenecks in graphs. Discret Appl Math 27(1–2):113–123
39. McSherry F (2001) Spectral partitioning of random graphs. In: IEEE symposium on foundations of computer science
40. Meznar S, Lavrac N, Skrlj B (2021) Transfer learning for node regression applied to spreading prediction. arXiv:2104.00088
41. Mnih V et al (2015) Human-level control through deep reinforcement learning. Nat 518:529–533
42. Myrvold W, Kocay W (2011) Errors in graph embedding algorithms. J Comput Syst Sci 2(77):430–438
43. Ng A, Jordan M, Weiss Y (2002) On spectral clustering: analysis and an algorithm. In: International conference on advances in neural information processing systems
44. Ng AY, Jordan MI (2002) On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In: Advances in neural information processing systems, pp 841–848
45. Norvig P, Russell S (2016) Artificial intelligence: a modern approach, 3rd edn. Prentice Hall
46. Onuki Y, Murata T, Nukui S, Inagi S, Qiu X, Watanabe M, Okamoto H (2019) Relation prediction in knowledge graph by multi-label deep neural network. Appl Netw Sci

47. Perozzi B, Al-Rfou B, Skiena S (2014) DeepWalk: online learning of social representations. In: ACM SIGKDD, pp 701–710

48. Qiu H, Hancock E (2004) Graph matching and clustering using spectral partitions. Pattern Recognit 39(1):22–34

49. Ramsauer H et al (2021) Hopfield networks is all you need. In: International conference on learning representations

50. Rao S (2009) Engineering optimization: theory and practice, 4th edn. ISBN: 978-0-470-18352-6

51. Robertson N, Sanders D, Seymour P, Thomas R (1997) The four color theorem. J Comb Theory Ser B 70:2–44

52. Ronneberger O, Fischer P, Brox T (2015) U-Net: convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention, pp 234–241

53. Ruj S, Pal A (2016) Preferential attachment model with degree bound and its application to key predistribution in WSN. In: IEEE international conference on advanced information networking and applications

54. Sabour S, Frosst N, Geoffrey E (2017) Hinton dynamic routing between capsules. In: The conference on neural information processing systems (NIPS), USA

55. Sarikaya R, Hinton GE, Deoras A (2014) Application of deep belief networks for natural language understanding. IEEE/ACM Trans Audio Speech Lang Process 22(4):778–784

56. Schaeffer S (2007) Graph clustering. Comput Sci Rev 1(1):27–64

57. Schoning U (1988) Graph isomorphism is in the low hierarchy. J Comput Syst Sci 37(3):312–323

58. Shervashidze N, Jan van Leeuwen E, Mehlhorn K, Borgwardt K (2011) Weisfeiler-Lehman graph kernels. J Mach Learn Res

59. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations

60. Tu L (2011) Introduction to manifold, 2nd edn

61. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2018) Graph attention networks. In: ICLR

62. Wang J (2012) Geometric structure of high-dimensional data and dimensionality reduction. Springer

63. Wu X, Pang H, Fan Y, Yang L, Luo H (2021) ProbWalk: a random walk approach in weighted graph embedding. Procedia Comput Sci 183:683–689

64. Wu X, Yan W (2022) Human identification based on gait manifold. Appl Intell. Springer

65. Xu S, Liu S, Feng L (2019) Neighborhood graph embedding for nodes clustering of social network

66. Yao W, Zeng Z, Lian C, Tang H (2018) Pixel-wise regression using U-Net and its application on pansharpening. Neurocomputing 312:364–371

67. Zhang M, Cui Z, Neumann M, Chen Y (2018) An end-to-end deep learning architecture for graph classification. In: AAAI-18, vol 32, no 1

68. Zheng N, Xue J (2009) Statistical learning and pattern analysis for image and video processing. Springer

69. Zheng S, Jayasumana S, Romera-Paredes B, Vineet V, Su Z, Du D, Torr PH (2015) Conditional random fields as recurrent neural networks. In: IEEE ICCV, pp 1529–1537

70. Zhu B et al (2018) Image reconstruction by domain-transform manifold learning. Nat 555:487–492

Transfer Learning and Ensemble Learning

7

In this chapter, we start from transfer learning and introduce the relationship between learners. We use ensemble learning to combine them together and hope to get a strong learner from a weak learner by changing the training dataset or adjusting parameters of networks. Our ultimate goal is to implement a robust classifier for pattern classification.

7.1 Transfer Learning

7.1.1 Concepts of Transfer Learning

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on stored parameters while solving one problem and applying it to another task but relevant problem [8, 29].

In this chapter, we will introduce how to use well-trained parameters to test a new model. We hope transfer learning [23, 45] is able to save our computing time and costs [46]. After implementing transfer learning, the performance of new models may be good, or may not, thus we have to train the new models again and improve the model by using new dataset.

Transfer learning is a new model of machine learning, which is employed to change the training targets. We reuse our previously trained models and save our computing resources. In the past, a lot of work has been developed for deep learning. The best paper from IEEE CVPR'18 is related to Taskonomy which is based on transfer learning. In transfer learning, the questions are as follows: What is transfer learning? what will be transferred? when to be transferred? how to implement transfer learning?

Transfer learning is a machine learning method where a model was developed for a task that is reused as the starting point for a model on a second task [44]. Transfer

learning extracts knowledge (i.e., parameters, features, samples, instance, etc.) from a task and applies it to a new task. Transfer learning is famous for unique features of a model and its parameters are available. After simple adjustment, the parameters could be confirmed and applied to new applications.

In transfer learning, corresponding to the discovered knowledge, we have sample transfer, instance transfer, parameter transfer, feature transfer, etc. According to the labels and domain knowledge, we group transfer learning to many categories, e.g., supervised learning, unsupervised learning, reinforcement learning. Transfer AdBoost (TrAdBoost) is a typical example.

Transfer learning is subject to the labels of source task and target task. Transfer learning allows domain $\mathbf{D}$, task $\mathbf{T}$, and distribution in training dataset and test dataset to be different. Because the domains are distinct, the tasks will be dissimilar.

