МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра авіаційних комп'ютерно-інтегрованих комплексів

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

Синєглазов В.М.

" ____ " _____2021.

ДИПЛОМНАРОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ

"БАКАЛАВР"

Тема: Гібридна згорткова мережа з використанням модулів стиснення-та-збудження та уваги

Виконав: **Павленко О. В.**

Керівник: **к.т.н.** **Синєглазов В.М.**

Нормоконтролер: к.т.н. **Туріцин М. Ф.**

Київ – 2021

EDUCATION AND SCIENCE MINISTRY OF UKRAINE
NATIONAL AVIATION UNIVERSITY
COMPUTER-INTEGRATED COMPLEXES DEPARTMENT

ADMIT TO DEFENSE
Head of department
Viktor M. Sineglazov
“____” _____ 2021.

BACHELOR WORK
(EXPLANATORY NOTES)

Topic: Hybrid convolutional network using Squeeze-and-Excitation and Attention Modules

Done by: **Pavlenko O.V.**
Supervised by: **Syneglazov V,M.**
Normcontrolled by: **Tupitsyn M. F.**

Kyiv 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра авіаційних комп'ютерно-інтегрованих комплексів

Освітній ступінь бакалавр

Спеціальність: 151 " Автоматизація та комп'ютерно-інтегровані технології"

ЗАТВЕРДЖУЮ

Завідувач кафедри

Синєглазов В.М.

" ____ " _____2021 р.

ЗАВДАННЯ

Павленко Олександра Володимировича

1. Тема проекту (роботи): " Гібридна згорткова мережа з використанням модулів стиснення-та-збудження та уваги "

2. Термін виконання проекту (роботи): з 5.01.2021 р. до 12.06.2021 р.

3. Вихідні данні до проекту (роботи): Згорткові штучні нейронні мережі, механізми уваги для нейронних мереж, алгоритми обробки зображень та створення штучних нейронних мереж в середовищі Matlab

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): 1. Аналіз, обґрунтування вибору методів ідентифікації об'єкту; 2. Огляд існуючих методів тай нейронних мереж.. 3. Розробка схеми штучної нейронної мережі на основі згорткової нейронної мережі; 5. Налаштування та навчання штучної нейронної мережі.

5. Перелік обов'язкового графічного матеріалу: 1.Структурна схема гібридної штучної нейронної мережі; 2. Код штучної нейронної мережі в середовищі Matlab; 3. Графік оцінки точності класифікації зображень нейронною мережею; 4. Приклад результатів навчання штучної нейронної мережі

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Отримання завдання	20.02.2021 – 5.03.2021	
2	Формування мети та основних завдань дослідження	6.03.2021 – 12.04.2021	
3	Аналіз існуючих методів	13.04.2021 – 25.04.2021	
4	Теоретичний розгляд рішення задачі	26.04.2021 – 16.05.2021	
5	Розробка структури штучної нейронної мережі	17.05.2021 – 28.05.2021	
6	Розробка штучної нейронної мережі за допомогою програмних засобів та оцінка результатів	29.05.2021 – 06.06.2021	
7	Оформлення пояснювальної записки	07.06.2021 – 09.06.2021	
8	Підготовка презентації та роздаткового матеріалу	10.06.2021 – 12.06.2021	

7. Дата видачі завдання: “10” травня 2021 р.

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

Завдання прийняв до виконання _____ Павленко О.В.
(підпис випускника) (П.І.Б.)

NATIONAL AVIATION UNIVERSITY

Faculty of aeronavigation, electronics and telecommunications

Department of Aviation Computer Integrated Complexes

Educational level bachelor

Specialty: 151 "Automation and computer-integrated technologies"

APPROVED

Head of Department

Sineglazov V. M.

"____" _____2021

TASK

For the student's thesis

Pavlenko Oleksandr Volodymyrovych

1. Theme of the project (work): " Hybrid convolutional network using Squeeze-and-Excitation and Attention Modules "

2. Term of project implementation (works): from 5.01.2021 to 12.06.2021

3. Initial data to the project (work): Convolutional artificial neural networks, mechanisms of attention for neural networks, algorithms of image processing and creation of artificial neural networks in the Matlab environment

4. Contents of the explanatory note (list of issues to be developed): 1. Analysis, justification of the choice of methods of object identification; 2. Review of existing methods of neural networks. 3. Development of an artificial neural network scheme based on a convolutional neural network; 5. Setting up and training an artificial neural network.

5. The list of obligatory graphic material: 1. Block diagram of a hybrid artificial neural network; 2. The code of the artificial neural network in the Matlab environment; 3. The schedule of an estimation of accuracy of

classification of images by a neural network; 4. Example of learning results of an artificial neural network.

1. 6. Planned schedule:

№	Task	Execution term	Execution mark
1	Task	20.02.2021 – 5.03.2021	
2	Purpose formation and describing the main research tasks	6.03.2021 – 12.04.2021	
3	Analysis of existing methods	13.04.2021 – 25.04.2021	
4	Analysis of existing systems	26.04.2021 – 16.05.2021	
5	Development of the structure of an artificial neural network	17.05.2021 – 28.05.2021	
6	Development of an artificial neural network using software and evaluation of results	29.05.2021 – 06.06.2021	
7	Making an explanatory note	07.06.2021 – 09.06.2021	
8	Preparation of presentation and handouts	10.06.2021 – 12.06.2021	

1. **Date of task receiving:** "10" May 2021

Diploma thesis supervisor _____ Head of ACIK, Doc. of Sc.

(Eng), Prof. Syneglazov V,M. (signature)

Issued task accepted _____ Pavlenko O,V.

(signature)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Гібридна згорткова мережа з використанням модулів стиснення-та-збудження та уваги»: с., рис., табл., літературних джерела.

Об’єктом дослідження є способи організації засобів та компонентів нейромережевої класифікації об’єктів на зображенні з використанням комп’ютерних процессорів.

Предметом роботи є методи та алгоритми реалізації гібридних згоркових нейронних мереж.

Метою роботи є розглялнути існуючі способи обробки та класифікації зображень з використанням гібридної згорткової нейронної мережі власної розробки, провести практичне експериментальне моделювання на наборі данних, використовуючи комп’ютерний процесор (англ. central processing unit, CPU).

Предметом дослідженння є аналіз та розробка структури штучної нейронної мережі з використанням механізмів уваги

Методи дослідження – теорія штучних нейронних мереж, теорія оборобки зображень та комп’ютерного зору.

