MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY

Department of Aviation Computer Integrated Systems

ACCEPT TO PROTECTION

Head of Department
Sineglazov V.M.
“____” _____ 2021
y.

GRADUATE WORK

(EXPLANATORY NOTE)
HIGHER EDUCATION STUDY
«BACHELOR»

Subject: «Deep pyramidal residual hybrid neural network»

Performer: Katrenko M.O.

Supervised: Sineglazov V. M.

Norm control: Tupitsyn M. F.

Kyiv 2021

NATIONAL AVIATION UNIVERSITY
Scientific faculty of aeronavigation, electronics and telecommunications

Department of Aviation Computer Integrated Systems

Education level: Bachelor

Specialty: 151, «Computer-integrated technological processes and production»

APPROVE
Head of
department
Sineglazov V.M.
"_____" _____ 2021 p.

TASK

for execution the bachelor work of student
Katrenko M.O.

1. Theme of the work: "Deep pyramidal residual hybrid neural network"

2. Term of execution of the work: from 20.02.2021 till 15.06.2021.

3. Initial data of the work: Topology of hybrid neural networks: residual neural network and pyramidal, solution type image-processing, machine learning – back propagation with gradient algorithm for example AdaGrad. Application problem – medicine diagnostics.

4. The contents of the explanatory note (list of issues to be developed):
1. Analysis and substantiation of the choice of methods of identification; 2. Review and analysis of existing developments in the field of neural networks. 3. Development of the block diagram of the intelligent system based on the convolutional neural network; 4. To learn the way how to set up and train the neural network.

5. List of compulsory graphical materials:
1. Topology of a deep pyramidal residual hybrid neural network. 2. The structure of the residual block for a hybrid neural network. 3. The results of a hybrid neural network on a set of images CIFAR-10. 4. The results of a hybrid neural network on real images for medical purposes.

6. Calendar Schedule

№	Task	Period of execution	Performance note
1	Selection and searching of literature and resources	23.02.2021 – 26.02.2021	done
2	Analysis of neural networks and justification of the choice of methods of identifying	27.02.2021 – 19.03.2021	done
3	Configure model of development and analysis of combined hybrid neural network system and its research by software modeling	20.03.2021 – 08.04.2021	done
4	Development of an algorithm for preparation of output data	09.04.2021 – 18.04.2021	done
5	Programming of a combined identification system	19.04.2021 – 11.05.2021	done
6	Obtaining and analysis of system results	12.05.2021 – 21.05.2021	done
7	Formation of conclusions on the performed work	22.05.2021 – 24.05.2021	done
8	Making an explanatory note	25.05.2021 – 30.05.2021	done
9	Presentation creating	31.05.2021 – 11.06.2021	done

7. Date of issuance of task: “21” December 2020.

Supervisor: _____ Head of ACIC Dr. Science (Eng.)

(supervisors sign) Professor Sineglazov V.M.

(П.І.Б.)

Task is accepted for execution _____ Katrenko M.O.

(graduates sign) (П.І.Б.)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра авіаційних комп'ютерно-інтегрованих комплексів

ДОПУСТИТИ ДО
ЗАХИСТУ
Завідувач кафедри

Синєглазов Віктор Михайлович
"____" _____ 2021 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО РІВНЯ "БАКАЛАВР"

Тема: "Глибока пірамідально-залишкова гібридна нейронна мережа"

Виконавець: _____ Катренко М.О.
Керівник: _____ Синєглазов В. М.

Консультанти з окремих розділів пояснювальної записки:

Нормоконтролер: _____ Тупіцин М. Ф.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Науково навчальний факультет аеронавігації, електроніки та телекомунікацій

Кафедра авіаційних комп'ютерно-інтегрованих комплексів

Освітній рівень: Бакалавр

Спеціальність: 151, «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Синєглазов В.М.
"_____" _____ 2021 р.

ЗАВДАННЯ

на виконання дипломної роботи студента

Катренка Максима Олександровича

1. Тема проекту (роботи): "Глибока пірамідальна залишкова гібридна нейронна мережа "

2. Термін виконання проекту (роботи): з 20.02.2021 р. до 20.06.2021 р.

3. Вихідні данні до проекту (роботи): Топологія гібридних нейронних мереж: залишкові нейромережі та пірамідальні, типи рішень обробки зображень, машинне навчання - зворотне розповсюдження помилки з градієнтним алгоритмом, наприклад AdaGrad. Проблема застосування - діагностика в цілях медицини.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): 1. Аналіз та обґрунтування вибору методів ідентифікації; 2. Огляд та аналіз існуючих розробок у галузі нейронних мереж. 3. Розробка гібридної нейронної мережі на основі існуючих згорткової нейронної мережі; 4. Вивчити спосіб налаштування та навчання нейронної мережі.

5. Перелік обов'язкового графічного матеріалу:
1. Топологія гібридної нейронної мережі. 2. Структура залишкового блоку для гібридної нейронної мережі. 3. Результати гібридної нейронної мережі на наборі зображень CIFAR-10. 4. Результати гібридної нейронної мережі на реальних зображеннях в медичних цілях.

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Підбір та пошук літератури та ресурсів	23.02 – 26.02	виконано
2	Аналіз нейронних мереж та обґрунтування вибору методів ідентифікації	27.02 – 19.03	виконано
3	Налаштування моделі розвитку та аналізу комбінованої гібридної нейромережевої системи та її дослідження за допомогою програмного моделювання	20.03 – 08.04	виконано
4	Розробка алгоритму підготовки вхідних даних	09.04 – 18.04	виконано
5	Програмування комбінованої системи ідентифікації	19.04 – 11.05	виконано
6	Отримання та аналіз результатів роботи	12.05 – 21.05	
7	Формування висновків щодо виконаної роботи	22.05 – 24.05	виконано
8	Оформлення пояснювальної записки	25.05 – 30.05	виконано
9	Створення презентації	31.05 – 11.06	виконано

7. Консультанти з окремих розділів

8. Дата видачі завдання: “21" _грудня_ 2020 р.

Керівник дипломної роботи _____ Синєглазов В.М.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Катренко М.О.

ABSTRACT

The aim of this work is to solve the problem of structural parametric synthesis of hybrid convolutional networks which include residual and pyramidal neural networks.

The topologies of modern convolutional neural networks, namely the residual and pyramidal neural network, are considered in the work. The necessity of using these neural networks to solve image processing problems is substantiated.

A study of these convolutional networks. The accuracy of image processing in solving classification problems, research on the size of the convolution, the number of feature maps and the depth and width of neural network data were analyzed.

An algorithm for training a hybrid neural network consisting of residual blocks and a pyramidal architecture has been developed and the conditions for improving the efficiency of the hybrid neural network have been determined, and examples are given in the CIFAR-10 data set and real examples of tumors detecting in medicine.

АНОТАЦІЯ

Метою роботи є розв’язання задачі структурно параметричного синтезу гібридних згорткових мереж які включають в себе залишкову та пірамідальну нейронну мережу.

У роботі розглядаються топології сучасних згорткових нейронних мереж, а саме залишкову та пірамідальну нейронну мережу. Обґрунтовано необхідність використання для розв’язання задач обробки зображень цих нейронних мереж.

Проведено дослідження даних згорткових мереж. Проаналізована точність обробки зображень під час розв’язування задач класифікації, дослідження відносно розміру згортки, кількість карт ознак та досліджена глибина та ширина даних нейронних мереж.

Розроблено алгоритм навчання гібридної нейронної мережі яка складається з залишкових блоків, та пірамідальної архітектури та визначено умови підвищення ефективності роботи гібридної нейронної мережі та наведено приклади на наборі даних ЦИФАР-10 та приклади на реальних задачах по виявленню пухлин у сфері медицини.

CONTENT

GLOSSARY

CNN - convolutional neural network

ANN - artificial neural network

CL - convolution layer

FC - face recognition

AI - artificial intelligence

AGI - artificial general intelligence

TT - Turing test

RGB - red-green-blue

MP - max pooling

AP - average pooling

LN - landmark normalization

GPU - graphical processing unit

RNN - recurrent neural network

ReLU - rectified linear units

INTRODUCTION

In today’s world, the need for computerized object recognition in images using neural networks grows every year. People do not stand still but are constantly looking for something new and unattainable. Therefore, today a certain type of neural network called convolutional neural network is used to recognize objects in an image.

Convolutional neural networks use various multilayer perceptrons designed in such a way that a minimal amount of preprocessing is required. Convolutional networks are based on a biological process, namely the connection pattern of neurons in the visual cortex of animals. Individual neurons in the cortex respond to stimuli only in a limited area the visual field, also known as the receptive field. Also, the receptive field data of many neurons is partially closed or overlapping, so that they cover the entire visual field. This type of neural network was proposed by French machine learning and artificial intelligence scientist Yann LeCun.

Today, convolutional neural networks are widely used, in areas such as autopilot in cars, facial recognition of crime scenes, recognition of diseases, symptoms, recognition of satellite images of the planet, and others. Recent developments in software combined with precise algorithms in artificial intelligence and computer vision allow software engineers to develop a subsystem for automatic object recognition.

In general, the convolutional neural network is today the "workhorse" in the field of neural networks. It is used primarily for computer vision tasks, although it can also be applied to audio and any data that can be represented as matrices.

In this thesis, the problem of hybrid parametric synthesis is solved and 2 neural network architectures, residual and pyramidal, are investigated and a hybrid based on them is created using the CIFAR-10 image collection dataset.

CHAPTER 1. CONVOLUTION NEURAL NETWORK AS THE MEAN OF IMAGE PROCESSING

1.1 State of modern neural network use in science and techniques

In the modern world, neural networks are used in many areas, for example:

- **Economy and business***:* time series forecasting (exchange rates, product prices, demand, sales volumes, ...), automated trading (currency, stock or commodity exchange trading), credit default risk assessment, bankruptcy forecasting, genuine estate valuation, identification of over- and undervalued companies, ranking, commodity and cash flow optimisation, cheque and document reading and recognition, plastic card transaction security.
- **Medicine and healthcare***:* patient diagnosis (disease diagnosis), processing of medical images, cleaning of instrument readings from noise, monitoring of patient condition, prediction of the results of various treatments, analysis of treatment effectiveness.
- **Avionics***:* learning autopilots, radar recognition, adaptive piloting of heavily damaged aircraft, unmanned aerial vehicles (drones).
- **Communications***:* video compression, fast encoding and decoding, and optimisation of cellular networks and packet routing schemes.
- **Internet***:* associative information retrieval, e-secretaries and offline agents on the Internet, spam filtering and blocking, automatic categorization of messages from news feeds, targeted advertising and marketing for e-commerce, captcha recognition.

- **Production automation***:* Optimisation of production processes, quality control, monitoring and visualisation of multi-dimensional dispatch information, prevention of accidents.
- **Robotics***:* recognition of the scene, objects and obstacles in front of the robot, route guidance, control of manipulators, maintaining equilibrium.
- **Political science and sociological researches***:* predicting election results, analysing polls, predicting rating dynamics, identifying significant factors, clustering the electorate, studying and visualising the social dynamics of the population.
- **Security, security systems***:* face recognition; identification of a person by fingerprint, voice, signature or face; number plate recognition; monitoring information packets and information flows in a computer network to detect intrusions; forgery detection; analysis of data from video cameras and various sensors; analysis of aerial images (for example, to detect forest fires or illegal logging).
- **Input and processing of information***:* recognition of handwritten texts, scanned postal, payment, financial and accounting documents; recognition of voice commands, voice text input into a computer.
- **Geological exploration***:* seismic data analysis, associative mineral prospecting methods, deposit resource estimation.
- **Computer and board games***:* creating neuro-players in draughts and chess (ratings confirmed by playing with humans - at the level of masters and international masters), winning in go with European and world champions, on average better than humans, getting through almost fifty old classic games with Atari (all kinds of pongs, paceman, ...).

Neural networks for doctors

The use of neural networks in medicine is actively evolving due to the large number of annotated images, the increase in computing power and the advent of cloud-based data storage. Neural networks influence the state of medicine on three levels:

1) To help doctors interpret images quickly and accurately;
2) Reduce the number of medical errors;
3) Help patients analyse their own data with sensors to monitor their condition.

However, researchers are still at an early stage of using neural networks in medical practice because of limitations that prevent them from being fully exploited.

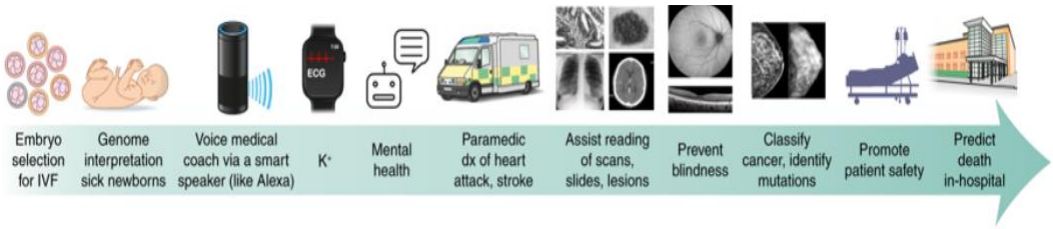

Figure 1.1 Examples of using NN in medicine

Deep Neural Networks (DNN) can help in the interpretation of medical scans of pathologies, electrocardiograms, and endoscopy. There is a particular focus on radiology - using neural networks to analyse X-ray scans. Google has used algorithms to interpret chest scans to make 14 different diagnoses, ranging from pneumonia to cardiac hypertrophy and collapsed lungs. DNNS can also diagnose certain cancers, fractures, haemorrhages, retinopathy, skin lesions and many other diseases. Algorithms can improve the work of dermatologists, cardiologists, ophthalmologists and even psychotherapists by monitoring the development of depression.

Neural network for patients

Algorithms that patients can use on their own are developing more slowly than algorithms used by doctors. In 2017, the US Food and Drug Administration (FDA) approved an algorithm to detect arrhythmias in a smartwatch, and then in 2018 Apple received FDA approval for the algorithm used in the Apple Watch Series 4. Sensors on the watch detect the user's heart rate at rest and during physical activity, and if there is a strong deviation from what is expected, the user is alerted to an ECG recording through the watch, the results of which are then interpreted by the algorithm.

Some smartphone apps use neural networks to monitor and control medication intake, such as AiCure causing a patient to take a selfie video while swallowing a prescribed pill.

Algorithms based on how glucose levels rise or fall are used by diabetic patients. They help to prevent episodes of hypoglycaemia. Thus, common chronic diseases such as hypertension, depression and asthma can theoretically be better controlled with supplements.

Problem of NN in medicine

A major challenge for the future of artificial intelligence in medicine is how well data privacy and security can be ensured. There is a risk of disclosing sensitive patient data from medical histories. There is also the risk of intentionally disrupting the algorithm to harm people on a large scale, such as an overdose of insulin in diabetics.

The second problem is the imprecise operation of algorithms. A recent example is IBM's Watson Health algorithm (known as Watson for Oncology).

Used by hundreds of hospitals around the world to make treatment recommendations for cancer patients, the algorithm was based on a small number of synthetic cases and very limited real-world data. Many of his treatment recommendations were flawed, such as suggesting the use of an incompatible drug for a patient with severe bleeding, which is a clear contraindication.

Another problem is bias. Low socioeconomic status is a major risk factor for premature mortality. The disproportionate use of AI among the 'haves' compared to the 'have-nots' may increase the existing health gap. Closely related to this problem is the bias in the results due to insufficient inclusion of minorities in data sets. An example would be algorithms in dermatology that diagnose melanoma but do not take skin colour into account. Prejudice must be eradicated and medical research that provides a truly representative representation of the population must be strived for.

Future abilities for NN in medicine

Scientists emphasise that careful research into the algorithms' performance and testing in clinical settings is essential. Human health is too valuable, so in the near future AI will only be able to perform routine tasks with minimal risk.

Despite the challenges, in the future researchers see the use of neural networks in software that will process vast amounts of data quickly and accurately, and machines that will see and do things that humans cannot. Ultimately, this will lay the foundation for highly efficient, data-driven medicine and reduce reliance on human resources.