Given domain $\mathbf{D} = \{\mathbf{X}, P(X), X \in \mathbf{X}\}$, $\mathbf{D}_S \neq \mathbf{D}_T$ implies $\mathbf{X}_S \neq \mathbf{X}_T$ or $P_S(X) \neq P_S(Y)$.

Given task $\mathbf{T} = \{\mathbf{Y}, P(Y|X), Y \in \mathbf{Y}\}$, $\mathbf{T}_S \neq \mathbf{T}_T$ implies $\mathbf{Y}_S \neq \mathbf{Y}_T$ or $P(Y_S|X_S) \neq P(Y_T|X_T)$.

If $\mathbf{D}_S = \mathbf{D}_T$, then $\mathbf{T}_S = \mathbf{T}_T$.

If $\mathbf{D}_S \neq \mathbf{D}_T$, then $\mathbf{T}_S \neq \mathbf{T}_T$ or $P(X_S) \neq P(X_T)$.

If $\mathbf{T}_S \neq \mathbf{T}_T$, then $\mathbf{Y}_S \neq \mathbf{Y}_T$ or $P(Y_S|X_S) \neq P(Y_T|X_T)$, $Y_S \in \mathbf{Y}_S, Y_T \in \mathbf{Y}_T$.

We have three kinds of transfer learning: Inductive learning [66], transductive learning, and unsupervised learning (e.g., clustering), their domains, tasks, and algorithms are all different. Correspondingly, there are multiple methods for transfer learning: Sample-based transfer learning, feature-based transfer learning, parameters-based transfer learning, etc.

Inductive transfer learning [44] aims at improving the target predictive function $f_T(\cdot)$ in $\mathbf{D}_T$ by using the knowledge in $\mathbf{D}_S$ and $\mathbf{T}_S$, where $\mathbf{T}_S \neq \mathbf{T}_T$. In inductive transfer learning, we transfer the samples, knowledge, and parameters.

Transductive learning [44] refers to the situation where all test data are required to be seen at training time, the learned model cannot be reused for future data. Transductive learning could be applied to knowledge and parameter transfer.

Transductive transfer learning aims at promoting the target predictive function $f_T(\cdot)$ in $\mathbf{D}_T$ by using the knowledge in $\mathbf{D}_S$ and $\mathbf{T}_S$, where $\mathbf{D}_S \neq \mathbf{D}_T$ and $\mathbf{T}_S = \mathbf{T}_T$.

Unsupervised transfer learning (e.g., clustering or dimensionality reduction) [44] aims to improve the target predictive function $f_T(\cdot)$ in $\mathbf{D}_T$ using the knowledge in $\mathbf{D}_S$ and $\mathbf{T}_S$, where $\mathbf{T}_S \neq \mathbf{T}_T$, $\mathbf{Y}_S$ and $\mathbf{Y}_T$ are not observable. No labeled data is observed in the source and target domains in training.

In transfer learning, a network is needed as well as training data and training algorithms. Training options should be specified when a network is being trained. Most of time, training data is needed, which needs huge workload to collect the training data and data augmentation. The training rate is related to how fast the network will be converged and what are the final trained parameters of the network. In transfer learning, we need to estimate and evaluate the outcomes of transfer learning. We need to ensure that the results will be much better [7].

Deep learning architectures mostly aim at the classification tasks and benefit from off-the-shelf models which can significantly simplify the model development

as they can be modified and adapted according to the new task [68]. Furthermore, this domain gains from transfer learning, where a model was built for a task that is used for another custom task [12]. In transfer learning, the model is trained based on a public dataset and the initial weights of this trained model are employed for the task instead of assigning random weights as conducted in a network designed from scratch. Usually, the last fully connected layer that is responsible for final classification is fine-tuned, i.e., presenting the images of new classification task to the network where weights of specific layers are adjusted as conducted in regular training process.

7.1.2 Taskonomy

Taskonomy is a word from the task taxonomy which refers to the rewarded paper: Disentangling Task Transfer Learning, published in IEEE CVPR 2018 [66]. Taskonomy is a fully computational approach for modeling the structure of visual tasks through transfer learning in a latent space. The word 'structure' means a collection of computational relations specifying which tasks supply useful information to another. The taxonomy was created using a four steps process:

- **Step 1**. **Task Specific Modeling**: A task-specific network for each task is trained.
- **Step 2**. **Transfer Modeling**: All feasible transfers between sources and targets are trained.
- **Step 3**. **Ordinal Normalization Using Analytic Hierarchy Process (AHP)**: The task affinities acquired from transfer function performances are normalized.
- **Step 4**. **Computing the Global Taxonomy**: A hypergraph is synthesized which can predict the performance of any transfer policy and optimize for the optimal one.

In taskonomy, the transfer operation is such a function that a small readout function $D_{s \rightarrow t}$ is trained to map representations of source task's frozen encoder to target task's labels.

Given a source task s and a target task t, where $s \in \mathbf{S}$ and $t \in \mathbf{T}$, a transfer network trains a small readout function for t given a statistic image computed for s.

$$D_{s \rightarrow t} = \arg \min_{\theta} \mathrm{E}_{I \in D}(\mathrm{L}_t(D_{\theta}(E_s(I)), f_t(I))) \qquad (7.1)$$

where $f_t(I)$ is ground truth of t for image I. $D_{s \rightarrow t}$ is parameterized by $\theta_{s \rightarrow t}$ minimizing the loss $\mathbf{L}_t$.

The dataset for taskonomy consists of 4 million images of indoor scenes from about 600 buildings, each image has an annotation for every task. Unknown labels were generated by using the methods of knowledge distillation [21].

The results of taxonomy are evaluated by using two metrics: Gain and quality. Gain is the win rate (%) against a network trained from scratch using the same training data as transfer networks. Quality is the win rate (%) against a fully supervised network that was trained with 120 K images.

7.2   Ensemble Learning

In statistics and machine learning, ensemble learning methods use multiple learning algorithms to obtain better predictive performance than any algorithms alone [42].

