МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп’ютерних систем та мереж

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
__________________ Жуков І.А.
«_____» _____________ 2020 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ
«МАГІСТР»

Тема: Система розпізнавання графічних об’єктів

Виконав: __________________________ Клименков Станіслав Віталійович

Керівник: __________________________ Надточій В.І.

Нормоконтролер з ЄСКД: ___________ Надточій В.І.

Київ 2020

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL AVIATION UNIVERSITY

Computer Systems and Networks Department

PERMISSION TO DEFEND GRANTED
The Head of the Department
__________________ Zhukov I.A.
«_____» _____________ 2020

GRADUATION PROJECT

(EXPLANATORY NOTE)

GRADUATE ACADEMIC DEGREE "MASTER"

Specialty – 6.050102 "Computer Engineering"

Topic: Graphic object recognition system

Completed by: ______________________ Klymenkov S.V.

Supervisor: _________________________ Nadtochii V.I.

Standards Inspector: __________________ Nadtochii V.I.

Kyiv 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп’ютерної та програмної інженерії

Кафедра Комп’ютерних систем та мереж

Освітньо-кваліфікаційний рівень Магістр

Спеціальність: 6.050102 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
______________ Жуков І.А.
“_____” ___________ 2020 р.

ЗАВДАННЯ

на виконання дипломного проекту

Клименкова Станіслава Віталійовича

1. Тема дипломного проекту «Система розпізнавання графічних об’єктів» затверджена наказом ректора від 25 вересня 2020 р. № 1793/ст.

2. Термін виконання роботи (проекту): з 1 жовтня 2020 р. до 25 грудня 2020 р.

3. Вхідні дані до роботи (проекту): Схематичне графічне зображення структури згорткової нейронної мережі та алгоритму її роботи у вигляді блок-схеми. Порівняння показників точності розпізнавання графічних об’єктів представлено у вигляді графіків.

4. Зміст пояснювальної записки: Вступ, огляд теми, огляд існуючих технологій розпізнавання графічних об’єктів та будування штучних нейронних мереж, розробка згорткової нейронної мережі, висновки по роботі.

5. Перелік обов’язкового графічного (ілюстративного) матеріалу: Матеріали представлені у вигляді презентації в Power Point.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Узгодження технічного завдання з керівником проекту	1.10-8.10.20	
2	Підбір та вивчення науково-технічної літератури за темою дипломного проекту	9.10-15.10.20	
3	Розроблення Розділу 1 ПЗ	17.05-21.05.19	
4	Розроблення Розділу 2 ПЗ	22.05-27.05.19	
5	Розроблення Розділу 3 ПЗ	28.05-30.05.19	
6	Розроблення Розділу 4 ПЗ		
7	Роздрукування ПЗ, отримання відгуку керівника і проходження нормоконтролю	31.05-05.06.19	
8	Підготування презентації та тексту доповіді	05.06-10.06.19	
9	Отримання рецензії, усунення зауваження, подання документації секретарю ЕК	10.06-15.06.19	
10	Захист	20.06.19	

7. Дата видачі завдання: 1 жовтня 2020 р.

Керівник дипломної роботи ____________________ Надточій В.І.

Завдання прийняв до виконання ________________ Клименков С.В.

NATIONAL AVIATION UNIVERSITY

Faculty of Cybersecurity, Computer and Software Engineering

Department: Computer Systems and Networks _____________

Educational and Qualifications level: Master Degree ________

The specialty: 6.050102 “Computer Engineering” ___________

APPROVED BY
The Head of the Department
______________ Zhukov I.A.
“_____” _____________ 2020

Graduate Student’s Degree Project Assignment

Klymenkov Stanislav Vitaliiovych

1. The Project theme: «Graphic Object Recognition System»

Approved by the Rector’s order of 25.09.2020 № 1793/st.

2. The Thesis to be completed between 1.10.2020 and 25.12.2020

3. Input data for the project (thesis): Schematic graphic representation of the structure of convolutional neural network and its working algorithm in a form of a flowchart. The comparison between accuracy parameters after graphic object recognition is represented in the form of charts.

4. The content of the explanatory note (the list of problems to be considered): Introduction, overview of the topic, review of existing technologies for graphic object recognition and building convolutional neural networks, design of the convolutional neural network, conclusions on work.

5. The list of mandatory graphical materials: Materials representation in the form of Power Point Presentation.

6. Timetable

#	Completion Stages of Degree Project	Stage Completion Dates	Remarks
1	Technical task coordination with the supervisor	13.05-14.05.19	
2	Selection and study scientific literature on the topic	15.05-16.05.19	
3	Preparing Part 1 of EN	17.05-21.05.19	
4	Preparing Part 2 of EN	22.05-27.05.19	
5	Preparing Part 3 of EN	28.05-30.05.19	
6	Preparing Part 4 of EN		
7	Printing EN, receiving supervisor's comment and completing standards inspection	31.05-05.06.19	
8	Preparing presentation and report text	05.06-10.06.19	
9	Receiving review, correcting mistakes, delivering documentation to the secretary	10.06-15.06.19	
10	Defense	20.06.19	

7. Assignment issue date: 25.12.2020

Diploma Thesis Supervisor ____________________ Nadtochii V.I.

Assignment accepted for completion ____________ Klymenkov S.V.

ABSTRACT

The Explanatory Note to on Master’s Degree Graduation Project – “Graphic Object Recognition System”: 92 pages, 38 figures, 9 tables, references, 9 appendices.

GRAPHIC OBJECT RECOGNITION, PATTERN RECOGNITION, CLASSIFICATION, CONVOLUTIONAL NEURAL NETWORK.

The Goal of Graduation Project: The goal of this project is to examine the existing technologies of graphic object recognition and artificial neural networks, and build a convolutional neural network for graphic object recognition.

Main Tasks: Design and successful training of a convolutional neural network for graphic object recognition tasks.

The Subject of Project: Examining the existing methods of graphic object recognition with the help of artificial neural networks, finding shortcomings and proposing ways to overcome them. Using technologies, which have already been studied, in order to build a convolutional neural network for solving graphic object recognition tasks.

Practical usage: Can be used in any detection or graphic object recognition system, the proposed methods to improve performance will help to maintain more accurate classification, detection and recognition.

Main metrics and Results: Graphic pattern recognition has been drawing attention for a long time of specialists in the field of applied mathematics and then computer sciences. The main problem of graphic object recognition is the human ability to classify and generalize objects – this mechanism still presents a huge challenge for researchers, which try to solve this problem by means of artificial neural networks. The project aims to find and analyze the existing methods of graphic object recognition by means of artificial neural networks. The basic model of a neural network devoted to solve the tasks of pattern recognition was designed, and methods of improving its performance were proposed.

CONTENT

CSN department				NAU 19 74 09 000 EN			
Done by	Klymenkov S.V.			Graphic object recognition system	Letter	Sheet	Sheets
Supervisor	Nadtochii V.I.				E	8	109
Consultant					6.050102 CS-422a		
S-inspector	Nadtochii V.I.						
Head of dep.	Zhukov I.A.						

LIST OF SYMBOLS, ABREVIATIONS, TERMS

ACAM – associative content addressable memory

ADALINE – Adaptive Linear Element (former Adaptive Linear Neuron)

ANN – artificial neural network

API – Application Programming Interface

AdaBoost – adaptive boosting

CIFAR - Canadian Institute For Advanced Research

CNN – convolutional neural network

CPU – central processing unit

Colliculus cranialis – structure lying on the roof of the midbrain

DL – deep learning

ECG – electrocardiogram

FFT – fast Fourier transform

FNN – feedforward neural network

GPU – graphics processing unit

gRPC – remote procedure call

GTSRB – German Traffic Sign Recognition Benchmark

HASYv2 – Handwritten Symbols version 2

HOG – histograms of oriented gradients

HPF – high-pass filter

HSV – “Hue, Saturation, Value” color model

LPF – low-pass filter

MADALINE – a multilayer network of ADALINE

ML – machine learning

MNIST – Mixed National Institute of Standards and Technology

NIST – National Institute of Standards and Technology

OpenCV – Open Source Computer Vision Library

PRP – pattern recognition problem

RBF network – a radial basis function network

RGB – “Red, Green, Blue” color model

RNN – recurrent neural network

ReLU – rectified linear unit

SIFT – scale-invariant feature transform

STL-10 – self-taught learning 10

SURF – Speeded Up Robust Features

SVHN – Street View House Numbers

TSP – travelling salesman problem

INTRODUCTION

During a rather long period of time graphic pattern recognition has been considered in terms of biological and psychological aspects. Furthermore, the subject of investigation included primarily qualitative, which do not allow giving an accurate description of the functioning mechanism.

As a rule, functional dependencies were mainly related to the investigation of visual, audial and haptic receptors. However, the principle of decision-making remained unsolved. The main mistake was considered the conception, that a human brain operates according to certain algorithms, thus discovering this set of rules could be recreated by means of computing and other technical devices.

Founded by Norbert Wiener at the beginning of the XX century, a new discipline called cybernetics (the science about general patterns of data transferring between machines, life forms and societies) allowed introducing quantitative methods into examining the graphic pattern recognition problem. In other words, the idea was to present the process of image recognition (being actually a natural phenomenon) by means of mathematical methods.

In most cases devices, which perform functions of diverse graphic pattern, recognition gives an opportunity to replace a human being with a specialized machine. Owing to this, complex data, logic and analytical systems receive a breakthrough in terms of possibilities. It is generally understood, that working productivity in case of people depends on many factors (competence, experience, consciousness, so on and so forth).

CSN department				NAU 19 74 09 000 EN			
Done by	Klymenkov S.V.			Graphic object recognition system	Letter	Sheet	Sheets
Supervisor	Nadtochii V.I.				E	12	109
Consultant					6.050102 CS-422a		
S-inspector	Nadtochii V.I.						
Head of dep.	Zhukov I.A.						

At the same time, any machine is supposed to operate monotonously with the same quality at each iteration. Automatic control over complex systems allows conducting monitoring and maintaining in-time service engineering, obstacle identification and self-operating exploitation of noise suppression methods or increasing the quality of data transmission. Apparently, the application of automated systems can maintain the needed processing speed in specific tasks, which is impossible for human.

Graphic pattern recognition has been drawing attention for a long time of specialists in the field of applied mathematics and then computer sciences. Made in 20s, works of Ronald Fischer were praised and recognized as a prominent part of pattern recognition theory.

In 50-60s of the XX century, a great bulk of correspondent works and conceptions became the basis for the decision theory. In the realm of the theory various algorithms were discovered – those ones which maintained the attribution of a new pattern to one of the given classes, which became the beginning of a new systematic scientific approach and practical prototypes.

In the realm of cybernetics, a new scientific discipline began to form, related to the development of theoretical basics and practical implementation of devices and then systems, dedicated to pattern, phenomenon and process recognition. This new scientific discipline was named “Pattern recognition”.

A human can consciously comprehend information after a long cycle of preliminary processing. First, the light hits an eye. Through the whole optical systems photons at last reach the retina – a layer of photosensitive cells, called rods and cones.

Here, still very far from the brain, the first stage of data processing occurs. Information transfers to the human brain via optic nerves into so-called “colliculus cranialis”, on which visual information is projected. Then visual information reaches brain regions, which distinguish separates components – horizontal, vertical, diagonal lines, edges, areas of light, shadow and/or color. Gradually patterns become more complex and blurred, but the graphic image will go through a long way until it reaches the level of consciousness.

The architecture of convolutional neural network (CNN) is designed specifically for efficient pattern recognition. In its core, there lies the work of convolutional layers called convolutions with non-linear activation functions like rectified linear unit (ReLU) or hyperbolic tangent *tanh* and connecting layers, called pooling layers.

On the contrary to feedforward neural networks (FNN), where each input neuron is connected with the output neuron of the next layer, CNN use convolutions over each input layer in order to receive output values. The convolution operation requires the matrix of weights of small size running through the current layer and after each shift forming the activation signal for next layer neuron in analogical position. This matrix is known as the convolutional core; it is applied for different neurons of the output layer. The result of CNN work depends on quality and quantity of learning data, the type of ANN architecture and quantity of parameters chosen for the given network. The task of this project is to design the CNN for pattern recognition problems.

The stages of the project structure are as follows:

1. Technical task coordination with the supervisor.
2. Selection and study scientific literature on the topic.
3. Work with theoretical materials.
4. Review of existing technologies for graphic object recognition and building convolutional neural networks.
5. Design of a convolutional neural network (CNN) to solve pattern recognition tasks.
6. Conclusions on the project.

The approach to the problem of pattern recognition can be based on analogues with biological processes. In certain situations, the animal capability of pattern recognition can greatly exceed the capacity of any machine ever built by humans.

PART 1
ANALYSIS OF THE METHODS OF GRAPHIC PATTERN RECOGNITION

1.1. Description of the subject field

During a rather long period of time graphic pattern recognition has been considered in terms of biological and psychological aspects. Furthermore, the subject of investigation included primarily qualitative, which do not allow giving an accurate description of the functioning mechanism.

As a rule, functional dependencies were mainly related to the investigation of visual, audial and haptic receptors. However, the principle of decision-making remained unsolved. The main mistake was considered the conception, that a human brain operates according to certain algorithms, thus discovering this set of rules could be recreated by means of computing and other technical devices.

Founded by Norbert Wiener at the beginning of the XX century, a new discipline called cybernetics (the science about general patterns of data transferring between machines, life forms and societies) allowed introducing quantitative methods into examining the graphic pattern recognition problem. In other words, the idea was to present the process of image recognition (being actually a natural phenomenon) by means of mathematical methods.

In most cases devices, which perform functions of diverse graphic pattern, recognition gives an opportunity to replace a human being with a specialized machine. Owing to this, complex data, logic and analytical systems receive a breakthrough in terms of possibilities. It is generally understood, that working productivity in case of people depends on many factors (competence, experience, consciousness, so on and so forth).

At the same time, any machine is supposed to operate monotonously with the same quality at each iteration. Automatic control over complex systems allows conducting monitoring and maintaining in-time service engineering, obstacle identification and self-operating exploitation of noise suppression methods or increasing the quality of data transmission. Apparently, the application of automated systems can maintain the needed processing speed in specific tasks, which is impossible for human.

Graphic pattern recognition has been drawing attention for a long time of specialists in the field of applied mathematics and then computer sciences. Made in 20s, works of Ronald Fischer were praised and recognized as a prominent part of pattern recognition theory. In 40s, Andrii Kolmogorov and Oleksandr Khinchin set the task of autocorrelation function of a wide-sense-stationary random process.

In 50-60s of the XX century, a great bulk of correspondent works and conceptions became the basis for the decision theory. In the realm of the theory various algorithms were discovered – those ones which maintained the attribution of a new pattern to one of the given classes, which became the beginning of a new systematic scientific approach and practical prototypes.

In the realm of cybernetics, a new scientific discipline began to form, related to the development of theoretical basics and practical implementation of devices and then systems, dedicated to pattern, phenomenon and process recognition. This new scientific discipline was named “Pattern recognition”.

In this way, as noted nowadays, results of the classic decision theory became the fundamentals for solving the task of attributing a pattern to one of given classes. In its realm, diverse algorithms, which indicated a certain class for the pattern to be attributed in terms of its experimentally measured characteristics, were built. Pattern recognition is the task of indicating a pattern or any of its features based on its image (so-called visual or optical recognition) or audio (audial recognition) and other characteristics (fig. 1.1).

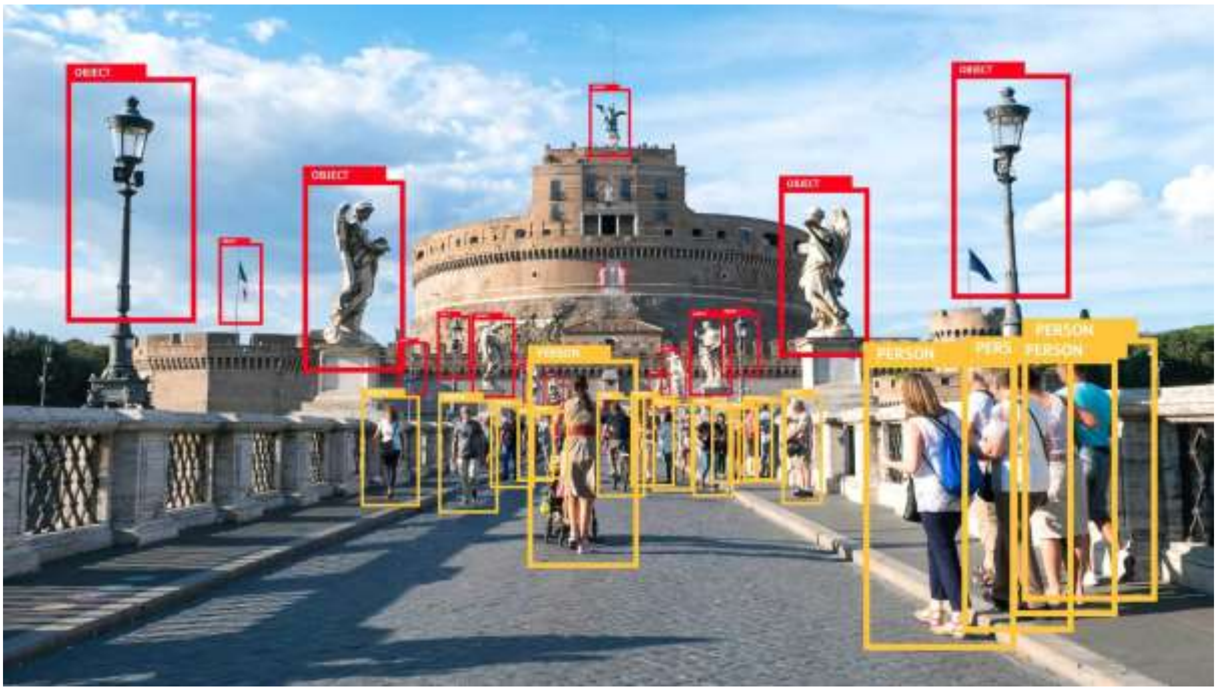


Fig. 1.1. The example of graphic pattern or pattern recognition

Pattern is a systematized grouping in the classification system, which contains (or distinguishes) a certain group of images according to a given feature. Images have characteristic features, defined by the fact, that receiving a finite number of events among the same array allows finding any big number of its representatives.

The method of attributing an element to a certain image is called a decision rule. Another significant notion – metrics – is a way of defining a distance between the elements of the universal set. The less this interval is, the more similar the patterns (symbols, sounds, etc.) to those ones, which we try to identify.

Usually elements are given in a form of a numeric set, and metrics – in a form of a function. The program efficiency in this case depends on the choice of image apprehension and metrics implementation: it means, the same algorithm with different metrics will output mistakes with different frequency. Adaptation is the process of changing parameters and system structure (sometimes even the controlling influences), based on current data in order to reach a certain state of system along with initial uncertainty and alternating working conditions.

Learning is the process, in a result of which the system is gradually reaching the capability to provide needed reactions to certain sets of external influences. At the same time, adaptation is the process of adjusting parameters and system structure in order to reach the needed quality under the condition of continuous changes in external influences.

The concept meant by learning is usually called as the development a certain system reaction to groups of external identical signals by means of repetitive influence on the system of external modification. Such external modification is usually referred to as “engagements” and “punishments”.

The learning algorithms defines the mechanism of modification generating to an almost full extent. Self-learning differs from learning by the fact that additional information about reaction fidelity is not transmitted to the system.

1.2. Concept of computer vision

As a human, we perceive the three-dimensional structure of the surrounding world or its two-dimensional projections easily. People are able to identify form or transparency of any object, precise relations between lighting and shadowing on the surface, which differ much. Looking at a group photo, a person can easily count and name all people on the image and even guess their emotions from how they look. Scientists have been spending decades trying to understand the functionality of our visual system. Though they can use different optical illusions in order to understand certain mechanisms, the final solution of this conundrum remains unknown.

Computer vision researchers developed mathematical models to create a three-dimensional form or objects on the image. Nowadays there are reliable methods of accurate computation of a partial 3D-model in the environment with thousands of points. It is possible to track someone moving against certain background. With alternating success, machines are able to find and name every person on the photo, using facial shapes, clothes and hair. However, despite all these achievements, the dream of computer being able to interpret objects as human or animal remains unreachable. Often recognition comes hard, since vision is a reverse problem, in which the task is to restore some unknown data with partial data quantity in order to generate a consistent decision.

Recognition models ambiguously regard potential solutions without a necessary number of initial parameters. However, modeling the visual world in its diversity is far more difficult than modeling, for instance, vocal cords.

Models used for computer vision are usually developed in the sphere of physics (optics, sensory design, etc.) and computer graphics. Both these fields have the task to simulate object motion and interaction – a common example is light reflecting from surfaces, dispersing in the atmosphere, refracting through camera lens or human eyes and finally projecting into a flat.

Computer vision tries to describe the world, which humans see as one or several images, and features, such as sizes, lighting, and color. Vision algorithms are so prone to mistakes that even the slightest changes in pixels sometimes can lead to very different results. Thus, computer vision is a set of methods and algorithms to interact with the visual environment, the goal of which is graphic object recognition and analysis.

1.3. Overview of existing methods

The methods of graphic object recognition can be divided into three groups: preliminary image filtration and preparation, logic processing of filtration results, algorithms of decision-making based on logic processing. Boundaries between these methods are rather relative. Depending on the task to be solved, sometimes it is not needed to use methods of all aforementioned groups – two or even one could be quite enough.

1.3.1. Filtration

This group consists of methods, which allow distinguishing needed areas on images without their analysis. A great number of these methods applies simple transformation to all image points. Image analysis is not conducted on the filtration level, but those filtrated points can be considered as areas with special characteristics.

Hypothetically, it is needed to perform an automatic pattern recognition on white sheet of paper: the threshold choice defines much the very process of binarization. In this case, the image would be binarized on the intermediate color. Usually binarization is performed by means of algorithms, which choose a threshold in an adaptive way.

Binarization can provide a very interesting output while working with histograms, especially in those cases, where we consider HSV images instead RGB. For instance, the task could be to segment colors. The very principle can become the basis to build marker or human skin detector.

The classical filtration contains Fourier, low-pass filter (LPF) and high-pass filters (HPF). Classical methods of filtrating radiolocation or signal processing can be successfully applied to the tasks of pattern recognition.

In radiolocation a traditional method, which is rarely applied to images pure and simple, is Fourier transformation (more accurately – Fast Fourier transform or FFT). One of few exclusions for applying one-dimensional Fourier transform is image compression. One-dimensional compression is usually not enough for image analysis, thus it is necessary to use a more demanding two-dimensional transform (1.1).

$$G_{uw} = \frac{1}{NM} \sum_{n=1}^{N-1} \sum_{m=1}^{M-1} x_{mn} e^{-2\pi j \left[\frac{mu}{M} + \frac{nw}{N}\right]}. \tag{1.1}$$

The transform is usually rarely solved – usually it is far more easier and simpler to use convolution of area with the filter already prepared, specified for high (HPF) or low (LPF) frequencies. Surely, such a method does not allow performing spectrum analysis, nevertheless, in certain tasks of video surveillance it is needed to receive the result, not the analysis. The most common examples are Kalman filter for low frequencies (fig. 1.2) and Gabor filter for high frequencies (fig. 1.3). For each image point, the window is chosen and multiplied with the filter of the same size. The result of such convolution is a new value of the point.

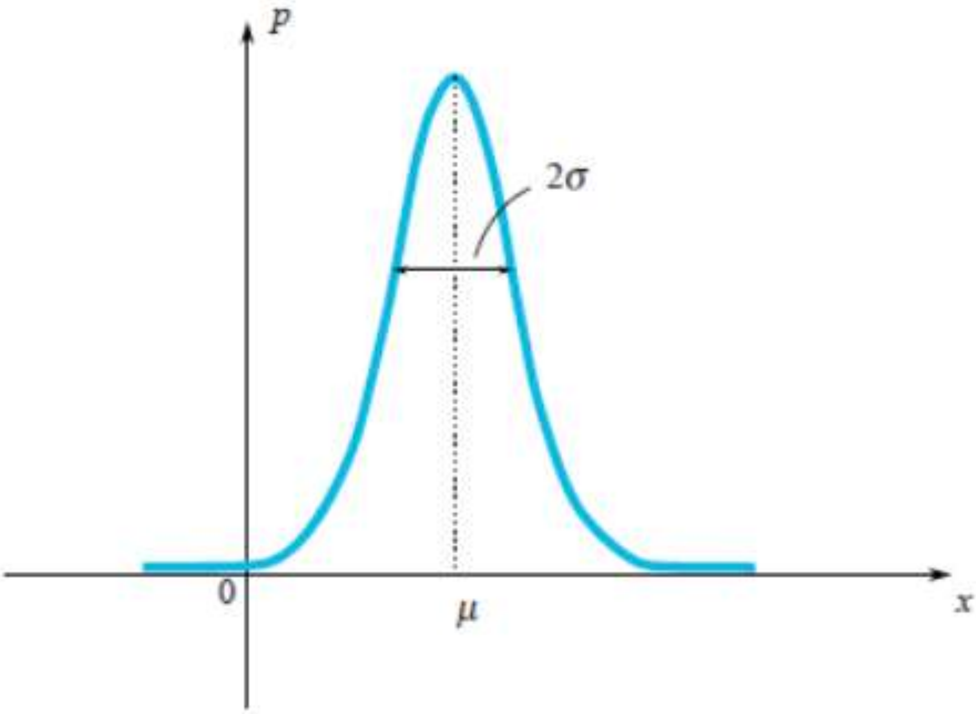


Fig. 1.2. The Kalman filter

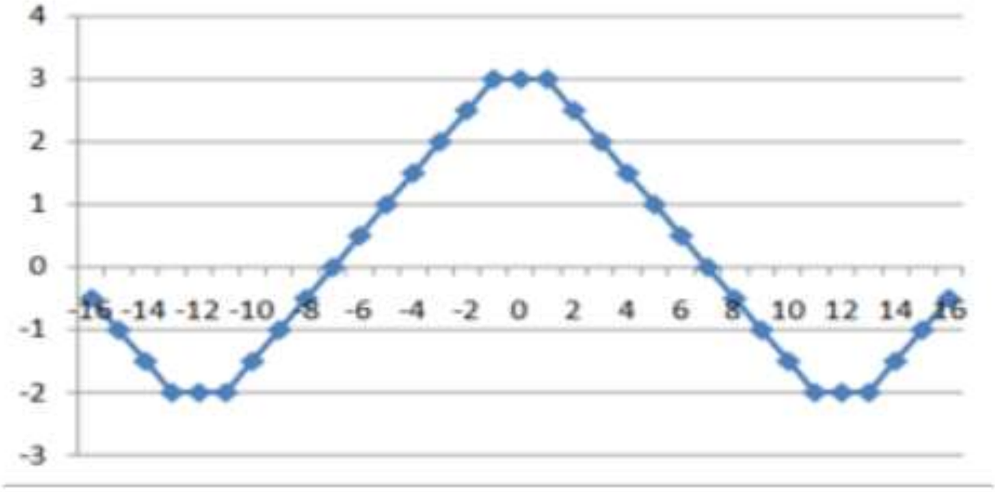


Fig. 1.3. The Gabor filter

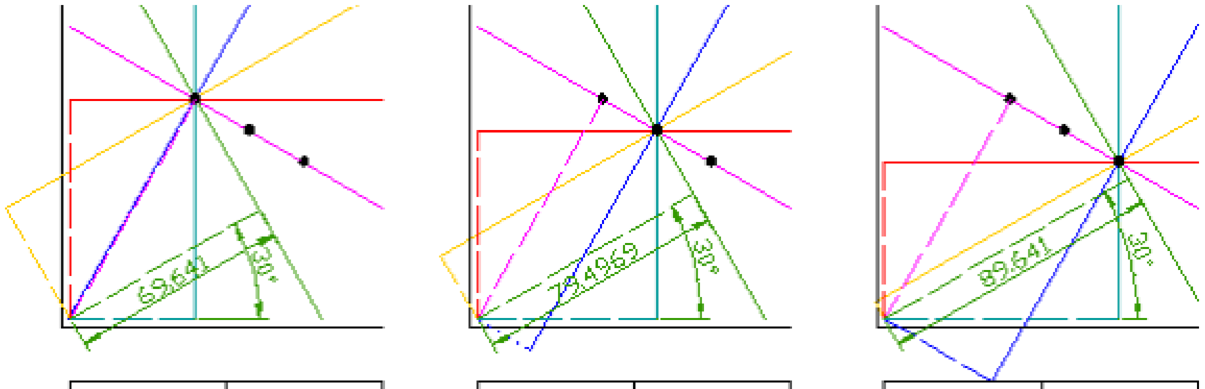
In case we take any arbitrary characteristic feature for signal convolution, we receive the so-called “Wavelet transform”. This definition of wavelets is not fully accurate, but traditionally in many commands, wavelet-analysis is the search of an arbitrary pattern on the image by means of convolution with the model of this pattern. There is a set of classical functions, used in wavelet-analysis, such as Haar wavelet, Morlet wavelet, Mexican hat wavelet, etc.