Ключові слова: комп’ютерний зір, рекурентна нейронна мережа, гібридна згорткова нейронна мережа, класифікація об’єктів, CPU

ABSTRACT

Explanatory note to the thesis "Hybrid convolutional network using Squeeze-and-Excitation and Attention Modules": p., figures, tables, literary resources

The object of the work is the ways of organizing the means and components of neural network classification of objects in the image using computer processors.

The subject of the work are methods and algorithms of realization of hybrid convolutional neural networks.

The aim of the work is to consider the methods of image processing and classification using a hybrid convolutional neural network of our own development, to offer an architecture that will best solve the problem - in terms of performance, computational volume and recognition. Perform practical experimental modeling on a data set using a central processing unit (CPU).

Research methods - the theory of artificial neural networks, the theory of image processing and computer vision.

Keywords: computer vision, recurrent neural network, hybrid convolutional neural network, object classification, CPU

CONTENT

Glossary

FCL - fully-connected layer

CNN - convolutional neural network

AI – artificial intelligence

RGB – red-green-blue

MP - max pooling

AP - average pooling

CPU – central processing unit

CL – convolution layer

RELU – rectified linear unit

RCNN – recurrent convolutional neural network

SEB – squeeze-and-excitation block

ANN – artificial neural network

Chapter 1

Problem statement

1.1 Actuality of image processing

The artificial neural networks for image classification purposes are impotant part of today's life. There a lot spheres of use, than utilizes this tools – from internet image search databases and to automated medical diagnostic systems.

The main purpose of this work is to research and find the optimal approach for application the artificial neural network to solve the image classification problem with restricted machine resources, and set the structure and parameters of such recognition system.

Main problem of this work is to discover the approaches to classify images from given dataset. Current problem could be solved in different ways - today, there a lot fo of already realized and trained neural networks, for example:

- squeezenet
- googlenet
- inceptionv3
- densenet201
- mobilenetv2
- resnet18
- resnet50
- resnet101
- shufflenet
- nasnetmobile
- nasnetlarge

Image recognition has become widespread in various spheres of human activity, and continues to expand and deepen its practical importance. [1] An example of such an application in term of numbers can be recently Marketsand Market's analysis (see fig. 1.1)

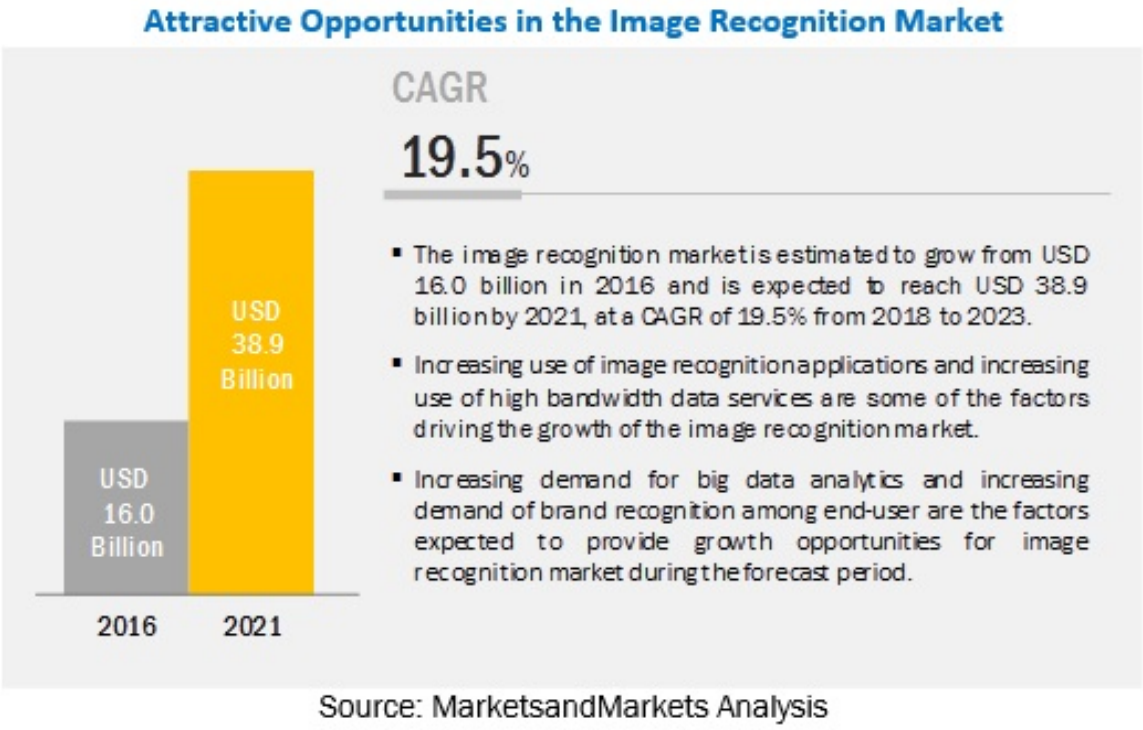

Fig. 1.1. Grown of image recognition market volume

Automotive industry: Not only classic car manufacturers are working on self-driving cars, but tech giants are taking over the production of such cars. The reasons for these cars include various reasons, such as fewer road accidents, adherence to traffic regulations, and more. At CES last year, Cisco announced a partnership with traditional car company, Hyundai. to provide wireless update capability for autonomous vehicles. Tesla's widely produced vehicles have an integrated control system, which, in addition to the autopilot functions, is capable of performing emergency prevention tasks, and the most important component of this system is optical systems using obstacle recognition systems.

Gaming: Image recognition has changed the dimension of the gaming industry. In the current scenario, advanced image recognition technology allows the player to use their real location as a battlefield for virtual adventures. An important component that reduces development time and improves product quality is the use of facial animation capture systems, whose use in high-budget game projects is now the de facto standard, and provides accelerated creation of

high-quality animations of human facial expressions - a similar technology came from the world of cinema, where, for example, was used in the creation of the Lord of the Rings trilogy.

Healthcare Industry: Image recognition technology is providing tremendous help to the healthcare industry. It helps make meaningful changes to the patient's journey. Robot-assisted microsurgical procedures in healthcare use computer vision and image recognition techniques. The use of this technique has expanded over the past decade thanks to advances in machine learning and artificial intelligence. Real-time emotion detection can also be used to detect patients' emotions in order to analyze how they feel during hospitalization or before discharge. Another important area is the development of auxiliary diagnostic systems that accelerate the detection of pathologies and changes when using X-ray, MRI and ultrasound research methods.

Unified Reality: This is a combination of virtual reality and augmented reality. To overcome the shortcomings of VR and AR, a unified reality is being created that provides a more dynamic and natural experience of the virtual world. For example, Intel Project Alloy or Microsoft Windows Holographic Shell, a wireless headset that allows you to bring real objects into the virtual world using 3D cameras. Such programs in the future will simplify the design and engineering of a wide range of things.