One classifier is not enough in machine learning, because it only reflects one aspect and has mistakes or shortcomings. Multiple learners work together will make the classification better. The previous weaker classifiers, after the ensemble learning, will be much stronger [49]. The classification results of a classifier at least should be above 50%, the stronger classification should be beyond 50%. The essential ensemble learning methods include averaging, weighted average, and majority voting [39].

Averaging is method based on a simple average over all the predictions from the different classifiers. Weighted average is based on the weights which are proportional to a classifier's capability and performance. In majority voting method, the prediction is based on the most frequent class.

A set of diverse learners differ in the decisions so that they complement each other [1]. We combine multiple learners and free ones for taking a decision [17]. (1) Draw random training sets from the given samples or training dataset (e.g., Bagging, etc.) (2) Train further base-learners (e.g., boosting, cascading, etc.) (3) Mix multiple experts.

$$y = f(d_1, d_2, \ldots, d_L | \Phi), \tag{7.2}$$

and

$$c = \underset{i=1,2,\ldots,K}{\arg \max} \; y_i, \tag{7.3}$$

where $f(\cdot)$ is the combining function with Φ denoting its parameters, c is the returned class number. Classifier combination rules include the operations $\sum(\cdot)$, $\max(\cdot)$, $\min(\cdot)$, $\prod(\cdot)$ and the simple voting $w_i = w_j \in \{1, 0\}$. If we use combination function $\sum(\cdot)$, mixing multiple learners is shown as

$$y_i = \sum_j w_j d_{ji}, \sum_j w_j = 1, w_j \geq 0. \tag{7.4}$$

We have multiple methods to combine learners together. The applications such as gait recognition will decide which way of the combination will be applied [16,61].

Stacking refers to stack standardization. All classes have the same attributes. Stacking combines multiple models and multiple attributes together. Stacking mixes multiple models via a meta-model [1]. The meta-model is trained based on outputs of the base-learners. Stacking achieves higher accuracy than using individual classifiers [4].

Stacking source code is available for public. WEKA also has the corresponding function. Besides, the ensemble learning methods which WEKA offers include Bagging, random forest, AdaBoost, voting. See the interface of WEKA with an ensembling method in Fig. 7.1.

Bootstrapping [1] is any test or metric that relies on random sampling with replacement. Bootstrapping performs inference about a sample from resampled data. Bootstrapping is a straightforward way to derive estimates of standard errors and confidence intervals.
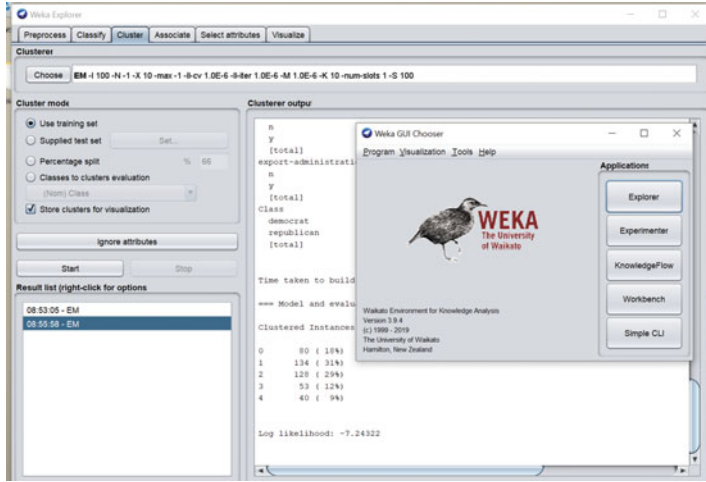
Fig. 7.1 An ensemble method provided by the famous machine learning software: WEKA

Bagging(i.e., Bootstrap Aggregating) [1] is a voting method whereby base learners are made differently by training them over various training sets.

Bagging has each model in the ensemble vote with equal weight. A learning algorithm is an unstable algorithm if the learning algorithm has high variance.

A learning algorithm is stable if different runs of the same algorithm based on the same dataset lead to learners with high positive correlation.

A forest is an ensemble of decision trees $\mathbf{F} = \{T_1, T_2, \ldots, T_n\}$, which delivers a prediction for a sample x by averaging the output of each tree,

$$p_{\mathscr{F}}(y|x) = \frac{1}{k} \sum_{h=1}^{k} p_{T_h}(y|x). \tag{7.5}$$

Boosting [1] is to generate complementary base learners by training the next learner boosting on the mistakes of the previous learners.

A boosting algorithm combines weak learners to generate a strong learner. Boosting involves incrementally building an ensemble learning by training each new model to emphasize the training that previous models misclassified. Reducing misclassification is an effective way for boosting classification. In the boosting classification, the weights of classifications are various.

Boosting is interpreted as an optimization algorithm based on a suitable cost function. For a training set $\{(x_i, y_i)\}, i = 1, \ldots, n,$

$$F(x) \stackrel{\Delta}{=} \sum_{i=1}^{M} \gamma_i \cdot h_i, \tag{7.6}$$

where h_i is the base learner.

$$\hat{F}(x) = \arg\max_{F} \mathbf{E}_{x,y}[L(y, F(x))], \tag{7.7}$$

where $L(\cdot)$ is the cost function.

By using the steepest descent $\nabla_{F_{m-1}} L(\cdot)$, $m = 1, 2, \ldots$, we have

$$F_m(x) = F_{m-1}(x) - \gamma_m \cdot \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)) \qquad (7.8)$$

where

$$\gamma_m = \arg\max_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i)) - \gamma \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))). \qquad (7.9)$$

Thus, the derivatives or gradients need to be calculated. This algorithm could be written in computable source code.

AdaBoost (i.e., Adaptive Boosting) makes use of the same training set over and over, the classifiers should be simplified so that they do not overfit [55]. The dataset should not be changed. AdaBoost is able to combine an arbitrary number of base learners based on weights. The success of AdaBoost is due to its salient property of increasing margin.

Cascading is a multistage method where there are a sequence of classifiers and the next one is applied only when the preceding ones are not confident [1]. Cascading has been applied to face detection, this ensemble learning could be applied to general visual object detection. Cascading generates a rule (or rules) to explain a large part of the instances as cheaply as possible and stores the rest as exceptions.