In fact, there are four examples of classical wavelets: three-dimensional Haar wavelet, two-dimensional Morlet wavelet, Mexican hat wavelet and Daubechies wavelet. A good example of applying an extended wavelet interpretation is the task of finding a glint in the eye, for which wavelet is the very glint. Classical wavelets are commonly used for images compression or their classification.

After such a simple interpretation of wavelets, it is necessary to mention correlation lying in their basis. It is an essential tool during image filtration. The classical application is the correlation of video stream to detect shifts and optical flows. The simplest shift detector is a subtractive correlation in its own way – the spot where images do not correlate indicates the presence of motion.

Function filtration is another interesting class of filters. They are purely mathematical filters, which allow indicating a simple mathematical function on an image (line, parabola, and/or circle). The image is constructed, in which each point of the initial image is drawn by an array of functions.

The Hough transform is the most widespread tool for linear functions (fig. 1.4). During this transform each point (x; y) is drawn by an array of other points (a; b), which confirm the equation: y = ax + b. The Hough transform allows finding any parametrized functions, for instance, circles.



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	−19.5

Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	−39.6

Fig. 1.4. The Hough transform for linear functions

An alternative for the Hough transform for linear functions can be the Radon transform. It is calculated via FFT, which gives an advantage in productivity in case of a great number of points. Besides, it can be used along with unbinarized images.

A separate class of filters constitute border and edge filtration. Outlines are very useful in case is it necessary to transfer from image processing to pattern processing on this image. When the pattern is quite complex but clearly distinguished, the only way to work with it becomes highlighting its edging. There is a whole set of algorithms which solve the task of edge filtration. The most common tool is Canny edge detector, which is also realized in OpenCV.

1.3.2. Definition of the Ethernet

Filtration gives a set of data, available for processing. Several typical methods allowing transferring from images to pattern features or the very patterns.

Methods of mathematical morphology serve the purpose of filtration-to-logic transform. To put it simply, those are the simplest operations of binary image dilation and erosion. These methods allow removing noises from binary images by increasing or decreasing the present elements. Diverse edge detecting algorithms exist on mathematical morphology, though often hybrid or joined algorithms are used.

The algorithms of border detection have already been mentioned, and received borders easily transform into edges. Canny edge detector automatically performs this function, but other algorithms require additional binarization. Edge is a unique feature of a pattern, which often enables its identification. There is a powerful mathematical tool allowing doing this, which is called edge analysis.

Special points are unique features of a pattern, which allow associating the pattern with itself or with similar pattern classes. Several dozens of ways allow highlighting these points, as distinguishing special points in nearby frames, over a certain interval or in case of illumination changes. Some methods allow finding special point, which remain as such even after pattern rotation. Generally, these methods can be divided in three classes. The first class represents special points, which remain stable during several seconds. These points exist in order to trace the pattern between nearby video frames. Such points can be local image maximums, angles (fig. 1.5), points with maximal dispersion, gradient, etc.

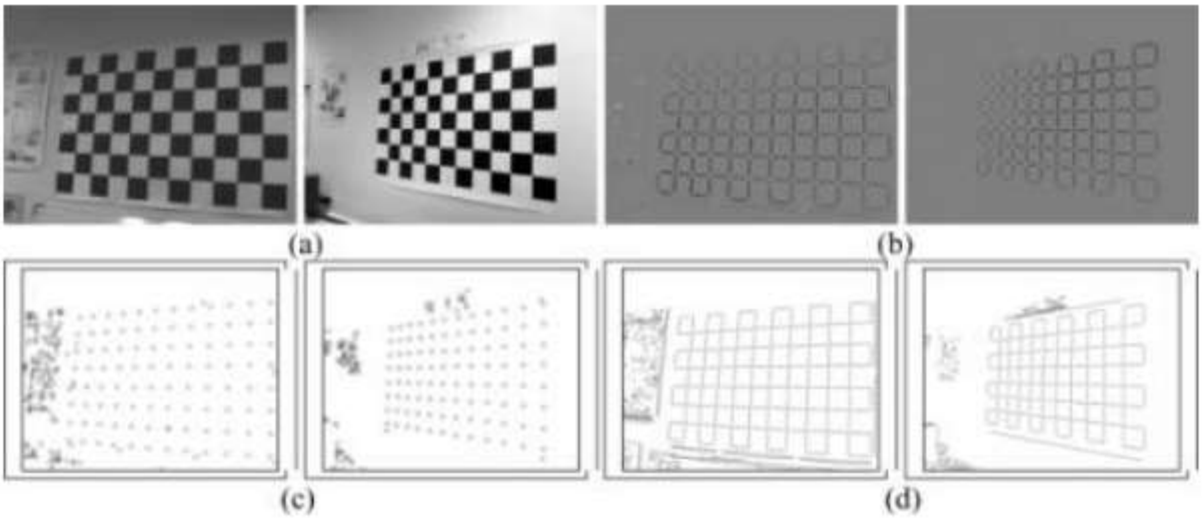


Fig. 1.5. Angle recognition on images

The second class represents special points, which remain stable while illumination changing or slight pattern motion. These points exist for learning and further pattern type classification. For instance, pedestrian or person classifier is a product of the system built on such points. Some of the aforementioned wavelets can become the basis for these points: Haar primitives, glint or other specific functions searching. In addition, these special points can be found by means of histograms of oriented gradients (HOG), shown in Figure 1.6.

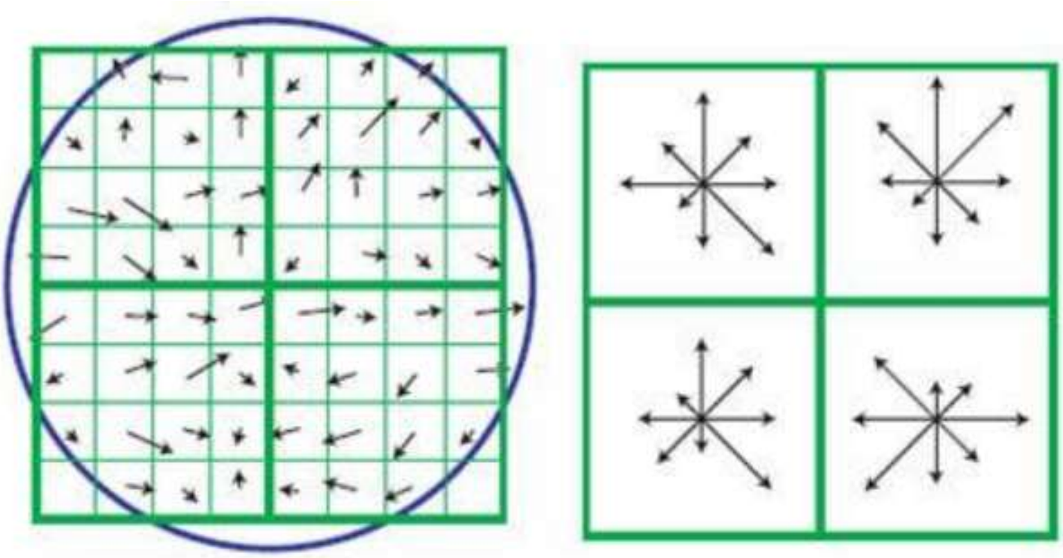


Fig. 1.6. A histogram of oriented gradients

The third class represents stable points. There are two methods giving full stability under modifications: Speeded Up Robust Features (SURF) and Scale-invariant feature transform (SIFT). They allow finding special points even during image rotation. Point calculation takes more time comparing with other methods, but still under a certain limit. Unfortunately, these methods are patented.

1.3.3. Learning

It is time to examine methods, which do not work with images directly, but allow decision-making. Usually they are diverse methods of machine learning (ML) and decision-making. As an example, let us imagine the following task: to detect the presence of human on different images. Each image has a set of features, distinguished by Haar, HOG, SURF or any other wavelet. A learning algorithms has to build a model by means of which it would be able to analyze a new image and decide, whether it contains human or not.

Each test image is a point in the space of features. Its coordinates stand for weights of each feature on the image. The given features could be “Presence of eyes”, “Presence of nose”, “Presence of two hands”, “Presence of ears”, so on and so forth. All these features can be highlighted with given detectors, trained on body parts resembling human ones. In such a space, a human would have a point [1; 1; 1; 1; ...], a monkey – [1; 0; 1; 0; ...], a horse – [1; 0; 0; 0; ...]. A classifier is trained on a set of examples, but some image does not display hands, the other does not display eyes, the third one had a monkey with a human nose because of a classifier mistake. The classifier under training automatically developed a space of features in order to indicate the following: if the first feature belongs the set $0.5 < x < 1$, the second feature belong the set $0.7 < x < 1$, etc., then the thing it detects is human.

In other words, the goal of the classifier is to draw areas of features, related to classification patterns, in a space. For example, AdaBoost classifier operates in a very similar way, though each specific task requires a correspondent classifier.

1.4. Artificial neural networks as a tool of graphic pattern recognition

1.4.1. Operating method of neural networks

The research on artificial neural networks (ANN) is related to the attempts to recreate the human brain processing capabilities, and human brain appears to be a tremendously complex, non-linear and parallel computer (or data processing system). A brain is able to organize its structural components (so-called neurons) in such a way, that they could perform concrete tasks (such as pattern recognition, processing signals from senses and/or motility functions) much faster than the most powerful modern computers. Nowadays there are many examples of using ANN for forecasting, classification, optimization, pattern recognition.

Neural Networks are computing structures, which simulate common biological processes associated with processes in human brain (fig. 1.7). They ensemble systems, prone to learning by means of analyzing positive and negative influences. An artificial neuron – or simply neuron – is the elementary converter in such networks, similarly to its biological prototype.

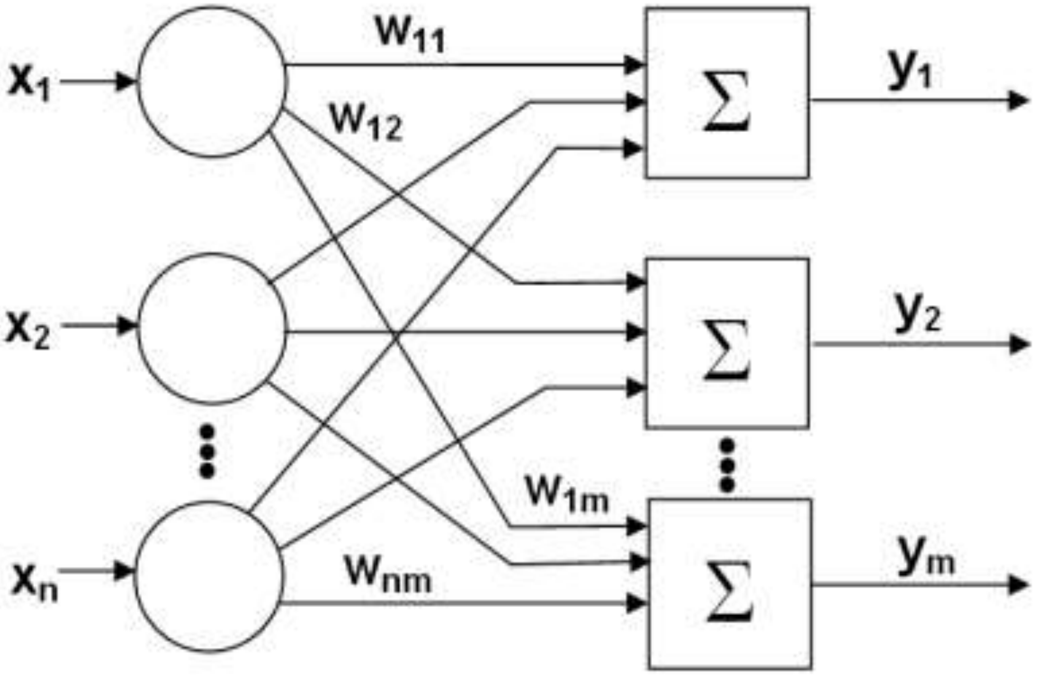


Fig. 1.7. Single-layer neural network

A biological neural has a body and a set of appendices – dendrites, via which a neuron inputs signals, and an axon, via which it outputs signals to other neurons or cells. The point of conjunction between dendrites and an axon is called a synapse (fig. 1.8).

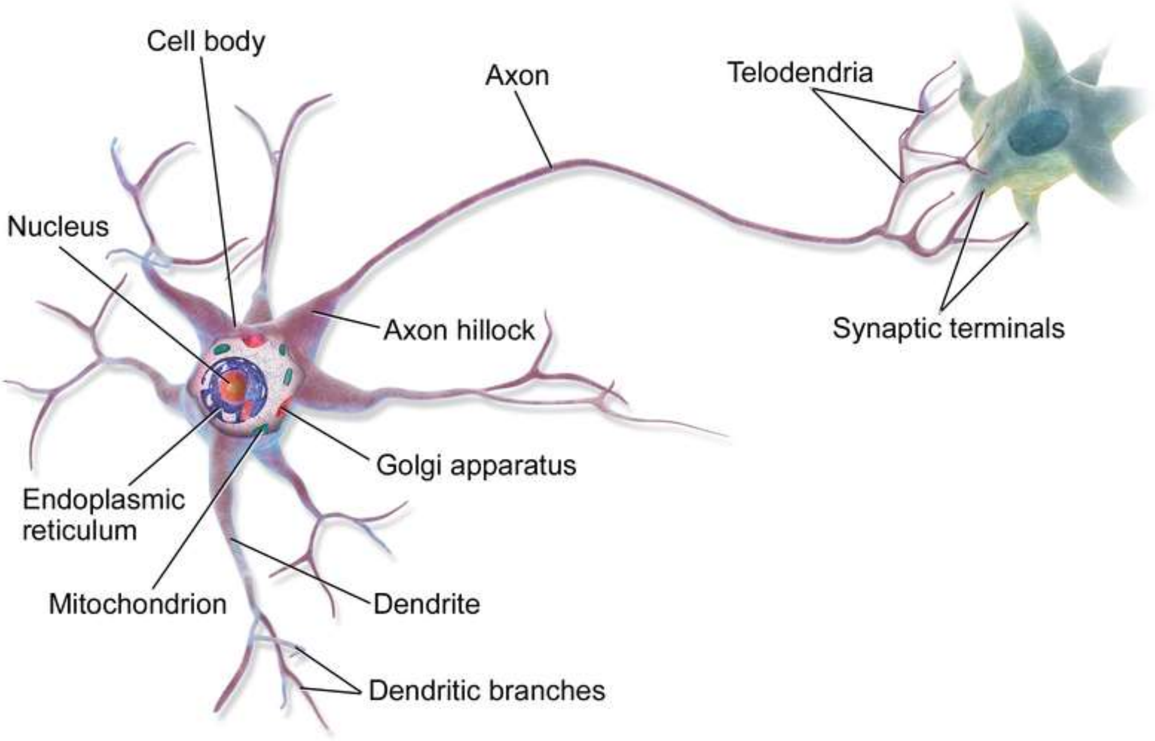


Fig. 1.8. A biological neuron

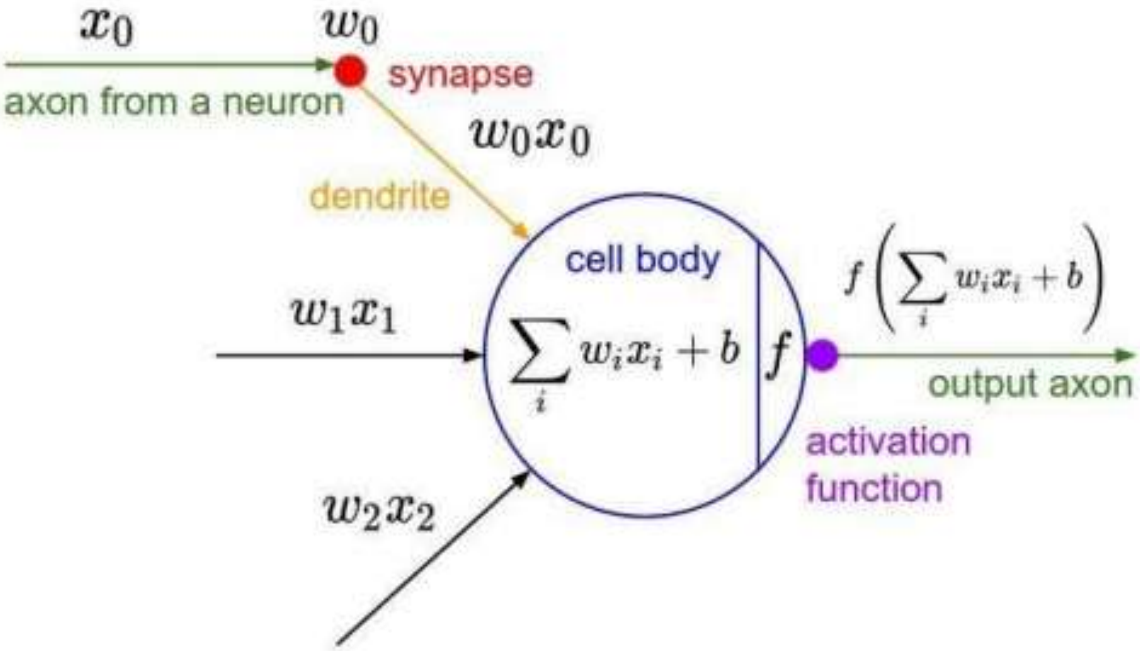


Fig. 1.9. A biological neuron in terms of mathematics

To put it simply, the functioning mechanism of a neuron can be depicted in the following way (fig. 1.9):

1. A neuron receives a set (or a vector) of input signals from dendrites.

2. Neuron’s body evaluates the summary value of input signals, though neuron entries are not equal. Each input is characterized by a certain weight coefficient, which indicates the importance of the transmitted data. In this way, the neuron not just summarizes the value of input signals, but calculates scalar product of input signal vectors and weight coefficient vector.

3. The neuron forms the output signal, and its intensity depends on the value of scalar product. If the product does not exceed a certain threshold, the output signal is not formed at all – the neuron “does not work out”.

4. The output signal is transmitted to the axon and further to dendrites of other neurons.

The most significant ANN feature is their ability to learn on environment data and increase its productivity. Higher productivity comes over time according to certain rules – the process of learning happens owing to the interactive process of modifying synaptic weights and thresholds. Ideally, an ANN receives knowledge about the outer worlds at each iteration of the learning process. The concept of learning can be associated with many different activities, thus it is hard to give this process an accurate definition. In terms of neural networks, learning can be defined as the process, during which ANN parameters are configured by means of modeling the environment, in which this system is integrated. The type of learning depends of the methods of tuning these parameters. Such a definition of the learning process foresees the following operating algorithm:

1. ANN receives stimuli out of the external environment.

2. As a result, parameters of ANN alternate.

3. After alternating the inner structure, the ANN gives a different response to the same stimuli.

This list of strict rules for problem solving is called the learning algorithm. Since neural networks are of different types and purpose, there is no universal learning algorithm. There is only a set of means in a form of various algorithms with their own advantages and disadvantages. Learning algorithms differ from each other by the way of configuring neuron's synaptic weights. Another distinguishing feature is the media of communication with the outer world. In this context, one usually says about learning paradigm, related to the model of environment, in which the given ANN operates. There are three types of learning: supervised, unsupervised and mixed.

ANN supervised learning presumes, that for each input vector of learning set there is a needed value of the output vector, also known as an objective vector. These vectors create a learning pair. Network's weights alternate until for each input vector there will be an acceptable deviation level from the objective vector. In other words, ANN possesses correct answers (network's output) for each input example. Weights keep on changing values to provide answers most close to the known correct answers. Advanced supervised learning presumes that there is only a critical evaluation of output correctness, not the answers.

ANN unsupervised learning provides a more natural approach in terms of the biological prototype. A learning set contains only input vectors. ANN learning algorithm fits weights into new conditions in order to present coordinated output vectors, i.e. rather similar input signals would provide similar output vectors. Unsupervised learning does not require correct answers for each example of a learning set. In this case, it is possible to see the inner data structure, or correlation between samples of the database – thus, allowing sorting samples into categories. During mixed learning, some weights are trained being supervised, while others provide results based on self-learning.

1.4.2. Types of algorithms: functions, tasks and applications

Image classification. The task lies in the attribution of input pattern (for instance, language signal or handwriting symbol, represented as a vector of feature) to one or several preliminary defined classes. It can include letter or language recognition, electrocardiogram (ECG) signal classification, blood cell classification, etc.

Clustering (or categorization). Also known as unsupervised pattern classification, clustering tasks exclude learning sets with class markers. Clustering is based on the image similarity, so it puts resembling patterns into one cluster. Sometimes clustering is used for data compression and examination of data parameters.

Function approximation. Hypothetically, there is a learning set generated by an unknown function F(x), which is distorted with noise. The task of approximation is to find a value of the unknown function F(x). Function approximation is essential for solving numerous engineering and scientific modeling problems.

Forecasting. The task presumes for ANN to predict some value at some point of time. Forecasting has a crucial importance in business, science and machinery. On top of that, predicting stock prices or weather are typical examples of forecasting algorithms.

Optimization. Numerous problems in mathematics, statistics, machinery, science, medicine and economics can be regarded as optimization problems. The task of the optimization algorithm is to find such a solution that would either satisfy system’s limitations or maximizing/minimizing the objective function. Travelling salesman problem (TSP) is a classical problem in this sphere.

Content addressable memory. In Von Neumann architecture, addressing memory is available only by means of the address, which is independent from memory storage. Moreover, a mistake in address computation can lead to different data. Associative content addressable memory (ACAM) is available as directed by the given content. ACAM is critically efficient while creating multimedia databases.

Adaptation to alternations of the environment. Neural networks have the ability to adjust to the environment; particularly, they can be easily retrained for slightly different environment after being trained under initial conditions. Furthermore, one can create neural networks to work in time-varying environment (stocks prices, for example, where statistics changes over time) – they can be retrained in real-time mode. The more adaptive the system is, the more stable work it will conduct in a non-stable environment.

However, one must mention, that adaptability not always results into stability – if often aggravates the work. An adaptive system with quickly changing parameters can distract itself on secondary signals, which results into decreased productivity. In order to benefit from adaptability, all the core parameters should be both stable enough not to distract on secondary noise and adjustable to maintain reaction in case of rapidly alternating environment.

Potentially neural networks are failure-resistant in terms of hardware implementation. It means, that in case of unfavorable conditions their productivity does not seem to demonstrate critical drop. If there is any damaged neuron or its links, data extraction can be problem. However, taking into consideration the distributed way of storing data, one can surely state that only severe and complex damages of ANN structure can cause a significant reduction in network's productivity.

Conclusions on Part 1

As a human, we perceive the three-dimensional structure of the surrounding world or its two-dimensional projections easily. People are able to identify form or transparency of any object, precise relations between lighting and shadowing on the surface, which differ much. Looking at a group photo, a person can easily count and name all people on the image and even guess their emotions from how they look. Scientists have been spending decades trying to understand the functionality of our visual system. Though they can use different optical illusions in order to understand certain mechanisms, the final solution of this conundrum remains unknown.

Computer vision tries to describe the world, which humans see as one or several images, and features, such as sizes, lighting, and color. Vision algorithms are so prone to mistakes that even the slightest changes in pixels sometimes can lead to very different results. Thus, computer vision is a set of methods and algorithms to interact with the visual environment, the goal of which is graphic object recognition and analysis. The main problem of graphic object recognition is the human ability to classify and generalize objects – this mechanism still presents a huge challenge for researchers. Part 2 is dedicated to cover this aspect and present methods of solving such tasks.

PART 2
THE PROBLEM OF CLASSIFYING GRAPHIC PATTERNS

2.1. Receptive structure of data apprehension

2.1.1. Biological prototype

A human can consciously comprehend information after a long cycle of preliminary processing. First, the light hits an eye. Through the whole optical systems photons at last reach the retina – a layer of photosensitive cells, called rods and cones.

Here, still very far from the brain, the first stage of data processing occurs. Information transfers to the human brain via optic nerves into so-called “colliculus cranialis”, on which visual information is projected. Then visual information reaches brain regions, which distinguish separates components – horizontal, vertical, diagonal lines, edges, areas of light, shadow and/or color. Gradually patterns become more complex and blurred, but the graphic image will go through a long way until it reaches the level of consciousness.