Neural networks in gaming industry

When it comes to applying machine learning to the games industry, there are a number of limitations that make it not always possible to use this type of architecture. These include system requirements, particularly those related to the CPU, which determine the computer's ability to process complex game structures and its suitability for creating game story and gameplay.

Thus, it turns out that in many games it is impossible to organize the necessary hardware to implement a complex AI system, much less a server cluster, which exists, for example, for image recognition networks on Facebook. Sometimes multiple AIs have to run simultaneously - and not only on computers, but also on mobile devices and other less productive platforms. All of this place’s constraints on the size and complexity of the machine learning architecture, as all calculations still need to be performed with frame durations on the order of 1 or 2 milliseconds. Of course, it is possible to use various optimization and load-balancing techniques between frames, but it is still not possible to completely get rid of these limitations.

The complexity of the game can cause serious problems for the AI. Indeed, games like StarCraft II have game mechanics many times more complex than Atari games. So don't expect machine learning to cope with learning the entire state of the game and be able to interact with it at a given frame rate and with known system requirements. Just as the player is often guided by intuition in the early stages of a game, the AI must learn to initially process the game's state to make it easier to play later on. For example, one recent API for StarCraft II displayed only information that the developers deemed important: In one case, the AI used a reduced view of the entire map area; in another, like the player, it could move the camera around, and then its perception was limited to the information on the screen.

Often, conventional methods of solving machine learning problems are not applicable to game artificial intelligence. For example, it does not usually need to win or do its best to win, as it did in the Atari games. More often, the AI's role is to make the game more fun. It may have to play a certain role and behave in a way that conveys the character it is responsible for. Thus, in-game AIs are more tied to the game design and story and should have the necessary tools to manipulate their behaviour to achieve its goal. Machine learning in its purest form is not always suitable for this - which means you have to look for something else.

Neural networks in science

One promising application area for artificial neural networks (ANNs) is industrial manufacturing. In this field, there is a marked trend towards production modules with a high level of automation, which requires an increasing number of intelligent self-adapting and self-regulating machines. However, production processes are characterised by a large variety of dynamically interacting parameters, which makes it difficult to create adequate analytical models. Modern production processes are becoming ever more complex. This slows down the introduction of new technological solutions. Moreover, in a number of cases, successful analytical mathematical models show failure due to the lack of computational power. Consequently, there is growing interest in alternative approaches to production process modelling using ANNs, which provide real-time models with small uncertainties that can be trained during use. The advantages of neural networks make their use attractive for tasks such as:

- forecasting;
- planning;
- ACS design;
- quality control;
- manipulators and robotics control;

- ensuring operational safety: identifying faults and preventing emergencies;
- process control: optimisation of production process modes; monitoring and visualisation of dispatch information.

Today, ANN-based forecasting is most fully realised in finance and economics. In industrial production, neural networks can be useful, for example, in creating enterprise risk management models, production cycle planning. Production modelling and optimisation is characterised by high complexity, large variables and constants that are not defined for all possible systems. Traditional analytical models can often only be built with considerable simplification and are mainly of an estimation nature. Whereas an LIS is trained on the basis of real or numerical experiment data.

Classical methods of building an automated process control system are based on formalised human knowledge of the control object. The option of building an ACS based on a neural network implements cognitive techniques inherent to humans. Examples of successful applications of ANN in this area include the control of complex processes and objects under conditions of information uncertainty, machining processes, robotic systems, etc.

Much experience has been gained in the use of ANNs in quality management in industry. For example, the neural network used at Intel to detect defects in chip production is able to reject a defective chip with 99.5% accuracy. By emitting sound waves and receiving the reflected signal and then processing the ANN, experts at the National Institute of Standards and Technology (NIST) check the quality of concrete with material thicknesses up to half a meter.

In the field of fault detection, the use of ANNs makes it possible to monitor the condition of equipment in real time, detect deviations and prevent the occurrence of emergency situations. The use of neural networks in pollution monitoring is very promising, which also reduces the risk of industrial accidents.

Neural networks in information technologies

Due to the increasing complexity and heterogeneity of modern information and communication systems, traditional measures to ensure their functioning are becoming increasingly inadequate. One of the promising directions for solving practical problems in the field of information technology is to investigate the possibilities of using ANNs.

Analysis of the relevant special literature has shown that in the information environment, neural networks have proven themselves in the following areas:

- network management and optimization;
- ensuring information security of communication networks;
- recognition of the entered information;
- information processing and retrieval.

ANN successfully solves an important problem in telecommunications - finding the optimal traffic path between nodes. Two features are taken into account: firstly, the solution must be adaptive, i.e., it must take into account the current state of the communication network and the presence of bad spots, and secondly, the optimal solution must be found in real time. In addition to flow routing control, neural networks are used to obtain efficient solutions in telecommunication network design.

In the field of information security, neural networks have also successfully solved problems that are inaccessible to traditional subsystems that focus on predefined classes of threats in the face of increasing complexity and dynamics of heterogeneous software. For example, the use of NS-based technologies for identification and authentication, anti-virus protection, intrusion detection and prevention, information security risk management, vulnerability detection, etc., has been well studied.

In addition to these examples of theoretical research, references to practical implementations can be found in open sources.

For example, the Vicarious recognition system developed by GoogleX Labs solves the problem of passing the CAPTCHA test. Researchers were able to achieve 90% accuracy in CAPTCHA recognition from Google, Yahoo, PayPal, Captcha.com and other projects. This research shows that modern CAPTCHAs are no longer an effective Turing test. In another

Google X project has created an experimental computer neural network capable of independently recognizing morphological objects, such as cats, in a video stream. Another example of a working project is the work of a number of firms focused on anti-fraud in the banking sector.

Speech recognition is one of the most popular applications of neural networks. A demonstration system of speaker-independent speech control with an integrated Windows calculator (developed by the Russian company NeuroProject) is capable of recognizing 36 commands spoken into a standard microphone, regardless of voice and pronunciation patterns. The use of neural network technology in cryptography is also represented by interesting developments.

The most frequent use of neural networks is associated with Merkle-Hellman public key cryptosystems.

Fein-Marquart Associates Inc. has developed a program for postcode recognition with automatic further sorting. The system recognizes both typeset and handwritten numbers. The recognition accuracy is estimated at 98%. AT&T Bell Laboratories achieved 0.14% NN errors in character recognition when presented with a training pair from a set of representative samples used in training, and 5.0% when recognizing "new" characters.

In the information society, the use of ANNs for coding and decoding information is a promising direction. The priority here is the task of processing speech information and images. The information compression method proposed in 1987 is well known. Much attention is paid to the construction of neural network receivers with multiple access.

1.2 Convolutional neural networks

In the modern world, the analysis and obtained some of the best results in the recognition of objects on different types of images convolutional neural network, which is quite relevant over time and developments in the field of neural architectures of neural networks, such as: cognitron and neocognitron. The key to its key success is its ability to analyze two-dimensional image architecture, as opposed to a multilayer perceptron.

Convolutional neural networks have a fairly good resistance to changes in the scale of the image, movement relative to the axis, rotation around the axis, changes in perspective relative to the place where this photo was taken and other possible distortions of the image. Hilly neural networks include three architectural / topological ideas to provide static scale, rotation about an axis, space shift, and other spatial distortions that may occur, including such as:

- local receptor fields (provide local two-dimensional connectivity of neurons);

- general synaptic coefficients (provide detection of some features anywhere in the image and reduce the total number of weight coefficients);

- hierarchical organization with spatial subsamples;

In the 21st century, convolutional neural networks and all their possible modifications prove that they are the best algorithms for finding objects in terms of recognition accuracy and recognition speed. Since 2000, neural networks have been ranked highest in a well-known international competition among image recognition developers and professionals called ImageNet.

Convolutional networks are a very good example of how the biological structure of humans helps to create similar technologies using a multilayer perceptron. To date, the most successful results in object recognition have been obtained with it. In the best cases, the recognition accuracy of these networks even exceeds the so-called already created ANN, which are already used in people's lives by 10-15%. Also, convolutional neural networks are the main branch of technologies that are associated with methods and approaches to deep learning.

The main reason for such a great success of convolutional neural networks was the so-called idea of common weights.

If you don't take into account the huge size of these neural networks, it still has fewer settings when compared to their parents, such as the neocognitron.

There is a very similar option, which is very similar to a convolutional neural network is a neocognitron. Some neural networks do not partially use the idea of common weights, but the learning algorithms use the same ones, the inclusive feature of which is the method of error backpropagation.

Convolutional neural networks have the ability to run quite fast on a machine that uses sequential operations, but if you use modules that can parallelize the learning processes of these networks, the speed can be increased by 2, 3, or even 5 times by parallelizing the convolution on each map signs, as well as deconvolution in the propagation of the error over the network.

1.2.1 Structure of convolutional neural network

A convolutional neural network consists of different types of layers: convolutional layers, subsampling layers and 'normal' neural network layers as shown on Figure 1.2.

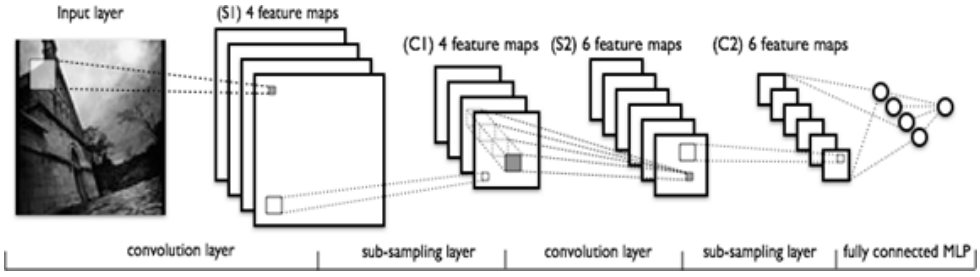

Figure 1.2 Topology of convolutional neural network

The first two types of layers (convolutional, subsampling), alternating with each other, form the input feature vector for the multilayer perceptron.

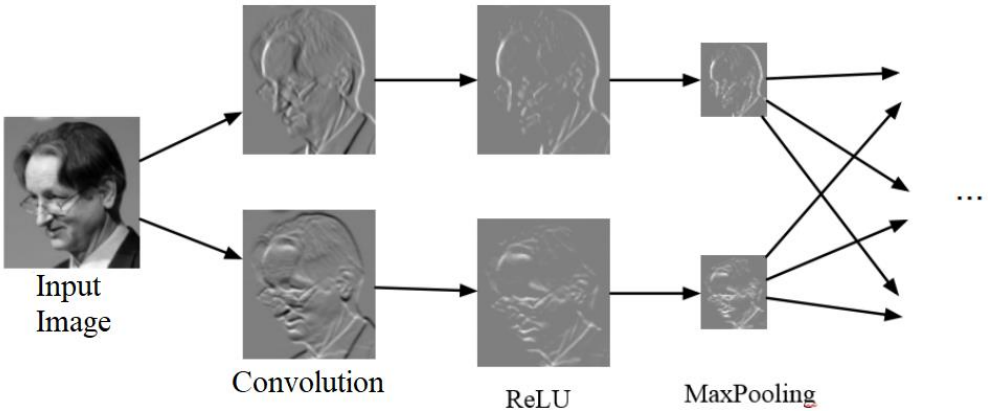

Figure 1.3 Visualization of convolution and subsampling

1.2.2 Topology and basic parameters of convolutional neural network

If we turn to scientific articles, the rule is that the topology of the neural network should be chosen depending on the specific problem and the solution to be found, but Figure 1.4 shows the standard default CNN topology.

The number of feature cards is usually set by the requirements for a particular task, for example, if you use a large number of feature cards, then the load on the system on which the neural network will study will increase, but the quality and accuracy of feature maps will increase several times. If you take the best way or the so-called rule of ratio of rapid learning and accuracy, there are recommendations in which it is usually suggested to take the average value of the feature maps.

In Figure 1.4 you can see the standard topology of the convolutional neural network which can be seen wherever their training is used.

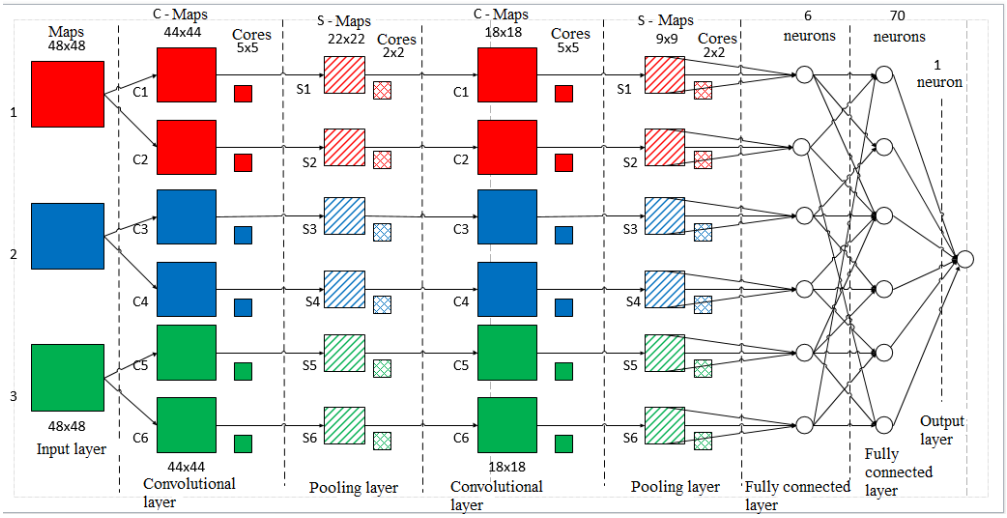

Figure 1.4 Topology of convolutional neural network

Input layer

Typically, the input image type in the input layer is a royal JPEG image with a different pixel type. But if the image size is larger, it is clear that the speed and complexity of processing will also increase several times, so in order for the calculation speed to be adequate, and within the limits we need, the image size must be determined by the problem to be chosen. . But at the same time, if the image is too small, or not high quality with a small expansion, then simply the convoluted layers will not be able to select the feature maps so that they correctly highlight the features of the image to further identify it in the work. Each image is standardly divided into 3 channels: red, blue, green, so-called RGB, but the image can be in other color and colorless spectra.

The input layer takes into account the two-dimensional architecture of the images and consists of several maps called matrices or core, the feature map can be one for one hill, but if the image in 3 spectra as described earlier, the maps should be 3, for each color spectrum color spectra, the maps may be larger than previously indicated. Each feature map corresponds to one color channel.

For example, 1 of the blue spectra corresponds to the maps for this channel, for the red spectrum the feature maps will correspond to the red channel maps, for green they should be green. If the image in the channel is gray, the feature map will be gray for one channel.

The input data for each specific pixel value is normalized to a range from 0 to 1, using the formula (1.1):

$$f(p, min, max) = \frac{p - min}{\max - min}, \qquad (1.1)$$

where

f - normalization function,

p – the value of a particular pixel color from 0 to 255,

min – minimum pixel value 0,

max – maximum pixel value 255.

Convolutional layer

A convolutional layer is a layer in which the process of selecting a set of maps takes place, in other words, they are function maps, or they are also called as regular matrices, each map has a so-called synaptic nucleus. Many other jares call these mathematical features of the kernel, filters, matrices, kernel algorithms.

The set and size of maps is determined by the specific task to be solved. The more feature maps, the more characteristics will be selected during the convolution process, but the load on the system will increase. As stated in the recommendations for neural networks, you need to use a ratio of one to two.

The size of all maps of the convolutional layer is the same and is calculated by the formula (1.2):

$$(w, h) = (mW - kW + 1, mH - kH + 1), \quad (1.2)$$

where

(w, h) – the computed size of the convolutional map,

mW – width of the previous map,

mH – height of the previous map,

kW – width of a core,

kH – height of a core.

A kernel is a filter or window, or convoluted kernel, that runs the entire area of the previous feature map and finds certain features and characteristics of the objects, saving them for future processing. For example, if a neural network has been trained on multiple faces, one of the many filters may emit the highest signal in the area of horizontal lines, circles, vertical lines, and other similar characteristics during training. At this time, other filters will highlight other features, such as lines at an angle of 45 degrees, or some other differences from those described earlier. The size of the convolution filter is usually taken in the range from 3x3 to 7x7. If the size of the core is not large, then a convolution with such a dimension may be useless, and not detect any signs.