Retail. There is a huge demand in retail for this revolutionary technique. Trax is designed for retail and consumer products that maintain a database of SKUs, processing 40,000 images per hour. The quality and price of a product can be compared using image recognition techniques. Tesco, Bangalore, works with biometric and image recognition technologies for numerous mechanisms covering

recognition, geometry, quality assessment to create store technologies, product assortment planning and more.

Security Industry: This evolving technology plays a vital role in the security industry. Whether it's an office, smartphone, bank or home, security measures are an integral part of every platform. Many security devices have been developed, including drones, CCTV cameras, biometric facial recognition devices, and more.CES 2019 showcased SimCam home security cameras and home automation cameras equipped with artificial intelligence for facial recognition, pet monitoring and more. by teaching location. Netatmo, a smart indoor camera, has a function that starts recording video only when the system detects any unknown persons.

Social Media Platforms: Image recognition works pretty well in this area as it has become easier for marketers to find visuals on social media. Image recognition tools can search social media for images and compare them against vast libraries to find the images you want at unprecedented speed and scale. As a result, it offers companies tremendous customer service benefits. In 2016, Facebook added a feature for visually impaired people, combining facial recognition and automatic text technologies to create an accurate description of the content of a photo, as well as a description of who exactly is in the photo without a tag. In addition, image recognition systems based on neural networks actually bear the main load in assessing downloadable content, allowing you to cut off most of the illegal or malicious content even at download.

Visual search engines: This technology uses image recognition to provide users with the best possible search results. Google and Bing are the oldest players on this platform. There are other visual search engines that perform the same functions as the larger players. For example, Picsearch is a traditional visual

search engine that offers a huge archive of images. UK retailer Marks and Spencer recently launched a visual search style search engine in its store.

1.2. Problem statement

The main purpose of this work is to consider the methods of image processing and classification using a hybrid convolutional neural network of our own development, to offer an architecture that will best solve the problem - in terms of performance, computational volume and recognition.

Then, to perform practical experimental modeling on a data set using a central processing unit (CPU). Paper presents a practical example of way to build such a system and its configuration for the correct classification depending on the category of the object depicted in the photo. Today in modern convolutional neural networks for independent processing of graphic data there is a problem of inaccuracy in classification and complexity of structure, the importance of this problem only increases over time due to the spread of increasingly complex and computing neural networks.

To increase the accuracy of the results, a system has been developed that includes a structurally modified residual convolutional neural network. A comparison was also made with the existing solution.

1.3 Neural network

Convolutional neural networks (CNN) are a most used logical tool that receives input parameters in the form of an image as a set of pixels, finds some features on it, and, thanks to this, sets parameters (weighted coefficients) for wide data objects in images and be able to highlight some features among all objects. CNN requires less raw processing power compared to other processing algorithms in performing classification tasks. Unlike standard filtering methods, which work

like a complex readymade unit, a convolutional neural network can achieve this through learning processes.

The structure of CNN (fig.1.2) is similar in its topology to the imaging device that is formed by biological neurons in the human brain, and was based on the "Visual Cortex" group. Each neuron responds to signals only in a specially limited area of the visual field, known as the receptive area. The set of such areas overlaps to cover the entire visual area. CNN can clearly capture spatial and temporal dependencies in any input image data through the application based on appropriate filters. The network structure optimally tunes the image dataset by reducing the number of parameters involved and the ability to reuse neuron weights. Thus, the neural network can be trained to better extract features from the original image for further processing and classification. [1]

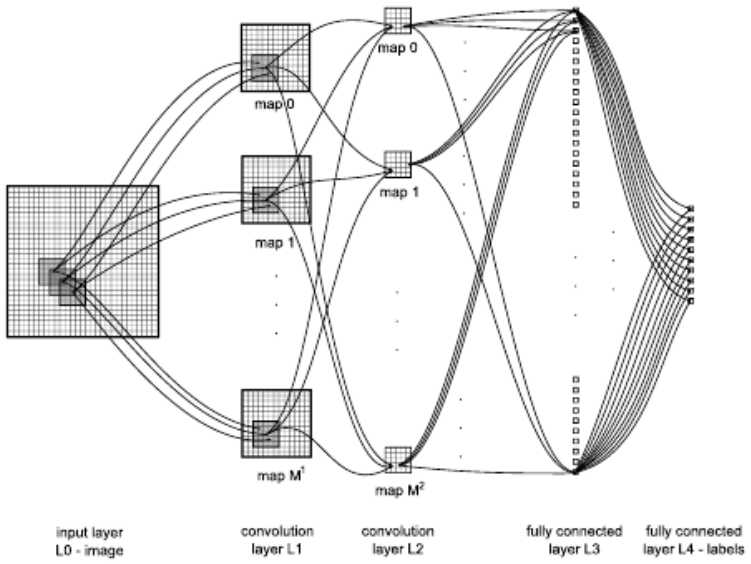

Fig. 1.2 Convolutional neural network principal structure

1.3 Convolutional neural network data input

Usually we get an RNG image as input, which means: R is red, G is green and B is blue. However, the use of other color schemes, shades of gray, etc. is not excluded.

However, the direct approach to using pixel-by-pixel analysis (fig. 1.3) is computationally expensive. So for processing an 8K image (7680 × 4320), the required number of parameters is tripled, and at the output we have a matrix with 99532800 (33177600 dots with 3 colors numbers in each) parameters. Therefore, care must be taken when we design a structure with low training machine power and large datasets - the main advantage of a convolutional network in this case is the ability to compress an image to a smaller array without losing important parts (features). [2]