The approach to the problem of pattern recognition can be based on analogues with biological processes. In certain situations, the animal capability of pattern recognition can greatly exceed the capacity of any machine ever built by humans.

Dealing with classification based on direct sensory experience, i.e. recognizing individuals or pronounced words, people easily exceed technical devices. In “non-sensory situations”, human actions are not so effective. For example, people cannot compete with pattern classification programs, if the right way of classification includes logic combinations of abstract features, such as colors, sizes and forms. Since pattern recognition has to be neurons’ function, it is possible to look for “the golden key” to biological pattern recognition in neurons’ features.

CSN department				NAU 19 74 09 000 EN			
Done by	Klymenkov S.V.			The problem of classifying graphic patterns	Letter	Sheet	Sheets
Supervisor	Nadtochii V.I.				E	32	109
Consultant					6.050102 CS-422a		
S-inspector	Nadtochii V.I.						
Head of dep.	Zhukov I.A.						

For many tasks, a neuron can be considered as the boundary element. It means that a neuron outputs a certain constant value, if the input sum reaches a certain value or remains passive. Warren McCulloch and Walter Pitts proved that any computational function could be realized by means of appropriately organized system of ideal neurons – threshold elements, logic features of which could be related to a real neuron. The problem lies in the question, whether it is possible to find a smart principle of network reorganization, which allows a randomly joined group of ideal neurons self-organize into "a computational device", able to solve an arbitrary task of pattern recognition. Such reorganizing principle became the fundamental of learning theory, applied on the level of a separate neuron.

2.1.2. Neural learning theory

Neural learning theory, proposed by Donald Hebb, was supposed to be used as a psychological model, but also had a great effect on artificial intelligence. Its modification was applied to define the principles of pattern recognition systems – which were later called as perceptron. Perceptron, described by Frank Rosenblatt, can exist in a form of a program or specifically constructed computing device.

Pattern (or class) is a classified group in the classification system, which connects a certain group of objects or images based on a certain feature. Image sensitivity is one of the characteristic features of an alive brain, which allows sorting all the endless stream of input information and remain self-orientated among miscellaneous data about the outer world. Perceiving the world, we always conduct data classification, i.e. sort out information into groups of similar, but not identical events. For example, despite huge differences, the same group contains all "A" letters, written with different handwritings, or all sounds correspondent to the same note in any octave or by any instrument. In order to form an image of this apprehended group it is quite enough to be acquainted with few representatives. Such brain functions allows forming such a notion as pattern.

Patterns have one very characteristic feature – acquaintance with a finite number of events from the same set allows found out as many as possible number of its representatives. Patterns possess certain objective features in terms of different people classifying objects in a similar way in spite of different learning material. Because of this, people from all over the world are able to understand each other. The ability to perceive the world in a form of patterns allows founding out infinite number of objects based on acquaintance with their finite number, and the objective character of primary object features allows modeling their recognition.

Pattern recognition is the task of identifying an object or defining its features according to its image (visual recognition) or audio (audial recognition). In the process of biological evolution, many living species resolved the task of visual and audial comprehension quite successfully. Development of artificial systems with similar pattern recognition possibilities appears to be a hard technical problem.

Generally, the pattern recognition problem (PRP) consists of two parts: learning and recognition. Learning is conducted by means of demonstrating separate objects referred to a certain method. Because of learning, the system has to obtain the capacity to provide one reaction to all objects of the same pattern and the other reaction to all distinguished patterns. It is crucially important to finish the learning process with demonstrating the finite number of objects. The material for learning sets could be either images or other visual objects, such as letters or numbers (fig. 2.1) [9].

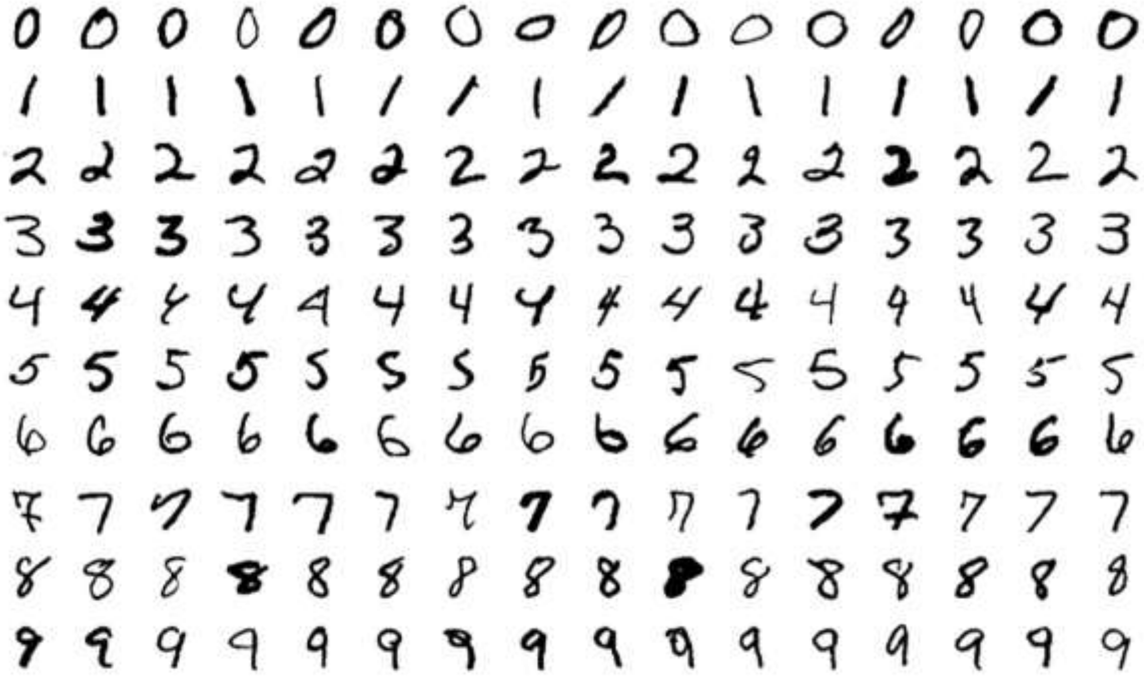


Fig. 2.1. An example of learning objects

It is also important to identify only objects and their relation to a pattern. Learning is followed by the process of recognizing new objects, which indicates the efficiency of the trained system. Automation of these processes becomes the very problem of pattern recognition learning. In case when a person takes charge of creating the rule of classification and then implying it to the machine, the PRP is solved only partially, since a great bulk of task is solved by a human.

A range of tasks that could be solved by recognition systems is truly great. It contains not only simple tasks of recognizing visual and audial patterns, but also classification of complex processes and phenomena, arisen, for example, in the choice of acceptable managing decisions in a company or optimal management in technical, economical, transport or military fields. Before launching the analysis of any object, it is necessary to receive certain organized data about it.

The choice of initial object description is among the core stages of PRP. With a fortunate initial description (the space of features), the problem could be quite a trivia and vice versa – wrong description model could either aggravate the task of data processing or result in the absence of solution at all.

Any image, resulted after observing any object in the learning process or examination, can be represented in a form of vector, i.e. a point of the given space of features. If one states that demonstrated images can be attributed to one pattern from two (or several), this also means that some space contains two or several areas with no common points, and these images are points from these areas. Each of these areas can be described, i.e. given with a name correspondent to the pattern.

It is possible to interpret the adaptation process of pattern recognition in terms of geometrical images, for now limited with two patters. The only known thing is that two areas must be distinguished in some space and demonstrated points belong only to these areas. The very areas are not defined beforehand, i.e. there are no information about their location and rules of attributing a point to one or another area.

While leaning, randomly chosen points from these areas are demonstrated with no information about their location or edges. The main goal of learning lies in building a surface, which would not only separate points shown while learning, but all other points which belong to these areas; or in building surfaces, which would limit the areas in a way that each of areas would contain the points of the same pattern. In other words, the task is to build such functions from vector-images, which would be positive in all points of one area and negative in all points of another area. Since these areas do not have any common points, there is a whole set of dividing functions, and one of them must be built during the learning process.

If the demonstrated images belong to more than two patterns, the task is to build a surface, which would separate all the areas related to these patterns from each other. One of the solution is to model such a function that has the same value over all points from each area and different values over the points from other areas.

At first, it may seem that knowledge about only a few number of points from an area is not enough to distinguish the whole area. Apparently, one can indicate an infinite number of different areas containing these points. Thus, whatever the method by which the distinguishing function built is, it is always possible to highlight the area, which crosses the surface but still contains demonstrated points. Nevertheless, it is well known that the task of function forthcoming according to its data to the limited, much more narrow set of points is a casual mathematical problem of function approximation. The solution to such problems requires introducing certain limitations on the class of regarded functions, and the choice of these limitations depends of data type, which could be added by a “supervisor’ while learning.

Function approximation would be even easier the more separated areas are in the space. For example, Figure 2.2 shows that the division in the left section is conducted far more easier than in the right section. In the first case, objects can be easily divided by the surface, and even serious discrepancies could still provide a successful result. In the second case, division is conducted by a complex function, and even the slightest infidelities lead to the mistakes in the result. The very intuitive concept about areas to be relatively divided at ease created the so-called compactness hypothesis.

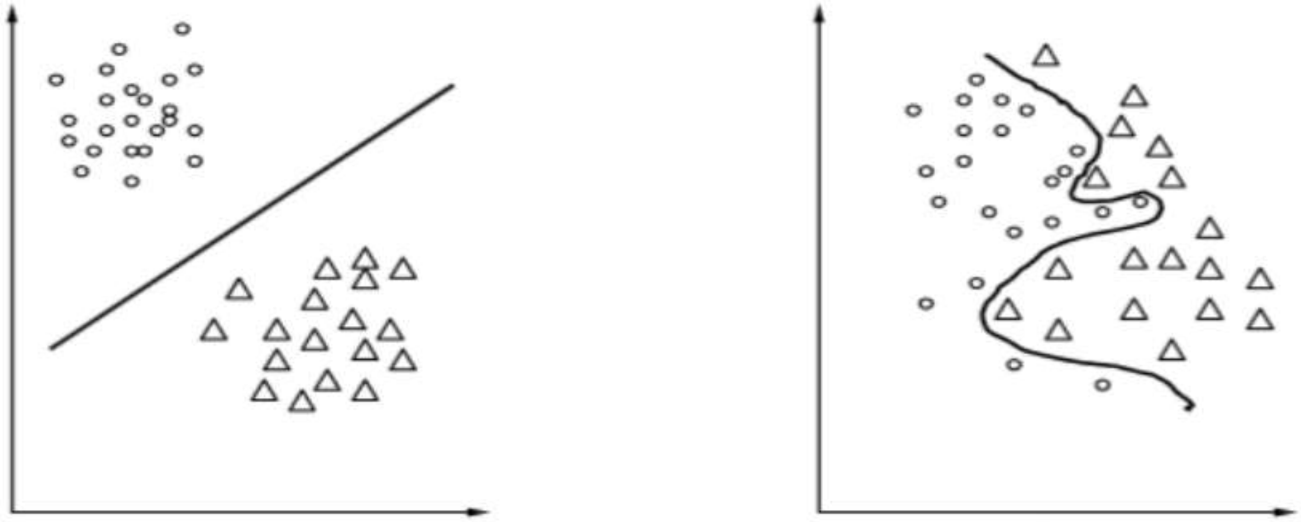


Fig. 2.2. Division of two patterns in the space

Along with geometrical interpretation of PRP, there is another approach, called structural or linguistic. The linguistic approach could be explained with the example of visual image recognition. Firstly, one distinguishes a set of initial descriptions – typical fragments, met on images and characteristics of mutual fragment location, i.e. “left”, “down”, “inside”, etc. These initial descriptions create a vocabulary, which allows building various logic statements. The task is to choose the most suitable statements for the given case among a great number of statements, which could be based on these descriptions.

Then, after looking through a finite and (if possible) small number of objects from each pattern, it is necessary to create description for these patterns. Created description so full and accurate in order to solve the questions of attributing object to the pattern. The linguistic approach arises to tasks: development of the initial vocabulary, i.e. a set of typical fragments, and development of describing rules from vocabulary’s elements.

In the realm of linguistic interpretation, the analogue between the image structure and linguistic syntax is stated. The aspiration to this analogue was determined by the opportunity to use tools of computational linguistics, which methods statistical by origin. The usage of computational linguistics to describe the image structure can be implemented only after images are segmented into components. After preliminary work highlighting the words, there appear the mentioned linguistic tasks, containing tasks of automatic grammatical breakdown of descriptions for image recognition. Here the selfsustained area of examining appears, which requires not only the knowledge of mathematical linguistics basics, but also mastering tools designed specifically for linguistic processing of images.

Assuming that during the learning process the space of features is formed according to the chosen classification, the space of features defines characteristic, under which patterns are easily divided in the given space. The compactness hypothesis states: compact pattern correspond to compact sets in the space of features. As a compact set, we understand some “clots” of points in the space of images, assuming these clots are divided. This hypothesis could not be always proved experimentally, but those tasks, in the realm of which the compactness hypothesis worked out could be easily solved. On the contrary, those tasks, in the realm of which the hypothesis failed, could be solved with very hardly or unsolved at all. This fact made at least doubt the consistency of compactness hypothesis, as one example is enough to bust it. Along with this, successful cases of applying the compactness hypothesis kept on drawing attention. It indicated the possibility of a reasonable outcome in recognition tasks.

Learning is the process, during the system is gradually obtaining the ability to react as expected to a set of external influences, and adaptation is the configuration of system parameters and structure in order to reach need control quality under constant changes in the environment. Each of these tasks has several sample (learning set) of correctly solved tasks. If it was possible to notice some general feature, related not to pattern or image origins, but to their ability to be divided, then another task could be set – so-called unsupervised learning.

At the descriptive level, this task could be stated in the following way: simultaneously or successively, the system receives objects without any designation to their pattern relation. The system input device depicts numerous objects for numerous images and develops its own classification to these objects. After such self-learning process without any hints by a “supervisor”, the system should obtain the ability to recognize no only known objects from learning sets, but new also [10].

The role of a “supervisor” is to provide some objective feature, universal for all patterns and defines the ability of numerous objects to be divided into patterns. Pattern compactness feature is that universal characteristic. Mutual location of points in the given space already contains information about the way of dividing them. This information defines the feature of pattern separability, which is enough for the self-learning process.

Therefore, learning is the process of developing certain reactions in a system to the groups of external identical signals by means of repeated influence on the system by external modification. Such external modifications in the learning happen to be called “engagements” and “punishments”.

The mechanism of generating such modifications almost fully defines the learning algorithm. Self-learning differs from learning by the fact that additional information about reaction fidelity is not given to the system.

The majority of self-learning algorithms are able to distinguish only abstract patterns (compact sets in the given spaces). The difference between them lies in the formalization of the compactness concept.

Self-learning result characterizes the consistency of the chosen space for each separate recognition task. If abstract patterns received in the self-learning process can be compared with the real ones, then the space is chosen as needed. The more abstract patterns differ from the real ones, the more inaccurate the chosen space is for the given task.

There is a possible method of building a recognizing machine, based on distinguishing all the features. As features, one can choose different peculiarities of figures, for instance, their geometrical or topological features. There are recognition machines, in which recognition of letter or digits is conducted according to so-called “probe method” (fig. 2.3), i.e. taking into account the number of crossing between figure’s edge with several lines in a certain order.

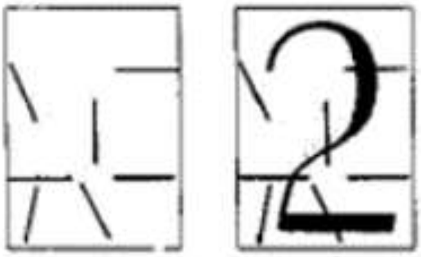


Fig. 2.3. The scheme of digit recognition

When projecting digits on the field with probes, each digit crosses certain probes and crossing combinations are different for each digit. These combinations are used as features for digit recognition. Such machines successfully read typescript, though their possibilities are limited by fonts (or a group of similar fonts), for which the feature system was developed [11]. The creation of a set with ideal figures or feature system should still be conducted by human. “Recognition” fidelity for proposed figures is defined by the quality of the preliminary human work and cannot be improved without human participation. Therefore, the described machine cannot learn.

To input image in the machine, it is necessary to encode it into machine language – represent it in a form of symbol combination for the machine to operate. Coding of plain figures can be conducted in many different ways. For example, one can draw figures on a filed, divided in squares by vertical and horizontal lines. Elements under the image are drawn black; all the others remain white. Black elements are indicated with “1”, white elements are indicated with “0”; and then it is necessary to introduce a successive numeration for all the elements. Then each figure drawn on such a field will be represented in a form of code, consisting of as many digits (ones and zeros) as there are elements in the field.

Such type of coding (fig. 2.4) is considered “natural”, since division of image into elements lies in the base of our visual apparatus. Human retina consists of a great number of separate sensory elements (rods and cones), connected with visual regions of our brain via nerve fibers.

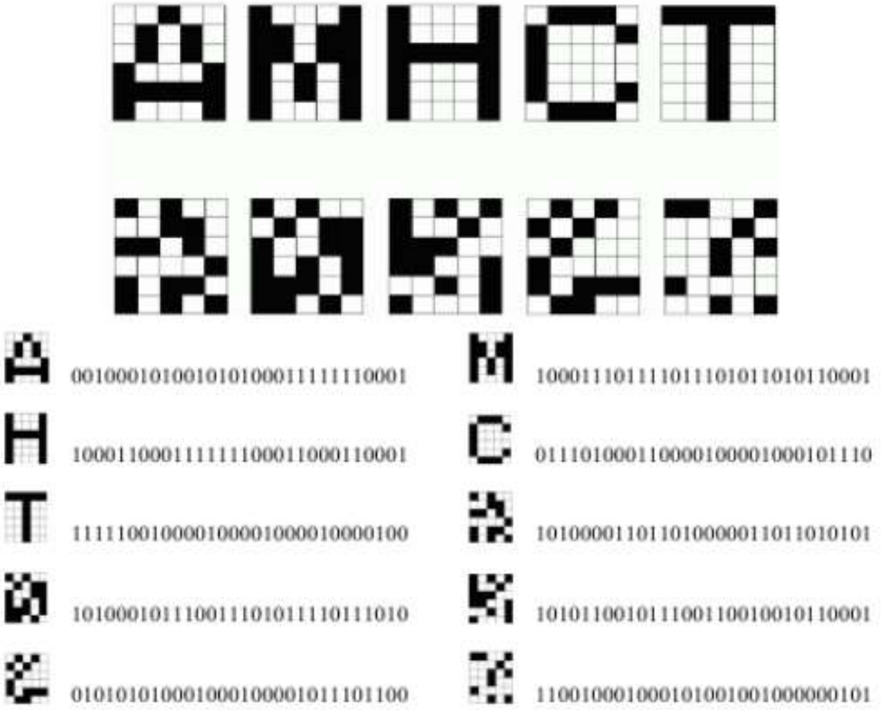


Fig. 2.4. An example of image coding

Retina sensory elements transmit signals, which intensity is based on lightness of a given element. In this way, the image projected on the retina with the eye optical system is divided by rods and cones into separate fields. Separate field elements are called receptors, and the very field is called the receptive field.

A group of all plain figures, which can be projected on the receptive field, is made of a certain set. Each figure from this group is the object of this set, correspondent to a certain code. Mutually definite correspondence between codes and images allows operating only codes, remembering that an image can always be recreated from its code.

ANN capacity is the number of patterns input for recognition. To divide a set of input patterns on two classes, it is enough to use only one output. Herewith each logic level – “1” and “0” – will designate a separate class. Two outputs allow coding four classes, so on and so forth. In order to increase classification efficiency, it is recommended to provide a surplus by means of reserving at least one neuron in output layer for each class. Each neuron learns the attribution of pattern to the class based on its fidelity degree, for example: low, middle and high. Such ANN allow conducting classification of input patters, joined into imprecise sets. The very feature advances similar ANN to the conditions of the real world.

2.2. Algorithmic models

The algorithmic universality of computers means that any algorithmic transformations can be implemented in a form of machine programs: computing, controlling, searching, graphic and pattern recognition algorithms. Nonetheless, one should not believe that machines and robots are able to solve all kinds of tasks. Some types of problems are unable to be solved by a unique and efficient algorithm; in this context, such tasks cannot be solved by machinery as well. This fact contributes to the understanding of what is possible for machinery and what is not.

Among noticeable features of ANN, the most important one is the ability to learn diverse methods and techniques [12]. The majority of learning methods are suitable for basic configuring and possess similar characteristics. ANN learning resembles the process of human learning, though ANN capabilities are quite limited [13]. A network is learning in order to provide a needed set of output for each set of inputs, and each of them is regarded as a vector. Learning is conducted by means of a successive demonstration of input vector with simultaneous configuration of weights according to a certain procedure. In the end network's weights gradually obtain values so that any input vector resulted into an output vector. Learning algorithms can be classified as algorithms of supervised or unsupervised training.

In the first case, there is a "supervisor", who provides input patterns for the network, compares resulting outputs with needed values and then configures weights in order to reduce differences. Such type of learning presumes that each input vector is correspondent to the so-called objective vector, which together form a learning pair [14]. Usually ANN is trained on several learning pairs. Input vector is given to the system, it computes output to be compared with the objective vector; difference (or error) is transmitted back into the network to configure weights according to the algorithm of error minimization. Vectors of a learning set are provided in a sequence, errors being computed and weight being configured for each vector until the error over the whole learning array drops to suitable level.

Supervised learning was greatly criticized because of its biological infidelity. It is quite hard to image a learning mechanism in the brain that would compare needed and real output values, performing correction by means of feedback methods. Unsupervised learning is a far more true-to-life model in the biological systems. Developed by Teuvo Kohonen and others, it does not require the objective vector, therefore, it does not require comparison with given ideal answers. A learning set contains only input vectors, and the algorithm reconfigures weights in order to provide consistent outputs, i.e. when providing quite similar input vectors resulted into the same outputs. This learning process thus distinguishes statistic features of a learning set and sorts vectors into classes. Each input provides a certain output, but it is impossible to forecast the output of a given class of input vector beforehand. Therefore, such network output have to transform in some understandable form, defined by the learning process.

ANN learning process on a new class of tasks includes the following stages [15]:

1. The task is assigned, the set of key parameters characterizing the subject filed are distinguished.

2. The paradigm most suitable for the current task by a neural network is chosen (a model, which contains input data type, a threshold function, network structure and learning algorithms). As a rule, contemporary neural packets allow implementing several basic paradigms at once [16].

3. A wide set of learning samples is prepared, which are organized in sets of input data, associated with known output values. Depending on the task, input values for training could be incomplete and/or partially controversial.

4. Successively input data is given to ANN, and the resulted value is compared with the template. This is followed by configuring weighing coefficients of interneuron connections in order to minimize the value of error between real and needed output values.

5. Learning process repeats until the total error over all input data reduces to an acceptable level, or ANN switches to a steady state condition. The examined ANN learning methods is called “error backpropagation) and referred to classical neural-mathematical algorithms.

Configured and trained ANN is available to be used on real input data with not only hinting a right solution to the user, but also evaluating its fidelity degree. However, there are loads of algorithms used for pattern recognition. The ANN learning algorithm of pattern recognition based on the method of sectional hyperplanes (fig. 2.5), lies in the approximation of hypersurfaces into "pieces" by hyperplanes and consists of the following stages:

- learning (formation of divided surface by drawing sectional planes, cutting out extra planes and their pieces);
- recognizing new objects.

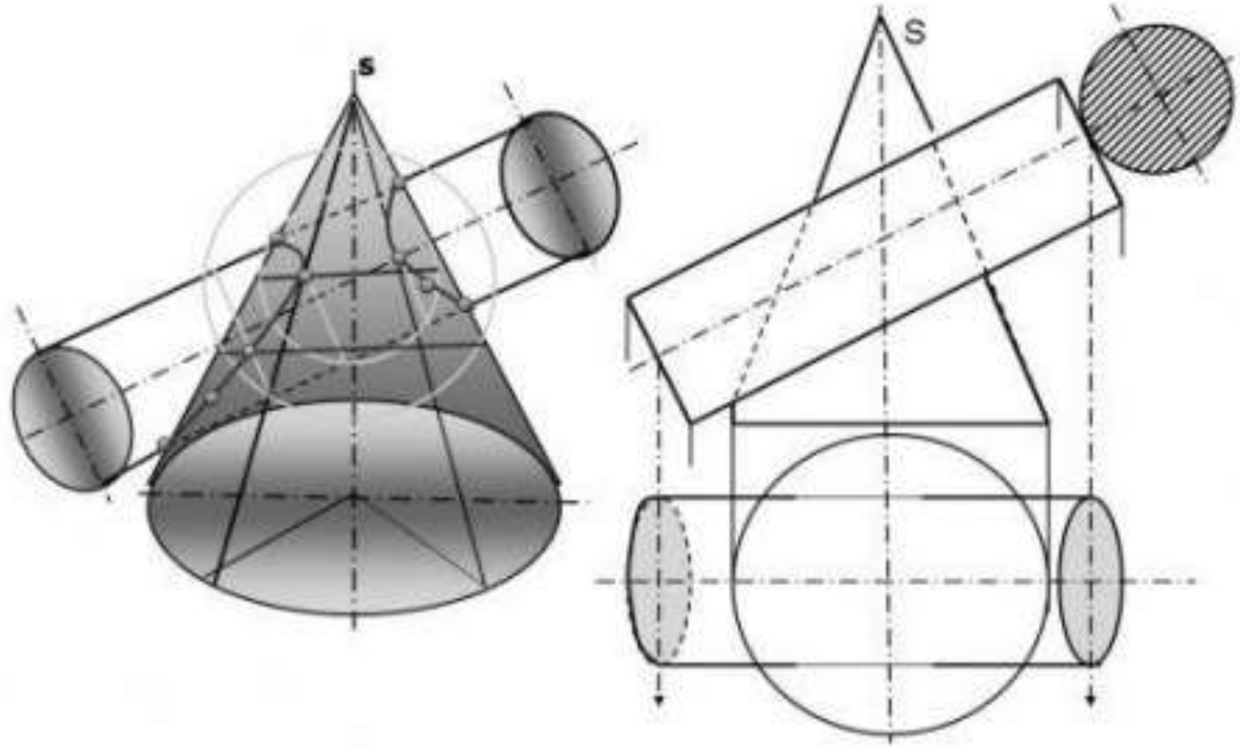


Fig. 2.5. The method of sectional hyperplanes

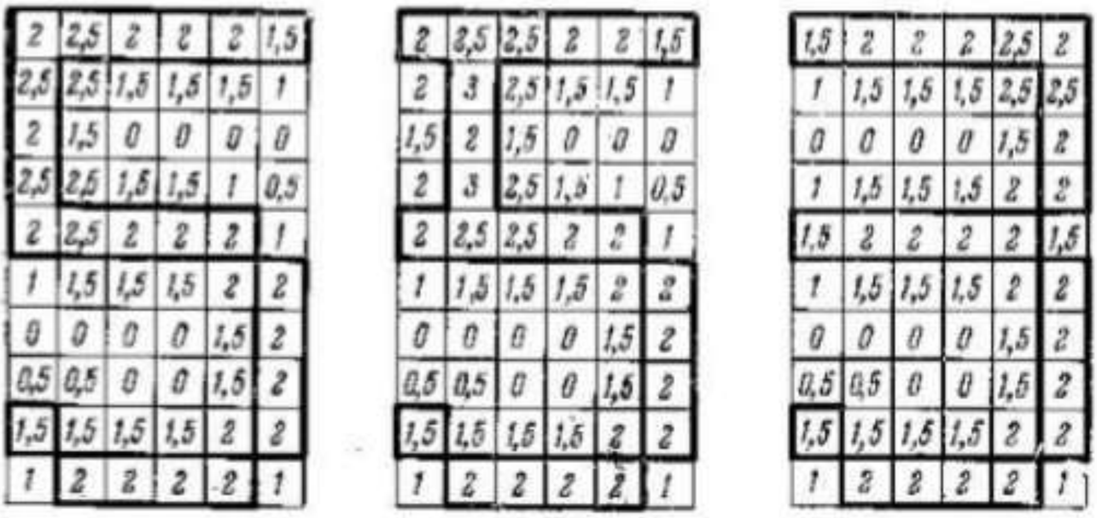
The method of parallel options provides simultaneous and independent learning on the same training material of several machines. After recognizing new objects, each machine would attribute these objects to certain patterns, sometimes different. The final solution is "voted" by machines, and the object is attributed to the pattern according to the majority of voices.