The same can happen if it is very large and this will further increase the number of neuronal connections. Another characteristic is that the size of the core should be chosen so that the size of the convolutional layer maps is uniform, this ensures that information is not lost when the dimension decreases or increases in the sample layer.

The kernel is a system of shared weights or synapses; this is one of the main features of a convolutional neural network. In a conventional multilayer network, there are many connections between neurons, i.e., synapses, which significantly slows down the recognition process. In the convolutional network, on the contrary, the total weights allow to reduce the number of connections and find the same feature over the entire image area.

Initially, the values of each map of the convolutional layer are equal to 0. The values of the weights of the kernels are set randomly in the range from -0.5 to 0.5. The kernel slides over the previous map and performs the convolution operation, which is often used for image processing, the formula:

$$(f * g)[m,n] = \sum_{k,l} f(m-k, n-l] * g[k,l], \tag{1.3}$$

where

f – initial image matrix,

g – kernel of convolution.

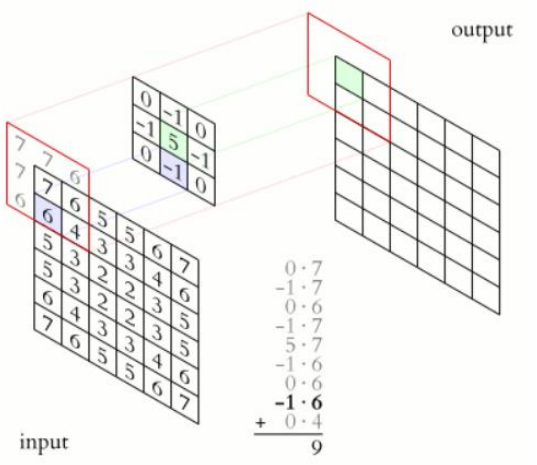

Figure 1.5 Operation of convolution and getting the values of convolutional map

Depending on the method of processing the edges of the original matrix, the result may be less than the original image (valid), the same size (same) or larger size (full), in accordance with Figure 1.6.

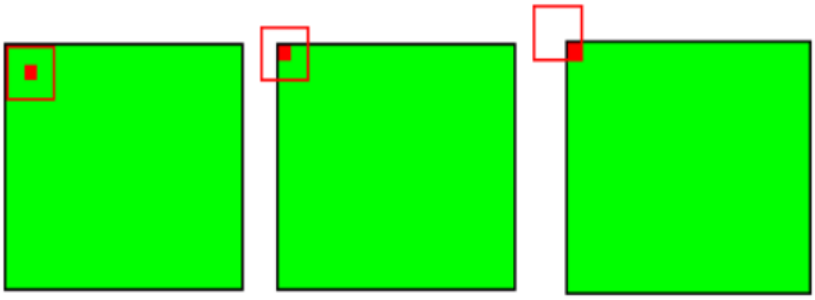

Figure 1.6 Three types of initial convolutional matrix

Pooling layer

The association layer, like the convolutional layer, has maps, but their number is the same as the previous (convolutional) layer. The purpose of the layer is to reduce the dimensionality of the maps of the previous layer.

If the previous convolution operation has already identified any features, such a detailed image is no longer needed for further processing, and it is compressed to a less detailed image. In addition, filtering out unnecessary detail helps to avoid overtraining.

While the kernel of a sub-sample (filter) layer scans the previous layer's map, the scanning kernel does not overlap, unlike a convolutional layer. Typically, each map has a 2x2 kernel, which reduces the previous convolutional layer maps by a factor of 2. The whole feature map is divided into 2x2 feature cells, from which the maximum value is selected.

Typically, the ReLU activation function is used in the subsample layer. Subsampling operation (or MaxPooling - selection of the maximum) in accordance with Figure 1.8.

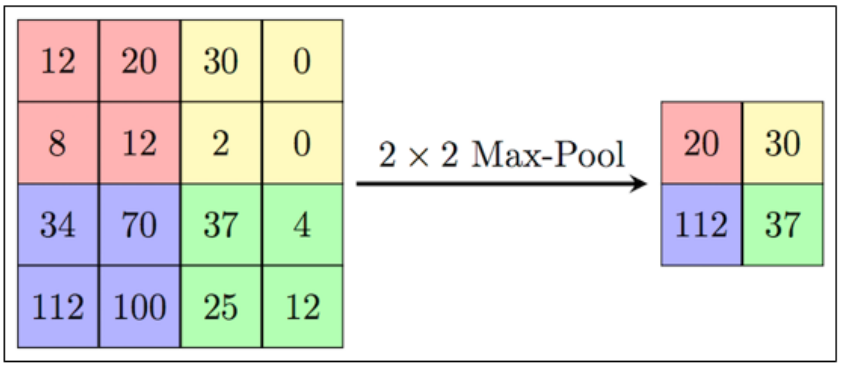

Figure 1.7 Generating a new sub-sampling layer map based on the previous convolutional layer map. Operation of subsampling (Max pooling)

It can be described by the formula:

$$x^l = f(a^l * subsample(x^{l-1}) + b^l), \quad (1.4)$$

where

x^l – output of l layer

f () – activation function

a^l, b^l - bias coefficients

subsample () - local maximum sampling operation

Fully connected layer

The last of the layer types is the layer of a conventional multilayer perceptron. The purpose of the layer is classification, it simulates a complex non-linear function, by optimizing which the quality of recognition is improved.

The neurons of each map of the previous sublayer are connected to one neuron of the hidden layer. Thus, the number of neurons in the hidden layer is equal to the number of cards in the subsample layer, but the connections can be not necessarily so, for example, only some of the neurons of any of the cards in the subsample layer will be connected with the first neuron of the hidden layer, and the rest with the second, or all neurons of the first cards are connected with neurons 1 and 2 of the hidden layer. The calculation of neuron values can be described by the formula:

$$x_j^l = f\left(\sum_i x_i^{l-1} * w_{i,j}^{l-1} + b_j^{l-1}\right), \quad (1.5)$$

where

x_j^l – feature map j (output from l layer),

f () – activation function,

b^l – bias coefficient of l layer,

$w_{i,j}^l$ - matrix of weighting coefficients of the layer l.

Output layer

The output layer is a fully connected layer with the number of classes whose size is determined by the number of classes our system has to identify. In our case it is 10 neurons, as our training set has 10 classes of images (CIFAR-10). It is designed to return the result of the whole neural network; it is called an identifier.

Activation function

One of the steps in neural network development is the selection of a neuron's activation function. The type of activation function largely determines the functionality of the neural network and the method for training this network. The classical backpropagation algorithm works well for two-layer and three-layer neural networks, but it starts to have problems when the depth is further increased. One reason for this is what is known as gradient decay. As the error propagates from the output layer to the input layer, on each layer the current result is multiplied by the derivative of the activation function. The derivative of a traditional sigmoid activation function is less than unity over the entire domain, so the error will become close to zero after a few layers. If, on the other hand, the activation function has an unbounded derivative (such as the hyperbolic tangent), the error can grow explosively as it propagates, making the learning procedure unstable. Consider the most common activation functions used in neural networks.

Sigmoid function

This function belongs to the class of continuous functions and takes an arbitrary real number as input and gives a real number in the range 0 to 1 as output. In particular, large (modulo) negative numbers are converted to zero and large positive numbers are converted to one.

Historically, the sigmoid has found widespread use because its output is well interpreted as a neuron's activation level: from no activation (0) to fully saturated activation (1). The sigmoid is expressed by the formula:

$$f(s) = \frac{1}{1 + e^{-s}}, \tag{1.6}$$

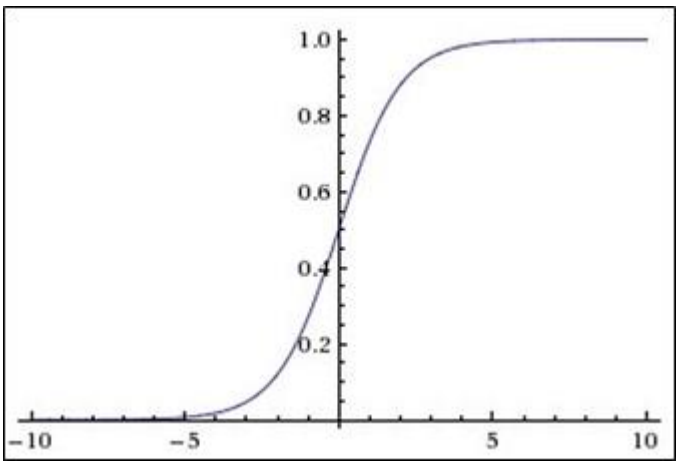

Figure 1.8 Sigmoidal function

An extremely undesirable property of the sigmoid is that when the function is saturated on one side or the other (0 or 1), the gradient in these areas becomes close to zero.

Recall that in the error back propagation process this (local) gradient is multiplied by the total gradient. Therefore, if the local gradient is very small, it effectively nullifies the overall gradient.

As a result, the signal will struggle to pass through the neuron to its weights and recursively to the data. Also, you must be very careful when initializing the weights of sigmoid neurons to prevent saturation. For example, if the initial weights are too high, most neurons will go into a saturation state, which will lead to bad learning for the network.

Activation function ReLU

These are known to be neural networks capable of inducing an arbitrarily complex function. Activation functions are similar to sigmoid or tangential induction, but lead to problems with attenuation or gradients.

However, a simpler version, the rectified linear activation function, can also be used, which is expressed as:

$$f(s) = \max(0, s)\,, \tag{1.7}$$

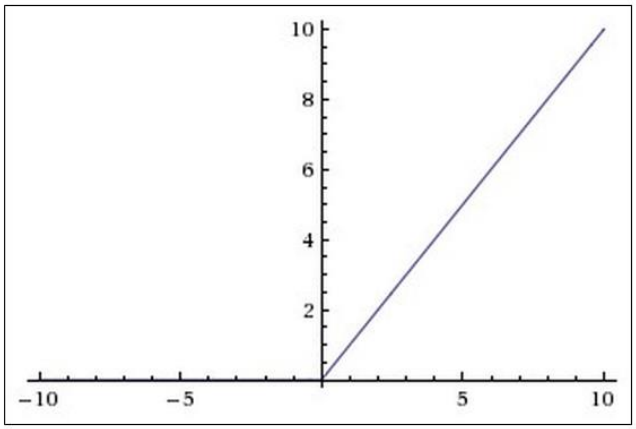

Figure 1.9 ReLU activation function

Benefits of using ReLU:

- its derivative is either one or zero, and therefore no growth or decay of gradients can occur, since multiplying one by the error delta gives us the error delta, but if we used another function, such as the hyperbolic tangent, the error delta could either decrease, increase, or remain the same, that is, the derivative of the hyperbolic tangent returns a number with a different sign and magnitude, which can strongly depend on the decay or spread of the gradient. Moreover, using this function leads to a decimation of the weights
- calculation of the sigmoid requires resource-intensive operations such as exponentiation, whereas ReLU can be realised by a simple threshold transformation of the activation matrix at the zero point;
- cuts off unnecessary parts in the channel when the output is negative

Among the disadvantages are that the ReLU is not always reliable enough and can fail ('die') during training. For example, a large gradient passing through ReLU may cause the weights to update so much that the neuron will never run again. If this happens, from that point on, the gradient going through that neuron will always be zero.

Accordingly, this neuron will be irreversibly turned off. For example, if the learning rate is too high, you may find that up to 40% of ReLUs are "dead" (i.e., never activated). This problem is solved by selecting a suitable learning rate.

1.3 Problem statement of image processing

Research and analysis in the field of neural networks, especially convolutional neural networks for different types of features and basic image processing parameters. To solve the problem of structural-parametric synthesis of convolutional neural networks and to find the most appropriate and most accurate parameters for a hybrid neural network, which will consist of 2 neural networks: a residual neural network and a pyramidal neural network.

Create a hybrid neural network at the software level using state-of-the-art neural network creation methods and tools.

Get better results compared to the state-of-the-art neural networks, explore the field of object identification by image.

Conclusion:

The first part of the article describes information about artificial neural networks, how they work and what they consist of. A convolutional neural network, its topology and the main parameters used to form it are described. The problem statement is also briefly described.

CHAPTER 2. MODERN CONVOLUTIONAL NEURAL NETWORKS

2.1 Classification of modern neural networks

To date, there are 26 types of neural networks. 12 of them are named after their inventors, the rest have names like chaotic, Siamese, oscillatory, adaptive-resonance, etc. In order to somehow systematise existing and future neural networks, attempts are being made in the modern world to divide them according to their classification type.

Classification by input data type:

- analog (real numbers at the input);
- binary (binary numbers at the input);
- figurative (signs, hieroglyphs, symbols at the input) neural networks;

Classification by the type of training:

- supervised learning (the output space of neural network solutions is known);
- unsupervised learning (the output space of solutions is formed only on the basis of input influences; such networks are called self-organizing);
- reinforcement learning (a system of assigning penalties and rewards obtained as a result of the interaction of the ANN with the environment is used);

Classification by the type of the synapse settings:

- networks with fixed connections (the weight coefficients of the neural network are selected immediately, based on the conditions of the problem);
- networks with dynamic connections (these networks have synaptic connections in the learning process);

Classification by the transmission time:

- synchronous networks (the transmission time for each synaptic connection is either zero or a fixed constant);
- asynchronous networks (the transmission time for each connection between the elements is different, but also constant);

Classification by the type of ties:

- feedforward networks (all connections are directed strictly from input neurons to output neurons);
- recurrent networks (the signal from the output neurons or neurons of the hidden layer is partially transmitted back to the inputs of the input layer neurons);
- Hopfield recurrent network (filters input data, returning to a stable state and, thus, allows solving problems of data compression and building associative memory);
- bidirectional networks (there are connections between layers both in the direction from the input layer to the output, and in the opposite direction);

In addition, radial-baseline networks (or RBF-networks), self-organizing maps (in particular, the self-organizing Kohonen map) and networks of other classes, which are not yet fully formed, are used.

Types of modern convolutional neural networks:

LeNet-5

A neural network proposed by Yann LeCun for MNIST dataset handwritten digit recognition.

AlexNet

Winner of ImageNet two thousand twelve with 84.6% accuracy. Implemented using CUDA for improved performance. Consists of two separate parts that loosely communicate with each other, allowing them to run in parallel on different graphical process units with minimal data exchange.

VGG

A family of neural network architectures, including VGG-11, VGG-13, VGG-16 and VGG-19, among others. The 2013 ImageNet winner (VGG-16) is 92.7% accurate. One distinguishing feature is the use of small convolution cores (3x3, as opposed to large 7x7 or 11x11 cores).

GoogLeNet

Also known as inception network, the 2014 ImageNet winner with 93.3% accuracy. Consists mainly of inception modules. Contains a total of 22 layers with configurable parameters (+5 pooling layers).

ResNet

2015 ImageNet winner. The winning network contained over 150 layers and scored 96.43% accuracy.

2.2 The main characteristics, structure, parameters and results of ResNet and Pyramidal neural networks

2.2.1 Residual Neural Network (ResNet)

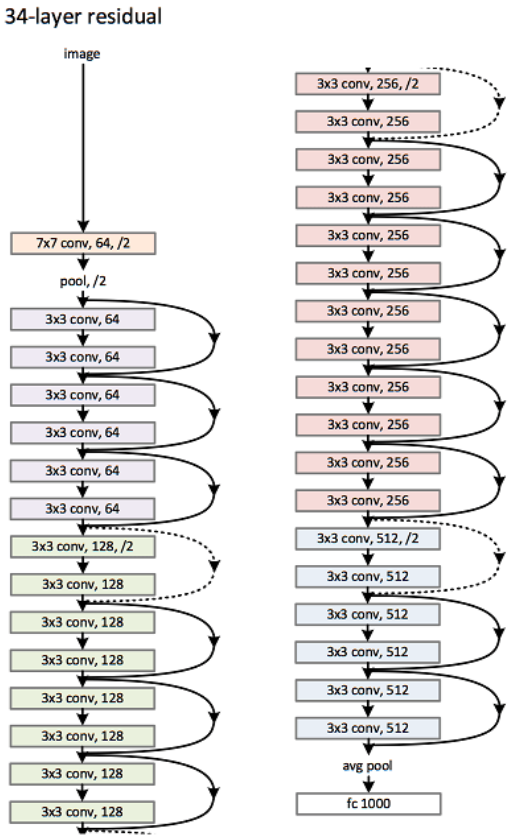

Figure 2.1 Topology of 34-layer ResNet (3.6 billion FLOP)

Dotted quick joins increase the dimension.