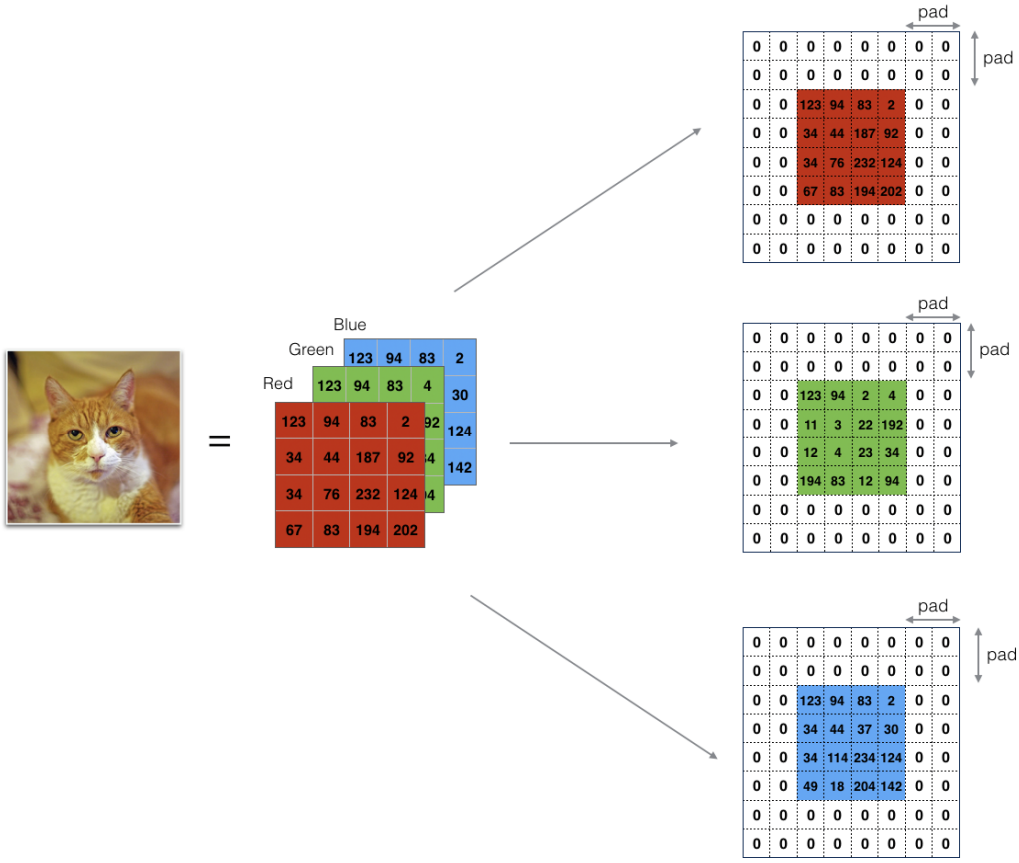

Fig. 1.3 Input data representation (RGB)

1.4 Convolution layer and its kernel filter

The main base layer of a convolutional network is the convolutional layer. This layer detects logic and features in pixel sets and extracts features (features)

from them. After that, in the layer, a logical operation processes the received signal - for this, the layer uses an input set of pixels (image) and a filter (kernel).

In figure 1.4, the blue area on the original array includes a portion of our input image. An element that operates in a convolutional operation of a convolutional layer is called a "kernel" or, in other words, a filter that extracts information from a specific region of the image, which is displayed as blue part of input matrix, and this filter looks like a shaded 3x3 matrix.

The kernel simply slides from left to right with the previously set stride value until it reaches the end of the current line. Then the next line will pop up and the process will repeat relatively. After a while, the filter will complete all rows and we will get the output matrix from the new values obtained by the filter. The use of a large number of sets of such filters allows us to obtain various results, such as the edges of an object, its sharpness, features of a concatenated image, detect bump maps, etc.

The main purpose of this operation is to obtain new, generalizing output elements in the form of matrices. Convolutional networks can contain any number of convolutional layers. Consequently, the first convolutional layer is useful for extracting "low-level" characteristics, such as edges, color, orientation of the gradient, etc., while the subsequent layers already perform more complex tasks of combining these features into groups, which is necessary in the case of tasks related to the detection and classification of objects

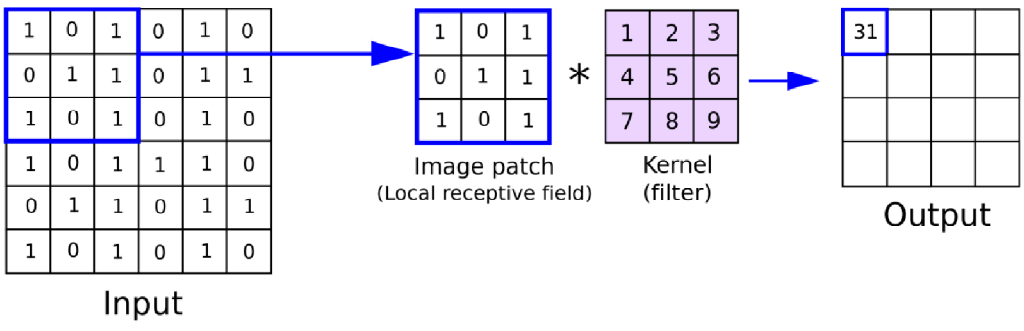

Fig. 1.4 Simple 3x3 kernel filter example

In the process of complementing and further expanding the network topology, the layers themselves also adapt to the "high-level" functions, giving us a network that has a complete understanding of the images in the dataset, just as the human brain automatically does. [3]

1.5 Pooling Layer

As a convolutional layer, a special unifying layer is used, based on the operation of reducing the spatial size of a convolutional object. By decreasing the dimension, it reduces the load on computational elements and partially allows you to discard unnecessary computational noise. In addition, you can highlight the dominant features at any angle and position to speed up the learning process, thus introducing additional nonlinearity into the structure of the network's computations. There are only two types of pooling: maximum pooling (MaxPool) and average pooling (AvPool, see fig. 1.5).

The maximum pooling finds the maximum value in the result matrix specified by the kernel filters. In turn, averaged pool finds the average value in the resulting matrix specified by the kernel filters.

Maximum pooling can also act as a noise suppression filter - by decreasing the dimensionality, it reduces the activation noise that occurs along it, while highlighting the most contrasting elements, which, moreover, is useful for processing images of poorly lit objects. The averaging pool also allows us to reduce the dimension as a noise suppression tool - as a result, we can get the same reduction in the dimension of the image.