For ensemble learning, all open Python source codes are available from the scikit-learn.org website at https://scikit-learn.org/stable/modules/classes.html♯ module-sklearn.ensemble. See the screenshot of the scikit emsembling methods in Fig. 7.2.

Scikit-learn is a free software machine learning library for the Python programming language. The library was developed based on NumPy, SciPy, and matplotlib.

MATLAB meld results from many weak learners into one high-quality ensemble predictor by using ensemble learning. The methods include bootstrap aggregation (Bagging), random forest, boosting algorithms, etc. Boosting algorithms encapsulate adaptive boosting, gentle adaptive boosting, adaptive logistic regression, linear programming boosting, least squares boosting, robust boosting, random undersampling boosting, etc. See the screenshot of MATLAB ensemble methods based on regression tree [15] in Fig. 7.3.

The motivation for adopting ensemble learning classification is to boost the generalization of the system. Single learner may have a limited capability to capture the distribution of data; therefore, an aggregated decision of multiple weak learners can improve the learning capability of classification system by overcoming the limitation of a single weak learner. Ensemble learning draws final decision from multiple diverse learners that may improve the robustness of the system [18,54].

The diversity of base-learners for classification is the basis of ensemble learning which is incorporated in multiple ways [31]. Usually, it is achieved by using: (1) Diverse learning algorithms or with their different configurations, (2) manipulation of feature spaces, (3) subsampling training instances.

In ensemble learning, the diversity is integrated by combining base learners to learn various features. An averaging method is harnessed to combine the base learners
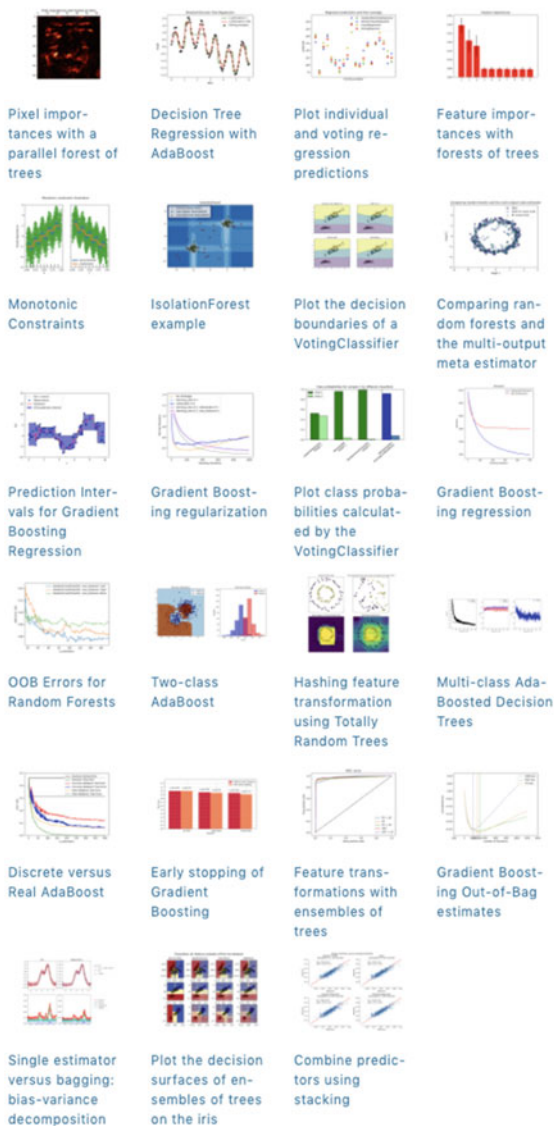
Fig. 7.2 Ensemble methods provided by scikit-learn
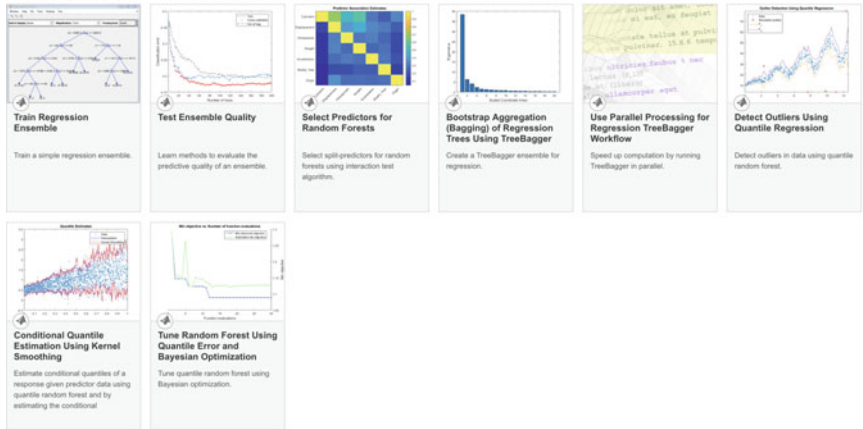
Fig. 7.3 Ensemble methods provided by MATLAB

and generate the final decision. Furthermore, weighted averaging method is also implemented and suitable weights were assigned to the base classifiers for improving classification outcomes.

Recently, with the development of deep learning, ensemble learning has been employed after the convolutional operations for improving the classification by combining outputs of deep learning classifiers [26,27]. Ensemble learning was employed for early AD diagnosis [2,25,56]. DenseNet-121, DenseNet-161, and DenseNet-169 were utilized as the base classifiers [26,27,43], then the results were ensembled by using the well-trained weights. In our project, our proposed method was evaluated based on a dataset from the Alzheimer's Disease Neuroimaging Initiative for the early diagnosis of this illness [26,27].

Deep Belief Networks (DBN) were proffered as the base classifiers, the final prediction was determined by using a voting scheme [39].

In our project of the early diagnosis of AD, we proposed an ensemble classifier based on the ConvNets [26]. ResNet-50, NASNet, and MobileNet as learners are combined together. The base learners were trained by using the end-to-end process instead of directly extracting convolutional features for training classifiers. The ensemble learning was implemented based on the base classifiers to improve the performance of early diagnosis of AD.