The way of improving the fidelity of recognition lies in the improving the method of sectional planes. One can assume that drawing sectional planes close to the plane cutting in half the line, connecting object and opponent, which is perpendicular to the line, than the resulting surface would be close to the real edge between patterns. Diverse experiments confirm this predication.

The algorithm, based on the methods of potentials, each excited element of the receptive field can be connected with a function equal to “1” on this element and decreasing in all directions from it; i.e. function φ, similar to electrical potential with R being a distance between too neighboring elements of the receptive field (2.1) [17].

$$\varphi(R) = \frac{1}{1 + \alpha R^2}. \qquad (2.1)$$

For calculation, one can use a simple rule: each excited element of the receptive field has its “own” potential equal to “1”, which increases all neighboring potentials (excited as well) by ½. However, such coding method must be improved. If each excited element of the receptive field is connected with this function (fig. 2.6), it can be approximated with an inter-stage function, which is constant in the range of one receptor, but changes spasmodically on receptor edges.



2	2,5	2	2	2	1,5
2,5	2,5	1,5	1,5	1,5	1
2	1,5	0	0	0	0
2,5	2,5	1,5	1,5	1	0,5
2	2,5	2	2	2	1
1	1,5	1,5	1,5	2	2
0	0	0	0	1,5	2
0,5	0,5	0	0	1,5	2
1,5	1,5	1,5	1,5	2	2
1	2	2	2	2	1

2	2,5	2,5	2	2	1,5
2	3	2,5	1,5	1,5	1
1,5	2	1,5	0	0	0
2	3	2,5	1,5	1	0,5
2	2,5	2,5	2	2	1
1	1,5	1,5	1,5	2	2
0	0	0	0	1,5	2
0,5	0,5	0	0	1,5	2
1,5	1,5	1,5	1,5	2	2
1	2	2	2	2	1

1,5	2	2	2	2,5	2
1	1,5	1,5	1,5	2,5	2,5
0	0	0	0	1,5	2
1	1,5	1,5	1,5	2	2
1,5	2	2	2	2	1,5
1	1,5	1,5	1,5	2	2
0	0	0	0	1,5	2
0,5	0,5	0	0	1,5	2
1,5	1,5	1,5	1,5	2	2
1	2	2	2	2	1

Fig. 2.6. The method of potentials

The simplest recognition algorithm, built on the method of potentials, can be conducted in two stages:

1. During the learning process, the codes of appeared points and indications of attribution are recorded.

2. During the recognition process, identification is conducted – as a result, one receives information, to which pattern each coded matrix is attributed.

Despite some limitations of the Rosenblatt perceptron and its output model, it become the fundamental for many most complex learning algorithms with a “supervisor” [18]. The scheme, shown in Figure 2.7, can be an example of a perceptron with non-recursive network.

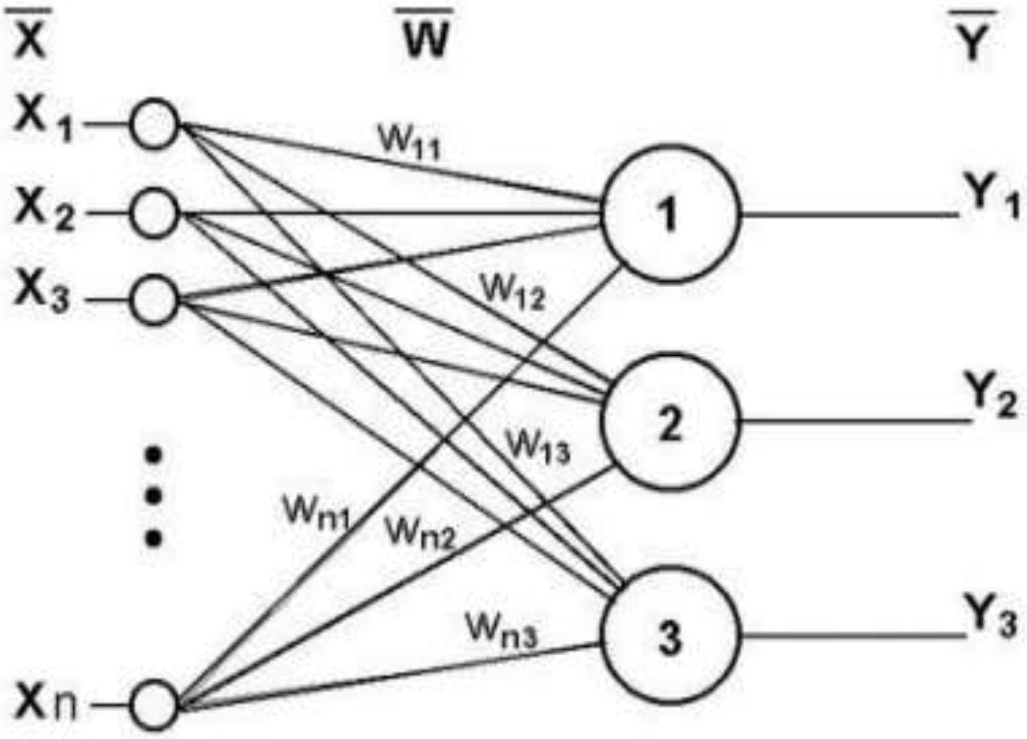


Fig. 2.7. Multilayer neural network

It uses algorithms of supervised learning; in other words, a learning set consists of input vectors, each of which is designated with a needed objective vector. Input vector’s components are represented as continuous range of values, and objective vector’s components are binary values (0 or 1). After learning, the network receives continuous input and develops a needed output in a form of vector with binary components [19].

Learning is conducted in the following way:

1. All network weights are randomized into small values.

2. Network input receives an input learning vector X, and by means of a standard formula, NET signal is computed for each neuron (2.2).

$$NET_j = \sum_i xw. \tag{2.2}$$

3. The value of a threshold activation function of NET signal for each neuron is calculated.

4. For each neuron the error is calculated by means of subtracting the resulted output from needed output (2.3).

$$error_j = target_j - OUT_j. \tag{2.3}$$

5. Each weight is modified according to the following formula (2.4).

$$W_{ij}(t+1) = W_{ij}(t) + ax_j error_j. \tag{2.4}$$

6. Steps 2-5 are repeated until the error is significantly less.

Rosenblatt perceptron is limited with binary outputs. Bernard Widrow together with Marcian Hoff extended the learning algorithm for perceptron in case of continuous output, using sigmoid function. They developed the mathematical proof that a network under certain circumstances would limit itself to the function being imagined. Their first model called Adaptive Linear Element (ADALINE) has one output neuron; the later model called a multilayer network of ADALINE (MADALINE) extends it for the case with many output neurons.

Formulas describing the work of ADALINE are very similar to perceptron ones. Significant differences are hidden in the fourth step, where continuous signals NET replace binary OUT. The modified fourth step in this case is implemented in the following way [20]:

4. For each neuron, the error is calculated by means of subtracting the resulted output from needed output.

The results of researching self-organized structures for pattern recognition by Kohonen classify patterns as vector values, in which each vector component corresponds to the pattern element. Kohonen algorithm is based on the method of unsupervised learning. After learning, sending input vector from a given class will lead to the development of an excited level in each output neuron; neuron with the maximal corruption represents the classification. As learning is conducted without denoting the objective vector, there is no opportunity to predict, which neuron will correspond to the given class in input vectors. However, such a planning is easily performed by means of testing the network after learning.

Algorithm interprets a set of n-number of input neuron weights as a vector in n-dimensional space. Before the learning, each component of this weight vector is initialized in a random value. Then, each vector is normalized into a vector with a length of one symbol in the space of weights. This is performed by dividing each random weight with a square root of sum of component's squared values in this weight vector.

All input vectors of a learning set are also normalized, and the network is trained according to the following algorithm [21]:

1. Vector X is sent to the network input.

2. D_j distance (in n-dimensional space) between X and weight vectors W_i of each neuron is calculated. In Euclidean space, this distance is calculated by the following formula.

$$D_j = \sqrt{\sum_i (x - w)}. \tag{2.5}$$

3. The neuron having the weight vector most close to X is stated as the winner. This weight vector called W_C become the main one in the group of vectors, which lie within distance D from W_C.

4. A group of weight vectors is configured according to the following formula (2.6).

$$W_j(t+1) = W_j(t) + \ \alpha\,[X - W]. \tag{2.6}$$

5. Steps 1-4 are repeated from each input vector.

During the learning of ANN, D and α value gradually decrease. Coefficient α at the beginning of learning can approximately equal to “1” and gradually reduce till “0”, while D could equal the maximal value of distance between weight vectors and become so little in the end, that only one neuron would be able to learn.

In terms of existing viewpoint, classification accuracy will improve with additional learning. According to Kohonen’s recommendation, in order to receive good statistical accuracy, the number of learning iterations should be at least 500 times more than the number of output neurons.

The learning algorithm configures weight vectors in the area of excited neuron in a way, that they would resemble the input vector. Since all vectors are normalized into vectors of one symbol length, they could be considered in terms of points on a single hypersphere surface. During the learning process, a group of nearby weight points relocated closer to the point of input vector.

Input vector are assumed to group into classes, according to their state in the vector space [22]. A certain class will be associated with a given neuron, moving its weight vector in the direction of the class center and maintaining its excitation in case of any vector of this class appearing at the input. After learning, classification is conducted by means of sending a testing vector to the network input, calculating the excitation of each neuron with further choice of neuron with the highest excitation as the indicator of correct classification.

2.3. Perceptron as recognition model

A great impulse to the development of cybernetics was given by Frank Rosenblatt – the neurophysiologist, who proposed pattern recognition mode called “Perceptron” (from Latin perceptio – comprehension). While developing it, Rosenblatt relied on some accepted concepts about the structure of brain and visual apparatus. Trying to recreate functions of a human brain, he used a simple mode of a biological neuron and the system of bonds between them (fig. 2.8).

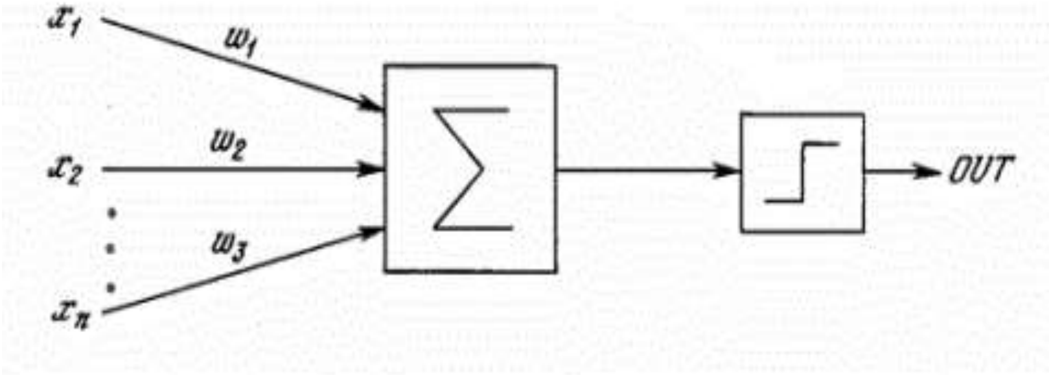


Fig. 2.8. The model of a perceptron neuron

A receptive device of a perceptron is inspired by the photoelectric model of eye’s retina – the receptive field with hundreds of photo-resistances (so-called S-elements). Each element of the receptive field can be in one of two states – excited and ground – depending on whether an edge of the projected figure is hit by a photo-resistance or not. Signal x_i appears in each element output (i = 1, 2, ..., n, where *n* is the number of elements) and equals “1”, if the element is excited, and “0” in the other case. The next perceptron stage are so-called associated elements or simply A-elements. Each A-element has several inputs and one output. While preparing perceptron for the experiment, receptor outputs are connected to the input A-elements, and every connection can be performed with either positive or negative sign (fig. 2.9).

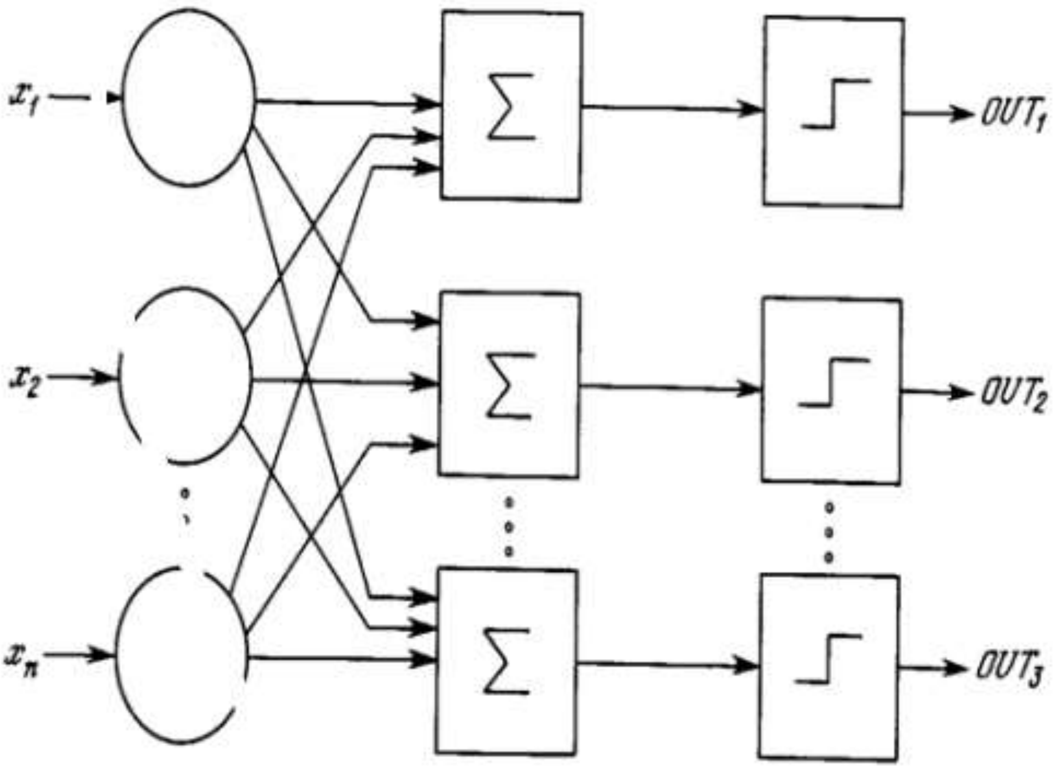


Fig. 2.9. A neuron with many outputs

The choice of receptors connected to a given A-element along with the sign of connection is conducted randomly. During the experiment, the connection of receptors with A-elements remains unchanged. A-elements perform algebraic signal summarizing [23], and received sum is compared with value θ, which is same for all A-elements.

If the sum is greater than θ, A-element is excited and outputs signals equal to “1” [24]. If the sum is less than θ, A-element remains grounded and its output equals “0”. A-element output signals are multiplied by alternating coefficients λ_j by means of special devices (intensifiers). Each of these coefficients can be positive, negative or equal to “0”, changing independently from other coefficients. Output signals of intensifier are summed, and the summarized signal is sent to the input – so-called reacting element or R-element. If σ is positive of equals “0”, the R-element outputs “1”, and if σ is negative – “0”.

Let us assume that figures belonging to two different patterns are projected on the receptive field. If it is possible to put a perceptron in such a state that he outputs “1” for figures of the same pattern with acceptable reliability, it means that a perceptron is able to learn pattern recognition [25].

The described structure of a perceptron allows distinguishing proposed objects into only two sets. In order to recognize a larger number of patterns – for example, A, B and C – one can apply a perceptron, build with the following scheme, demonstrated in Figure 2.10.

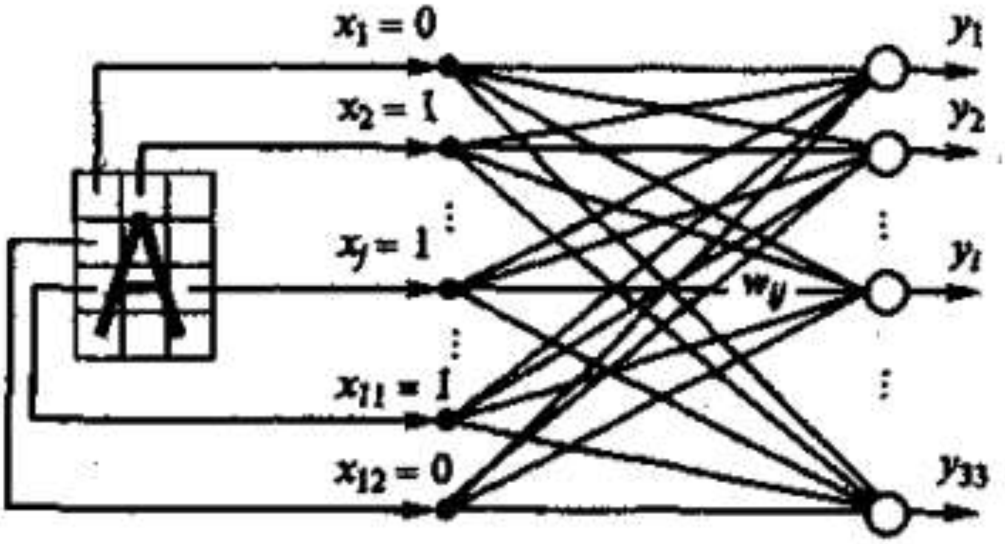


Fig. 2.10. A perceptron with several outputs

The output signal of each A-element is sent to several intensifiers, according to the number of patterns to be recognized). After multiplying by λ, output signals are sent to Σ adders, the number of which equals the number of patterns to be recognized. R-element is replaced by the device, which compares output signals of adders. The resulted object is attributed to the pattern, which accumulator has the strongest signal.

In order to recognize several patterns, it is possible to use a perceptron of a bit another structure. In such a perceptron A-elements are divided into several groups, each connected with its adder and R-element. A group of output signals by R-elements can be regarded as a pattern number represented in a binary code, thus, giving the perceptron an opportunity to distinguish objects into several classes. For instance, three groups are enough to perform classification into eight classes. In this case, there are eight possible combinations for the output signals of R-elements: 000, 001, 010, 011, 100, 101, 110, and 111. The appearance of each combinations can be considered as the attribution of a proposed figure to one of eight patterns by the perceptron.

In terms of structure and performance, each group of A-elements connected with its R-element is quite similar to a perceptron, which is able to distinguish objects into two classes. Perceptron's learning represents a set of sequential iterations. Each iteration the perceptron is given an object from one of the patterns. Depending on perceptron's reaction to the given object, alteration of λ_j coefficients is conducted according to certain rules. Over a finite quantity of iterations, it is possible to put the perceptron in such a state, when it can recognize proposed objects with quite accurately.

There are two possible ways to train a perceptron. The first one does not take into account the correctness of perceptron's answers during the learning process, and changing λ_j coefficients occurs despite perceptron "knowing" or "not knowing" the proposed figure. The algorithms of the second way presume that λ_j coefficients change according to perceptron's answers and their accuracy.

Algorithms of the first type are conducted in the following way. Beforehand assumed that the perceptron has to output "1" in case of recognizing objects from pattern A, and "0" in case of pattern B. Then the perceptron is given objects from both patterns. At each iteration, the perceptron answers to a given object by exciting some A-elements. Learning lies in the concept, that λ_j coefficients of disturbed A-elements increase by some value (for instance, by "1"), if the object belongs to pattern A. And they decrease by the same value, if the object belongs to pattern B. Naturally, such λ_j coefficients alterations have to increase perceptron's accuracy, since increasing λ_j in A-elements leads to increasing signal at R-element input, and vice versa. According to the assumed condition, the perceptron will produce correct answers, if positive signals correspond to pattern A, and negative signals at R-element input – to pattern B.

Developing a perceptron, Frank Rosenblatt tried to simulate some functions of a human brain [26]. Perceptron or any program imitating recognition process work in two modes: learning mode and recognition mode. First, in contradiction to aforementioned algorithms, the perceptron algorithm does not require remembering all the proposed objects during learning – it also does not require looking through all "known" objects during recognition. In this way, perceptron performance is rather similar to brain performance, which forms an idea of a pattern without remembering all of its representatives and recognizes objects without comparing it with all previously seen. Then, perceptron's structure has common features with nervous system. Particularly, perceptron receptors are quite analogue with visual apparatus receptors, and A-elements can be compared with neurons. It is known that neurons are able to be disturbed, the signal intensity received by a neuron exceed certain threshold value. The ability of a perceptron to allow random connections "receptor – A-element" can be compared with similar features of human brain structure.

It is possible that connections between neurons in a brain is random in the majority of cases, i.e. randomly vary for different animals of the same species. If opposite assumed (all connections between brain neuron are fixed and the same for all animals of a given species) and alternating these connections can lead to damage in the brainwork, then it can be assumed, that all these connections are inherited. Taking into account billions of neurons, it must be a tremendous volume of genetic information. Yet, with a perceptron example one can see, how natural the compact set notion is, since learning to recognize such sets can be possible with random connections between receptors and neurons.

The brain is able to store and recreate many functions in case of a damage or ailment. Perceptron consistency resembles this function of human brain. All these assumptions cannot lead to the conclusion that brain and perceptron operate under the same algorithm. However, a perceptron may be the most true-to-nature brain model right now.

With the criterion of linear division, it is possible to state whether single-layer neural network is able to implement the needed option. Even if the answer is positive, the question is still to find the reasonable way of obtaining values for weights and thresholds. To make the network present practical value, it is necessary to use the systematic method (algorithm) to calculate these values. Rosenblatt implemented this in his learning algorithm for the perceptron along with the proof, that it can be trained with anything it is able to implement.

Learning, as mentioned, can be either supervised or unsupervised. Supervised learning requires an "external" spectator, who would evaluate system's behavior and controlled its further modifications. Unsupervised learning assumes the networks is able to realize the needed alterations by itself [27].

Perceptron learning algorithms can be implemented on a digital computer or any electronic device – the network becomes self-adjusting in a certain way. This is where the notion “learning” comes from. Rosenblatt’s works and developments became a major milestone in the discipline and gave a strong impulse for its further investigation.

Conclusions on Part 2

The algorithmic universality of computers means that any algorithmic transformations can be implemented in a form of machine programs: computing, controlling, searching, graphic and pattern recognition algorithms. Among noticeable features of ANN, the most important one is the ability to learn diverse methods and techniques.

The majority of learning methods are suitable for basic configuring and possess similar characteristics. ANN learning resembles the process of human learning, though ANN capabilities are quite limited. A network is learning in order to provide a needed set of output for each set of inputs, and each of them is regarded as a vector.

Learning is conducted by means of a successive demonstration of input vector with simultaneous configuration of weights according to a certain procedure. In the end network’s weights gradually obtain values so that any input vector resulted into an output vector. Learning algorithms can be classified as algorithms of supervised or unsupervised learning.

Convolutional neural network (CNN) is the most widespread, suitable and flexible implementation of different machine learning algorithms in the sphere of pattern recognition. The work of CNN will be thoroughly covered in Part 3.

PART 3
CONVOLUTIONAL NEURAL NETWORK

3.1. Hybrid architectures of convolutional neural network

The architecture of convolutional neural network (CNN) is designed specifically for efficient pattern recognition. In its core, there lies the work of convolutional layers called convolutions with non-linear activation functions like rectified linear unit (ReLU) or hyperbolic tangent *tanh* and connecting layers, called pooling layers.

On the contrary to feedforward neural networks (FNN), where each input neuron is connected with the output neuron of the next layer, CNN use convolutions over each input layer in order to receive output values. The convolution operation requires the matrix of weights of small size running through the current layer and after each shift forming the activation signal for next layer neuron in analogical position. This matrix is known as the convolutional core; it is applied for different neurons of the output layer. The result of CNN work depends on quality and quantity of learning data, the type of ANN architecture and quantity of parameters chosen for the given network.

There is a great number of ANN architectures, and apart from perceptron and CNN, one can highlight radial basis function (RBF) networks and recurrent neural networks (RNN). RBF network is ANN working with radial basis activation function ρ(γ). These networks contain only three layers: input, hidden with RBF as the activation function and output with the linear activation function. In most cases, Gaussian function is used as activation RBF ρ in most cases (3.1).

$$\rho(\gamma) = \exp(-\gamma^2). \tag{3.1}$$

CSN department				NAU 19 74 09 000 EN			
Done by	Klymenkov S.V.			Convolutional neural network	Letter	Sheet	Sheets
Supervisor	Nadtochii V.I.				E	56	109
Consultant					6.050102 CS-422a		
S-inspector	Nadtochii V.I.						
Head of dep.	Zhukov I.A.						

ANN is called recurrent, if neuron within it have feedback connections, i.e. output signal is transmitted to other neuron input, which is located in the layer with smaller index. The presence of feedback connection maintains receiving data not only out of the previous layers, but also from previous learning iterations – this guarantees saving temporary data. It means that the result of learning will depend also on the sequence of transmitted learning data.