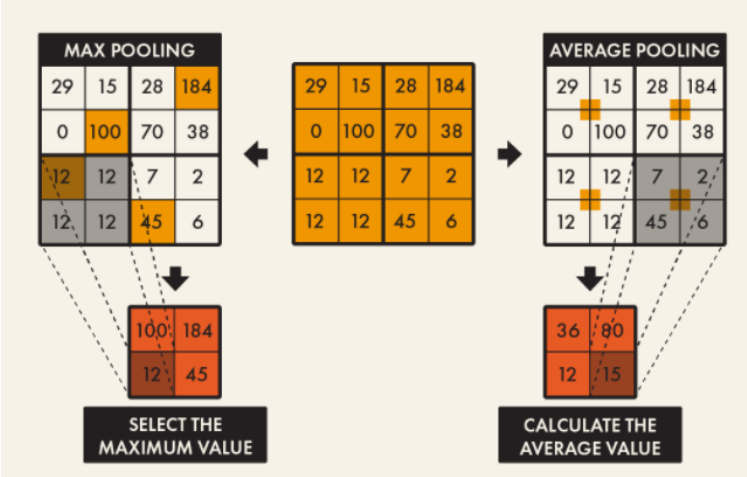

Fig 1.5 Example of polling layers

Usually, merge layers always work in pairs with a convolutional layer and form the ith convolutional layer of the CNN structure. Given the input image, its dimension and the number of functions it uses, it depends on the number of such convolutional layers, as well as on the computational power when training the neural network. [4]

1.6 Fully Connected Layer (FC layer) and Softmax Layer

A fully connected layer is a standard layer - a tool for finding nonlinear relations in object structures, the final matrix of signals after processing, we get from convolutional layers. Thus, it is always applied as the last layer of convolutional neural networks. Functionally, the task of a fully connected layer is to find possible nonlinear functions (see fig.1.6).

To work with a fully connected layer or also with a multilayer perceptron, it is necessary to convert the image to a vector with one column using a flat layer - this is usually done using the flattering function, which converts the input signal to a one-dimensional data array. The resulting matrix of vectors is moved to the fully connected layer and it starts the backpropagation training process due to each iteration.

For the final classification of the output signal, the softmax classification function is widely used, a specialized layer that assigns to each signal a set of values from 0 to 1, which characterize the probability that an object belongs to one or another image class.Currently, there are a huge number of different structures of convolutional neural networks with a different number of layers, as well as different performance. The most popular structures: GoogleNet, LeNet, AlexNet, ResNet, etc. [5]

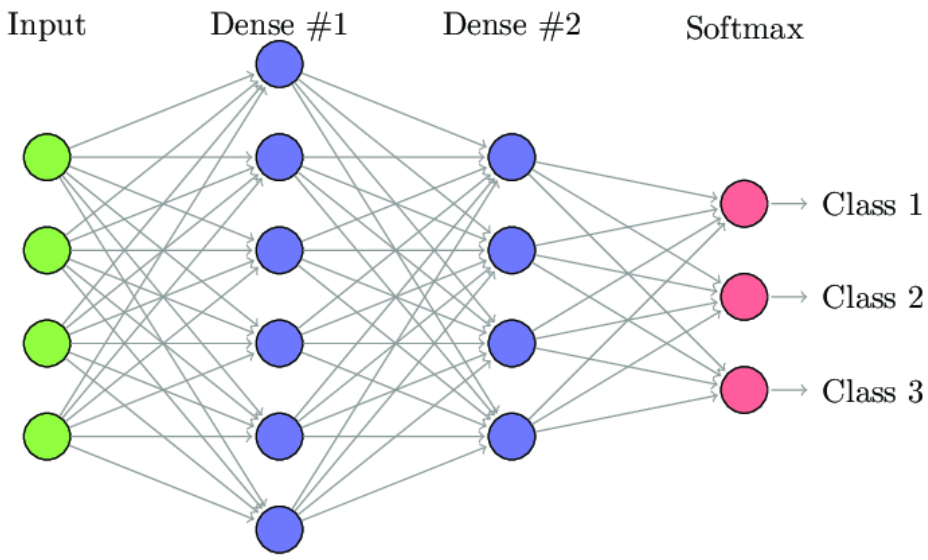

Fig. 1.6 Structural scheme of fully connected layer

1.7 Learning of neural network

Backpropagation is probably the most fundamental building block of a neural network. It was first introduced in the 1960s and nearly 30 years later (1989) popularized by Ramelhart, Hinton, and Williams in an article titled "Exploring Representations with Backpropagation Errors." The algorithm is used to efficiently train a neural network using a technique called chain rule. In simple terms, after each forward pass through the network, backpropagation performs a backward pass while adjusting the model parameters (weights and biases). Now

the algorithm is actually the basis for training a wide range of neural networks. The practical implementation of computations that implement the algorithm in practice in a specific network uses the following methods:

- Stochastic gradient descent. It is the main algorithm responsible for the convergence of neural networks, that is, we are shifting towards the optimum of the cost function. There are several gradient descent algorithms and I have mixed them together in previous posts. I'm not talking about batch (vanilla) gradient descent or mini-batch gradient descent here. The main difference between Batch Gradient Descent (BGD) and Stochastic Gradient Descent (SGD) is that we only calculate the cost of one example for each step in SGD, but in BGD we have to calculate the cost for all training examples into the dataset. Oddly enough, this significantly speeds up the work of neural networks. This is where SGD's motivation lies. The equation for SGD is used to update the parameters in the neural network - we use the equation to update the parameters in the back pass, using backpropagation to compute the gradient ∇ by formula 1.7.1:

$$\theta = \theta - \eta * \nabla(\theta) * J(\theta; x; y) \qquad (1.1)$$

Where θ is a parameter (theta), such as your weights, prejudices and activations. Note that here we update only one parameter for the neural network, that is, we can update one weight, η is learning speed, ∇ is gradient, which is taken from J, J is formally called an objective function, but is most often called a cost function or a loss function. A very popular technique that is used in conjunction with SGD is called Momentum. Instead of using only the gradient of the current step for the direction of the search, momentum also accumulates the gradient of the past steps to determine the direction of travel.

- RMSprop, or Root Mean Square Propagation, has an interesting history. It was developed by the legendary Geoffrey Hinton when he came up with a random idea during a lesson on Coursera. RMSProp also tries to dampen the wobble, but in a different way than momentum. The RMS prop also removes the need to adjust the learning rate and does it automatically. Moreover, RMSProp chooses a different learning rate for each parameter.

In the RMS prop, each update is performed according to the equations described below (fig. 1.7). This update is performed separately for each parameter.