The training of base classifier includes two stages. Firstly, the ConvNet is fixed as we train the fully connected network as a classifier in order to fit the weights of the fully connected layers classifier. The fully connected layer on the top was randomly initialized, and a large weight updates would be propagated. Otherwise, the representation that was previously trained by using the ConvNet will be modified. The base classifier will be trained based on fine tuning between fully connected layers and released layers of ConvNet to fit more features on the early diagnosis of AD by

using the end-to-end process. These base classifiers are eResNet-50, eNASnet, and eMobileNet.

We trained our base classifiers by using the end-to-end process. We trained the base classifier in order to fit the model more robust. In the end, the output of the base learner ConvNet was combined to improve the accuracy and stability.

In the project regarding human behavior recognition [36,37], we explore and exploit the state-of-the-art methods for human behavior recognition. More importantly, in order to attain our goal, spatiotemporal information was collected and employed to the implementation of our research project. We adopted ensemble learning with deep learning methods. We proposed Selective Kernel Network (SKNet) and ResNeXt with attention mechanism, which generate positive results to recognize human behaviors.

In the project related to human gait recognition [61], we put forward an effective method of gait recognition: Cross-view gait recognition based on ensemble learning. The proposed method greatly enhances the effectiveness and reduces the sensitivity of gait recognition under various view angles conditions. An algorithm based on ensemble learning combines several gait learners together, which utilizes a well-designed gait feature based on area average distance. The contribution of this research work is to resolve the multiview angles problem of gait recognition through assembling several gait learners.

7.3 Knowledge Distillation

Conducting classification or making predictions using a whole ensemble of models is cumbersome due to a large number of users. It is possible to compress the knowledge in an ensemble into a single model. A possible way is to distilling Knowledge in an ensemble of models into a single model [21]. Neural networks typically produce class probabilities by using a 'softmax' output layer,

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{7.10}$$

where T is a temperature that is normally set to 1.0, q_i is a probability of each class.

The general solution 'distillation' is to raise the 'temperature' of the final softmax until the cumbersome model produces a suitably soft set of targets. We train the small model to match these soft targets.

The distillation strategy achieves the desired effect of distilling an ensemble of models into a single model that works significantly better than a model of the same size that is learned directly from the same training data. The distillation approach is able to extract more useful information from the training set than simply using the hard labels to train a single model. The distilling works very well for transferring knowledge from an ensemble or from a large highly regularized model into a smaller and distilled model.

Exercises

Question 7.1. What's knowledge transfer? How to use the learned knowledge in transfer learning?

Question 7.2. Based on data labels, how to group transfer learning models?

Question 7.3. How to train multilabel classification?

Question 7.4. What are the applications of deep neural networks in ensemble learning?

Question 7.5. How to understand reinforcement learning is the third machine learning method which is different from supervised learning and unsupervised learning?

Question 7.6. Please list data dimensionality reduction methods.

References

1. Alpaydin E (2009) Introduction to machine learning. MIT Press
2. Andres O, Munilla J, Gorriz J et al (2016) Ensembles of deep learning architectures for the early diagnosis of the Alzheimer's disease. Int J Neural Syst 26(7)
3. Andrews PL, Loftsgaarden DO, Bradshaw LS (2003) Evaluation of fire danger rating indexes using logistic regression and percentile analysis. Int J Wildland Fire 12(2):213–226
4. Ayerdi B, Savio A, Graña M (2013) Meta-ensembles of classifiers for Alzheimer's disease detection using independent ROI features. LNCS 7931:122–130
5. Baum LE, Petrie T (1966) Statistical inference for probabilistic functions of finite state Markov chains. Ann Math Stat 37(6):1554–1563
6. Baum LE, Sell G (1968) Growth transformations for functions on manifolds. Pac J Math 27(2):211–227
7. Bird J, Kobylarz J, Faria D, Ekart A, Ribeiro E (2020) Cross-domain MLP and CNN transfer learning for biological signal processing: EEG and EMG. IEEE Access 8:54789–54801
8. Bozinovski S (2020) Reminder of the first paper on transfer learning in neural networks. Informatica 44:291–302
9. Collobert R, Weston J (2008) A unified architecture for natural language processing: deep neural networks with multitask learning. In: International conference on machine learning, pp 160–167
10. Cho K (2013) Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images. In: International conference on machine learning, pp 432–440
11. Dabney W et al (2020) A distributional code for value in dopamine-based reinforcement learning. Nature 577:671–675
12. Do C, Ng A (2005) Transfer learning for text classification. In: Neural information processing systems foundation
13. Dosovitskiy A et al (2021) An image is worth 16×16 words: transformers for image recognition at scale. In: International conference on learning representations