CNN are very similar to FNN like perceptron: neurons of such a type have weight and shift parameters learned. However, CNN take into account that input data is consisted of images and after obtaining this information, it is possible to encode certain features in ANN structure, which allows reducing the number of parameters in the ANN. CNN and RNN are often mentioned together, therefore, it is necessary to understand differences between them, described in Table 3.1.

Table 3.1

Main differences between convolutional and recurrent neural networks

CNN	RNN
• possess inputs and outputs of the fixed size; • belong to the type of ANN with feedback connection – a variation of multilayer perceptron, designed to use minimal volumes of preliminary processing; • use the scheme of interaction between neurons similar to the organization of animal's visual cortex; • can be implemented for image and video recognition.	• can process arbitrary size on input/output; • on the contrary to related ANN can use its internal memory to process arbitrary sequences of inputs; • use information of time sets; • can be implemented for text and language recognition.

As a rule, any ANN architecture has its own advantages and disadvantages. Network hybridization can be one of the methods to solve the problem of network shortcomings. What is meant by here is the convergence of several architectures into one, in order to improve certain problems. Hybridization with non-network models can be also found and this presumes very different methods of pattern recognition. The main principle while building a hybrid architecture is using strong features of the models and avoiding their weak sides.

In other words, hybrid ANN are the merge of several neural networks to solve certain tasks. It allows dividing complex tasks intro simple subtasks, and the architecture of ANN can be optimized for the given task. Another great advantage of hybrid ANN is that with software implementation the speed work can vary in wide range depending on the chosen structure, which allows using them to solve tasks in real time.

Hybrid ANN join several "experts" different in their structure and origin, therefore, each of them can generate different responses to the same input set. In order to form the general answer in hybrid ANN, the most used method is to apply weighted result summarizing or dynamic choosing of certain ANN with further using its result as the output value. There are algorithms, which allow picking out the optimal number of neural networks, performing their correct organization and training to solve the given tasks with predefined accuracy.

3.2. The logical and physical topologies of the corporate computer network

3.2.1. General structure

It is well known that multilayer ANN receive input data (for example, one vector), and then transform this data by running it through a set of hidden layers. Each hidden layer consists of a set of neurons, where each neuron has strong connection with all the neurons from the previous layer and where neuron as one layer are fully independent from each other without any connections. The last fully connected layer is called the output layer, and in classifying configuration, it demonstrates a number of classes.

Common ANN (fig. 3.1) are badly scaled in case of images of big sizes. In the system of computer vision CIFAR-10, images have the size of [32 × 32 × 3], where “32” is width, “32” is height and “3” is the color channel. Therefore, one fully connected neuron in the first hidden layer of ANN has the weight 32 * 32 * 3 = 3 072. This value seem to be changeable, however, the fully connected structure is not scaled up to large images. An image of a higher volume – for example, [200 × 200 × 3] – will lead to the fact, the fully connected neuron will weigh 120 000. In addition, it is necessary to invoke several neurons, which results into adding parameters. The major disadvantage of full connection is a large number of parameters resulting into overfitting.

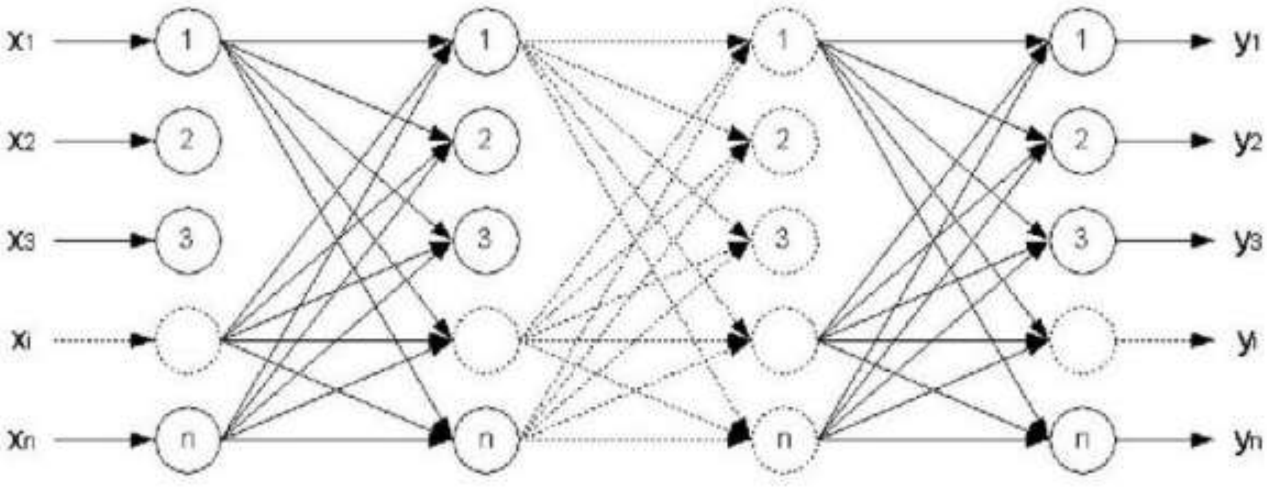


Fig. 3.1. A multilayer neural network

CNN take the advantage on input data consisting of images, and they limit building the network with a more reasonable way (fig. 3.2). On the contrary to common ANN, CNN layers consist of neurons, located in three dimensions: width, height and depth, i.e. in dimensions forming the volume.

For example, images at the CIFAR-10 input are input activation volume, and the volume is formed with dimensions 32 × 32 × 3. As described further, neuron will be connected only a small area of the layer. Besides, the resulting output layer of this system of computer vision will be 1 × 1 × 10, since the image will transform in the single vector of class values, located according to measuring depth, by the end of the CNN work.

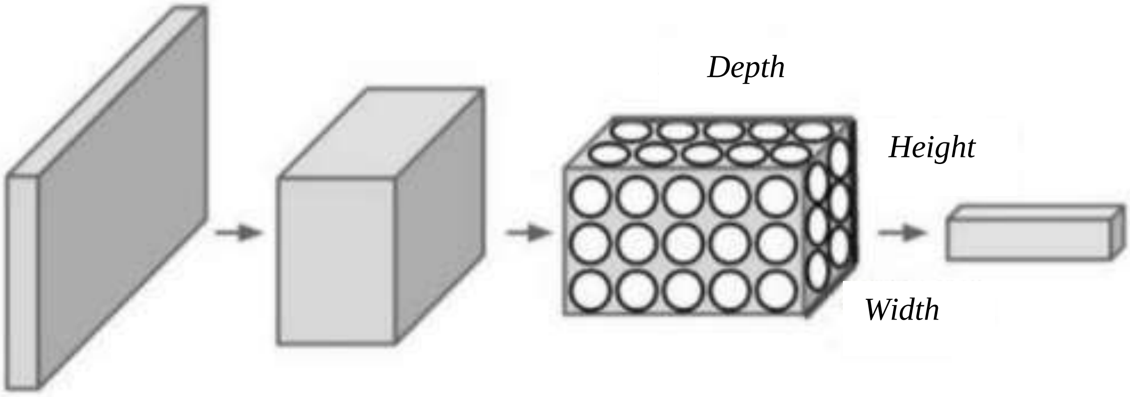


Fig. 3.2. A convolutional neural network

Thus, the base of CNN is layers. Each layer is characterized with a simple set of tasks. It transforms input data in a form of 3D-volume into output 3D-volume with some differentiated function, which be either with or without any parameters.

3.2.2. Convolutional layers

Unconditionally, the main layer for the work of CNN is the convolution layer, which is the basic of CNN operation. Convolutional layer parameters consist of a set of learning filters (another common name is Kernel cores). Each filter has special dimensions (width and height), running through the whole depth of input volume.

For example, a standard filter of the first layer of CNN can be of size [5 × 5 × 3]. While moving forward, a somewhat sliding of each filter over width and height of input data occurs, and scalar product between filter records and input in any state is calculated. To the extent of filter running through image’s height and width, a two-dimensional activation map is constructed, which provides filter’s response over each spatial position (fig. 3.3).

The network trains filters, activated at a certain visual peculiarity. It can be the edge of certain direction, maculation of a certain color on the first layer or circle-like pattern on higher machine levels. After this, the network deals with a large set of filter on each convolutional layer, and each filter will form a separate two-dimensional activation map or a feature map. Feature maps are built along the “depth” of input volume, forming the output volume.

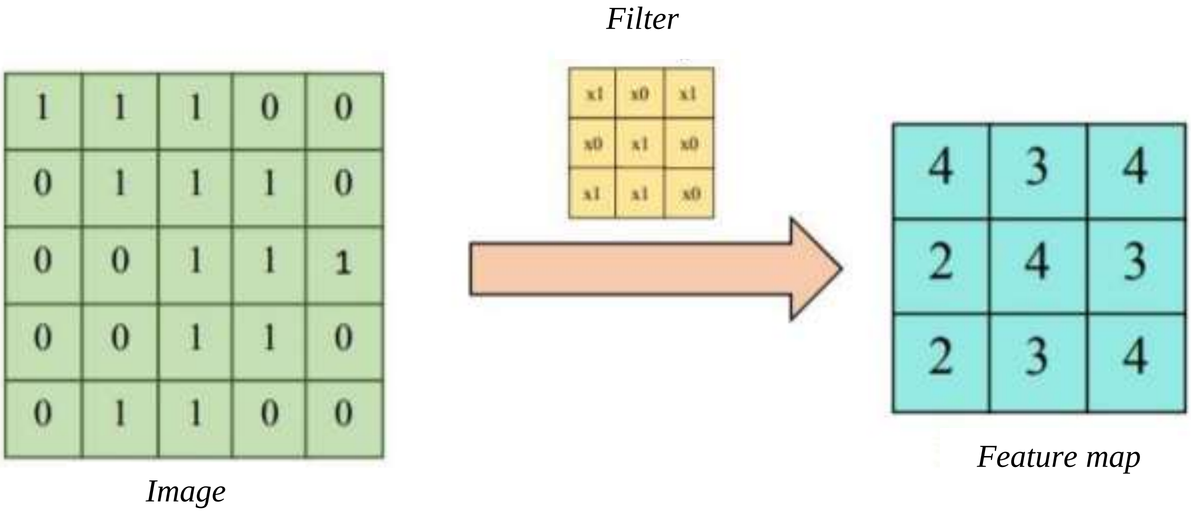


Fig. 3.3. The operation of convolution (forming feature maps)

Working with input information of big sizing, setting connection between a given neuron with all the previous volume is irrelevant. Instead, each neuron should be used only for the local area of the input volume. Spatial elongation of its connection is a hyper-parameter called the receptive field. The idea of the receptive field lies in joining neurons of the hidden layer with the similar ones from the previous layer, located within an area n × n. Moreover, for each neuron the separate area is chosen, which is called as the local receptive field. Receptive field indicate elementary visual features such as angle or edge, and their combinations allow receiving features that are more complex.

It goes without saying, that receptive field are equals the filtering area. Spatial elongation along axis of depth is always equivalent to the depth of input volume. The asymmetry of spatial dimensions like width, height and depth should be additionally emphasized – connections over width and height are local, but run through the whole depth of input volume.

An example of input value (a rectangle with parameters [32 × 32 × 3], CIFAR-10 image) and the possible volume of neurons in the first layer of CNN is shown in Figure 3.4. Each neuron in the convolution layer is connects with the local input area, where spatial elongation is defined by width and height, but the neuron runs through the whole depth encompassing all color channels.

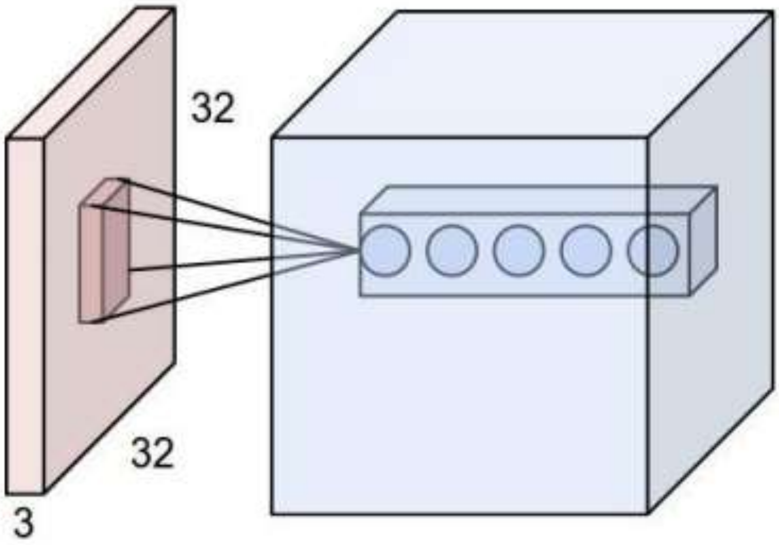


Fig. 3.4. Connection between the convolutional layer and the local input volume (a part of the input image)

Neurons of CNN remain unchanged: as common ANN neurons, they compute the scalar product of their weights and input data with further non-linearity. However, their connection is limited with local spatial dimensions.

3.2.3. Parameters and hyper-parameters of the network

There are three hyper-parameters of the network, which control the size of output information: depth, stride and zero extension. The hyper-parameter of depth is equivalent to the number of applied filters; each filter is trained to search for different features at the input. For instance, the first convolution takes unprocessed image as input data, where different neurons along the depth can activate in case of clots of certain color. A set of neurons “looking” at the same area of the input volume can be noted as depth column.

Another significant hyper-parameter is stride, with which a filter performs its run. The size of a stride defined the value of filter shift at each iteration. The larger the stride is, the less usage of filter is, resulting in a smaller output matrix. Usually, a stride equal to “1” is used, though a larger stride can allow building a model resembling RNN (i.e. CNN with a large will look like a tree); this will allow forming smaller output volumes of spatial dimensions.

While filter is running, there are certain situations when it is convenient to fill the edge of the input image with zeroes. The size of zero extension is the third hyper-parameter. The key feature of zero extension is that it allows controlling the spatial size of output volumes (often this feature is used in order to save the spatial size of input data so that input and output values of width and height remained the same). Without using zero extensions narrow convolution is received.

The spatial size of output volume can be calculated as the function if the input volume, the size of receptive field of convolution layers neurons, a stride of moving and a number of zero extensions on input image edges. The parameters of the convolution layer are:

- K – number of filters;
- F – size of a filter;
- S – stride;
- P – number of zero extensions;
- W_1 – width;
- H_1 – height;
- D_1 – depth of the input data.

The size (or volume) of output data W_2 × H_2 × D_2 is calculated by means of the following formulas (3.2 – 3.4). Shared usage of parameters is applied in the convolutional layers in order to control the number of parameters.

The size (or volume) of output data W_2 × H_2 × D_2 is calculated by means of the following formulas (3.2 – 3.4). Shared usage of parameters is applied in the convolutional layers in order to control the number of parameters.

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1. \tag{3.2}$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1. \tag{3.3}$$

$$D_2 = K. \tag{3.4}$$

Another exceptional parameter of CNN is the layers of composition. Layers of composition help to reduce the sizing of output data, while keeping the most significant information. For example, if the filters defines whether there is an obvious feature. In case of an obvious feature, the result of applying the filter to this image area will result in the big value, and it will result with small values for other parts of the image.

Color channels represent different "viewpoints" on input data. For instance, in image there are usually three channels in a form of RGB.

3.2.4. Maximal and averaged subsampling layers

CNN architectures use the subsampling layer between sequences of convolutional layers. Its function lies in the gradual reduction of spatial dimensions of an image in order to reduce the number network parameters and computations overall, and control the overfitting. Subsampling layer works despite the section of depth in input data, and scales the volume in space using the function of maximum.

Often the layer with filters of 2 × 2 size with stride 2; such a layer reduces the discretion of each input depth section twofold over width and height. Each operation of maximum in this case will choose the maximal value from four numbers. The size of depth remains the same. Generally, the subsampling layer receives the data volume of size $W_2 \times H_2 \times D_2$ and requires two hyper-parameters: elongation and stride.

In each convolutional layer for each channel, there is a separate filter, which convolutional core processes the previous layer by fragments. The result of applying different filters are composed. This is how the composing layers are formed.

The operation of sub-discretion performs reduction of sizes of formed feature maps. In the given network architecture it is assumed, that information about the presence of needed feature is more important than the exact knowledge of its coordinates. Therefore, from several neighboring neurons of a feature map one maximal is chosen of a compressed feature map of smaller dimension. By means of this operation, apart from boosting future computations, the network becomes flexible to the scale of input image.

Along with subsampling maximum, given layers can perform other functions, represented in Table 3.2, for example, averaging subsampling or even L_2-normalized subsampling. Usually, the averaging subsampling has been used quite often, but lately it fell by the wayside comparing to the maximal subsampling, which works better in practice.

Table 3.2

Types of activation *a* subsampling layers in CNN of set *A*

Function name	Function definition
Maximal subsampling	max {a ∈ *A*}
Averaging subsampling	$\frac{1}{\vert A\vert}\sum_{a \in \mathrm{A}} a$
l_2 subsampling	$\sqrt{\sum_{a \in \mathrm{A}} a^2}$
Stochastic subsampling	Probability $p_i = \frac{a_i}{\sum_{a_j \in \mathrm{A}} a_j}$

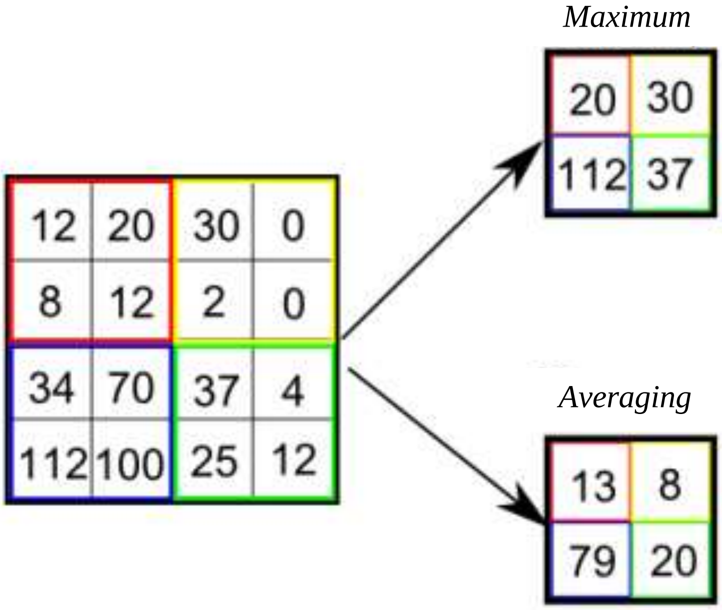


Fig. 3.5. Subsampling operation

Figure 3.5 demonstrates the subsampling operation. In this case, subsampling operation uses maximizing and averaging function.

3.2.5. Key methods for CNN implementation

One of biggest problems of using deep neural network, whether multilayer or convolutional is overfitting. Overfitting happens when the created model starts explaining only example from the learning set adapting to them instead of learning to classify samples, which did not take part in the learning (losing the ability for generalization). Over the recent years, numerous solutions have been proposed to solve the overfitting problem, but only one exceeded others due to its simplicity and achievements in practice.

The dropout technique can be described as the method used to prevent overfitting and certain adaptation of neurons towards input data, by means of setting output signal of each neuron as “0” with probability p. Inoperative neurons do not contribute to the learning process at any stage of error backpropagation algorithms, which are often used for network learning. Therefore, switching off at least one neuron is almost equivalent to the training of a new neural network.

The graphic representation of the dropout method is taken from the article, where this technique was published for the first time (fig. 3.6). The main idea of the dropout method is to train the ensemble of several ANN instead of training only one, and then average received results.

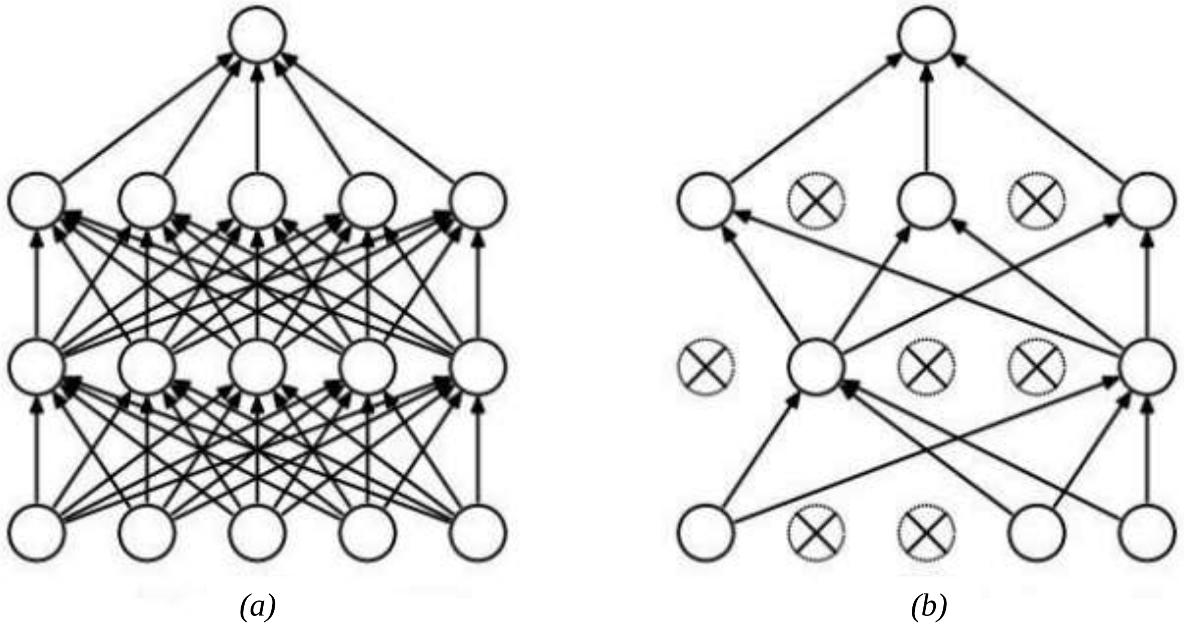


Fig. 3.6. ANN before (a) and after (b) implementation of the dropout method

In a standard ANN, a derivative received by each parameter reports the very parameter how it should change in order to minimize the loss function. Thus, blocks can alternate along with correcting the mistakes of other blocks. This can lead to excessive co-adaptation, which results into overfitting, since these co-adaptations are unable to be generalized for the data that did not take part in the learning. The dropout method prevents co-adaptation for each hidden block by creating a small percent of reliability of other hidden blocks existence. That is why the hidden block cannot rely on other block while correcting its own mistakes.

The probability of switching off is the same for all neurons. Implementation of the dropout method during learning can be represented as alternated activation function (2.5):

$$out = D \ \cdot a(h), \tag{3.2}$$

where a(h) is activation function, and "•" symbol is Hadamard product.

In multilayer networks, each layer receives data from the previous one. Data can be represented in a form of any tensor, which coordinates are defined beforehand. As a result, parameters in further layers become worse. The necessity of choosing a lesser speed of learning comes up – it is essential to configure the fact that input functions can rapidly alternate over the time.

For the receiving layer it would convenient to get data in a form of tensors with coordinates of fixed distribution – the same for all layers – then training transformation to distribution parameters become useless. It becomes relevant to set impositions between layers, which would normalize previous layer outputs and reduce pressure on the next layer, significantly simplifying network's performance. Thus, the way to solve this problem lies in the normalization of "batches".

Two methods of normalization are widespread today: local response normalization and batch-normalization. Both normalizations are used to prevent reduction in learning speed of network's parameters. Local response normalization of correspondence is the individual network layer. The given operation normalizes each value of the input matrix via the channel.

Batch-normalization applies standard normalization to received values and then perform their linear transformation (3.3):

$$y = \frac{x - \mu}{\sqrt{\sigma^2 - \in}} \cdot \gamma + \beta, \tag{3.3}$$

where ∈, γ and β are parameters to be configured. Average value μ and dispersion σ depend on the state, in which the network is now. If learning is in the process, these values are taken from values of the current moment. While network testing, these values are taken from the completely gathered statistics. The question of where normalization (normalizing layer) to be used is still opened. For the dropout it is insignificant whether normalization is applied before or after the activation function. Considering this, several options are possible:

- Convolution layer / Classifier (fully connected layer) → BN (normalization) → activation function → Dropout → …;
- Convolution layer / Classifier (fully connected layer) → activation function → BN → Dropout → …;
- Convolution layer / Classifier (fully connected layer) → activation function → Dropout → BN →…;
- Convolution layer / Classifier (fully connected layer) → Dropout → BN → activation function → ….

Each of these scenarios has its own advantages and disadvantages depending on the given task to solve. The most widespread option is to perform batch-normalization right after using the convolution operation and before applying the operation of non-linearity (activation function).

Like common ANN, in CNN neurons of fully connected layers (classifiers) have the link with all activation functions of the previous layer. Activations of classification layers can be calculated by means of multiplying matrices followed by shift. The difference between classifying and convolutional layers lies in the fact that convolutional layer neurons are connected only with the local area at input, and they can co-use the parameters.

However, neurons in both layers despite their peculiarities obtain the scalar product, so their functionality is identical. Furthermore, one can perform conversion between fully connected and convolutional layers.

A classifier has to remember all learning data and store them for further comparison with data from the testing input. It is very resource demanding, since datasets can weigh up to gigabytes.

A linear classifier (fig. 3.7) evaluates class as the weighted sum of all pixel values in three color channels. Depending on which values are set on these weights, the classifying function is able to evaluate (depending on the sign of each weight) certain color in certain image locations as positive or negative. For instance, a human will state that the class named “ship” can be more possible, if there is a lot of blue color at the sides (which may presume water).