ResNet: Based on the simple network described above, a fast connection has been added (Figure 2.1), which turns the network into its residual version. The fast connection of the identity F (x {W} + x) can be used directly when input and output are of the same dimension. When increasing the dimensionality (dashed lines in Figure 2.1), two options are considered:

1) quick join performs ID mapping with additional zeros added to increase the dimensionality. This option does not introduce any additional para-metrics.

2) the fast connection projection in F (x {W} + x) is used for dimension matching (done with 1 × 1 convolutions).

Each ResNet block has two levels of depth (used in smaller networks such as ResNet 18, 34) or 3 levels (ResNet 50, 101, 152). For either option, if fast connections are made over feature maps of two dimensions, they are performed in steps of 2.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2.2 Layer ResNet description

50-layer ResNet: Each 3-layer block is replaced in the 34-layer network by this 3-layer bottleneck, resulting in a 50-layer ResNet (see table above). They use variant 2 to increase the dimensionality. This model has 3.8 billion FLOPs.

ResNet with 101 and 152 layers: ResNet with 101 and 152 layers uses more three-layer blocks (see table above). Even after increasing the depth, the 152-layer ResNet (11.3 billion FLOPs) has less complexity than the VGG-16/19 networks (15.3 / 19.6 billion FLOPs).

Skip connection (shortcut connection)

When a deeper network begins to add up, there is a problem: as the depth of the network increases, the accuracy first increases and then rapidly decreases. The decline in learning accuracy shows that not all networks are easy to optimise.

To overcome this problem, Microsoft introduced a deep "residual" learning structure. Instead of hoping that every few layers correspond directly to the desired baseline representation, they explicitly allow these layers to correspond to the "residual". The formula F (x) + x can be implemented using neural networks with fast connections (feedforward connections, fast connections).

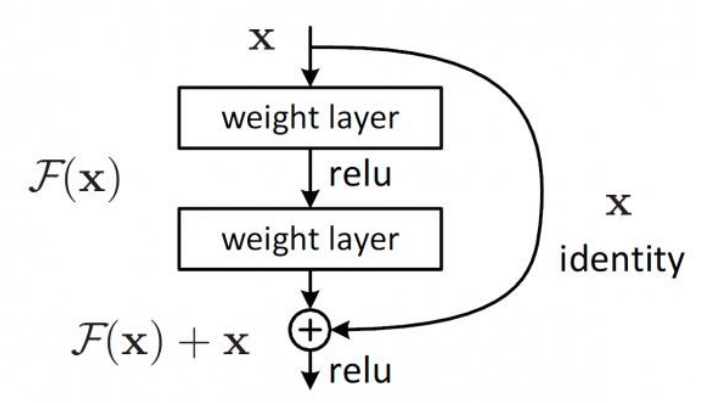

Figure 2.3 Skip (shortcut) connection structure

A shortcut connection skips one or more layers and performs ID mapping. Their outputs are added to the outputs of the stack layers. Many problems can be solved with ResNet, such as

- ResNet is relatively easy to optimise: 'simple' networks (which simply stack layers) show large learning errors when the depth is increased;
- ResNet makes it relatively easy to improve accuracy by increasing depth, which is more difficult to achieve with other networks;

Results

ResNet converges faster than its simple counterpart. Figure 2.4 shows that a deeper ResNet achieves better learning results than shallow networks.

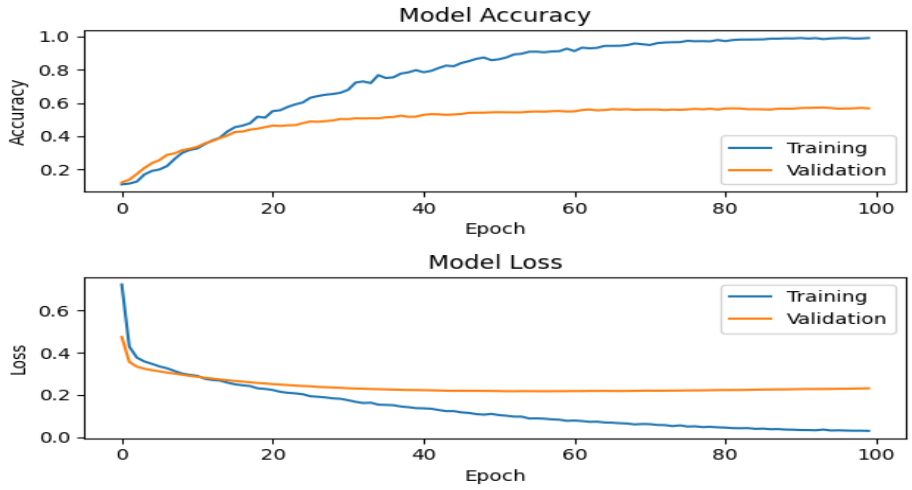

Figure 2.4 Results of ResNet50 on CIFAR-10 dataset

As we can see, the residual neural network performs very well with a high accuracy factor.

During training, the image is resized by randomly sampling its short side by [256,480] to enlarge it. A 224 × 224 crop is randomly selected from the image or its horizontal offset by subtracting the average value for each pixel. The training rate starts at 0.1 and is divided by 10. When the error variation reaches a plateau,

the models are trained to sixty × ten thousand iterations. A weight loss of 0.0001 and a moment of 0.9 is used.

2.2.2 Pyramidal Neural Network

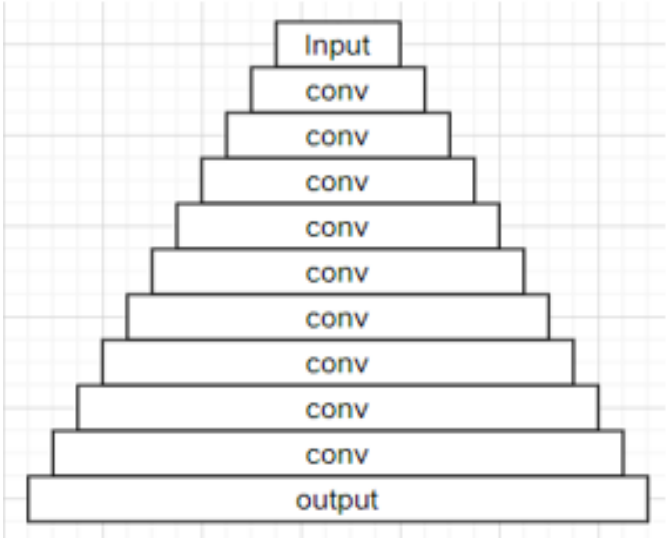

Figure 2.5 Topology of pyramidal neural network (PNN)

The main difference that is immediately apparent is that each successive layer of the pyramidal convolutional neural network increases the number of feature maps to improve identification accuracy.

Most deep convolutional networks use a method in which the size of the feature maps increases along the edges of the image, but the size of the feature map itself does not increase until it reaches the subsampling layer. In this case, the size

of the feature map Dk of the k-th residual unit belonging to the n-th grouping can be described as follows:

$$D_k = \begin{Bmatrix} 16 \\ 16 * 2^{n(k)-2} \end{Bmatrix}, if \begin{matrix} n(k) = 1 \\ n(k) \geq 2 \end{matrix}, \quad (2.1)$$

where n(k) ∈ {1, 2, 3, 4} denotes the index of the group to which the k-th residual unit belongs. Residual units belonging to the same group have the same size of the feature map and the n-th group contains Nn residual units.

The first group has only one convolutional layer which converts the RGB image into multiple feature maps. For the nth group, after passing through Nn residual units, the feature size is halved and the number of dimensions is doubled.

A pyramidal neural network is that the dimensionality of the additive network's feature map gradually increases linearly, while the dimensionality of the multiplicative network increases geometrically. That is, the dimensionality increases slowly in the input layers and increases sharply in the output layers. This method is similar to the original deep network architectures such as VGG or ResNet.

Group	Output size	Building Block
conv 1	32×32	$[3 \times 3, 16]$
conv 2	32×32	$\begin{bmatrix} 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \\ 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \end{bmatrix} \times N_2$
conv 3	16×16	$\begin{bmatrix} 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \\ 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \end{bmatrix} \times N_3$
conv 4	8×8	$\begin{bmatrix} 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \\ 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \end{bmatrix} \times N_4$
avg pool	1×1	$[8 \times 8, 16 + \alpha]$

Figure 2.6 Structure of pyramidal neural network

The building block (convolutional filter stacks with BN- and ReLU-layers) in the block is the core. Obviously, to maximise the capability of the network

architecture, an excellent building block is required. The pyramidal neural network incrementally increases the feature map. Size instead of doubling in one of the residual blocks evenly distributes the load, increasing the function of the map. To illustrate, let's compare PNN with ResNet.

Activation function ReLU:

After using ReLU for this neural network it’s improved performance:

$$x_k^l = ReLU\left(F_{(k,l)}\left(x_{k-1}^l\right) + x_{k-1}^l\right), \tag{2.2}$$

where the ReLU seem to have the function of filtering nonnegative elements.

Results

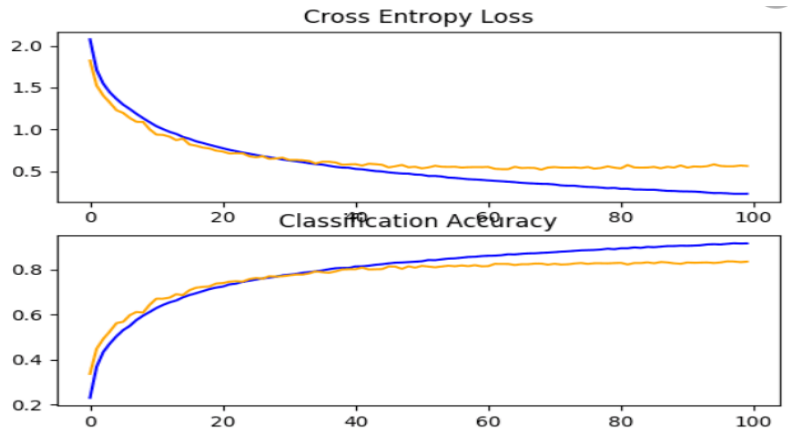

Figure 2.7 Results of Pyramidal neural network on CIFAR-10 dataset

As can be seen from the results on the CIFAR-10 dataset, the pyramidal neural network performed much better than ResNet. Finally, the experimental results show that PNN has excellent generalisation ability, showing better results than other deep models.

2.3 Learning Image Sampling (Datasets)

CIFAR-10

The CIFAR-10 training set was used to train and test the neural networks, which consists of sixty thousand colour images of thirty-two-by-thirty-two pixels of ten classes, six thousand images per class.

This training set also includes fifty thousand training images and ten thousand test images to test the already trained network.

The data set is divided into five training and one test batch, each containing ten thousand images. The test batch contains exactly one thousand images of each class selected at random. The training batches contain the remaining images in random order, but some training batches may contain more images of one class than another. Between training batches contain exactly five thousand images of each class.

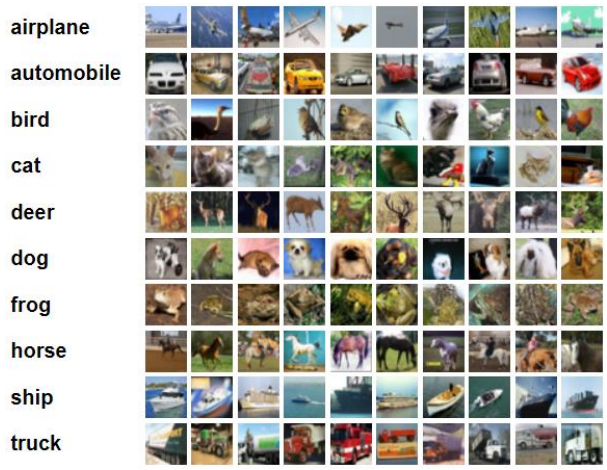

Figure 2.8 Types of images in CIFAR-10

The classes are completely exclusive of each other. There is an intersection of numbers between cars and trucks. "Car" includes sedans, SUVs and the like. "Truck" includes only big trucks. None of them include pick-up trucks.

Each image is chosen so that it only shows the object being studied, so that the convolutional neural network can select only those features that are specific to that type of object.

Computer algorithms for object recognition in photographs are often trained on examples of datasets that have already been selected to begin with. CIFAR-10 is a set of images that can be used to train a neural network for object recognition.

Since the images in CIFAR-10 are of low resolution (thirty-two by thirty-two pixels), this dataset allows users to train their network quickly and get results quickly, try different algorithms and immediately see what works and what doesn't. In general, it is good practice to use the CIFAR-10 image set to test the overall performance of your network, and only then use the larger image sets.

CIFAR-100

This dataset contains all the same images as in CIFAR-10, but additionally contains 90 new image classes for more detailed and advanced training of your network and verification of results. Each image class contains a fairly large number of images, equal to six hundred images. Five hundred of these are set aside for training and one hundred of the six hundred for network verification. All classes in this dataset are divided into 20 superclasses, each image has a label "fi-ne" (the class it belongs to) and a label "rough" (the superclass it belongs to).

In general, it contains the following superclasses: aquatic mammals, fish, flowers, food containers, fruits and vegetables, household appliances, home furry

furniture, insects, large predators, large outdoor man-made objects, large outdoor nature scenes, large omnivores and herbivores, medium sized mammals, non-insect invertebrates, non-insect invertebrates, humans, reptiles, small mammals, trees, type 1 vehicles, type 2 vehicles. And other big amount of different images.

Conclusion:

This chapter describes the types of neural networks, their parameters and features. ResNet and Pyramid neural networks are described with their topologies, main parameters and additional parameters. The results of ResNet and Pyramid neural networks on CIFAR-10 and CIFAR-100 datasets are included. The image data sets used in training and testing of these networks are described.

CHAPTER 3. CREATION OF HYBRID NEURAL NETWORK

3.1 Deep pyramidal residual hybrid neural network structure

3.1.1 Zero-padded skip connection type

ResNets and pre-activation ResNets have been studied several types of labels, such as the identity display label or the projection label. The experimental results showed that the identity mapping label is a more suitable choice than the other labels. Because the identity mapping label has no parameters, it has a lower probability of rearrangement compared to other types of labels; this provides an improved ability to generalize. Moreover, it can pass cleanly through a gradient according to the identity mapping, and therefore provides greater stability during the learning phase.

In the case of a hybrid pyramidal residual neural network, only identity mapping cannot be used for reduction because the dimensionality of the feature map differs for the individual residual units. Therefore, only a null substitution skip or projective reduction can be used for all residual units. However, as already mentioned, projective labeling can make information dissemination difficult and lead to optimization problems, especially for very deep networks.

On the other hand, we have found that the zero-addition label does not lead to overfitting problems, as there are no additional parameters and, surprisingly, it shows significant generalisation power compared to other labels.