14. Einicke G, White L (1999) Robust extended Kalman filtering. IEEE Trans Signal Process 47(9):2596–2599
15. Gashler M, Giraud-Carrier C, Martinez T (2008) Decision tree ensemble: small heterogeneous is better than large homogeneous. In: International conference on machine learning and applications, pp 900–905
16. Guan Y, Li C, Roli F (2015) On reducing the effect of covariate factors in gait recognition: a classifier ensemble method. IEEE Trans Pattern Anal Mach Intell 37(07):1521–1529
17. Polikar R (2006) Ensemble-based systems in decision making. IEEE Circuits Syst Mag 6(3):21–45
18. Gomes H, Barddal J, Enembreck A, Bifet A (2017) A survey on ensemble learning for data stream classification. ACM Comput Surv
19. Ham Y, Kim J, Luo J (2019) Deep learning for multi-year ENSO forecasts. Nature 573:568–572
20. He K, Gkioxari G, Dollar P, Girshick R (2017) Mask R-CNN. In: ICCV, pp 2980–2988
21. Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. Statistics 1050:9
22. Hinton G, Salakhutdinov R (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
23. Hoo-Chang S, Roth H, Gao M, Lu L, Xu Z, Nogues I, Summers RM (2016) Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE Trans Med Imaging 35(5):1285
24. Huang G, Liu Z, Weinberger KQ, van der Maaten L (2017) Densely connected convolutional networks. In: IEEE CVPR, p 3
25. Islam J, Zhang Y (2017) An ensemble of deep convolutional neural networks for Alzheimer's disease detection and classification. In: NIPS workshop on machine learning for health
26. Ji H, Liu Z, Yan W, Klette R (2019) Early diagnosis of Alzheimer's disease using deep learning. In: ICCCV, pp 87–91
27. Ji H, Liu Z, Yan W, Klette R (2019) Early diagnosis of Alzheimer's disease based on selective kernel network with spatial attention. ACPR 2:503–515
28. Jordan MI, Mitchell TM (2015) Machine learning: trends, perspectives, and prospects. Science 349(6245):255–260
29. Karimpanal T, Bouffanais R (2019) Self-organizing maps for storage and transfer of knowledge in reinforcement learning. Adapt Behav 27(2):111–126
30. Klette R (2014) Concise computer vision: an introduction into theory and algorithms. Springer, London, UK
31. Kuncheva L, Whitaker C (2003) Measures of diversity in classifier ensembles. Mach Learn 51:181–207
32. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444
33. Lin TY, Dollar P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: IEEE CVPR, vol 1, no 2, p 4
34. Littman M (2015) Reinforcement learning improves behavior from evaluative feedback. Nature 521:445–451
35. Liu Z et al (2021) Swin Transformer: hierarchical vision transformer using shifted windows. In: IEEE ICCV
36. Lu J (2016) Empirical approaches for human behavior analytics. Master's thesis, Auckland University of Technology, New Zealand
37. Lu J (2021) Deep learning methods for human behavior recognition. PhD thesis, Auckland University of Technology, New Zealand
38. Mnih V et al (2015) Human-level control through deep reinforcement learning. Nature 518:529–533
39. Mu X, Lu J, Watta P, Hassoun M (2009) Weighted voting-based ensemble classifiers with application to human face recognition and voice recognition. In: International joint conference on neural networks, pp 2168–2171

40. Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Networks 1(1):4–27
41. Niculescu-Mizil A, Caruana R (2007) Inductive transfer for Bayesian network structure learning
42. Opitz D, Maclin R (1999) Popular ensemble methods: an empirical study. J Artif Intell Res 11:169–198
43. Ortiz A, Munilla J, Gorriz M (2016) Ensembles of deep learning architectures for the early diagnosis of the Alzheimer's disease. Int J Neural Syst 26(7):1132–1140
44. Pan S, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
45. Pratt LY (1993) Discriminability-based transfer between neural networks. In: Advances in neural information processing systems, pp 204–211
46. Rajat R, Ng A, Koller D (2006) Constructing informative priors using transfer learning
47. Reddy G et al (2018) Glider soaring via reinforcement learning in the field. Nature 562:236–239
48. Reichstein M et al (2019) Deep learning and process understanding for data-driven Earth system science. Nature 566:195–204
49. Rokach L (2010) Ensemble-based classifiers. Artif Intell Rev 33(1–2):1–39
50. Ronneberger O, Fischer P, Brox T (2015) U-Net: convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention. Springer, pp 234–241
51. Rriedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting. Ann Stat 38(2):337–374
52. Rumelhart D, Hinton G et al (1986) Learning representations by back-propagating errors. Nature 323:533–536
53. Shrivastava A et al (2017) Learning from simulated and unsupervised images through adversarial training. In: IEEE CVPR
54. Sohail A, Khan A, Nisar H, Tabassum S, Zameer A (2021) Mitotic nuclei analysis in breast cancer histopathology images using deep ensemble classifier. Med Image Anal 72
55. Sollich P, Krogh A (1996) Learning with ensembles: how overfitting can be useful. In: Advances in neural information processing systems, pp 190–196
56. Suk H, Lee S, Shen D (2017) Deep ensemble learning of sparse regression models for brain disease diagnosis. Med Image Anal 37:101–113
57. Sutton R, Barto A (2018) Reinforcement learning: an introduction, 2nd edn. MIT Press
58. Takeda F, Omatu S (1995) A neuro-paper currency recognition method using optimized masks by genetic algorithm. In: IEEE international conference on systems, man and cybernetics, vol 5, pp 4367–4371
59. Valmadre J, Bertinetto L, Henriques J, Vedaldi A, Torr P (2017) End-to-end representation learning for correlation filter based tracking. In: IEEE CVPR, pp 2805–2813
60. Vinyals O et al (2019) Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575:350–354
61. Wang X, Yan W (2020) Multi-perspective gait recognition based on ensemble learning. Springer Neural Comput Appl 32:7275–7287
62. Webb S (2018) Deep learning for biology. Nature 554:555–557
63. Wei J, He J, Chen K, Zhou Y, Tang Z (2017) Collaborative filtering and deep learning based recommendation system for cold start items. Expert Syst Appl 69:29–39
64. Welling M, Kingma D (2019) An introduction to variational autoencoders. Found Trends Mach Learn 12(4):307–392
65. Yao W, Zeng Z, Lian C, Tang H (2018) Pixel-wise regression using U-Net and its application on pansharpening. Neurocomputing 312:364–371
66. Zamir A et al Taskonomy: disentangling task transfer learning. In: IEEE CVPR
67. Zarchan P, Musoff H (2000) Fundamentals of Kalman filtering: a practical approach. American Institute of Aeronautics and Astronautics, Incorporated

68. Zoph B, Vasudevan V, Shlens J, Le Q (2018) Learning transferable architectures for scalable image recognition. In: IEEE CVPR, pp 8697–8710
69. Zhu B et al (2018) Image reconstruction by domain-transform manifold learning. Nature 555:487–492

Glossary

Activation function   In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs.

AdaBoost   Adaptive Boosting, a voting method for training a boosted classifier.

Autoencoder   An autoencoder is a neural network that learns to copy its input to its output.

Average pooling   Calculating the average for each patch of the feature map.

Atlas   A specific collection of charts which covers a manifold.

Bagging   Bootstrap AGGregatING, a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.

Banach spaces   Complete normed vector spaces.

Bayesian inference   A method of statistical inference in which Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available.

Bayesian learning   Using Bayes' theorem to determine the conditional probability of hypotheses given evidence or observations.