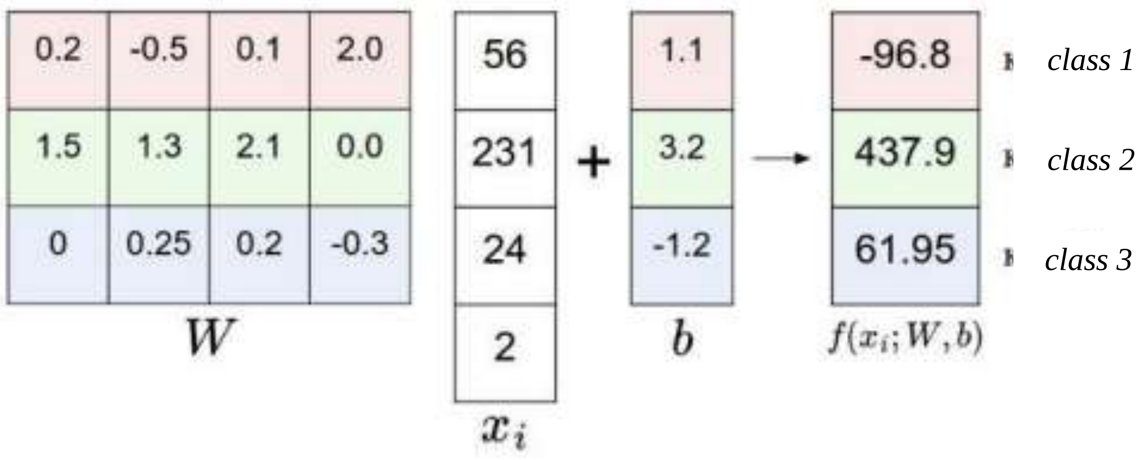


Fig. 3.7. A linear classifier

Another way to organize a classifier is the templating. The feature of this method relates to weights W and lies in the fact that each W-row corresponds to the template (which is also sometimes called as prototype) for one of classes. Evaluation of each class of images is received by means of comparing each template step-by-step with the image with the inner feature (or point-feature) in order to find the most suitable. With the help of this comparison, a linear classifier finds the correspondence to the template.

A popular choice of classifier is the Softmax classifier, which has a different loss function and the base of which is the normalized exponential function. This classifier in its origin is the generalized classifier of binary logistic regression.

On the contrary to the method of support vector, which calculates results of the vector *f (x$_i$, W)* as the evaluation of each class, the Softmax classifier is controlled by a bit more intuitive approach, also having probability interpretation. In the Softmax classifier the loss function (3.4) of representation *f (x$_i$, W)* = *W x$_i$* remains unaltered, and the function of cross-entropy is connected (3.5).

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right), \tag{3.4}$$

$$L(X,Y) = -\sum_{n}^{i=1} y^i \ln a(x^i) + (1-y^i)\ln\left(1-a(x^i)\right), \tag{3.5}$$

where X = {x(1),…, x(n)} is the set of input samples for learning in the input set. Y = {y(1),…, y(n)} is the correspondent set of marks for the input set. Function a(x) is the output values of ANN with input value x.

Conclusions to Part 3

The architecture of convolutional neural network (CNN) is designed specifically for efficient pattern recognition. In its core, there lies the work of convolutional layers called convolutions with non-linear activation functions like rectified linear unit (ReLU) or hyperbolic tangent *tanh* and connecting layers, called pooling layers.

The convolution operation requires the matrix of weights of small size running through the current layer and after each shift forming the activation signal for next layer neuron in analogical position. This matrix is known as the convolutional core; it is applied for different neurons of the output layer. The result of CNN work depends on quality and quantity of learning data, the type of ANN architecture and quantity of parameters chosen for the given network.

This structured and detailed information will come in handy in practice. Part 4 will demonstrate the power of ANN in solving pattern recognition tasks.

PART 4
IMPLEMENTATION OF CNN LEARNING METHOD

4.1. Learning algorithms for ANN

The ability of learning is a fundamental peculiarity of a human brain that can develop. In terms of neural networks, one should consider configuring ANN architecture, neuron weights and their connections. All this has its influence on the efficient solution of the given task. Usually, ANN must configure its parameters (such as a weight for each neuron) according to the given learning sets. The feature of ANN to learn according to the given samples makes them more profound comparing with systems working according to predefined rules. Among all learning methods, two can be distinguished: determined and stochastic.

The class of determined methods interactively configures network parameters relying on the state of current parameters, input values, current and objective outputs. A common illustration for this method is the error backpropagation algorithm.

The class of stochastic methods alters network parameters randomly. Only those changes, which led to the improvement of the work, are kept. As an example of a stochastic method, the following algorithm can be an example:

1. Choose network parameters randomly and configure them on some small random value. Calculate received values according to proposed output values.

2. Compare these outputs with the objective results and calculate the difference of values. This difference is the error itself. The goal of learning is to minimize the value of error.

3. Error values reduced, configuration kept. If necessary, configuring value is put aside and chosen again.

Steps 2 and 3 are repeated until the network is trained. It is necessary to mention, that stochastic methods can fall in the “trap” of local minimum (fig. 4.1).

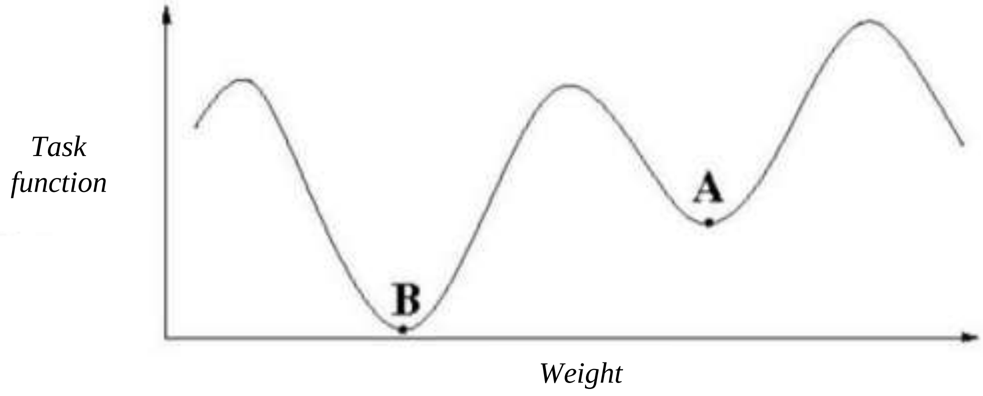


Fig. 4.1. The problem of local minimum

Let the initial value of error be equal or close to the value in point A. If randomly chosen strides are too small, deviations from point A increase the error and they should be ignored. In this way, the smallest value in point B will not be found. In case of random strides of network configuration being too big, the error will alternate so rapidly, that it will not stick to any of the minimums.

In order to avoid the problem of local minimum, it is acceptable to reduce gradually the random average size of configuring sizes. When the average size of strides is too big, the error value will reach any values with the equal probability. While gradually reducing the size of strides, one meets the condition at which the error value will “stick” to the point B. When the stride value will be reduced more, the error value will “stop” for a small period in both A and B points.

While reducing stride constantly, in the end it will reach such a value, which will be enough to overcome the local minimum A and not B. The mathematical representation of network learning can be depicted in the following way.

Therefore, there is an equation $Y = F(X)$, which solves the task, and input/output data set $(X^1, Y^1), (X^2, Y^2) \dots (X^n, Y^n)$. While training the network, it is necessary to find such a function, which will be close to the function with the error. I.e., knowing all these parameters, the task reckons up to the multidimensional function optimization.

The following algorithms are used to solve it:

1. Local optimization algorithms of computing partial first-order derivatives:

- gradient algorithm (the method of the fastest descent);
- methods with one- or two-dimensional optimization of task function in the direction of anti-gradient;
- methods, which take into account the direction of anti-gradient on several stages of algorithm.

2. Local optimization algorithms of computing partial first- and second-order derivatives:

- Newton method;
- optimization method with thin Hessian matrices;
- Levenberg-Marquardt method and others.

3. Stochastic optimization methods:

- random direction search;
- Monte Carlo methods and others.

Another separate class of algorithms is known as global optimization methods. They are designed to solve tasks with global optimization by means of brute-forcing values of variables, which the task function depends on.

4.1.1. Supervised and unsupervised learning

Supervised learning can be defined in the following way: the task of machine learning lies in the implementation of output function by virtue of known learning data. The structure of learning data consists of a set of samples to perform training. In this learning method, each sample form a pair (usually a vector) and a task output value (control signal). It is necessary to calculate function value at the input moment, then, compare this value with objective values, obtain the error and configure network parameters (neuron weights) on this error.

The supervised learning algorithm requires the following stages:

1. Divide the learning data for network in two parts. The first part will form the learning set, the second one – a testing set (for example, in proportion 70/30).

2. Calculate the function value for the learning set and find the value function error.

3. Conduct configuring weights of synapses in the network.

4. Repeat steps 2 and 3 until the error value is minimized. These actions can be repeated either for each sample of the learning set or the whole set. In the first case, learning will be conducted slower but more accurately. In the second case, index of accuracy reduces, but learning will be conducted faster.

5. Check all values over the testing set in order to understand whether the system could successfully generalize data, i.e. to learn.

If the ANN is trained using beforehand known correct answers, then such an algorithm is called supervised. It is necessary to note that supervised learning requires a large set in order to form a rather working and flexible neural network.

Unsupervised learning, also known as the method of uncontrolled learning, is able to find a structure or correlation between different inputs. The main difference from supervised learning is the presence of input data only. Unsupervised learning algorithm is applied when only input data is known. By virtue of input data, the network is trained to produce the best output results. The notion of “the best results” is defined by the learning algorithm itself. Usually, the algorithm configures parameters in such a way, that the network produced the same results for rather similar input values.

The most significant method of unsupervised learning is clustering, which creates different input clusters and can introduce any new data to the correspondent cluster. Apart from clustering, there are other methods of unsupervised learning: anomaly detection, Hebb learning theory and hidden changeable models, such as algorithms of expectation maximization, method of moments or method of dividing bling signals.

Though this method is often used for applied tasks, it is also often criticized by scientist for its biological inclination. It is hard to imagine that human brain has the mechanism of comparing received results with objective results.

4.1.2. Error correction method

This method of perceptron training, proposed by Frank Rosenblatt in 1957, assumes the weight of connection remains the same unless a perceptron makes a mistake. Parameters introduce changes in case of divergence between input and output values. For correction value calculation, the difference between current and needed output values is applied.

The given model uses supervised learning, i.e. a learning set consists of a set of input vectors with an output vector for each. Despite certain limitations, the model became the basis for many modern complex learning algorithms.

4.1.3. Competitive learning

The method of competitive learning lies in the rivalry between each output neurons over activation. This leads to the conclusion, that among all output neurons, only neuron with the biggest output come to work. Such an algorithm resembles biological “neural networks”. With the help of competitive method, one can classify output data, where similar samples are grouped together in one class and represented as one pattern element. Each neuron from the set of neurons is responsible for only one class.

The overall number of classes, which the network works with, is equal to the number of output neurons. Competitive learning is quite suitable to be applied for tasks of input pattern classification. In this case, each neuron of output layer is responsible for one pattern.

4.1.4. Hebb learning

Hebb learning is based on physiological and psychological researches. The learning algorithm can be represented in a form of rules in two parts.

The first part is as follows: if two neurons aside the synapse are activated simultaneously, the strength of the connection increases. The second rule states, that if two neurons aside the synapse are activated asynchronously, then this synapse is weakened or totally ceases to exist.

4.2. Error backpropagation algorithm

CNN, according to their type of activation signal propagation between neurons, are feedforward; therefore, implementation of error backpropagation method is quite relevant for these networks. This algorithm is also known gradient descent method, since the strategy of weight obtaining for each neuron of multilayer ANN is based on the gradient method. Constant task function as the index of network success is defined as squared difference of sum between the current result and expected output value.

Error backpropagation method during the learning process applies two types of propagation – forward and backward. At the very beginning, the forward running occurs, where input data in a form of vector implement propagation between layers, from the first to the last. Because of forward running, a set of output signals is generated, which defines network reaction on input data. All synaptic network weights are fixed during forward running. The second stage of the algorithm is backward running, where parameters (all synaptic weights) are configured according to the rules of error correction. The essence of the mentioned rule is as follows: the value of current output is subtracted from the value of expected output, and the result of this operation forms the signal of error. Error signal propagates as echo in the opposite way of synaptic connection, thus receiving its name. Synaptic weights in their turn are adjusted, so that the output signal maximally closes to the expected result.

A shortcoming of error backpropagation is that it does not allow reaching the global minimum. This is one of the main challenges in ANN learning, which is the method of leaving local minimums. Among other disadvantages there are:

1. Network “paralysis”. Weight values in the end of correction can reach very big values. As the error sent backwards while learning is inverse to the derivative of the squashing function, the learning process can stop. This can be avoided by reducing the stride; however, the learning process will take much longer.

2. Size of a stride. If the value of a stride does not change and it is quite small, then the method is going to work slowly. If the stride is too big, the network can end up being paralyzed.

4.3. ANN parameters: accuracy, loss and quality

4.3.1. Overview of main parameters

Epoch is the forward and backward running through all training samples. The size of a set (batch) is the number of training samples for one iteration of forward and backward running. The number of iterations is the number of running, each using batches. One running equals forward and backward ways. Therefore, with 1000 samples and 500 batches, it is necessary to make two iterations to finish one epoch.

In terms of mathematics, ANN learning is the multi-parameter task of non-linear optimization. The ANN learning process is described by means of a line chart, where horizontal axis depicts the number of training samples and vertical axis represents the value of error, received during classification.

Figure 4.2 demonstrates two lines: the curve of error values over the learning set (horizontally oriented value) and the curve of error values over the training or testing set (of a fixed size). The more data is used during the learning, the more errors the network architecture will produce, correspondent to the learning data. At the same time, learning data become similar to the real data division, which should be noted according to the learning data. At a certain moment, the error over learning and testing sets should be almost the same.

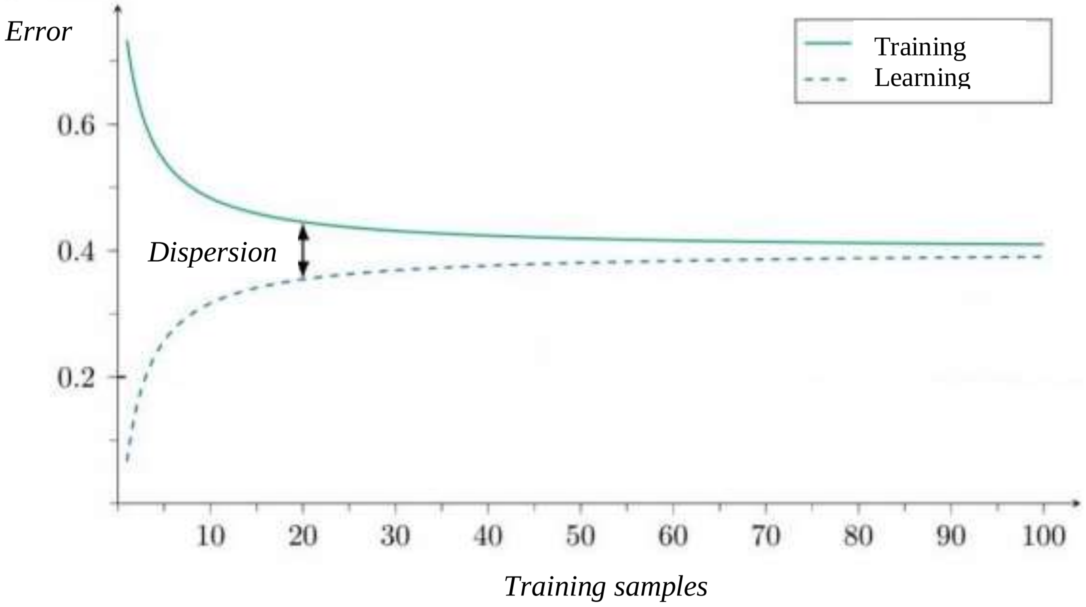


Fig. 4.2. The process of ANN learning

The curve of learning is the index of network successful performance, if there is a great quantity of learning data without any alternations. After building the curve of learning, one can evaluate, whether network architectural model is able to perform the needed error classification. Error value over the training set should never be much smaller, than error value over the learning set. If the error over the learning set is too big, than increasing data volumes will not improve the situation. Instead, it becomes clear that the very architecture or algorithm have to be reconfigured. In case when the learning curve is much greater than the training curve, it is necessary to either reduce the resolution of images or introduce additional learning data samples, or increase regularization to solve the problem.

Network parameters come in handy to computation and reflection of hyper-parameters (for example, of the learning epoch). Parameters can be presented in a form of hyper-parameter chart on the horizontal axis and quality value on the vertical one. The recognition accuracy represented as error or loss are typical quality parameters. In case when the number of learning epochs is regarded as the main hyper-parameter, parameters act as indexes of long-time training and machine's performance. After building the chart of error values over the training and learning sets, one can evaluate the possibility of overfitting.

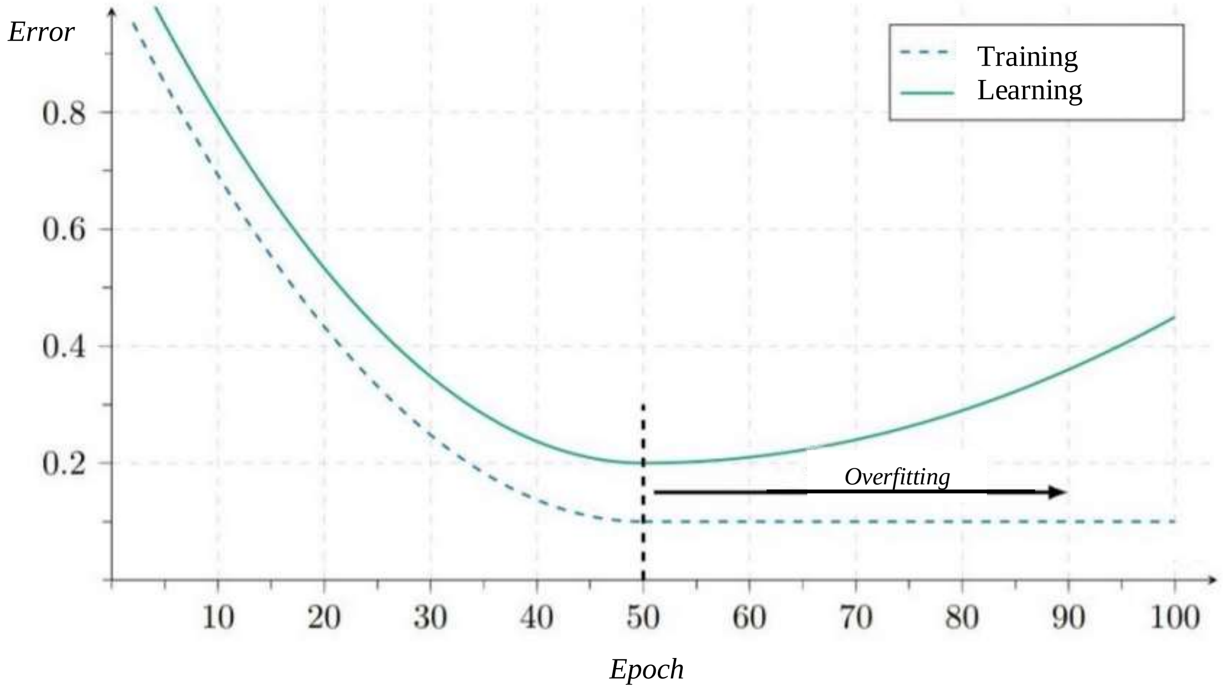


Fig. 4.3. The problem of network overfitting

Figure 4.3 elaborately demonstrates a typical situation of overfitting. The number of learning epochs act as a network hyper-parameter, and quality evaluation is represented with the error value. The longer network is trained, the more it gets used to the training set. At certain point, the network will perform successful recognition of only learning data and lose the ability to generalize. On this stage, qualitative curves over learning and training sets disperse. While the classifier continues to improve the quality index on the training set, it worsens the recognition of learning set. When the quality parameter does not improve after quite a number of epochs, it is relevant to change the value of initialized neuron weights in order to apply model normalization.

4.3.2. Loss function

Used in linear classifier, functions of attributing object to a certain class do not provide control over input data, as it is parametrized with the value of weight gaining in neuron W. Thus, it is necessary to set control over these weights. This should be done in a way that forecast class evaluation corresponded to the value of learning sets.

Practical implementation of classifiers demonstrated that the evaluation of the class, which really relates to the right object identification, is not always correct; thus, the solution is to compute not the positive result, but the negative one. This is where loss function or objective function comes in handy. Intuitively, loss function value will be big, if learning data classification ended with errors, and low, if everything works fine.

$$E_{CE}(W) = \underbrace{\sum_{x \in \mathrm{X}} \sum_{k=1}^{K} [t_k^x \log(o_k^x) + (1 - t_k^x) \log(1 - o_k^x)]}_{Cross-entropy} + \tag{4.1}$$

$$+ \underbrace{\lambda_1 \cdot \overbrace{\sum_{w \in W} |w|}^{L_1} + \lambda_2 \cdot \overbrace{\sum_{w \in W} w^2}^{L_2},}_{Network\ values\ of\ loss}$$

where W – weight, X – a set of learning data, $K \in N \geq 0$ is the number of classes, t_k^x indicates, whether x is an example of class k. Value o_k^x is the output value of classification algorithm, which depends on weights. Values $\lambda_1, \lambda_2 \in [0, \infty)$ is regularization weight – usually, this value is less than 0.1. Value of data loss is positive, when classification is wrong, but loss value is a bigger number for more complex models. If two network models recognize the same dataset, the better model is the one with a simpler architecture.

The reason to represent the loss function with a line chart instead of other parameters is that the chart will contain more information about the network quality. The main disadvantage of the loss function is that it does not have the top edge as the accuracy and it is hard to be interpreted. The loss function only relative learning process, while accuracy shows the absolute progress.

There are criteria of improvement in implementation of the loss function, which can help in the development of ANN. If losses during several epochs do not reduce, then the learning rate can be too low. The optimization process can be also fixed in the local minimum.

The wrong value of the loss function in a form of non-existing number can be related to very high learning rates. Another reason for this could zero division or logarithm of zero. In both cases, adding a small constant like 10^{-7} can solve the problem. If the loss function curve depending on the number of epochs reached its limit at the beginning, bad values of initialized weights can be the source of troubles.

4.3.3. Accuracy as the criteria of quality

Among ANN architectural models, which performance is dedicated to image recognition, there are several criteria to evaluate the quality of this performance. The majority of criteria are based on the so-called forecasting matrix (indicated as c_{ij}), which diagonal contains a number of correct prognoses. Let $t_i = \sum_{j=1}^{k} c_{ij}$ is the number of learning example for class i, then the most generalized criteria of quality is accuracy, represented in the following formula (4.2).

$$accuracy(c) = \frac{\sum_{j=1}^{k} c_{ii}}{\sum_{j=1}^{k} t_i} \in [0,1]. \tag{4.2}$$

One of the accuracy problems as quality criteria is the deviation in class evaluation. If one class is more generalized than others are, the simplest way to evaluate any other class with a high rank will be its constant classification as generalized. To solve this problem, one can use the averaged accuracy value, and other parameters, such as the following ones:

- the speed of evaluation and analysis of new images given to the network;
- learning time delay;
- demand of resources;
- persistence to (non)random deviations in learning data;
- dimension of network architecture.

In practice, apart from such criteria as classification accuracy, the mentioned criteria are significant as well. It is noticeable, that giving accuracy as a floating-point number allows process more data on the same device.

4.4. Overview of deep learning libraries

Today, the important role in the implementation of ANN belongs to specialized packets, which provide fully implemented basic structures and algorithms for convenient recognition. The most widespread tools are libraries, such as Theano, TensorFlow, Torch and others. The given packets implement software interface for low-level computations, thus, decision-making should be based on contradictions about productivity and design convenience.

It is worth noting that Torch implements the interface for Lua programming language, not so popular among developers, which reduces convenience and speed. According to researches, Theano and Torch demonstrate almost the same productivity over the same datasets.

Apart from aforementioned inconvenience of using Torch library, there is another problem in this realm. Another two tolls are able to work with a high-level Keras library, which provides general, beforehand-defined algorithms and structures for TensorFlow and Theano.

Keras library is the software product written in Python, which provides high-level API to work with ANN. The product was created in order to prevent difficulties in developing ANN, related to syntax peculiarities of concrete packets. In 2016, this tool was purchased by Google and adapted as the main high-level configuration for TensorFlow library. According to the compared results and analysis of library documentation, it is suitable to use TensorFlow and Keras for CNN implementation (Table 4.1).

Table 4.1

The comparison of Deep Learning libraries

Name of library	Theano	Caffe	Torch	TensorFlow
Programming language	Python	Python, C++	Lua	Python
Specialization in ML methods	Diverse regression types, ANN	ANN	–	Diverse regression types, ANN
DL feature	+	+	+	+
Working speed	Medium	High	Low	High
Quality of paralleling among several devices	–	–	–	+
Additional features	–	+ (by means of additional Python libraries)	+ (by means of additional Python libraries)	+

Theano library was primarily developed as the extension for Python language, which allows efficient computation of mathematical expressions with multi-dimensional arrays. The basic set of tools for ANN building is implemented in this library. The process of model creation and definition of its parameters requires writing a long code, which includes implementing model class, self-determining its parameters, implementing method that defined error function, gradient computing rule, and ways of changing of neuron weights.

Caffe library is implemented with C++ programming language. ANN topology, output data and learning method are defined by means of configuring protocols (data transmission technology and protocol is applied) in the form of a prototype. Caffe library is supported by a large community of developers and users, and today it is the most widespread library of deep learning programs with a wide range of application.

Python is a flexible programming language of high-level, created in 1991, which is aimed for low "input level" and easy readability of resulting programs. Python language has several versions. Nowadays, versions 2.7 and 3.5 are mostly used.

TensorFlow is the library of software maintenance with open-sources code for machine learning tasks, developed by Google. It allows creating and training ANN of different architectures, which find their implementation in pattern detection and recognition, and search of interconnections. TensorFlow also contains TensorBoard, which represents visualizing tools in browser to evaluate learning efficiency and network model parameters. TensorFlow reaches its productivity due to parallel data processing between central processing unit (CPU) and graphics processing unit (GPU) or a bunch of them, and even horizontal scaling by means of gRPC (remote procedure call). TensorFlow supports Python and C++ programming languages.

Each computation in TensorFlow is represented in a form of dataflow chart, computation chart. Computation chart is the model, which describes the method of computing. It is important to note that computation chart composition and operation in the given structure are two different processes. A chart consists of placeholders (tf.Placeholder), variables (tf.Variable) and operations. It is a place for computing tensors – multi-dimensional arrays, which can be either number or vector.

Charts are conducted in sessions (tf.Session). There are two types of sessions: casual and interactive (tf.InteractiveSession); the interactive session is suitable to be conducted in the terminal. A session preserves the state of variables and queues. The obvious composition of sessions and charts guarantees appropriate memory resources release. In the chart, each top has zero or more input and zero or more outputs, representing operation implementation. Tensors are chart edges, particularly the arrays of arbitrary size (the array type is defined during chart composition). Special tops, that control dependencies, can also be in the chart: they indicate that output node of control dependency can finish its work before the node of control dependency receiver starts working.