We now consider the effect of reducing the null substitution identity mapping to the k-th residual unit belonging to the n-th group with the reformed vector xl k of the l-th feature map:

$$x_l^k = \begin{matrix} F_{(k,l)}(x_{k-1}^l) + x_{k-1}^l, & if\ 1 \le l \le D_{k-1} \\ F_{(k,l)}(x_{k-1}^l), & if\ D_{k-1} < l \le D_k \end{matrix} \tag{3.1}$$

where $F_{(k,l)}$ denotes the l-th residual function of the k- the residual unit and Dk represents the pre-defined channel

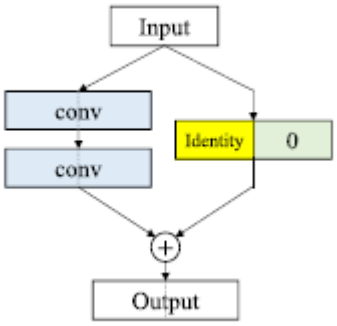

Figure 3.1 Structure of residual block with zero-padded identity-mapping skip

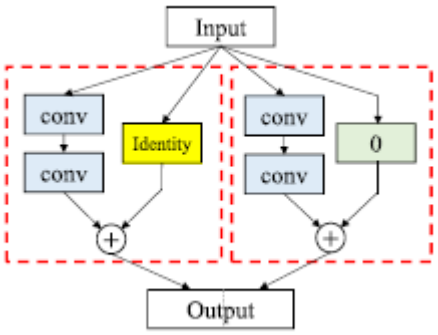

Figure 3.2 Structure of unraveled view of Figure 3.1

is the dimensionality of the k-th residual unit. It follows from equation (3.1) that the null elements of the identity mapping label for increasing dimensionality let xl k contain the outputs of both residual networks and simple networks.

Therefore, we can assume that each null-filled identity mapping label can provide a mixture of residual network and simple network, as shown in (3.1, 3.2).

Moreover, our hybrid pyramidal residual neural network increases the channel dimensionality in each residual unit, and the effect of mixing the residual network and the ordinary network is markedly increased.

3.1.2 New building block

To maximise the capabilities of the network, it is natural to ask the following question: "Can I design a better building block by changing the stacked elements within the building block in a more principled way?" The first types of building blocks were proposed in the original article on ResNets, and another type of building block was proposed later in the article on pre-activated ResNets to answer this question. Moreover, pre-activated ResNets attempted to solve the problem of reverse gradient overflow by redesigning the residual modules; this proved successful in tests. However, although the pre-activation residual module has been found to have empirically improved performance, further investigation into possible combinations has yet to be carried out, leaving potential room for improvement. In the following, we will attempt to answer this question from two perspectives, looking at straightened linear units (Re-LU) and batch normalisation layers (BN).

3.1.3 ReLU function in the building block

The inclusion of ReLUs in the building block residuals blocks are necessary for non-linearity; however, we have found empirically that performance can vary depending on the location and number of ReLUs. This can be discussed

This can be discussed with the original ResNets, for which it has been shown that performance increases as the network deepens; however, if the depth exceeds 1,000 layers, overfitting still occurs, and the result is less accurate than that obtained by smaller ResNets.

First, we note that using ReLU after adding residual units have a negative impact on performance:

$$x_k^l = ReLU\left(F_{(k,l)}\left(x_{k-1}^l\right) + x_{k-1}^l\right), \tag{3.2}$$

where ReLUs are removed after adding to the identity path creation. Consequently, the overall performance increased by a significant amount without excessive fitting, even at depths exceeding 1000 layers. In addition, Shen et al. proposed a weighted residual network architecture that places ReLUs inside the residual unit (instead of placing ReLUs after addition) to create an identical path. ReLU after addition) to create an identity path, and showed.

This structure also does not rebuild, even with a depth of more than 1000 layers. Secondly, we found that using a large number of Re-LUs in blocks of each residual block can have a negative impact on performance. Removing the first ReLU in the blocks of each residual block was found to improve performance over blocks. Experimentally, we found.

That removing the first ReLU in the stack is preferable and that the second ReLU must remain to ensure non-linearity. However, when we remove the first ReLU, the blocks are changed to BN-conv-BN-ReLU-conv, in this case the two convolution layers are separated by a second ReLU, thus guaranteeing non-linearity. Consequently, provided an appropriate number of ReLUs are used to ensure non-linearity of the feature space, the remaining ReLUs can be removed to improve network performance.

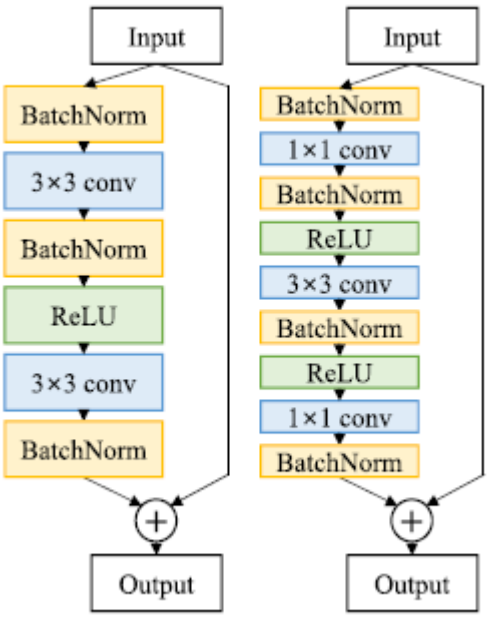

Figure 3.3 Pre-activation function ResNet with removed the first ReLU with a Batch Normalization layer after the final convolutional layer

3.1.4 Batch Normalization layers in the building block

The main role of the BN layer is to normalise activations for rapid convergence and improve performance.

Experimental results from four structures, show that the BN layer can be used to maximise the capabilities of a single residual block. The BN layer performs an affine transformation according to the following equation:

$$y = \gamma x + \beta, \tag{3.3}$$

where γ and β are studied for each activation in the feature maps. We find experimentally that the computed values of and can approach 0. This means that if the computed values of and are close to 0, the corresponding activation is not considered useful.

Weighted ResNets, in which the learned weights are at the end of their building blocks, are also similarly trained to determine whether the corresponding residual block is useful. Thus, the BN layers at the end of each residual block represent a generalized version, among other things, so that decisions can be made as to whether each residual block is useful. In this way, degrees of freedom obtained by involving and from batch normalisation layers can improve the capabilities of the network architecture. The results support the conclusion that adding a batch normalisation layer at the end of each building block and improves performance. Note that the aforementioned network, which removes the first ReLU, is also improved by adding a batch normalization layer after the last convolutional layer.

3.1.5 Training parameters

The hybrid neural network was trained by stochastic gradient descent (SGD) back propagation with Nesterov pulse for 300 epochs on the CIFAR-10 dataset. The initial learning rate is set to 0.1 for CIFAR-10, and decreases by a factor of 0.1 at 150 and 225 epochs, respectively. The filter parameters are initialized with "msra". We use a weight attenuation of 0.0001, an attenuation of 0, a pulse of 0.9 and a batch size of 128.

3.2 Application of multicriteria optimisation for structural-parametric synthesis of neural networks

3.2.1 Pareto-optimality

Optimisation criteria can be narrow, neutral or overriding. In the first case, optimizing one of the criteria leads to improving the others. In the second case, optimizing one criterion does not affect the others.

The case of conflicting (overlapping) criteria is of interest, when an attempt to improve one of them leads to deterioration of the others. In this case, a solution can only be achieved on the basis of a compromise. The mathematical model of trade-offs in optimisation is often based on the concept of Pareto multiplicity, named after the Italian economist who first investigated such models in the early 20th century.

A solution x ∈ D is called an effective solution (non-dominant, Pareto-like, irreducible) if there is no solution in the set of admissible alternatives D which is not worse than x by objective functions, or at least one objective function is strictly better than x. (A Pareto point cannot be polished by the sum of all objective functions).

This definition of an efficient solution can be refined by introducing the notion of Pareto dominance, which is basic in the theory of bagatocriteria choice and more precisely reveals the essence of the formulation 'one solution is better than another'.

Pareto's concept of domination

We will assume that the problem of minimization for N criteria is solved. Then the notion of Pareto-dominant for two any vectors is defined by three possible variants:

1. Solution a domains solution *b*: *a*? ‰»*b* if $f_i(a) < f_i(b) \forall i = \overline{1,N}$.

2. Solution a weakly domains solution *b*: $a \succcurlyeq b$, if $f_i(a) \leq f_i(b) \forall i = \overline{1,N}$.

3. The solutions a and b are non-integer: a ≈ b if $f_i(a) \prec f_i(b) \wedge f_i(a) \succ f_i(b) \forall i = \overline{1,N}$

Thus, if vector x is undominant with respect to D, it is called Pareto-optimal, i.e., if y ∈ D: y x does not exist, then x is Pareto-optimal.

The set of all efficient points is called the Pareto set in the variation space, and their image in the criterion space is called the Pareto front.

From all of the above, the following conclusion can be made. For any admissible point behind the Pareto set, there is a point in the Pareto set which gives for all objective function's values not worse than at this point and at least one objective function is strictly better. Thus, it turns out that the solution to the bagatocriterial optimization problem is appropriate to choose from the Pareto multiplicity, since any other point can obviously be improved by a Pareto point at least by one criterion without degradation of the other criteria.

Mathematically speaking, a given Pareto solution cannot be regarded as superior to another Pareto solution, so once the Pareto multiplicity has been formed, the problem of bagatocriterial optimisation can be regarded mathematically solved.

3.2.2 Approaches to the designation and selection

Target function interchange (NCGA)

In the evolutionary algorithms of this class, instead of combining the target functions to obtain a scalar value of adjacency, the target functions are interchanged during the selection phase. Each time an individual is selected for reproduction, the decision to accept him or her into the intermediate population is made on the basis of different target functions. For example, the intermediate population can be populated with equal portions of different objective functions. Change the order or select by chance the order in which the target functions are viewed.

Aggregation (alignment) of objective functions

In this case, the objective functions are combined into one parameterized objective function as in classical approaches, but the parameters of the objective function do not change during different optimization runs, but vary during a single run.

Some approaches use a valid method, where each individual is evaluated using a given amount combination. Thus, all members of the population are evaluated using different objective functions, and the optimization is done simultaneously from a variety of perspectives.

Use of the concept of Pareto-Dominion (SPEA, SPEA2)

All undomesticated individuals will be ranked 1 and will be removed from the population at the same time. Further, additional undomesticated individuals will be assigned rank 2 and will also be removed from the population. This process continues until all of the individuals have been assigned a similar rank to other members of the population. As a result, an individual's rank determines his or her value of belonging.

Thus, membership is linked to the entire population, whereas in other approaches the membership value of an individual is calculated independently of the other individuals.

3.3.3 Approaches to maintaining population diversity

Underpopulations are formed and maintained, so called niche, in which individuals are obliged to share resources. The more individuals are in the closest proximity to an individual, the more the value of his or her affiliation decreases.

A niche is a group of individuals who have the same affiliation. When there is a laggard-criteria optimization, an approach called dilution of total commitment is often used to reduce the commitment of the individuals who are in the same

niche. This is done to prevent the population from converging to a single solution. Thus, stable subpopulations can be formed, each corresponding to different optima of the function.

Limited Scoping

Two individuals are only allowed to join forces if they are distant from one another. As in the case of a separation of proximity, the distance between two individuals can be determined using different metrics. It should be noted that this approach is not widespread in practice.

Isolation by means of distance control

In this mechanism, each individual is assigned a position in the population. And, in general, two approaches are distinguished: either the spatial structure of a single population is defined, such that spatial niche is created; or the population is divided into distinct subpopulations that only occasionally exchange individuals, with so-called migrations of rare individuals occurring.

Redesignation

In line with this method, identities are composed of active and inactive parts: the first identifies the vector of solutions and the inactive part is not used. Thus, the information can be "stored" in the individual, as the active and inactive parts can change places during the evolution.

Restarting

Another approach used to counteract leapfrogging is to "re-start" either the entire population or a portion of it after a certain period or when the search has stalled (no gain over several generations).

In this case, "restarting" means replacing all or part of the individuals in the population with those generated accidentally, i.e., repeating the initial initiation of the population or part of it.

Consolidation (clustering)

This is where new individuals (soviets) replace similar individuals in the population. In this case, unlike the general evolutionary algorithm, not the entire population participates in selection, recombination and mutation, but only a small fraction of individuals.

3.3.4 Using elitism to maintain population diversity

The use of elitism ensures that the maximum adaptability of the population will not decrease from generation to generation, but can only increase. In contrast to the methods discussed in the previous paragraph, the application of the concept of elitism in the genetic algorithm usually promotes faster convergence of the population. Therefore, this approach is good for optimizing unimodal functions, and when solving multicriteria optimization problems can cause premature convergence.

Accordingly, in contrast to single-criteria optimization, the use of the concept of elitism in multi-criteria optimization is a more complex procedure associated with the concept of dominance.

Instead of one best individual, an elite set is created here, the size of which can be compared with the size of the population itself. This raises questions: What

kind of individuals and how long to keep in the elite set? What individuals and when to return to the population?

In general, there are two main elitist approaches:

1. Directly copy non-dominant individuals to the next population. Sometimes variants with restrictions are used, when only N individuals optimal for a certain function are transferred to the next population, regardless of the values of other criteria.

2. The concept of supporting an external set of individuals (archive) is often used. Thus, in each generation, a certain percentage of the population is filled or replaced by representatives of the archive, who are selected randomly or according to some criterion, for example, after the end of the time during which the individual must remain in the archive.

3.3.5 Basic algorithms of multicriteria optimization

NCGA

A feature of the NCGA algorithm (Neighborhood Cultivation Genetic Algorithm) is the use of a special type of crossover - neighborhood crossover:

The choice of individuals for crossbreeding is not accidental - the two closest individuals are chosen. Due to this approach, obtained after crossing individuals are close to their parents.

Algorithm description:

Step 0. Initialization: Set the parameters of the algorithm: population size - N, the maximum number of iterations of the search - T. An initial population of P0 is generated.

The initial population P0 is copied to the archive A0. The size of the archive is also N. Set the iteration number t = 0.

Step 1. Start a new iteration of the search: t = t +1. We generate a new population Pt = At – 1 (i.e., P1 = A0 = P0 is the initial population).

Step 2. Calculate the value of the fitness function for each individual in this step.

The optimization criterion used to calculate the fitness function varies in turn at each iteration. For example, if the problem includes three criteria for optimization, then the first criterion uses the first criterion to calculate the fitness function, the second iteration uses the second criterion, the third iteration uses the third criterion, the fourth iteration uses the first criterion again, and so on.

Step 3. Ranking: Individuals in the population are sorted by the values of adaptation functions in descending order (from the best individual to the worst).

Step 4. Grouping: individuals are divided into groups, each of which consists of two individuals. These two individuals are selected from the beginning of the list of sorted individuals.

Step 5. Crossing and mutation: in each of the formed groups there is crossing (crossover) and mutation.

This algorithm uses a variant of a single-point crossover - two selected "parents" are cut at a randomly selected point, after which their chromosomes exchange their fragments.

The mutation is associated with a random change in one or more genes on a chromosome. As you can see, hear a randomly selected bit changes its state to the opposite. Two daughter individuals are formed from two parental individuals. Parents are removed from the group.

Step 6. All daughter individuals are grouped into one group, which becomes the new population of Pt.

Step 7. The values of the fitness function for each individual from the new Pt population are calculated. The evaluation criterion used is the same criterion used in the second step of this iteration.

Step 8: Update the archive: The Pt population and the At-1 archive are merged together. From the formed association (which will contain 2N individuals) it is necessary to choose N individuals that form a new At archive. The SPEA2 - Environment Selection algorithm approach is used to make this choice.

Step 9. Check the stop condition: if the stop condition is reached (t> Tmax), the search ends. Otherwise, go to step 1.

SPEA2 algorithm

In comparison with the classical genetic algorithm, the SPEA method has the following features:

• to calculate the value of the fitness function, use the concept of "Pareto-optimality" (Pareto optimality);

• individuals who are not dominant in relation to other members of the population are stored externally in a special external archive;

• to reduce the number of individuals stored in the external archive, clustering is performed, which in turn does not affect the properties of individuals acquired in the search process.

The main improvements of the SPEA2 algorithm in comparison with the classic SPEA:

• improved procedure for determining the suitability of individuals, which for each individual takes into account the number of individuals that dominate the individual and the number of individuals that it dominates

• a new method of reducing the size of the archive, which guarantees the preservation of boundary solutions

Description of the algorithm

Exit: N (population size), NA (archive size), T (maximum number of generations),

Output: A (set of non-dominant individuals).