Bayesian network   A decision network, which is a type of statistical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG).

Boltzmann machine   Stochastic Hopfield network with hidden units, a type of stochastic recurrent neural network.

Boltzmann distribution   The distribution maximizes the entropy.

Boosting   An ensemble meta-algorithm for primarily reducing bias, and also variance [1] in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones.

Bootstrapping   Test or metric that uses random sampling with replacement.

CapsNet   Capsule Neural Network, which is a type of artificial neural network (ANN) for better modeling hierarchical relationships of an object.

Capsule   A set of neurons that are individually activated for various properties of a type of objects.

Cascading   A particular case of ensemble learning based on the concatenation of several classifiers, using all information collected from the output from a given classifier as additional information for the next classifier.

Chart   An invertible map between a subset of the manifold and a simple space such that both the map and its inverse preserve the desired structure.

Clique tree   The junction tree algorithm, a method in machine learning to extract marginalization from general graphs.

CNN   Convolutional neural network.

Convex   The line segment between any two points on the graph of the function lies one side of the graph between the two points.

ConvNet   Convolutional neural network.

Convolution   A mathematical operation for two functions produces a third function expressing how the shape of one is modified by the other.

DAG   Directed acyclic graph, a finite directed graph with no directed cycles.

DBM   Deep Boltzmann machine, a type of binary pairwise Markov random field which is a undirected probabilistic graphical model with multiple layers of hidden random variables.

Decision tree   A decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.

Decision rule   A function which maps an observation to an appropriate action.

Deep learning   Deep neural network has powerful ability of nonlinear processing using a cascade of multiple layers network for feature transformation and end-to-end learning.

DRL   Deep reinforcement learning, using deep learning and reinforcement learning principles to create efficient algorithms.

Double Q-learning   An off-policy reinforcement learning algorithm, where a different policy is used for value evaluation than what is used to select the next action.

Dynamic Bayesian network   A Bayesian network represents sequences of variables.

EKF   Extended Kalman filter, nonlinear Kalman filter which linearizes about an estimate of the current mean and covariance.

Ensemble learning   The process by which multiple models are strategically generated and combined to solve a particular computational intelligence problem.

Entropy   A measure of the unpredictability of the state, or equivalently, of its average information content.

Event   In textual topic detection and extraction, an event is something that happened somewhere at a certain time.

Exponential family   A set of distributions, where the specific distribution varies with the parameter.

Factorization   A product of factors over cliques in the graph.

Fourier transform   A mathematical transform that decomposes a function into its constituent frequencies.

Fuzzy optimization   A mathematical model which deals with transitional uncertainty and information deficiency uncertainty.

GCD   The greatest common divisor of two or more integers, which are not all zero, is the largest positive integer that divides each of the integers.

Genetic algorithm   A metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms.

Gibbs distribution   A probability distribution or probability measure.

Gibbs measure   The unique statistical distribution that maximizes the entropy for a fixed expectation value of the energy.

Global optimization   The task of finding the absolutely best set of admissible conditions to achieve your objective, formulated in mathematical terms.

Hausdorff space   A topological space in which each pair of distinct points is separated by a disjoint open set.

Hilbert spaces   An inner product space which is complete as a metric space.

Induced subgraph $G[S]$ is the graph whose vertex set is S and whose edge set consists of all of the edges in E that have both endpoints in $S \subset G = (V, E)$.

Influence diagram   A compact graphical and mathematical representation of a decision situation.

Isometry   congruence, a distance-preserving transformation between metric spaces, usually assumed to be bijective.

Joint entropy   A measure of the uncertainty associated with a set of variables.

Kalman filter   The optimal linear estimator for linear system models with additive independent white noise in both the transition and the measurement systems.

KL divergence   Kullback–Leibler divergence or relative entropy, a measure of how one probability distribution is different from a second and reference probability distribution.

Latent variables   The variables that are not directly observed but are rather inferred (through a mathematical model) from other variables that are observed (directly measured).

LCM   The smallest common multiple of two integers a and b is the smallest positive integer that is divisible by both a and b.

Lipschitz continuity   A strong form of uniform continuity for functions.

Linear Dynamical System   A dynamic Bayesian network where all of the dependencies are linear Gaussian.

Linear programming   A technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints.

LSTM   Long short-term memory network.

MAP   Maximum a posteriori probability estimate, an estimate of an unknown quantity, that equals the mode of the posterior distribution.

Manifold   A topological space with the property that each point has a neighborhood, a second countable Hausdorff space that is locally homeomorphic to Euclidean space.

Manifold learning   An approach for nonlinear dimensionality reduction.

Markov chain   A stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

Markov process   A stochastic process that satisfies the Markov property.

Max pooling   Applying a max filter to non-overlapping subregions of the initial representation.

MDP   Markov decision process, a discrete-time stochastic control process.

Metadata   Data about data, namely additional information of a given set of data.

Metric   A function that defines a concept of distance between any two members of the set, which are usually called points.

Metric space   A set together with a metric on the set.

MGU   Minimal Gated Unit.

MLE   Maximum likelihood estimation, a method of estimating the parameters of a probability distribution by maximizing a likelihood function, under the assumed statistical model, the observed data is most probable.

MNIST   The Modified National Institute of Standards and Technology (NIST) dataset.

MRF   Markov random field which is a set of random variables having a Markov property described by an undirected graph.

Multiobjective programming   A part of mathematical programming dealing with decision problems characterized by using multiple and conflicting objective functions that are to be optimized over a feasible set of decisions.

Mutual information   A measure of the mutual dependence between the two variables.

Naive Bayes Model   A family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions.

NLAR   Nonlinear autoregressive model.

Norm   A real-valued function defined on the vector space.

Normed space   A vector space over the real or complex numbers, on which a norm is defined.

Observable variable   Manifest variable in statistics, a variable that can be observed and directly measured.

Orbifold   A generalization of manifold allowing for 'singularities' in the topology.

Padding   The filled region of an image boundary is applied to fill up the edge region with zero.

Parameter estimation   The process of using sample data to estimate the parameters of the selected distribution.