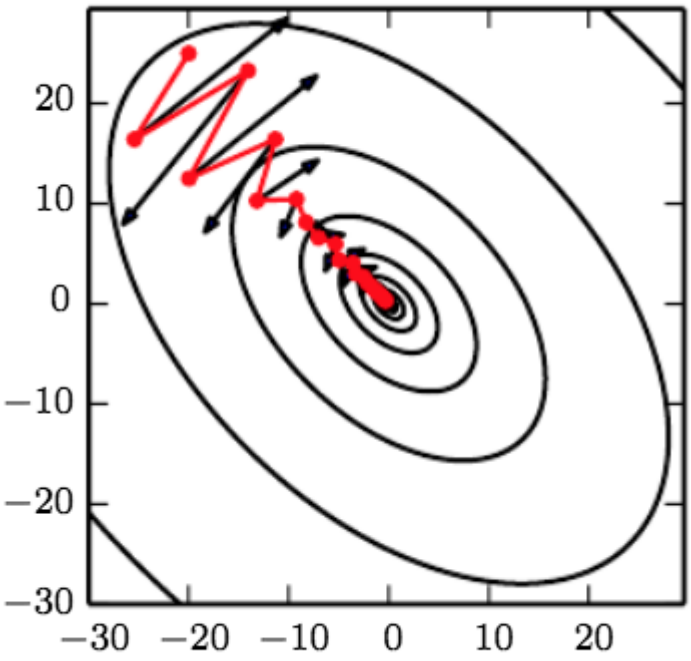

Fig. 1.7 RMS propagation algorithm visualizationn

We calculate the exponential mean of the square of the gradient. Since we do this separately for each parameter, the gradient Gt here corresponds to the projection or component of the gradient along the direction represented by the parameter we are updating. To do this, we multiply the exponential average calculated before the last update by the hyperparameter, represented by the Greek symbol nu. Then we multiply the square of the current gradient by (1 - nu). We

then add them together to get the exponential average up to the current time step. The reason we use exponential mean is because, as we saw in the impulse example, it helps us weigh more recent gradient updates more than less recent ones. In fact, the name "exponential" comes from the fact that the weight of the previous terms falls off exponentially (the very last term has weight p, the next is the square of p, then the cube of p, and so on).

Notice our pathological curvature diagram, the gradient components along w1 are much larger than along w2. Since we square them and add them, they don't cancel out and the exponential mean is large for w2 updates. Then, in the second equation, we determined our step size. We are moving in the direction of the gradient, but the exponential mean affects our step size. We chose an initial learning rate, eta, and then divided it by the mean. In our case, since the mean of w1 is much larger than w2, the learning step for w1 is much smaller than for w2. Hence, it will help us avoid bouncing between the ridges and move towards the lows.

The third equation is simply a renewal step. The hyperparameter p is usually set to 0.9, but you may need to tweak it. Epsilon is equation designed to prevent division by zero and is usually set to 1e-10. It should also be noted that RMSProp implicitly simulates annealing. Suppose we are moving to lows and want to slow down so as not to go beyond them. RMSProp will automatically reduce the size of the gradient steps to a minimum if they are too large (large steps tend to overstep our limits).

- Adam or Adaptive Moment Optimization algorithms combines the heuristics of both Momentum and RMSProp. Here are the update equations.

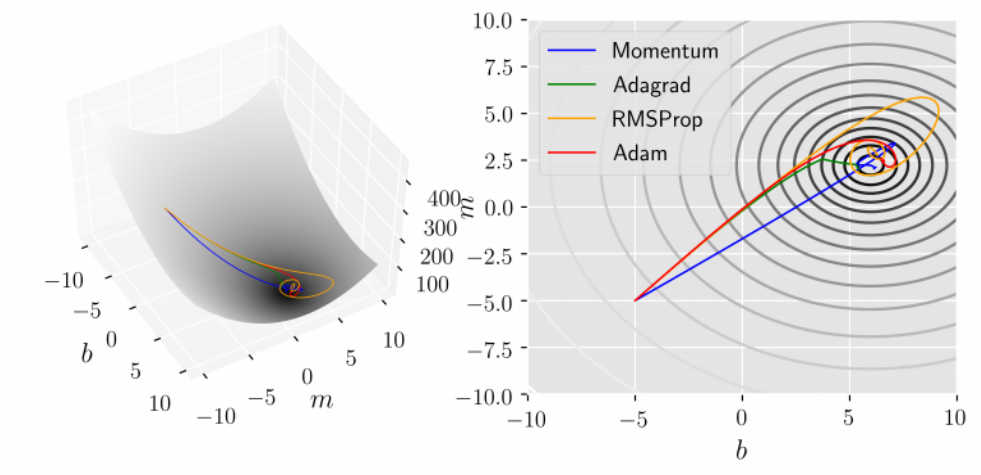

Fig. 1.8 ADAM algorithm compared to other visualization

Here we calculate the exponential mean of the gradient as well as the squares of the gradient for each parameter. To determine the learning step, we multiply our learning rate by the gradient mean (as with momentum) and divide by the mean square of the exponential mean of the squares of the gradients (as with momentum). Then we add an update. The result of such operation overhelm another methods in terms of performance (see fig. 1.8). [6]

Chapter 2

The subject of the work

2.1. The means of artificial neural network synthesis

Deep learning and machine learning are part of the group of artificial intelligence strategies. Deep learning refers to a class of artificial intelligence computing that uses multiple layers where nonlinear processing modules are used to extract and modify image information. Each next layer in such an architecture uses the result of the work of the previous layer as information.

Deep neural systems, deep learning systems, and convolutional neural systems have historically emerged in the fields of computer vision, human language processing and machine translation, sound processing and bioinformatics, where they gave results that were almost identical, and at some point surpasses the human ones.

This sphere often uses the programming language and libraries Python.In this part, we will learn about the Earth, created for Python Deep Learning. For this purpose, the programming language currently has a number of extensions, so-called libraries, which contain ready-made tools and functions designed to work with neural networks. Here are some examples:

- PyTorch
- Theano.
- Matplotlib
- TensorFlow
- Keras
- TensorFlow
- Sci-Kit Learn

Python belongs to one of the fastest growing programming languages. It is so popular that it is relatively easy to use, free, and optimized for some deep learning tasks. Its advantages, along with the growing interest in AI and the breadth of application of neural networks in the modern world, provide such a variety of libraries that already poses the problem of choosing a specific implementation method.

The open source library available today has its qualities and drawbacks when we look at them comprehensively. Accordingly, the optimal choice of the system in a particular case will depend on the specific tasks that will be set at the development stage.

Considering a structure that is optimized to perform mathematical calculations, you can turn your attention to the Theano library. A library well prepared for numerical processing, which has ensured its widespread use and integration - it is part of other deep learning libraries, such as Keras or Tensorflow. It allows you to efficiently work with tasks that include multidimensional clusters. The advantages of the library include:

- Efficiency when processing multidimensional datasets or datasets with large amounts of information;
- A fairly flexible tool that allows you to make more precise adjustments to the original algorithms, just like the creation of new structures.
- Supports GPUs

Cons:

- Steep learning curve, potentially making it difficult to correctly train the artificial neural network;
- Does not support parallel computing with multiple GPUs.
- The complexity of working with the library

Theano philosophy is intended for advanced learning specialists, since the library is quite low-level - which implies a sufficient degree of familiarity with ANN.The Theano language structure uses a library for working with large data arrays and NumPy matrices, so the code from it can efficiently process washing as in CPU and GPU.