Each operation has its name and represents an abstract computation (for instance, the summing operation). Operations can have attributes. The core is the specific operation implementation, which can be executed in a certain type of devices (CPU or GPU). Variable is a special type of operation, which returns counter to a constantly altering tensor: such a variable does not disappear after single chart usage. To similar tensors, counters are transmitted with numerical operation, which then alternate the given tensor. In machine learning tasks, model parameters usually keep tensors in variables, which are refreshed at each learning stage.

4.5. Stage of preliminary dataset preparation

For network work evaluation according to certain criteria, one uses datasets. In the current case, datasets in a form of archive of classified images will be used (Appendix A):

1. CIFAR-10 is the archive of 10-class set of color images of 32 × 32 pixel size. All images are distinguished in the following classes: aircraft, vehicle, bird, cat, deer, dog, frog, horse, ship, truck. The value of recognition accuracy is 96.54%.

2. CIFAR-100 is the set of 100 classes of color images of 32 × 32 pixel size. These classes are grouped in 20 super-classes. Groups include animals, humans, plants, transport and other objects. CIFAR-100 is not the super-set of CIFAR-10, since CIFAR-100 does not include the “aircraft” class. The value of recognition accuracy is 82.82%.

3. German Traffic Sign Recognition Benchmark (GTSRB) is the dataset of 43 traffic signs. 51 839 images are colored, sized from 25 × 25 to 266 × 232 pixels. The value of recognition accuracy is 99.46%.

4. Handwritten Symbols version 2 (HASYv2) is the set of 369 classes of black-and-white images of 32 × 32 pixel size. Classes contain Latin and Greek letters, arrows and mathematical symbols. The value of recognition accuracy is 82%.

5. Self-taught learning 10 (STL-10) is the 10-class set of colored images of 96 × 96 pixel size. It represents 10 classes: aircraft, bird, vehicle, cat, deer, dog, horse, monkey, ship, truck. The value of recognition accuracy is 74.80%.

6. Street View House Numbers (SVHN) dataset exists in two forms. The following experiments were based on volumetric digit format. It contains 10 digits, cut out from Google Street View photos. All images are colored, of 32 × 32 pixel size. The value of recognition is 98.41%.

7. Mixed National Institute of Standards and Technology (MNIST) dataset contains handwritten digits – the learning set contains 60 000 examples and the testing set consists of 10 000 examples. It is a subset of a bigger set, available in NIST. Digits are normalized by size and centered in images of fixed size. The set is a great example to try learning and recognition over real data, applying minimal efforts for preliminary processing and image formatting. The value of recognition accuracy is 98.7%.

As the stage of preliminary processing, all images from these datasets have to be changed before transmitting to CNN input. Despite a small size of images, relation of sides is kept original to avoid distortion.

Pixel values for red, blue and greed channels, located in the 0-255 range were normalized by dividing each value on 255. The resulting set of coefficients will receive values from 0 to 1.

The initial value format was integer, changed to float-pointing numbers for the best conduction of the normalization. In order to solve classification problems, pixel values are represented as vectors, correspondent to each of 10 classes, which are transformed into a binary matrix. To achieve better object recognition for different image location, the dataset was added with modified input images.

For CIFAR-10, CIFAR-100, STL-10, MNIST and HASYv2 modification is understood as horizontal image reflection, vertical image reflection, height and width changes. For GTSRB dataset images were scaled to 32 × 32 pixel size. Only SVHN does not require any changes.

4.6. Hierarchical model of the classifier

Configuring classifier’s working principles for separate datasets provides certain difficulty for developers. Many examples, illustrating classifier structures, are not the best example among quality parameters.

Evaluation of each example can last rather long time, since CNN learning can take days or even weeks. Besides, some analyzing methods of input dataset become worse to be used while increasing class number and adding learning samples. The proposed solution for this problem is to build the hierarchy of classifiers.

The main classifier (or root classifier) distinguishes class clusters (subclasses), which distinguish singular classes. The main classifier has to distinguish 6 obvious classes (transport, road signs, digits and others) or 17 hidden classes.

If the main classifier presumes a vehicle, another classifier must predict, whether it is two-wheeled (bicycle) or four-wheeled (automobile). However, if the main classifier recognizes road, another one would not activate. The hierarchal approach distinguishes 7 class clusters and therefore uses 8 classifiers. Such a hierarchical model requires formed class clusters.

There are two ways for class clustering: by similarity or semantics. If the semantic clustering requires either additional data or processing by hand, then classification by similarity can automatically be conducted from data. Sometimes, semantically similar classes can be visually similar.

For example, in the ImageNet dataset the majority of dogs semantically and visually resemble each other. An example of unobvious classification can be symbols: the sum symbol visually resembles a Greek letter, but semantically is much close to the adding operator.

One of the approaches to clustering of classes by similarity is classifier learning or examining the values of its forecasting. Each class is represented in the matrix of forecasting with one row. These rows can be represented by means of standard clustering algorithms, such as k-values.

The matrix of classifier forecasting $c_{ij} \in N^{n\,x\,k}$ is built on how often i-class was classified and j-class was forecast. On this forecasting matrix, classes can be clustered with the algorithm, working by the following principles:

1. Column and row sequences in the forecasting matrix is arbitrary. It means rows and columns can be relocated. If i- and j-rows are relocated, then i- and j-columns must be also relocated, in order to perceive the structure.

2. If two class are often forecast, they are similar to the classifier.

Classifier training was conducted over 100 classes of CIFAR-100 dataset. Each class has 100 examples to be tested. The accuracy of class attribution to clusters recognition is demonstrated in Table 4.2.

The fact that the main classifier achieves better results within the cluster, than specialized classifier in 13 events out of 14, can relate to limited data characteristics, overfitting or small image size. The experiment also shows that the majority of error is connected with undefined class (Appendix D).

Table 4.2

The accuracy of the main classifier while training on the full set of 100 classes with 14-cluster division

Cluster	Classes	Classifier’s accuracy of attributing a class to a cluster	
		Main classifier, %	Specialized classifier, %
1	3	82.26	72.98
2	5	57.56	42.45
3	2	90.15	81.65
4	2	86.39	84.72
5	3	78.3	72.21
6	2	77.63	73.28
7	2	90.14	88.51
8	2	86.22	82.69
9	2	89.81	86.99
10	2	85.59	73.13
11	2	86.09	74.42
12	2	93.69	77.37
13	2	84.59	85.36
14	2	89.91	88.50

Therefore, in this case, it is necessary to do something more than improving densifying classes in the cluster. Though classes inside the cluster identify the majority of classifications, there are loads of wrongly defined classifications outside the clusters.

4.7. The choice of CNN architecture

There are several options to compute the number of parameters for each network layer, particularly, the number of parameters depending on configurations of feature map size. If n_i (i = 0, …, n) equals the number of output feature maps of i-layer, where at i = 0 it is the input layers, and all filters have a size of 3 × 3, then the computation of parameters quantity for the following layers looks like this (4.3).

$$number\ of\ parameters = \sum_{i=1}^{k} \left((n_{i-1} \cdot\ 3^2 + 1) \cdot n_i \right). \tag{4.3}$$

A fully connected layer with n-nodes and k-inputs has $n \cdot (k + 1)$ parameters, where "1" is the bias neuron. The convolution i-layer with k_i filters of n × m size will be used for k_{i-1} feature map and have $k_i \cdot k_{i-1} (n \cdot m + 1)$ parameters, where "1" is the bias neuron. A fully connected layer with n-nodes after applied k-maps of features of size $m_1 \times m_2$ will have $n \cdot (k \cdot m_1 \cdot m_2 + 1)$ parameters. A dense block with L-depth, n-speed of growth and feature maps of 3 × 3 size will have $L + n\ \cdot\ 3^2 +\ 3^2 \cdot\ n^2\ \sum_{i=0}^{L} L - i = L + 9n + 9n^2\, \frac{L^2 - L}{2}$ parameters.

The basic architecture of CNN model can be described according to the following template: Convolutional block (n) = (Convolution – Batch-normalization – Activation function)^2 – Subsampling. Activation function ReLU is used in all convolutional layers, except for the last fully connected layer, where Softmax function is used. Before the latest convolutional layer, exception layers are applied, where the probability of the dropout equals 0.5. The detailed architecture of the basic model is represented in Table 4.3. A graphic representation of the basic CNN model is demonstrated in Appendix C.

Table 4.3

The architecture of the basic model for CNN

№	Type of layer	Channels / filter / stride	Parameters	Output size
	Input data		0	3 / 32 × 32
1	Convolutional	32 / 3 × 3 × 3 / 1	896	32 / 32 × 32
2	BN + ReLU		64	32 / 32 × 32
3	Convolutional	32 / 3 × 3 × 32 / 1	9248	32 / 32 × 32
4	BN + ReLU		64	32 / 32 × 32
	Subsample (max)	2 × 2 / 2	0	32 / 16 × 16
5	Convolutional	64 / 3 × 3 × 32 / 1	18 496	64 / 16 × 16
6	BN + ReLU		128	64 / 16 × 16
7	Convolutional	64 / 3 × 3 × 64 / 1	36 928	64 / 16 × 16
8	BN + ReLU		128	64 / 16 × 16
	Subsample (max)	2 × 2 / 2	0	64 / 8 × 8
9	Convolutional	64 / 3 × 3 × 64 / 1	36 938	64 / 8 × 8
10	BN + ReLU		128	64 / 8 × 8
	Subsample (max)	2 × 2 / 2	0	64 / 4 × 4
11	Convolutional (v)	512 / 4 × 4 × 64 / 1	**524 800**	512 / 1 × 1
12	BN + ReLU		1024	512 / 1 × 1
	Dropout 0.5		0	512 / 1 × 1
13	Convolutional	512 / 1 × 1 × 512 / 1	262 656	512 / 1 × 1
14	BN + ReLU		1024	512 / 1 × 1
	Dropout 0.5		0	512 / 1 × 1
15	Convolutional	k / 1 × 1 × 512 / 1	k * (521 + 1)	k / 1 × 1
	Subsample (avg)	1 × 1	0	k / 1 × 1
16	BN + Softmax		2k	k / 1 × 1

The basic network architecture receives data from three channels of 32 × 32 size. All convolution layers use such network parameter as zero extension, apart from convolution layer 11, since the task here is to reduce the size of feature map to 1 × 1. If the input feature map exceed 32 × 32, the capacity is divided between two block of convolution layers (normalization and activation function), and added with one layer of maximal subsampling. From Table 4.3 it is clear that the first convolution layer of the basic model has 896 parameters. Assuming it is undesired to use less than three filters for each layer, all layers use only filters of 3 × 3 size, and then the maximal depth will equal 1. Furthermore, if more than 800 parameters are about to be used, there are another 120 combinations of possible layers. The training of the basic model was conducted configuring the following parameters:

- learning algorithm – error backpropagation method (Appendix B);
- learning rate: 10^{-4};
- batch size – 64;
- number of epochs – 1000.

If the dataset does not define relation between learning and testing sets, 67% to 37% ratio is used. If the dataset does not include the checking set, the relation is 90% to 10%. Configuring the number of epochs (cycles) by means of a procedure, known as early stopping – the learning process is merely stopped, if over the given number of epochs (patience) the loss is not reduced. Since the set is relatively small and quickly saturates, the patience can be equal to 10, with the maximal number of epochs as 1000.

All experiments are conducted with the help of Keras 2.0 and TensorFlow 1.0 libraries. Experiments are conducted on GPU GeForce 940MX, the results of which are given in Table 4.4. According to the results, the percentage of accuracy for the datasets do not exceed results, mentioned in other architectures, therefore, it is worth thinking about optimizing the hierarchical classifier and the very architecture to improve the results. The programming code of the CNN is demonstrated in Appendix E.

Table 4.4

The parameter of recognition accuracy of the basic model on different datasets

№	Dataset	Network learning		Network testing	
		Accuracy, %	Error, σ	Accuracy, %	Error, σ
1	CIFAR-10	91.25	1.10	85.90	0.87
2	CIFAR-100	76.64	1.41	82.82	0.55
3	GTSRB	100	0.01	99.46	0.11
4	HASYv2	88.48	0.41	81	0.10
5	MNIST	99.93	0.07	98.70	0.06
6	STL-10	94.12	0.87	74.80	0.34
7	SVHN	99.02	0.07	98.41	0.10

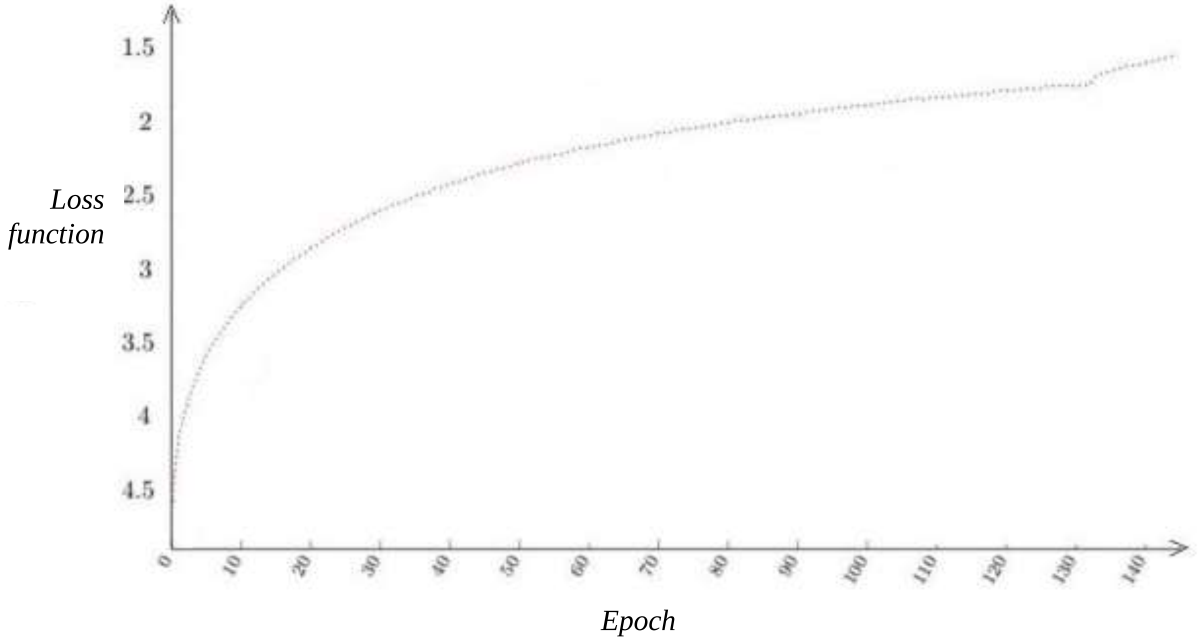


Fig. 4.4. The curve of loss function values over epochs

Owing to the procedure of early stopping, there is a difference in epoch number while network learning. For example, on the dataset, where data was altered or extended, the number of epoch ranges from 133 to 182, with standard deviation of 17.3 epoch with CIFAR-100. Figure 4.4 demonstrates the chart with changes in loss functions values by epochs on the example of CIFAR-100 (Appendix D).

4.8. Methods of improving the accuracy

Bias neuron or just bias is the third type of neurons among already mentioned, used in the majority of ANN. The peculiarity of this neuron is that its input and output always equal "1", and they never have input synapses. Bias neurons can be either present for each layer or absent, or be in half of layers in the ANN. Bias neuron has the same connections as usual neurons – will neurons of the next level, except that there can be no synapse between two bias neurons. Correspondently, they can be located on the input layer of all hidden layers, but not on the output layer, since there will be nothing to form a connection.

Bias neuron is necessary to receive the output result by means of shifting the chart of activation function to the left or right. When weights of hidden and output neurons are configured while learning, the activation function angle on the chart also changes. However, configuring bias neuron weights can give the opportunity to change the activation function on axis X and obtain new areas. In other words, if the point responsible for solution will be located to the left of the chart, ANN will never solve the problem without using bias neurons. Therefore, ANN without bias neurons are rare to be met.

Bias neurons also help in the case, when all input neurons are 0 at the input despite their weight and all of them are transmitted to the next layer. Presence or absence of bias neuron is another hyper-parameter of ANN.

According to the results of basic model's architecture of CNN working, it can be stated, that the system can be optimized to improve the results. The ways to so are as follows:

- removing bias neurons from the last network layers: for all layers, which output feature maps in 1 × 1 size, biasing is removed;
- increasing dimension of maximal subsampling layer up to 3 × 3;
- increasing the number of filters for the input layer of the CNN.

Taking into account all the proposed changes to the basic architecture, one can receive a modified new architecture of CNN. In theory, it is able to provide far more better results of recognition for the same datasets.

Conclusions on Part 4

In classification or recognition tasks, the great advantage of ANN over specialized algorithms is their ability to learn, therefore, Part 4 depicts the basic ideas and implementation methods of this complex process. In order to improve or boost up the speed of learning, and avoid the overfitting, it is convenient to apply mathematical methods and tools, such as error backpropagation algorithm or normalization procedures.

To evaluate the quality of ANN, the following parameters come in handy: accuracy and quality (calculated by the loss function), which can be analyzed in terms of received results in a form of charts or tables. In Part 4 the architecture of CNN with hierarchical classifier (basic model) was proposed, with learning conducted and accuracy index calculated for each given dataset by means of modern deep learning libraries. According to the results, it is worth noting that the basic model of CNN is prone to improvement in order to achieve higher accuracy values than the current ones.

CONCLUSIONS

The primary goal of the Graduation Project – “Graphic Object Recognition System” – is the design of the convolutional neural network, built to solve the tasks of pattern recognition and classification. The topicality of the graduation project is the necessity to analyze the existing methods and techniques of graphic object recognition, choose the most suitable for the given task and apply it with recommendation of further improvement. During the graduation project preparation, a great volume of information was collected and analyzed.

Part 1 of the graduation project contains the detailed description of the subject field, the overview of existing methods for pattern recognition and the introduction to artificial neural networks (ANN) as a tool of graphic pattern recognition. In addition, the concept of computer vision is introduced.

Computer vision researchers developed mathematical models to create a three-dimensional form or objects on the image. Nowadays there are reliable methods of accurate computation of a partial 3D-model in the environment with thousands of points. It is possible to track someone moving against certain background. With alternating success, machines are able to find and name every person on the photo, using facial shapes, clothes and hair. However, despite all these achievements, the dream of computer being able to interpret objects as human or animal remains unreachable. Often recognition comes hard, since vision is a reverse problem, in which the task is to restore some unknown data with partial data quantity in order to generate a consistent decision.

Computer vision tries to describe the world, which humans see as one or several images, and features, such as sizes, lighting, and color. Vision algorithms are so prone to mistakes that even the slightest changes in pixels sometimes can lead to very different results. Thus, computer vision is a set of methods and algorithms to interact with the visual environment, the goal of which is graphic object recognition and analysis.

Part 2 of the graduation project describes the problem of classifying graphic patterns. The approach to the problem of pattern recognition can be based on analogues with biological processes. In certain situations, the animal capability of pattern recognition can greatly exceed the capacity of any machine ever built by humans.

Dealing with classification based on direct sensory experience, i.e. recognizing individuals or pronounced words, people easily exceed technical devices. In “non-sensory situations”, human actions are not so effective. For example, people cannot compete with pattern classification programs, if the right way of classification includes logic combinations of abstract features, such as colors, sizes and forms. Since pattern recognition has to be neurons’ function, it is possible to look for “the golden key” to biological pattern recognition in neurons’ features.

Among noticeable features of ANN, the most important one is the ability to learn diverse methods and techniques. The majority of learning methods are suitable for basic configuring and possess similar characteristics. ANN learning resembles the process of human learning, though ANN capabilities are quite limited. A network is learning in order to provide a needed set of output for each set of inputs, and each of them is regarded as a vector. Learning is conducted by means of a successive demonstration of input vector with simultaneous configuration of weights according to a certain procedure. In the end network’s weights gradually obtain values so that any input vector resulted into an output vector. Learning algorithms can be classified as algorithms of supervised or unsupervised training.

Part 3 of the graduation project contains the description of the structure of convolutional neural networks (CNN). The architecture of CNN is designed specifically for efficient pattern recognition. In its core, there lies the work of convolutional layers called convolutions with non-linear activation functions like rectified linear unit (ReLU) or hyperbolic tangent *tanh* and connecting layers, called pooling layers. On the contrary to feedforward neural networks (FNN), where each input neuron is connected with the output neuron of the next layer, CNN use convolutions over each input layer in order to receive output values.

The convolution operation requires the matrix of weights of small size running through the current layer and after each shift forming the activation signal for next layer neuron in analogical position. This matrix is known as the convolutional core; it is applied for different neurons of the output layer.

Part 4 contains the design of CNN to solve graphic object recognition tasks. The main idea of improving the method of classification in the convolutional neural network is the usage of hierarchical classifier, which becomes the universal tool of computing the accuracy, applied on many input datasets. The core of hierarchy in classifiers lies in the clustering of the given classes.

The significant stage of improving the process of classification is the choice of the basic model of convolutional neural network and calculation its accuracy parameters, depending on the size of learning stride and applied algorithm. This allows the further understanding of what exact architectural solutions can be introduced in order to optimize the model and improve the performance. In terms of software implementation, the model of convolutional neural network was realized by means of Keras library written in Python programming language. TensorFlow library was used to conduct complex mathematical calculations. The chosen basic model was trained with seven input datasets.

The result of the work appears to be the developed basic model architecture of the convolutional neural network for the task of graphic pattern recognition. The model can be simultaneously applied to seven different datasets owing to such a universal instrument as the hierarchical classifier. The developed convolutional neural network successfully performed the given tasks, providing quite high accuracy results for each proposed dataset, containing different number and types of images.

However, it became clear that the current convolutional neural network could be further modernized and improved in order to provide better performance. Therefore, the current model could be developed and advanced by the following methods: removing bias neurons from the last network layers, increasing dimension of maximal subsampling layer, increasing the number of filters for the input layer of the convolutional neural network. According to the learned theoretical information, such alterations can successfully empower performance of the current model.

REFERENCES

1. ДСТУ 3008:15. Інформація та документація. ЗВІТИ У СФЕРІ НАУКИ І ТЕХНІКИ Структура та правила оформлювання.

2. ДСТУ 3321:2003. Система конструкторської документації. Терміни та визначення основних понять.

3. ДСТУ ГОСТ 2.001:2006. Єдина система конструкторської документації. Загальні положення (ГОСТ 2.001-93, IDT).

4. ДСТУ ГОСТ 2.001:2006. Єдина система конструкторської документації. Основні написи (ГОСТ 2.104-2006, IDT).

5. ДСТУ 3582-97. Інформація та документація. Скорочення слів в українській мові в бібліографічному описі. Загальні вимоги та правила.

6. ДСТУ ГОСТ 7.1:2006. «Бібліографічний запис, бібліографічний опис. Загальні вимоги та правила складання».

7. ДСТУ ISO 5457:2006. Документація технічна на вироби. Кресленики. Розміри та формати.

8. Zhukov I.A., Kudrenko S.O., Fomina N.B. Graduation Project Guidelines / Compilers. K.: NAU; 2019.

9. Chen C. H. Handbook of pattern recognition and computer vision / C. H. Chen, L. F. Rau // Singapore-New Jersey-London-Hong Kong: World Scientific Publishing Co. Pte. Ltd., 1995. – 256-258.

10. Duda R.O., Hart P.E. Pattern Classification and Scene Analysis / John Wiley & Sons, Inc.; 1973.

11. Duda R.O., Hart P.E., Storck D.G. Pattern Classification / 2nd ed., Wiley; 2001.

12. Bishop C.M. Pattern Recognition and Machine Learning / Springer Verlag, Singapore, 2006.

13. Bishop C.M. Neural Networks for Pattern Recognition / Clarendon Press, Oxford, 1996.

14. Schalkoff R. Pattern Recognition. Statistical, Structural, and Neural Approaches / John Wiley & Sons, Inc., 1992.

15. Oja E. Subspace Methods of Pattern Recognition / John Wiley & Sons, Inc., 1983.

16. Hyvärinen A., Karhunen J., Oja E. Independent Component Analysis / Wiley, 2001.

17. Sklansky J. and Wassel G.N. Pattern Classifiers and Trainable Machines / Springer Verlag, New York, 1981.

18. Niemann H. Klassifikation von Mustern / Springer Verlag, Berlin, 1983.

19. Devijver P. A., Kittler J. Pattern Recognition: A Statistical Approach / Prentice-Hall International, Englewood Cliffs, NJ, 1980.

20. Fukunaga K., Introduction to Statistical Pattern Recognition. 2nd Ed. / Academic Press, New York, 1990.

21. Hand D. J. Discrimination and Classification / John Wiley and Sons, Chichester, UK, 1981.

22. Haykin S. Neural Networks / MacMillan, NY, 1993.

23. Gose E., Johnsonbaugh R., Jost S. Pattern Recognition and Image Analysis / Prentice-Hall, Upper Saddle River, NJ, 1996.

24. Ripley B. Pattern Recognition and Neural Networks / Cambridge University Press, Cambridge, 1996.

25. Schuermann J. Pattern classification, a unified view of statistical and neural approaches / John Wiley & Sons, New York, 1996.

26. McLachlan G.J. Discriminant Analysis and Statistical Pattern Recognition / John Wiley and Sons, New York, 1992.

27. Therrien C.W. Decision, Estimation, and Classification: An Introduction to Pattern Recognition and Related Topics / Wiley, New York, 1989.

28. Clowes, M. B. On seeing things / Artificial Intelligence. 1971; 2:79 – 116.

29. Connell J. H., Brady M. Generating and generalizing models of visual objects / Artificial Intelligence. 1987; 3:159–183.

30. Guzman, A. Decomposition of a visual scene into three-dimensional bodies / AFIPS Proceedings of Fall Joint Computer Conference. 1968; 33:291 – 304.

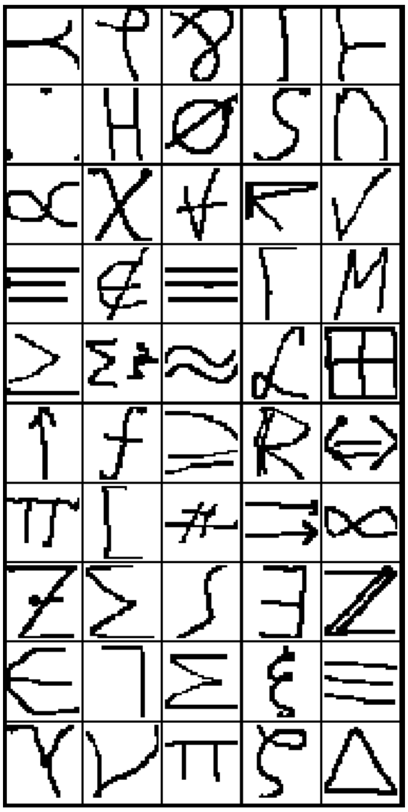
31. Huffman, D. A., Impossible objects as nonsense sentences / 6. Edinburgh University Press, 1971:295–324.