Particle swarm optimization   A computational method that optimizes a problem by iteratively improving a candidate solution with regard to a given measure of quality with a population of candidates, moving the particles in the search-space according to the position and velocity.

Partition functions   A generating function for expectation values of various functions of the random variables.

PGM Probabilistic graphical model or structured probabilistic model which is a model for a graph to express the conditional dependence structure between random variables.

Q-learning A model-free reinforcement learning algorithm to learn a policy telling an agent what action to take under what circumstances.

Random forests An ensemble learning method for classification and regression by constructing a multitude of decision trees at training time and outputting the class that is the mode of classification regression of the individual trees.

RBM Restricted Boltzmann machine, a variant of Boltzmann machine with the restriction that their neurons must form a bipartite graph.

Regularization The process of adding information in order to solve an ill-posed problem or to prevent overfitting.

Reinforcement learning Approximate dynamic programming or neuro-dynamic programming, which is teaching a software agent how to behave in an environment by telling it how good it's doing.

ResNet Deep residual network.

Reward function A function defines the goal for an agent.

RNN Recurrent neural network.

Roots of a polynomial Those values of the variable that cause the polynomial to evaluate to zero.

SARSA State-action-reward-state-action, an algorithm for learning a Markov decision process policy, used in the reinforcement learning area of machine learning.

Siamese neural network Twin neural network, an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors.

Single shot The tasks of visual object localization and classification are done in a single forward pass of the network.

Squashing function A function that squashes the input to one of the ends of a small interval.

SSD Single shot multibox detector.

Stride The step length of convolution operations.

Target variable The variable whose values are to be modeled and predicted by other variables.

Temporal difference learning A class of model-free reinforcement learning methods which learn by bootstrapping from the current estimate of the value function.

Tensor A generalization of vectors and matrices to potentially higher dimensions.

TensorFlow A framework to define and run computations involving tensors, represent tensors as n-dimensional arrays of base datatypes.

Time series analysis Analyzing time series data in order to extract meaningful statistics and other characteristics of the data.

Time series forecasting The use of a model to predict future values based on previously observed values.

Transfer function This function is used for transformation purposes, from input nodes to the output of a neuron.

Transfer learning   A machine learning method where a model is developed for a
task which is reused as the starting point for a model on a second task.

Transformer   A deep learning model that adopts the mechanism of self-attention,
differentially weighing the significance of each part of the input data.

YOLO   You only look once.

Names in This Book

Stefan **Banach**   (1892–1945)
Leonard Esau **Baum**   (1931–2017)
Reverend Thomas **Bayes**   (1701–1761)
Richard Ernest **Bellman**   (1920–1984)
Yoshua **Bengio**   (1964–)
Augustin-Louis **Cauchy**   (1789–1857)
Dennis **Gabor**   (1900–1979)
Johann Carl Friedrich **Gauss**   (1777–1855)
Evariste **Galois**   (1811–1832)
Josiah Willard **Gibbs**   (1839–1903)
Jean-Baptiste Joseph **Fourier**   (1768–1830)
Jacques Salomon **Hadamard**   (1865–1963)
Felix **Hausdorff**   (1868–1942)
David **Hilbert**   (1862–1943)
Geoffrey **Hinton**   (1947–)
Paul **Jaccard**   (1868–1944)
Johan Ludwig William Valdemar **Jensen**   (1859–1925)
Nikola **Kasabov**   (1948–)
Rudolf Emil **Kalman**   (1930–2016)
Reinhard **Klette**   (1950–2020)
Joseph-Louis **Lagrange**   (1736–1813)
Yann Andre **LeCun**   (1960–)
Rudolf Otto Sigismund **Lipschitz**   (1832–1903)
Prasanta Chandra **Mahalanobis**   (1893–1972)
Andrey Andreyevich **Markov**   (1856–1922)
David Courtenay **Marr**   (1945–1980)
Karl **Pearson**   (1857–1936)

Andrey Nikolayevich **Tikhonov**   (1906–1993)
Marc-Antoine **Parseval**   (1755–1836)
Andrea Giacomo **Viterbi**   (1935–)
Brook **Taylor**   (1685–1731)
Alan **Turing**   (1912–1954))
Lloyd Richard **Welch**   (1927–)

Index

A

Abnormal detection, 110
Absolute loss function, 13, 89
Acceptance-rejection method, 102
ACM Turing award, 9
Activation function, 90, 91, 110
AdaBoost, 8, 15, 196
Adam optimizer, 105
Adaptive boosting, 196
Adaptive logistic regression, 196
AdBoost, 192
Adding noises, 52
Additive seasonality, 109
Adjacency matrix, 177, 181, 186
Adjacency tensor, 180
Affine transformation, 53
Affinity matrix, 166
Age estimation, 17
Aging recognition, 47
AlexNet, 9, 10, 46, 72
Algebra, 16
AlphaStar, 29
Alzheimer's disease, 20
Analytic Hierarchy Process, 193
Analytic manifold, 163
Anchor box, 74, 75, 78
ANN toolbox, 45
Anomaly detection, 110
Answer intent, 101
Ant colony optimization, 154
Applied mathematics, 143
Approximate inference, 168
Area Under the Curve (AUC), 45, 46

ARIMA, 46
ARMA, 108
Artifact removal, 52
Artificial intelligence, 47
Artificial Neural Network (ANN), 45, 110
Associative, 58
Associativity, 113
Atlas, 163
Attention mechanism, 20
Attention network, 45, 47
Autocorrelation, 108
Autocorrelation analysis, 46
Autocorrelation correlogram, 108
Autocovariance, 108
Autocovariance lag, 108
Autoencoder, 3, 13, 30, 31, 130
Autonomous vehicle, 47
Autoregressive (AR), 46
Autoregressive integrated moving average, 46
Autoregressive language model, 100
Average cost function, 89
Average loss function, 13
Average pooling, 73

B

Backward pass, 2, 30
Bagging, 15, 194–196
Banach space, 115
Bandit problem, 143
Bart, 93
Basic algebra, 16
Batch normalization, 76
Baum–Welch algorithm, 86