Step 0. Initialization: Create an initial population P0, according to stage 1 of the scheme of the general evolutionary algorithm, and an empty archive A0 = ∅. Set t = 0.

Step 1. Determination of fitness: Calculate the value of fitness of individuals in Pt and At.

Step 2. Upgrading the archive (Environmental selection): Copy all non-dominant individuals in Pt and At to At + 1.

If the size of At + 1 exceeded NA, then reduce At + 1 using the Environmental selection procedure, otherwise, if the size of At + 1 is less than NA, then fill At + 1 with dominant individuals with Pt and At.

Step 3. Stop: If $t \geq T$ or some other stop criterion is met, then At + 1 is the required set of solutions.

Step 4. Selection: Put At * = ∅ (mating pool) and for s = 1, N perform:

a) randomly select two individuals $i, j \in At + 1$,

b) if F (i) <F (j), then At * = At * + {i}, otherwise At * = At * + {j} (in the case of minimization).

Step 5. Crossbreeding: Crossbreeding is performed similarly to crossbreeding in the classical genetic algorithm.

Step 6. Mutation: Mutation is carried out similarly to mutations in the classical genetic algorithm

Step 7. Save the result of crossbreeding and mutation operations on the set of individuals At * as a new population of Pt + 1. Increase the iteration counter t = t + 1 and go to step 1.

It should be noted that the stage of determining the fitness functions in the scheme of the CREA / SPEA2 method is not carried out as in the classical genetic algorithm, but is a modified procedure based on the concept of Pareto-dominance.

Along with this stage, the clustering mechanism used to reduce the number of individuals in the archive is also carried out according to a special scheme. Algorithms for step-by-step implementation of both procedures are presented below.

Calculation of fitness in the CREA2 method:

To avoid a situation where individuals dominated by the same members of the archive have the same values of fitness functions, the SPEA2 algorithm for each individual takes into account the number of individuals that dominate it and which it dominates.

1. Each individual i ∈ At, is assigned the value S (i) ∈ [0,1), which is called the "strength" of the individual (reflects the adaptability of the non-dominant solution), which is proportional to the number of members of the population j ∈ Pt, for which f (i) ≽ f (j).

Let n be the number of individuals in the archive At that are dominated by the individual and, N be the total number of individuals in the archive. Then the "strength" of the individual will be defined as

$$S(i) = \frac{n}{N+1} \tag{3.4}$$

2. Based on the value of S (i) is calculated "rough" (raw) value of the fitness function R (i) of individual i, which is calculated by summing the "forces" of all individuals j, which dominate or weakly dominate the individual.

$$R(i) = \sum_{j \in P_t + A_t, j \succ i} S(j) \tag{3.5}$$

3. Also to calculate the value of the fitness function is used the value of the density of the location of individuals:

For each individual and the Cartesian distance from it to the rest of the individuals j in the archive and population is calculated. All calculated distances

for this individual are added to the list and sorted in ascending order. The k-th element of such a list for an individual and denote as σki.

Calculate the value of density D (i) for individual i:

$$D(i) = \frac{1}{\sigma_i^k + 2}, k = \sqrt{(N + N_A)} \quad (3.6)$$

4. The final value of the fitness function F (i) for an individual and is defined as F (i) = R (i) + D (i).

The mechanism of clustering in the method SPEA2 (Environmental Selection):

In some cases, the archive of Pareto-optimal solutions may be too large or even contain many decision points. However, from the point of view of the final decision-maker on choosing the optimal solution, too many options to choose from is a disadvantage. Moreover, the number of individuals in the archive affects the behavior of SPEA / SPEA2 - too many non-dominant solutions reduce "competition" between individuals (selection pressure) and can slow down the search. Also, to obtain qualitative results that would cover the entire allowable Pareto region, it is necessary that all individuals be evenly distributed in the space of criteria: if the points in the archive A are unevenly distributed, then during the calculation of fitness functions (and the formation of a new population) towards certain areas of space, which leads to an imbalance of the population and as a consequence - the unrepresentativeness of the resulting solution. Thus, the reduction and redistribution of individuals from the archive (while maintaining the basic characteristics of the individuals found) is desirable or even mandatory.

Unlike the clustering mechanism used in the SPEA algorithm, the corresponding procedure in the SPEA2 algorithm has two features:

• the number of individuals stored in the archive has become constant and does not change over time;

• boundary solutions are not removed from the archive.

Conclusion:

This chapter describes a new idea of convolutional neural networks so called “hybrid”. Also described an increase in the function map measurement gradually to build so-called pyramid structure alongside the ResNets concept.

It was also created a new residual unit that includes a new building block for residual unit with zero filled label. This design leads to significantly improved generalization capabilities. Also described briefly parameters of structural parametric synthesis of neural networks.

CHAPTER 4. SOFTWARE COMPLEX

4.1 Development tools

4.1.1 TensorFlow library

TensorFlow is an open-source machine learning library developed by Google to meet its needs for systems that can build and train neural networks to detect and decode images and correlations, similar to human learning and understanding d. It is currently used for both research and product development by Google, often replacing its closed-source predecessor DistBelief. TensorFlow was originally developed by the Google Brain team for Google's internal use until it was released under the open-source Apache two-point zero license on nine of November two thousand fifteen.

TensorFlow is Google Brain's second-generation open-source machine learning system. While the reference implementation runs on a single device, TensorFlow can run on multiple CPUs and GPUs (including optional CUDA extensions for general purpose graphical process unit computing). TensorFlow is available for 64-bit Linux, macOS, Windows and mobile computing platforms including Android and iOS.

TensorFlow computation is expressed as data flow graph states. The name TensorFlow comes from the operations that such neural networks perform on multidimensional data sets. These multidimensional arrays are called 'tensors'. In June 2016, Jeff Dean of Google stated that TensorFlow is mentioned in 1500 repositories on GitHub, of which only 5 are owned by Google.

Among the applications for which TensorFlow is the basis are automatic image description programs such as DeepDream. On 26 October 2015, Google officially introduced RankBrain, which supports TensorFlow.

RankBrain now handles a significant number of search records, replacing and complementing traditional static algorithms based on search results.

Another application is using FakeApp to seamlessly combine photos and videos to create fake but believable videos, known as Deepfake.

TensorFlow provides a library of off-the-shelf numerical computation algorithms implemented as data flow graphs. The nodes of such graphs implement mathematical operations or input/output points, and the edges of the graph represent multidimensional data sets (tensors) that are passed between nodes. Nodes can be assigned to computing devices and operate asynchronously, processing all suitable tensors together in parallel, allowing simultaneous operation of nodes in a neural network, analogous to the simultaneous activation of neurons in the brain.

Basic elements of TensorFlow tool

Computation graph

Working with TF is built around the construction and execution of a calculation graph. A computation graph is a construct that describes how the computation will be executed. In classical imperative programming, we write code that is executed line by line. In TF, the conventional imperative programming approach is needed only for some auxiliary purposes. The basis of TF is to create a structure which defines the order of calculations. Programs are naturally structured into two parts - making a computation graph and performing computations in the created structures.

In TF, the graph consists of placeholders, variables and operations. From these elements it is possible to assemble a graph in which tensors will be computed.

Tensors are multidimensional arrays; they serve as "fuel" for the graph. A tensor can be a single number, a feature vector from the problem being solved or an image, or a whole batch of object descriptions or an array of images. Instead of a single object, we can pass an array of objects to the graph and an array of responses will be computed for it. TF's work with tensors is similar to how numpy works with arrays, in which functions we can specify the axis of the array relative to which the computation will be performed.

Sessions

Computational graphs are executed in sessions. The session object (tf.Session) hides the context of graph execution - the necessary resources, auxiliary classes, address spaces.

There are two types of sessions - regular sessions, which are implemented in tf.Session, and interactive sessions (tf.InteractiveSession). The difference between the two is that an interactive session is more suitable to run in the console and is immediately identified as the default session. The main effect is that the session object doesn't need to be passed as a parameter to the calculation function. In the examples below, I will assume that the interactive session we declared in the first example is running at the moment, and when I need to refer to the session, I will refer to the sess object.

Artificial intelligence is the process of modelling human intelligence using machines and specialised computer systems. Examples of artificial intelligence include learning, reasoning and self-correction. Applications of artificial intelligence include speech recognition, expert systems, pattern recognition and machine vision.

Machine learning is a branch of artificial intelligence that deals with systems and algorithms that can learn any new data and data patterns.

Machine learning encompasses the machine learning section, and deep learning is part of machine learning. The ability of a programme that follows machine learning concepts is to improve the characteristics of the observed data.

The main motive for transforming data is to improve its knowledge to achieve better results in the future, to provide an output that is close to the desired result for that particular system. Machine learning includes 'pattern recognition', which involves the ability to recognise patterns in data.

It is also provided free of charge. The code is open source. The tasks that this library solves involve information flow and programming of differentiated types. Despite the specific range of tasks, this resource is widely used by specialists - mainly in the field of machine learning. It is also suitable for commercial activities and research work.

Using this library, it is possible to work on

- with projects designed for machine learning;

- with neural network projects;

- naming projects.

It loses out to CNTA and MxNet, for example, in terms of performance. It has a higher entry level for beginners than PyTorch or Keras. Naked Tensorflow is fairly low-level and requires a lot of boilerplate code, and the "discover and run" mode for Tensorflow makes debugging much harder.

This form of library makes it possible to write new algorithms, including those that contain tensor operations in large numbers. As you know, neural networks can be expressed in different ways.

They are often mapped in the form of computational graphs, which is easily implemented using the Tensor Flow resource. In this case it will look like a series of operations applied in a certain sequence.

Python is the fastest growing programming language. This is not surprising given that it is simple, easy to use, free and, relevant to some, allows for shape tasks. Researchers of information are particularly well versed in Python's literate grammar, its teachability, and its ease of combining with various dialects such as C and C++.

All of these positive characteristics, along with the ongoing surge of enthusiasm for AI and artificial intelligence, may help explain the abundance of incredible open-source libraries and frameworks for AI and data science applications. There are libraries that can be used in a myriad of applications. Open-source systems have arisen for most of the above tasks, and the choice of which library to use for a particular enterprise can now be stumped.

Each open-source framework available today has its own advantages and disadvantages when evaluating these components. Moreover, choosing the best system for your needs will depend on exactly what you need to achieve. In any case, whatever your particular circumstances, this guide will help you understand which system is ideal.

Sci-kit Learn is a library with a large group of old-style AI classifiers, such as Support Vector Machines (SVMs), KNN maps, K-Nearest Neighbours (KNN)

classifiers, random forests and reciprocity calculations. It includes choices for supervised and unsupervised learning.

Thus, it is ultimately a successful machine for measurable mapping. It has been based on many other Python libraries such as SciPy, Numpy and Matplotlib, and part of its central computation is additionally composed using Python.

Advantages:

- Very suitable for people who are new to neural networks and computer vision;

- Very suitable for analysing intelligent data, such as predicting small or labelled datasets.

Weaknesses:

- Does not support artificial neural networks;

- No support for graphical process unit computing;

What sets this framework apart is the usual interface for engineers and the abnormal state of reflexivity that allows particularly delicate feet in artificial intelligence to effectively familiarise themselves with the stage without having to manage the low levels of real computation.

It's not difficult to run and learn, and there are some decent and common training exercises to help you understand the calculations when you have to work with them. Be that as it may, Sci-unit Memories has a few obstacles.

First of all, it does not support artificial neural networks. It is also suitable for small enterprises with small datasets and for tasks that are not particularly

computationally intensive. This is because the framework does not support GPU registration. For more trained or "bad to the bone" engineers, it may be perceived as somewhat reticent, as reflection does not take into account tweaking hidden computations.

Ideal for: Hardcore designers who require fast computation on a single graphical process unit.

The system allows productive work with scientific formulas involving multidimensional clusters.

Advantages:

- Efficient handling of compressed images of large and multidimensional data sets;

- Allows for extensive system flexibility to fine-tune the parameters of the main algorithms and algorithms to create new models

Weaknesses:

- Very unclear and harsh learning curve;

- Does not support scaling by more than 1 graphical process unit.

In any case, it continues to be an unusual solution for energetic deep learning enthusiasts.

Although unusual for CNN and image processing, the system is not suitable for recurrent neural network (RNN) and applications involving content, sound and temporal coordinate information. In addition, although there are a large number of

layers ready to execute, the formation of new layers can be monotonous, since for each new layer a full forward, backward and angular update must be characterised.

In addition, the structure offers a medium level of thought - it is in a not normal state, sufficient for quick research, and adapted enough for you to tweak some of the perspectives. This detail may be an advantage for mid-level designers, but it seems rather limiting for non-serious engineers.

Created by Facebook's team (FAIR), genuinely new still rivals TensorFlow, and many foresee it becoming a best option in the near and far future than many other structures.

Advantages:

- Coding is simple, so the learning curve is flatter;

- Supports dynamic graphics, so you can adjust them on the fly;

- Supports graphical process unit acceleration.

Weaknesses:

- Quite new, so has a less community and small amount of resources available over the internet.

For the most part, this is common code for writing code, but the structure is a mixture of high-level and low-level APIs. In fact, it's suitable for both school-youth and deep learning. It includes many pre-prepared models, and when coding you don't have to arrange numbers into 'int', 'short' or 'twofold' information types as in other coding dialects. This makes the representation of activities and possibilities in this system progressively more instinctive than other alternatives.

What makes this framework special, however, is that it offers engineers the ability to use dynamic diagrams. TensorFlow. This framework also provides robust support for increasing graphical process unit speed, so you get both skill and speed. However, the main drawback is that the system is still under development, and you might run into some bugs.

Also, which is indicative of its younger age, the means to populate its official documentation is still very scarce. In any case, as common shows, this will not be a problem for a long time, as more and more developers change and the network evolves gradually but consistently.

TensorFlow is currently being hailed as the best system. In a short period of time, it has become the favourite of some designers, and there is now a regular development of the network and a significant energy of improvement. The structure was developed by the Google Brain group and supports all stages, from Linux to An-droid. It's a non-normal state structure that allows you to run low-level code using helper libraries. Ultimately, it allows you to track the progress of the preparation procedure, while respecting a ton of measurements and not worrying about the vast majority of the different subtleties.

Advantages:

- Quite a lot of flexibility and customisation possibilities;

- Contains several ready-to-use models and a ready-to-use library, allowing you to get started without additional customisation;

- Ability to expand with hardware or software;

- Quite a big online community over the internet.

Weaknesses:

- Only supports NVIDIA graphical process units;

- Very steep learning curve.

TensorFlow engineering and UX differs from other systems in that the nodes in a TensorFlow diagram represent numerical actions and the edges of the diagram represent multidimensional exponents (tensors). These tensors move between nodes, which gives you greater adaptability with respect to creating new nodes, unlike, for example, the design.

In addition, the framework has a large set of models to look at: there are pre-installed packages that allow for voice recognition and machine interpretation, as well as models that allow for recursion, characterisation, neural systems and a combination of different computations. TensorFlow can be used for many internal AI applications.

However, the component that truly boggles the mind is TensorFlow's data logging capabilities. So far, TensorFlow is the most reasonable contender in the field of dispersed data processing. It provides amazing adaptability and gives you the ability to send your computations to different CPUs, GPUs, different servers, mobile phones, and the Google Cloud Machine Learning Engine. You can do this without changing any code - it's truly incredible. However, the main drawback is that the technology currently only supports NVIDIA GPUs. Similarly, in terms of RNN support, it's at least weaker than some other frameworks, and the wait for information absorption can be a bit more extreme than in Sci-unit.

Overall, with Google's solid support and huge online network, TensorFlow is here to stay.

4.1.2 Primary data processing for training samples

In our daily lives, we work with a lot of information, but this information is roughly structured. To provide information as input to AI calculations, we must transform it into important information. This is where information preprocessing comes into play. In other simple words, we can say that we have to process the information before we can give it to the AI computation.