Unlike simpler libraries, Theano provides a high degree of adaptability, allowing the developer to develop and execute structures of non-standard topology or manually optimize bottlenecks existing for a specific task.A big plus for the library is the presence of a large array of technical documentation, but for some time the library no longer supports updates which is its disadvantage

Caffe (Convolutional Architecture for Fast Function Inlining) was designed to work with convolutional neural networks (CNNs), with tasks like image recognition, computer vision, and feedforward systems. Structurally, it is a Python-based API integrated to work with the GPU.

Pros:

- Contains built-in models for image recognition;
- The parameters of the models are configurable, not set each time by the code.
- Supports GPUs
- Fast implementation of convolutional networks - optimal for weak resources;

Cons:

- Limited amount of documentation;
- Does not support multithreading;
- Not suitable for recurrent neural networks;
- Limited amount of working documentation;
- Creating new layers and structures is inconvenient;

The main advantage of the library from the point of view of a beginner is the presence of a library of ready-made network architectures, in particular, the library contains such pretrained networks as GoogleNet and AlexNet. [14]

Sci-kit Learn library is a library that contains a large group of image processing methods, including support vector machines (SVMs), K-nearest neighbors (KNN) classifiers, and also includes a choice of both administrative and unsupervised learning. As a result, we have a flexible image processing tool. The library is partially based on other Python libraries such as SciPy or Matplotlib,

Pros:

- Suitable for beginners in programming.
- Suitable for initial learning algorithms
- Suitable for machine data analysis and fairly simple projects

Cons:

- Does not contain support for ANN operation;
- Does not support GPU computing;

The main obstacle to using this method is its organic inability to support artificial neural networks. It is reasonable to use this library for small programs with relatively small data sets or even for tasks that do not require special computing resources - due to the fact that the structure does not support the use of GPUs, which is a performance limitation for programs using this library. [7]

Pytorch was built using Python, C ++ languages and is optimized to use CUDA kernels. Created by the Facebook Artificial Intelligence Research Group (FAIR), it is a new product,

Pros:

- Supports GPU acceleration.
- Supports dynamic charts, so you can customize them on the go;

Cons:

- A small amount of documentation ..

Tensorflow is a framework that was created by the Google Brain group and supports all operating systems. A feature of the library is the ability to run low-level code with supporting libraries.

Pros:

- Flexibility;
- Contains several ready-to-use ANN models.
- Scalability - multithreading support and optimized software;
- A large amount of documentation.

Cons:

- Supports only NVIDIA GPUs;
- Not suitable for beginners;

Deep learning toolbox in MATLAB engineering medium Deep learning toolbox in MATLAB engineering medium is an additional extension to the main MATLAB package, which allows modeling neural networks based on ready-made functional blocks, with the subsequent translation of the architecture into C ++ code, supported by the basic development environment.

Pros:

- Using the complex development environment MATLAB

- Availability of ready-made models of neural networks, such as RESNET and ALEXNET.

- Support for multithreading and GPU

- Built-in artificial neural network pre-validation

-Integrated network training tools;

- Ease of use

Cons:

- Less flexibility than tensorflow libraries.

- Average computation optimization

Therefore, the optimal choice for the task set in the framework of the thesis is the development in the software environment MATLAB. [8]

2.2. Datasets

Each convolutional neural network needs quite large arrays of images to train. Such collections are called datasets, and a large number are now available for a variety of tasks. For example. Machine Learning Datasets, such as

Mall Customers Dataset: The Mall customers dataset contains information about people visiting the mall in a particular city. The dataset consists of various columns like gender, customer id, age, annual income, and spending score. It's generally used to segment customers based on their age, income, and interest.

IRIS Dataset: The iris dataset is a simple and beginner-friendly dataset that contains information about the flower petal and sepal width. The data is divided into three classes, with 50 rows in each class. It's generally used for classification and regression modeling.

MNIST Dataset: This is a database of handwritten digits. It contains 60,000 training images and 10,000 testing images. This is a perfect dataset to start implementing image classification where you can classify a digit from 0 to 9.

Boston Housing Dataset: Contains information collected by the US Census Service concerning housing in the area of Boston Mass. It was obtained from the StatLib archive and has been used extensively throughout the literature to benchmark algorithms.

Fake News Detection Dataset: It is a CSV file that has 7796 rows with four columns. There are four columns: news, title, news text, result.

Wine quality dataset: The dataset contains different chemical information about the wine. The dataset is suitable for classification and regression tasks.

SOCR data — Heights and Weights Dataset: This is a basic dataset for beginners. It contains only the height and weights of 25,000 different humans of 18 years of age. This dataset can be used to build a model that can predict the height or weight of a human.

Titanic Dataset: The dataset contains information like name, age, sex, number of siblings aboard, and other information about 891 passengers in the training set and 418 passengers in the testing set.

Credit Card Fraud Detection Dataset: The dataset contains transactions made by credit cards; they are labeled as fraudulent or genuine. This is important for companies that have transaction systems to build a model for detecting fraudulent activities.

Computer Vision Datasets:

xView: xView is one of the most massive publicly available datasets of overhead imagery. It contains images from complex scenes around the world, annotated using bounding boxes.

ImageNet: The largest image dataset for computer vision. It provides an accessible image database that is organized hierarchically, according to WordNet.

Kinetics-700: A large-scale dataset of video URLs from Youtube. Including human-centered actions. It contains over 700,000 videos.

Google's Open Images: A vast dataset from Google AI containing over 10 million images.

Cityscapes Dataset: This is an open-source dataset for Computer Vision projects. It contains high-quality pixel-level annotations of video sequences taken in 50 different city streets. The dataset is useful in semantic segmentation and training deep neural networks to understand the urban scene.

IMDB-Wiki dataset: The IMDB-Wiki dataset is one of the most extensive open-source datasets for face images with labeled gender and age. The images are collected from IMDB and Wikipedia. It has five million-plus labeled images.

Color Detection Dataset: The dataset contains a CSV file that has 865 color names with their corresponding RGB(red, green, and blue) values of the color. It also has the hexadecimal value of the color.

Stanford Dogs Dataset: It contains 20,580 images and 120 different dog breed categories.

2.3. Modern convolutional neural networks

In the area of convolutional networks, there are several architectures that have a name. The most common (shown in fig. 2.1) are:

- Lenet. The first successful applications of convolutional networks were developed by Yann LeCune in the 1990s. Of these, the most famous is the LeNet architecture, which was used to read zip codes, numbers, etc.