Appendix A

The examples of images from datasets used in CNN training

CIFAR-10 / CIFAR-100

GTSRB



HASYv2

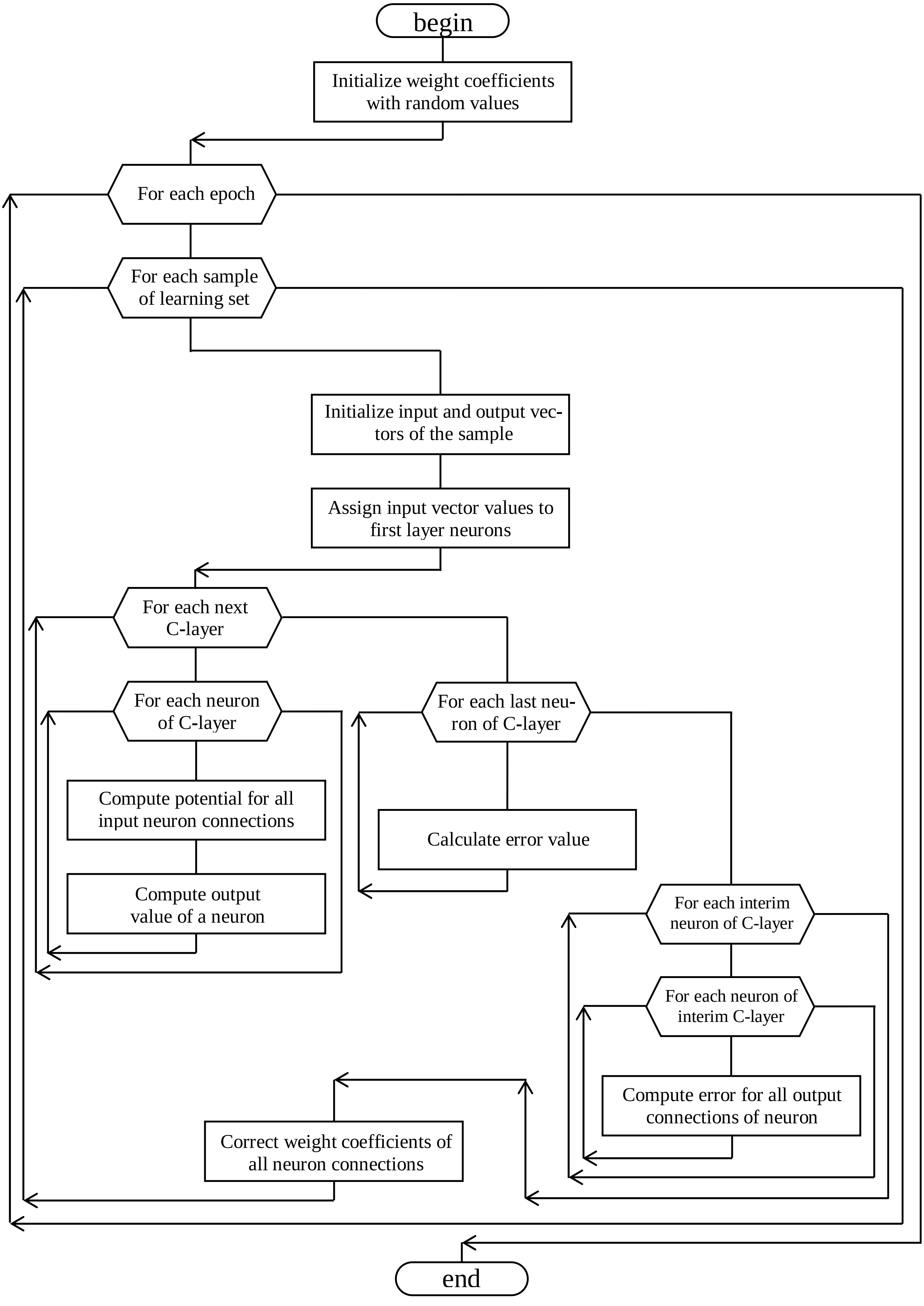
STL-10

SVHN

MNIST

Appendix B

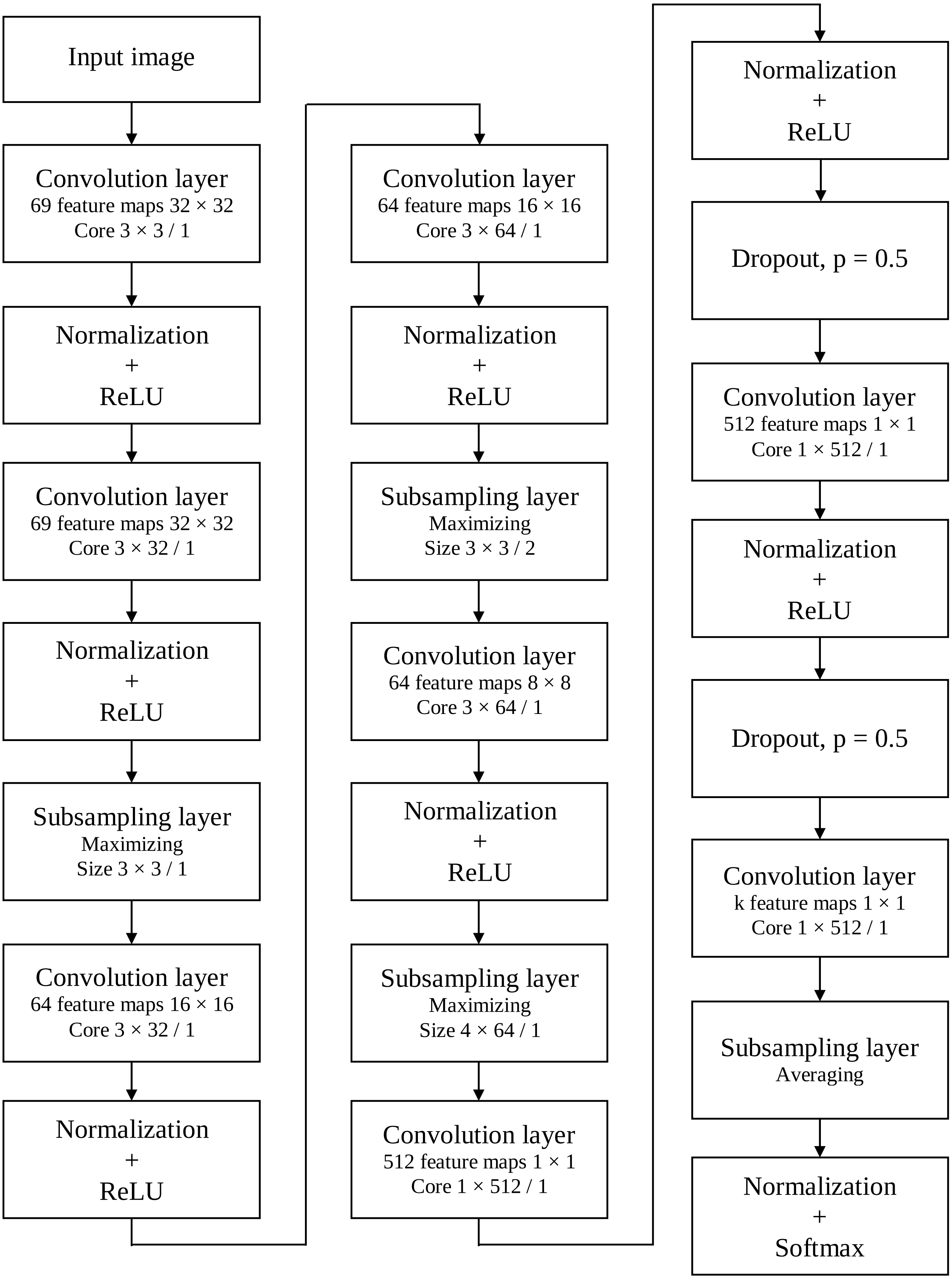
The flowchart of learning method by means of error backpropagation



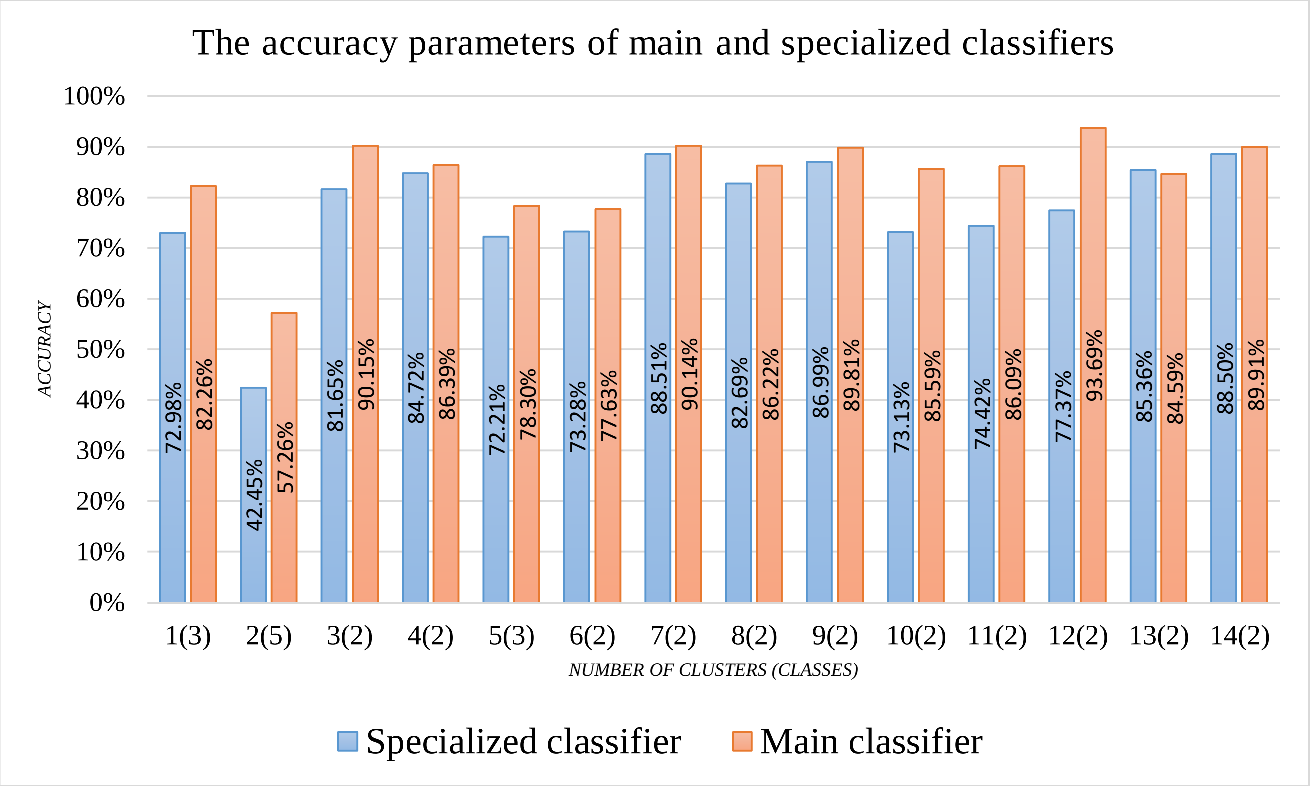
begin
Initialize weight coefficients with random values
For each epoch
For each sample of learning set
Initialize input and output vectors of the sample
Assign input vector values to first layer neurons
For each next C-layer
For each neuron of C-layer
Compute potential for all input neuron connections
Compute output value of a neuron
For each last neuron of C-layer
Calculate error value
For each interim neuron of C-layer
For each neuron of interim C-layer
Compute error for all output connections of neuron
Correct weight coefficients of all neuron connections
end

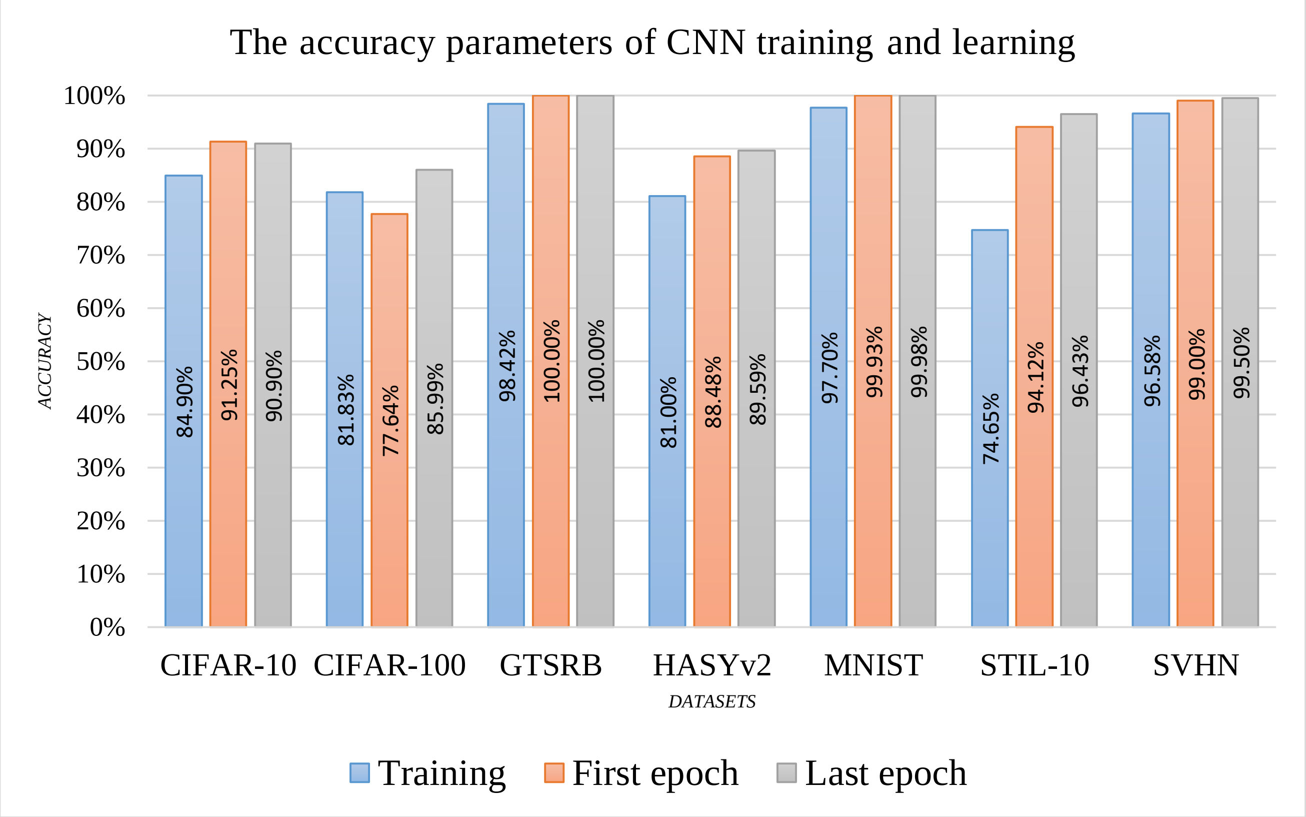
Appendix C

The structural scheme of CNN basic model architecture



Appendix D


The accuracy parameters of main and specialized classifiers
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
ACCURACY
72.98%
82.26%
42.45%
57.26%
81.65%
90.15%
84.72%
86.39%
72.21%
78.30%
73.28%
77.63%
88.51%
90.14%
82.69%
86.22%
86.99%
89.81%
73.13%
85.59%
74.42%
86.09%
77.37%
93.69%
85.36%
84.59%
88.50%
89.91%
1(3)
2(5)
3(2)
4(2)
5(3)
6(2)
7(2)
8(2)
9(2)
10(2)
11(2)
12(2)
13(2)
14(2)
NUMBER OF CLUSTERS (CLASSES)
Specialized classifier
Main classifier


The accuracy parameters of CNN training and learning
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
ACCURACY
84.90%
91.25%
90.90%
81.83%
77.64%
85.99%
98.42%
100.00%
100.00%
81.00%
88.48%
89.59%
97.70%
99.93%
99.98%
74.65%
94.12%
96.43%
96.58%
99.00%
99.50%
CIFAR-10
CIFAR-100
GTSRB
HASYv2
MNIST
STIL-10
SVHN
DATASETS
Training
First epoch
Last epoch

Appendix E

The programming code for CNN installation and configuration

File: analize_model.py

```
import logging
import sys
import keras.layers.convolutional
import keras.layers.normalization
from keras import backend as K
from keras.models import load_model
import seaborn as sns
sns.set_style("whitegrid")
import matplotlib.pyplot as plt
import operator
from functools import reduce
import numpy as np
import yaml
import imp
from run_training import make_paths_absolute
import os
import pprint
import scipy.misc
import scipy.stats
import glob
logging.basicConfig(format='%(asctime)s %(levelname)s %(message)s',
                    level=logging.DEBUG,
                    stream=sys.stdout)
from msthesis_utils import make_mosaic
def get_activations(model, layer_index, X_batch):
    get_activations = K.function([model.layers[0].input, K.learning_phase()],
    [model.layers[layer_index].output, ])
    activations = get_activations([X_batch, 0])
    return activations
def show_conv_act_distrib(model, X, show_feature_maps=False):
    X_train = np.array([X])
    layer_index = 0
    activations_by_layer = []
    labels = []
    for layer_index in range(len(model.layers)):
        act = get_activations(model, layer_index, X_train)[0]
        if show_feature_maps:
            scipy.misc.imshow(X_train[0])
            mosaik = make_mosaic(act[0], 8, 4)
            scipy.misc.imshow(mosaik)
         data = act[0].flatten()
         if isinstance(model.layers[layer_index],
                keras.layers.convolutional.Conv2D):
            print("\tlayer {}: len(data)={}".format(layer_index, len(data)))
            activations_by_layer.append(data)
            labels.append(layer_index)
        layer_index += 1

    #Activations
    for label, fw in enumerate(activations_by_layer):
        print("99% filter weight interval of layer {}: [{:.2f}, {:.2f}]"
            .format(label, np.percentile(fw, 0.5), np.percentile(fw, 99.5)))
    f, ax1 = plt.subplots(1, 1)
    p = sns.violinplot(data=activations_by_layer, orient="v",
                palette=sns.color_palette(palette="RdBu", n_colors=1),
                ax=ax1)
    p.tick_params(labelsize=16)
    ax1.set_xticklabels(labels)
    ax1.set_title('Convolution activations by layer')
    sns.plt.show()
def show_conv_weight_dist(model, small_thres=10**-6):

    layer_index = 0
    filter_weights = []
    bias_weights = []
    filter_weight_ranges = []
    labels = []
    for layer in model.layers:
        if not isinstance(layer, keras.layers.convolutional.Conv2D):
            layer_index += 1
            continue
        weights = layer.get_weights()

        print("{}: {} filter weights in {}th layer"
        .format(weights[0].shape,
            reduce(operator.mul, weights[0].shape, 1),
            layer_index))

        #Filter
        ranges = []
        w, h, range_i, range_j = weights[0].shape
        if w > 1 or h > 1:
            for i in range(range_i): # one filter, but different channel
                for j in range(range_j): # different filters
                    elements = weights[0][:, :, i, j].flatten()
                    ranges.append(elements.max() - elements.min())
            ranges = np.array(ranges)
            filter_weight_ranges.append(ranges)

        data = weights[0].flatten()
        labels.append(layer_index)
        filter_weights.append(data)
        data_small = np.array([el for el in data if abs(el) < small_thres])
        print("< {}: {}".format(small_thres, len(data_small)))
        #Bias
        if len(weights) > 1:
            print("{}: {} bias weights in {}th layer"
                .format(weights[1].shape,
                    reduce(operator.mul, weights[1].shape, 1),
                    layer_index))
            data = weights[1].flatten()
            bias_weights.append(data)
            data_small = np.array([el for el in data if abs(el) < small_thres])
            print("< {}: {}".format(small_thres, len(data_small)))
        else:
            print("No bias in layer {}".format(layer_index))
        layer_index += 1
    labels = [1, 3, 5, 7, 9, 11, 13, 15] # baseline

    #Filter weight ranges
    f, ax1 = plt.subplots(1, 1)
    p = sns.violinplot(data=filter_weight_ranges, orient="v",
                palette=sns.color_palette(palette="RdBu", n_colors=1),
                ax=ax1)
    p.tick_params(labelsize=16)
    p.set_xlabel('Layer', fontsize=20)
    p.set_ylabel('Weight range', fontsize=20)
    ax1.set_xticklabels(labels)
    ax1.set_title('Filter weight ranges by layer')
    sns.plt.show()

    #Filter weights
    for label, fw in enumerate(filter_weights):
        print("99% filter weight interval of layer {}: [{:.2f}, {:.2f}]"
            .format(label, np.percentile(fw, 0.5), np.percentile(fw, 99.5)))
    f, ax1 = plt.subplots(1, 1)
    p = sns.violinplot(data=filter_weights, orient="v",
            palette=sns.color_palette(palette="RdBu", n_colors=1),
            ax=ax1)
    p.tick_params(labelsize=16)
    p.set_xlabel('Layer', fontsize=20)
    ax1.set_xticklabels(labels)
    ax1.set_title('Filter weight distribution by layer')
    sns.plt.show()

    #Bias weights
    for label, fw in enumerate(bias_weights):
        print("99% bias weight interval of layer {}: [{:.2f}, {:.2f}]"
            .format(label, np.percentile(fw, 0.5), np.percentile(fw, 99.5)))
    f, ax1 = plt.subplots(1, 1)
    p = sns.violinplot(data=bias_weights[:], orient="v",
        palette=sns.color_palette(palette="RdBu", n_colors=1),
        ax=ax1)
    p.tick_params(labelsize=16)
```

```
    p.set_xlabel('Layer', fontsize=20)
    ax1.set_xticklabels(labels[:])
    ax1.set_title('Bias weight distribution by layer')
    sns.plt.show()

def show_batchnorm_weight_dist(model):

    # analyze
    gamma_weights = []
    beta_weights = []
    layer_index = 0
    labels = []
    for layer in model.layers:
        if isinstance(layer, keras.layers.normalization.BatchNormalization):
            labels.append(layer_index)
            weights = layer.get_weights()
            data = weights[0].flatten()
            gamma_weights.append(data)
            data = weights[1].flatten()
            beta_weights.append(data)
        layer_index += 1
    labels = [2, 4, 6, 8, 10, 12, 14, 16] # baseline

    # Gamma weights
    if len(gamma_weights) > 0:
        for label, fw in zip(labels, gamma_weights):
            print("99% gamma interval of layer {}: [{:.2f}, {:.2f}]"
                .format(label,
                    np.percentile(fw, 0.5),
                    np.percentile(fw, 99.5)))
    f, ax1 = plt.subplots(1, 1)
    p = sns.violinplot(data=gamma_weights, orient="v",
                    palette=sns.color_palette(palette="RdBu",
                                n_colors=1),
                    ax=ax1)
        p.tick_params(labelsize=16)
        p.set_xlabel('Layer', fontsize=20)
        ax1.set_xticklabels(labels)
        ax1.set_title('Gamma distribution by layer')
        sns.plt.show()

    # beta weights
    if len(beta_weights) > 0:
        for label, fw in zip(labels, beta_weights):
            print("99% beta interval of layer {}: [{:.2f}, {:.2f}]"
                .format(label,
                    np.percentile(fw, 0.5),
                    np.percentile(fw, 99.5)))
        f, ax1 = plt.subplots(1, 1)
        p = sns.violinplot(data=beta_weights, orient="v",
               palette=sns.color_palette(palette="RdBu",
                                          n_colors=1),
               ax=ax1)
    p.tick_params(labelsize=16)
    p.set_xlabel('Layer', fontsize=20)
    ax1.set_xticklabels(labels)
    ax1.set_title('Beta distribution by layer')
    sns.plt.show()
def main(config, data_module, model_path, image_fname):

    model = load_model(model_path)
    print("## Activation analyzation")
    if image_fname is not None:
        image = scipy.misc.imread(image_fname)
    else:
        data = data_module.load_data(config)
        X_train = data['x_train']
        X_test = data['x_test']
        # y_train = data['y_train']
        # y_test = data['y_test']
        X_train = data_module.preprocess(X_train)
        X_test = data_module.preprocess(X_test)
        image = X_train[0]

    show_conv_act_distrib(model, image)
    print("## Weight analyzation")
    show_conv_weight_dist(model)
    print("## BN analyzation")
    show_batchnorm_weight_dist(model)

def get_parser():

    from argparse import ArgumentParser, ArgumentDefaultsHelpFormatter
    parser = ArgumentParser(description=__doc__,
               formatter_class=ArgumentDefaultsHelpFormatter)
    parser.add_argument("-f", "--file",
           dest="filename",
           help="Experiment yaml file",
           required=True,
           metavar="FILE")
    parser.add_argument("--model",
           dest="model_path",
           help="Model h5 file",
           metavar="FILE")
    parser.add_argument("--image",
           dest="image_fname",
           help="A single image",
           metavar="FILE")
    return parser
```

```
if __name__ == "__main__":
    args = get_parser().parse_args()

    # Read YAML experiment definition file
    with open(args.filename, 'r') as stream:
        config = yaml.load(stream)

    # Make paths absolute
    config = make_paths_absolute(os.path.dirname(args.filename),
                                 config)
    # Print experiment file
    pp = pprint.PrettyPrinter(indent=4)
    pp.pprint(config)

    # Load data module
    dpath = config['dataset']['script_path']
    sys.path.insert(1, os.path.dirname(dpath))
    data = imp.load_source('data', config['dataset']['script_path'])

    # Load model
    if args.model_path is not None:
        model_path = args.model_path
    else:
        artifacts_path = config['train']['artifacts_path']
        model_path = os.path.basename(config['train']['artifacts_path'])
        model_paths = glob.glob("{}/*.h5".format(artifacts_path))
        model_paths = [m for m in model_paths if "_chk.h5" not in m]
        model_path = model_paths[0]
    logging.info("Take {}".format(model_path))
    main(config, data, model_path, args.image_fname)
```

File adam_keras.py

```
from keras.optimizers import Adam

def get_optimizer(config):
    lr = config['optimizer']['initial_lr']
    optimizer = Adam(lr=lr) # Using Adam instead of SGD to speed up training
    return optimizer
```

File sgd.py

```
from keras.optimizers import SGD

def get_optimizer(config):
    lr = config['optimizer']['initial_lr']
    optimizer = SGD(lr=lr) # Using Adam instead of SGD to speed up training
    return optimizer
```