Used these tools to preprocess the information in Python:

Step 1: Import libraries: when using the Python programming language, this will be the first step to convert the information into a particular structure, i.e., preprocessing:

import tensorflow as tf

import tf.keras as keras

from keras.datasets import cifar10

from keras.utils import np_utils

Step 2: Selection of the training sample: after adding the libraries, we have to offload the images into 4 variables so that we can apply preprocessing strategies to this information. We will characterize the accompanying sample of information:

(x_train, y_train), (x_test, y_test) = cifar10.load_data()

Step 3: Apply the preprocessing procedure: In this step we need to classify our images using the built-in method in the tensorflow.keras library and convert the la-bels class into categories. The following area shows the preprocessing information method:

#num_classes=10 (for CIFAR-10), 100 (for CIFAR-100)

y_train = np_utils.to_categorical (y_train, num_classes)

y_test = np_utils.to_categorical (y_test, num_classes)

Step 4: Pixel intensity data must be normalized as follow to get pixels in range from 0 to 1:

x_train = x_train.astype('float32')

x_test = x_test.astype('float32')

x_train /= 255

x_test /= 255

4.2 Results of hybrid convolutional neural network based on ResNet and pyramidal neural network structure

Features of the image preprocessing program are:

– the image from the device comes in the form of a video sequence and not sufficiently contrast;

– the programs automatically increase the contrast to the desired level (by checking the echogenic scale).

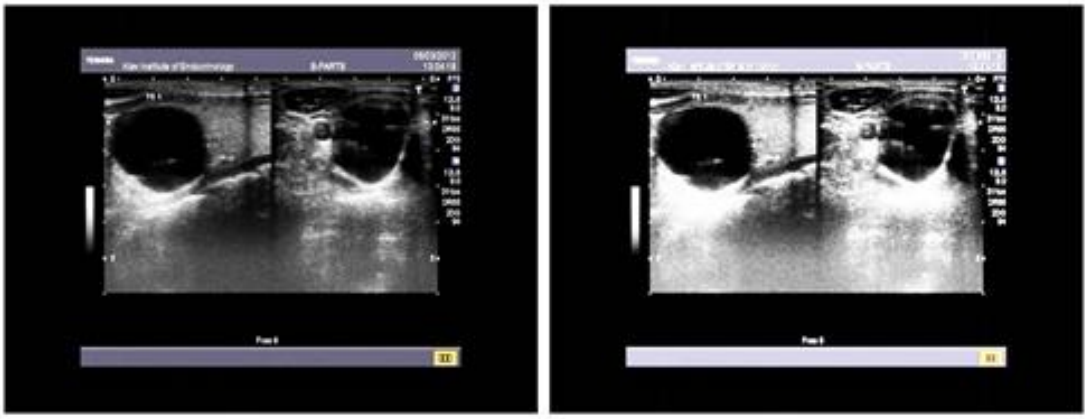

Figure 4.1 The image pre-processing sample

An example of image preprocessing is shown in fig. 4.1.

Contour Highlightening

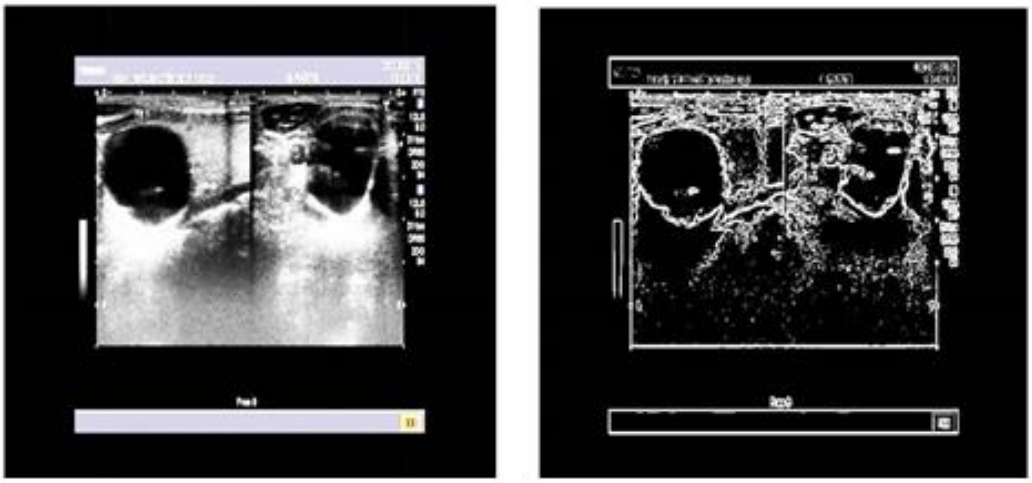

Figure 4.2 The contour highlightening

An example of contour selection is shown in fig. 4.3, where the highlighted contours are shown in the right image.

The suspicious outline is highlighted in red. The problem of discontinuities and irrelevant data inside the circuit is solved.

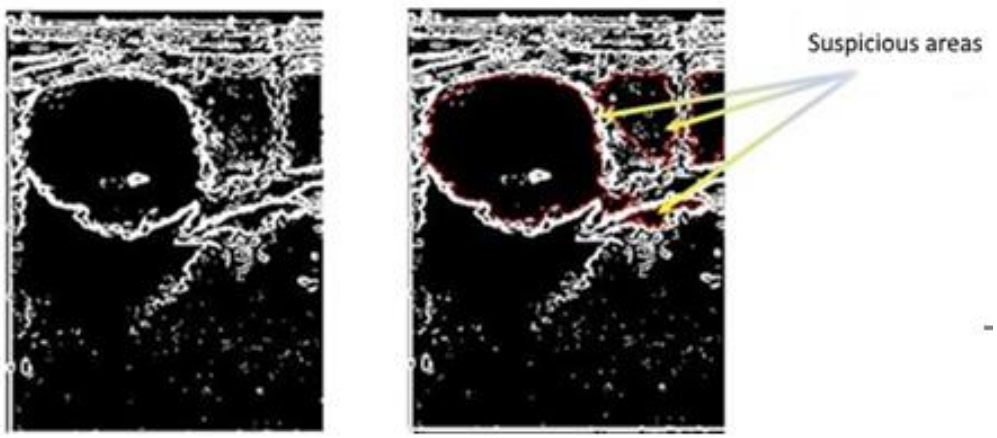

Figure 4.3 Selection of suspicious areas

Obtaining autocorrelation function

Obtaining autocorrelation functions (ACF) corresponding to certain contour shapes is shown in Figure 4.4.

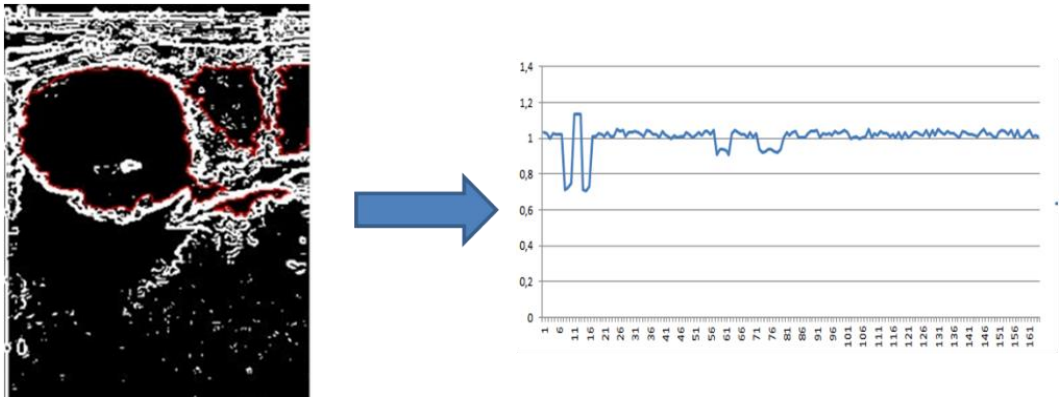

Figure 4.4 Obtaining autocorrelation function

After image processing, four contours were obtained. One of them managed to classify. The neoplasms are isoechogenic, as the middle tone is in the appropriate place on the echogenicity scale. The echostructure is heterogeneous.

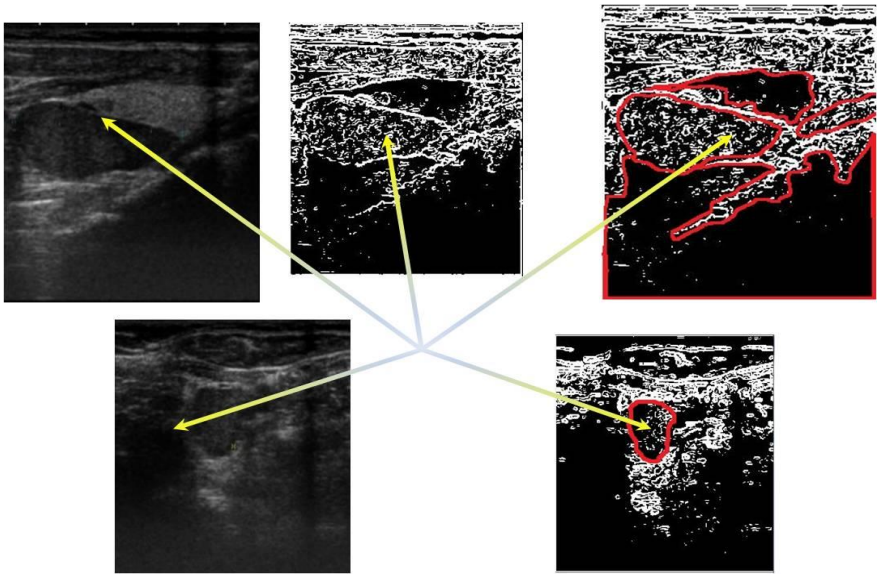

Figure 4.5 Outlines of patient ultrasound scan results

Based on the correspondence of a certain pattern of ACF, it was concluded that the "wrong" form of neoplasm shown in Figure 4.6.

On the basis of the analysis of the outline of the contour, it was concluded that the contour is without clear boundaries. Using the second projection and the Brunn method, the volume of the tumor was determined: 11.03.

Additionally, the following data were entered into the neural network: age: 58 years; female. Neural Network Output Diagnosis: Papillary Carcinoma. The results of ultrasound of the patient BV are shown in Fig. 4.6, 4.7. After image processing, 6 contours were obtained. One of them managed to classify

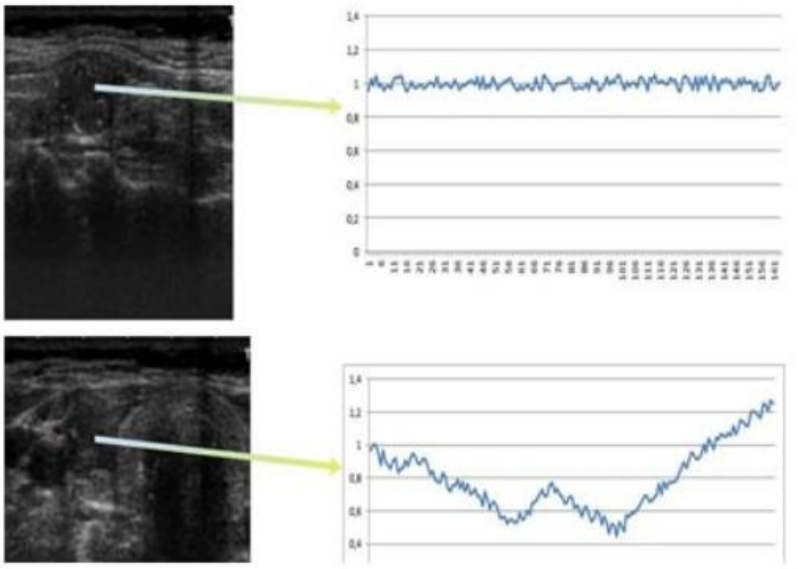

Figure 4.6 The conclusion about the “wrong” form of neoplasm of the patient’s ultrasound scan results

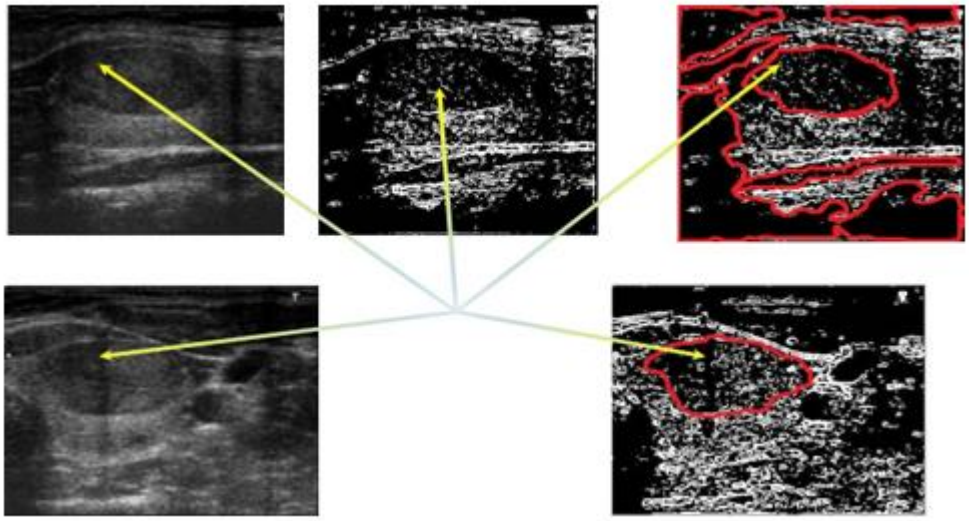

Figure 4.7 Outlines of patience ultrasound scan results

The neoplasm is hypoechoic because the middle tone is in the appropriate place on the echogenicity scale. The echostructure is homogeneous.

Based on the correspondence of a certain pattern of ACF, it was concluded that the "correct" form of neoplasm shown in Fig. 4.8.

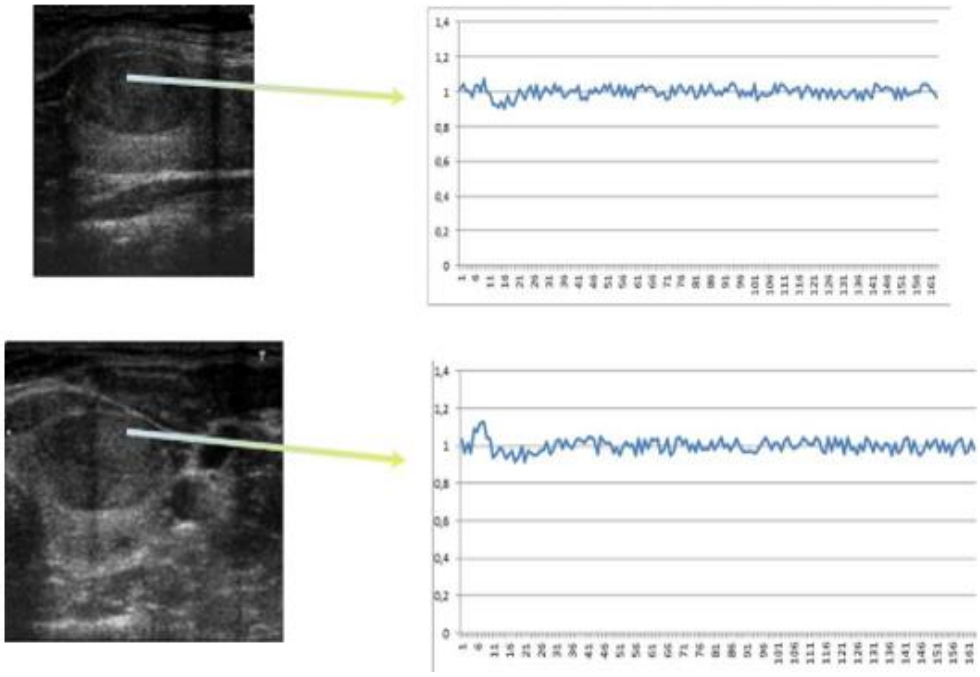

Figure 4.8 The conclusion about the “wrong” form of neoplasm results of ultrasound scan of the patient

Additionally, the following data were entered into the neural network: age: 34 years; female. Neural Network Output Diagnosis: A-Cell Tumor (Follicular Adenocarcinoma).

The results of ultrasound scan of the patient are shown in Fig. 4.9.

After image processing, four contours were obtained. One of them managed to classify.

The neoplasm is hypoechoic because the middle tone is in the appropriate place on the echogenicity scale. Echostructure is heterogeneous, there are hydrophilic formations.

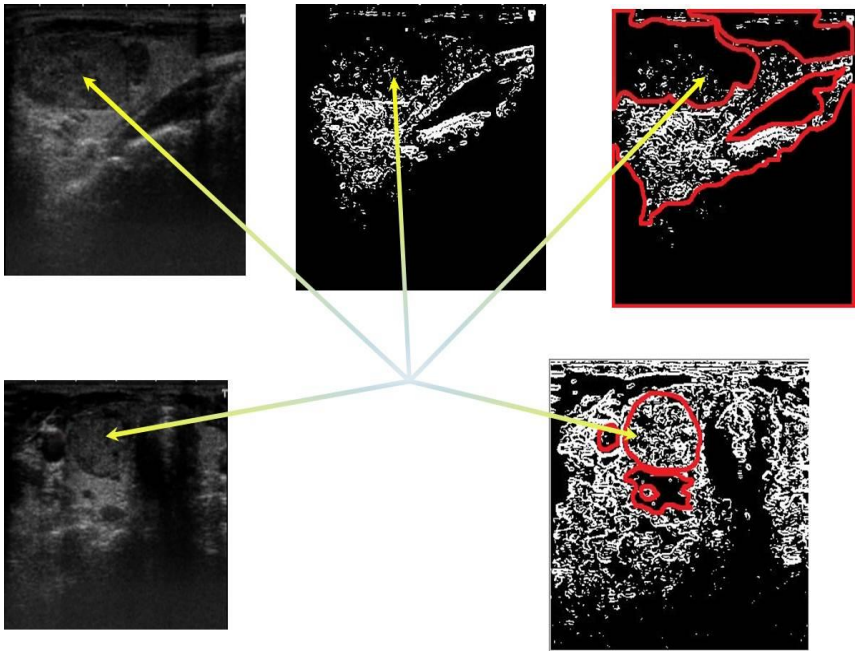

Figure 4.9 Outlines of the results of ultrasound scan of the patient

On the basis of the analysis of the outline, it was concluded that the outline has clear hydrophilic boundaries. Using the second projection and the Brunn method, the volume of tumors was determined: 43.57.

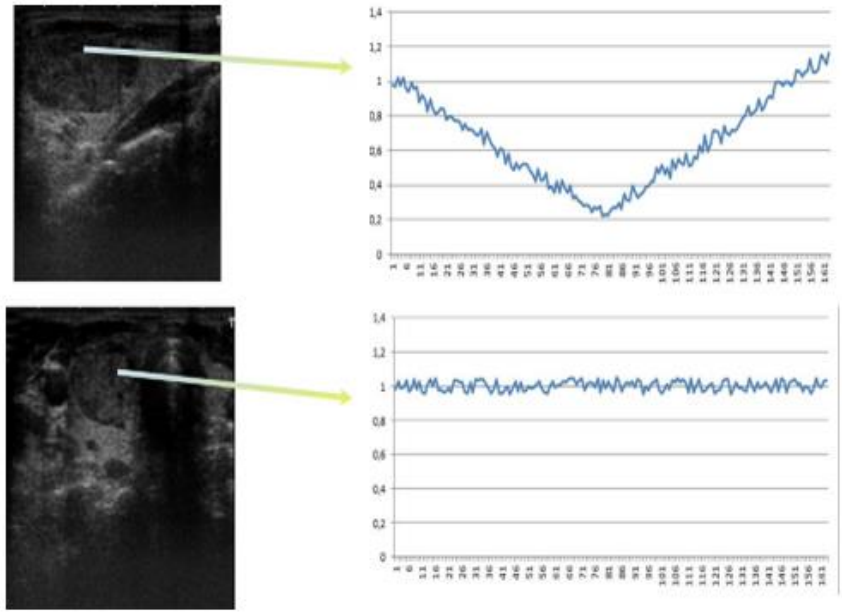

Figure 4.10 The conclusion about the “correct” form of neoplasm results of ultrasound scan of the patient

Additionally, the following data were entered into the neural network: age: 53 years; gender: male. Diagnosis at the output of the neural network: nodal goiter.

The results of ultrasound scan of the patient are shown in Fig. 4.11.

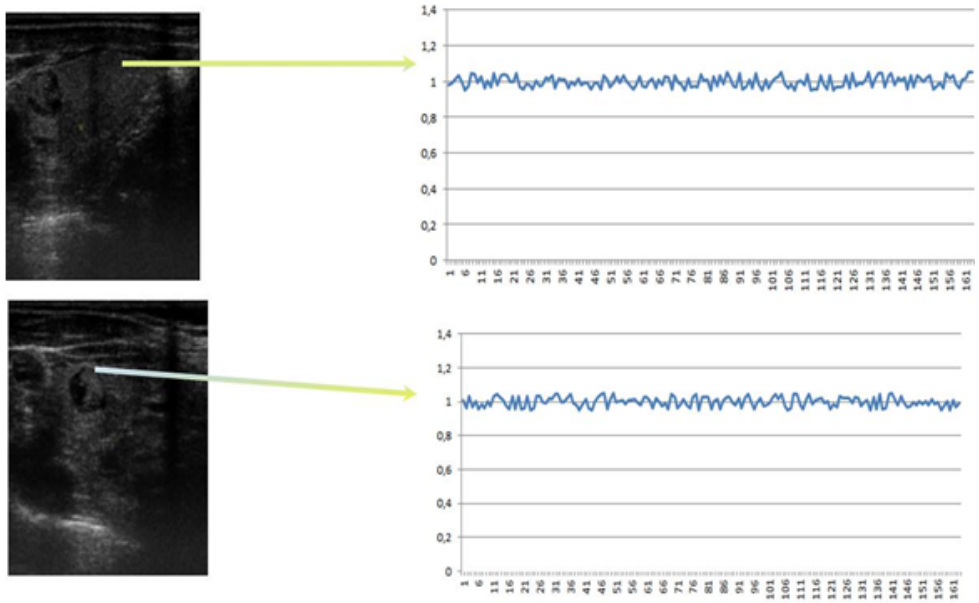

Figure 4.11 Another results of the “right” new creation of the ultrasonic examination of the patient

The neoplasms are isoechogenic because the middle tone is in the appropriate place on the echogenicity scale. The echostructure is heterogeneous. There are massive cystic lesions.

Based on the correspondence of a certain pattern of ACF, it was concluded that the "correct" form of neoplasm shown in Fig. 4.12.

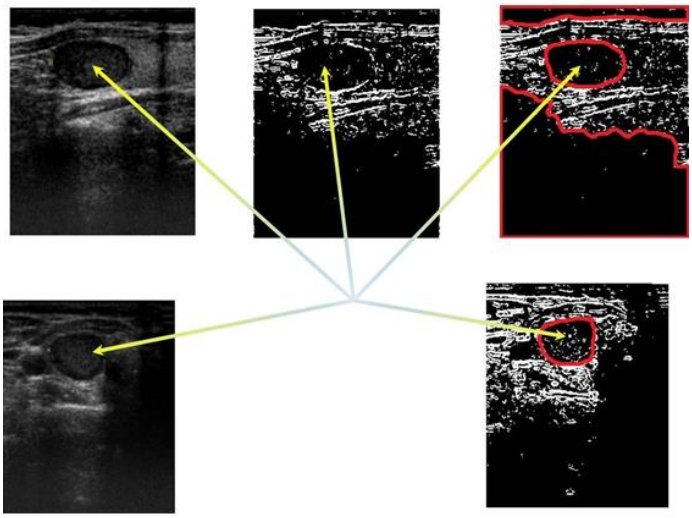

Figure 4.12 Contours results of ultrasound of the patience

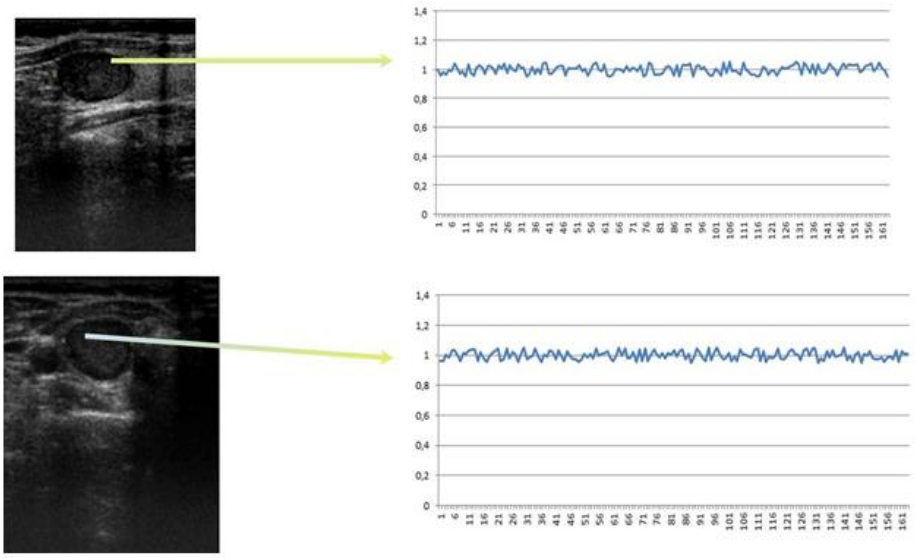

Figure 4.13 Conclusion about the “right” form of new creation of the ultrasound examination of the patient

On the front side of the analysis around the contour of the boules of the crusts of the gallows, the contour is with clear hydrophilic spaces. For another alternative projection and the Brunn method, a new appraisal oath was made: 13.69.

Additional in the neural measure I have boules introduced so far: age: 44 rocks; become: Female. Diagnosis of neuronal outcome: an adenomatous nodular goiter (actually cystadenoma). Results of ultrasound of the patient AV image config.

After processing the image, three contours were shaded. One of them went into classify.

Neoplasms are isoechogenic, so as the middle tone can be found at the mainstream on the exogenous scale. The echo structure is not uniform. The presence of masculine approval.

Based on the correspondence of a certain pattern of ACF, the conclusion was made about the "correct" form of neoplasm.

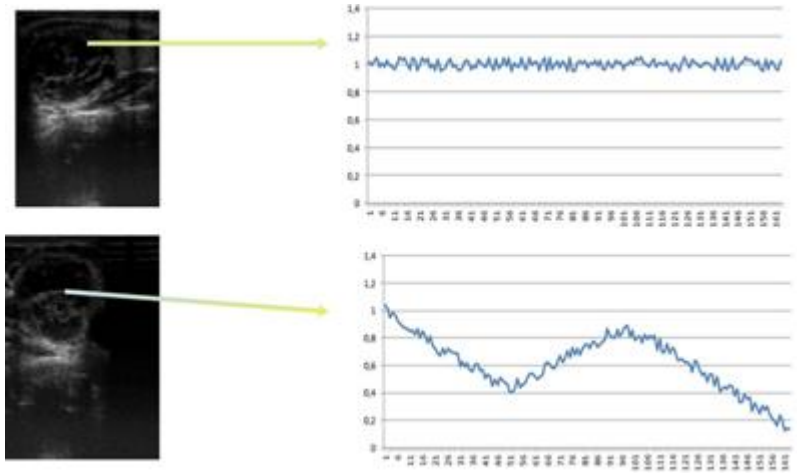

Figure 4.14 Another conclusion of the creation of the “right” new creation results of ultrasonic examination of the patient

On the basis of the analysis of the outline of the contour, it was concluded that the contour with clear hydrophilic boundaries. Using the second projection and the Brunn method, the volume of neoplasm was determined: 15.51.

Additionally, the following data were entered into the neural network: age: 13 years; female. Diagnosis at the output of the neural network: nodal goiter.

In accordance with the structural diagram of the intellectual diagnostic system after isolation of anomalous areas as a result of video processing of ultrasound images, it is necessary to evaluate their parameters, namely: type of rim of tumor, inclusion, structure of tumor, size and echogenicity of tumor. The type of rim, the structure of the neoplasm and the presence of inclusions are determined by constructing autocorrelation functions of fragments of the anomalous region with subsequent comparison with the reference correlation functions corresponding to the possible types of these parameters, for example, for inclusions these are the following options: no, point inclusions, linear inclusions, tail comets ». The

echogenicity of the neoplasm is determined by the use of the Piton Image Library (PIL).

Conclusion:

This section describes how data is processed before neural network training, and provides real results of neural network operation on images of malignant tumors in the human body.

CONCLUSION

In this work with application of modern technical approaches to data processing the GPU-basic method of functioning of a neural network was chosen. To compare the methods of image recognition, a graphically based method of calculation and verification of the original data of the convolutional neural network was chosen. Compared with the results in this work, an analysis of modern literature and scientific papers on image recognition was performed and the basic structure of the neural network was obtained.

Depending on the results of the research, it follows that reducing the core filters will reduce the processing power, as well as increase the number of processed frames per second. Research has revealed that all relevant programming libraries for the formation of a neural network cannot allow it to be easily used in more complex systems, where the neural network is generally just a structural part.

A new complex deep structural neural image identification network is proposed and formed using the recently proposed structure of a convolutional neural network together with the use of residual blocks and a pyramidal architecture. As a result, compared to the tests, we get a more accurate image recognition system.

REFERENCES

1. “Информационные технологии и нейронные сети в профессиональной деятельности” URL: https://studref.com/336179/informatika/neyronnye_seti
2. “Информационные технологии и нейронные сети в науке и образовании” URL: https://studref.com/434192/pedagogika/neyronnye_seti
3. Aleix M.Matinez and Avinash C. Kak, " PCA versus LDA"IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol.23,No. 2, February 2001.
4. The CIFAR-10 dataset URL: https://www.cs.toronto.edu/~kriz/cifar.htmlMcKenna
5. “ResNet (34, 50, 101): ‘остаточные’ CNN для классификации изображений” URL: https://neurohive.io/ru/vidy-nejrosetej/resnet-34-50-101/
6. Patterson W. D., “Artificial Neural Network: Theory and Applications”, Prentice Hall, 1999.
7. “Искусственные нейронные сети (ИНС)” URL: https://www.it.ua/ru/knowledge-base/technology-innovation/iskusstvennye-nejronnye-seti-ins
8. Lin Sadrina .W Gao Yang, Liu Ray K.J “Template matching for image prediction: A game-theoretical approach”, IEEE ICASSP, 2012.
9. “Нейронные сети в промышленности и информационных технологиях” URL: https://u.to/mxFjGw
10. P.N.Belhumeur, J.P.Hespanha and D.J. Kriegman, "Eigenfaces Vs Fisherfaces : Recognition using Class specie linear projection" IEEE Trans. Pattern Anal . Machine Intel, Vol 19, pp 711-720, July1997.
11. “A comparison of training algorithms when training a Convolutional Neural Network for classifying road signs” URL: https://kth.diva-portal.org/smash/get/diva2:1336299/FULLTEXT01.pdf

12. Smeets Dirk, Claes Peter, HermansJeroen , Vandermeulen Dirk, Suetens Paul, “A comparative Study of 3-D Face recognition under expression variations”, IEEE transactions on system , man, andCybernetics,vol-42,no-5,2012.

13. “Разработка системы распознавания визуальных образов в потоке данных” URL: https://kubstu.ru/data/fdlist/FDD0445.pdf Hurshudov Artem Aleksandrovich, 05.13.01

14. “Learning Multiple Layers of Features from Tiny Images”, Alex Krizhevsky, April 8, 2009 URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

15. “A Beginner's Guide To Understanding Convolutional Neural Networks”, Adit Deshpande, URL: https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

16. “Свёрточные нейронные сети” URL: https://u.to/WRNjGw

17. Scott E., “Computer Vision and Image Processing: A Practical Approach using CVItools”, Prentice-Hall, 1998.

18. J,-S, R. Jang, C, -T. Sun & E. Mizutani, ”Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence”, Prentice Hall, 1997.