- AlexNet. The first work to popularize convolutional networks in computer vision was AlexNet, developed by Alex Krizhevsky, Ilya Sutskever, and Jeff Hinton. AlexNet was represented in the 2012 ILSVRC ImageNet competition and significantly outperformed the second runner-up (top five error of 16% versus the runner-up with a 26% error). The network had a very similar

architecture to LeNet, but was deeper, larger and contained convolutional layers stacked on top of each other (it used to be common to have only one CONV layer, immediately followed by a POOL layer).

- ZF Net. The ILSVRC 2013 winner is the Convolutional Network from Matthew Zeiler and Rob Fergus. It became known as ZFNet (short for Zeiler & Fergus Net). This was an improvement to AlexNet by tweaking the hyperparameters of the architecture, in particular by increasing the size of the middle convolutional layers and decreasing the step and filter size on the first layer.

- GoogLeNet. The ILSVRC 2014 winner is the Convolutional Network by Szegedy et al. from google. His main contribution was the development of the starter module, which drastically reduced the number of parameters on the network (4M versus AlexNet from 60M). Also, this document uses a middle pool instead of fully connected layers at the top of ConvNet, which removes a lot of options that don't seem to matter much. There are also several versions of GoogLeNet, the latest of which is Inception-v4.

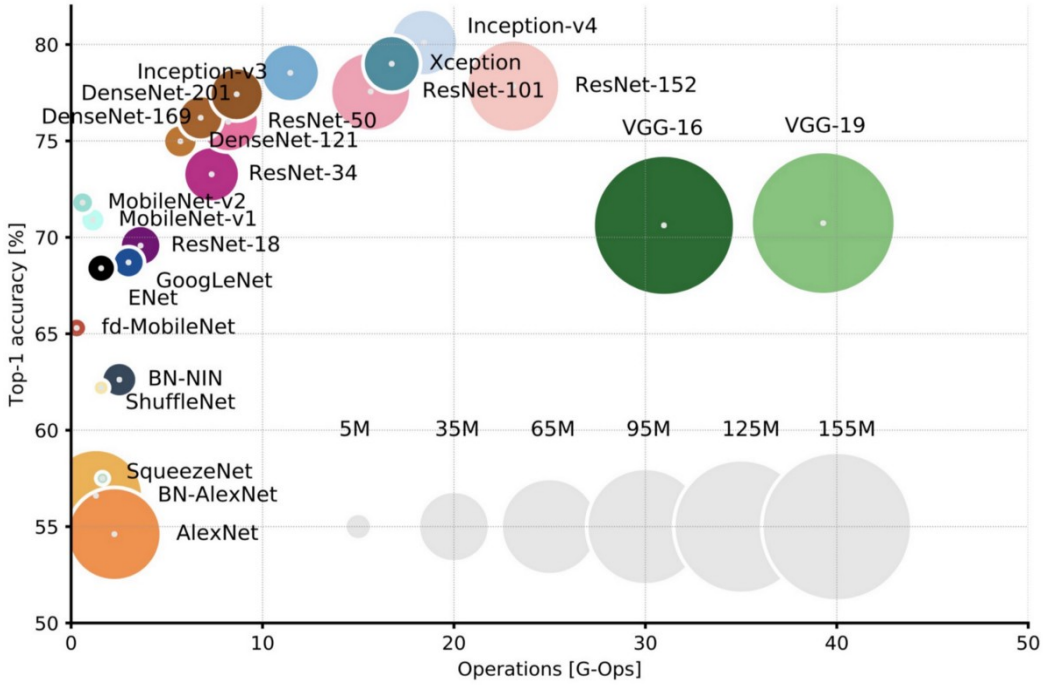

Fig. 2.1 Comparison of different modern neural networks performance

- VGGNet. The second place in ILSVRC 2014 was taken by the network of Karen Simonyan and Andrey Zisserman, which became known as VGGNet. His main contribution was to show that network depth is a critical component of good performance. Their latest best network contains 16 CONV / FC layers and, interestingly, has an extremely uniform architecture that only performs 3x3 convolutions and 2x2 aggregation from start to finish. Their pre-trained model is available for plug and play use in Caffe. The downside of VGGNet is that it is more expensive to evaluate and uses much more memory and parameters (140MB). Most of these parameters are in the first fully connected layer, and it has since been found that these FC layers can be removed without sacrificing performance, greatly reducing the number of parameters required.

- ResNet. The residual network developed by Kaiming He et al. was the winner of ILSVRC 2015. It features special bandwidth connections and heavy use of batch normalization. The architecture also lacks fully connected layers at the end of the network. The reader is also encouraged to check out Kaiming's presentation (videos, slides) and some recent experiments replicating these networks in Torch. ResNets are currently modern convolutional neural network models and are the default choice for using ConvNets in practice.

2.4. Architecture problem

In the tasks of image classification, the so-called residual neural networks, which are hybrid networks based on ANN of direct propagation, have achieved success. [9]

Deep convolutional neural networks led to a series of breakthroughs in image classification. They naturally integrate functions of different levels of level, using classifiers in an end-to-end multi-level way. At the same time, the depth of the network is crucial for the performance and efficiency of

classification, and the most outstanding results have been achieved by neural networks that use very deep models, with a depth of sixteen to thirty layers. Many other non-trivial visual recognition problems with 20-layer and 56-layer "simple" networks.

At the same time, the deeper network has a higher error in the learning process and, therefore, the error in the final result. Similar phenomena in the example of the ImageNet network are shown in Fig. 2.2.

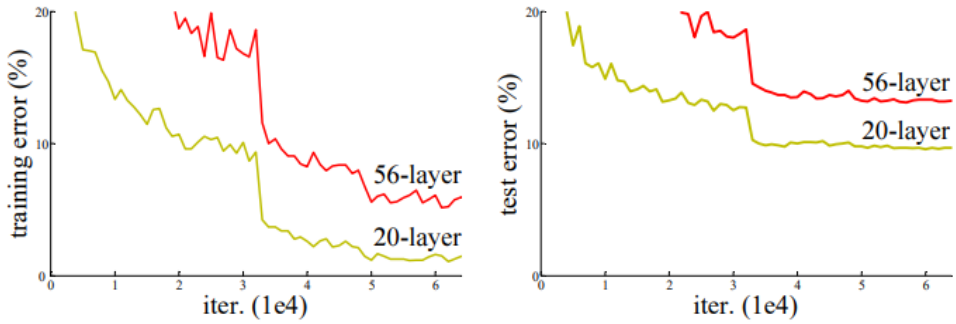

Fig. 2.2 Deep neural networks error comparison

In view of the identified problem, the question of contradictions arises between the need to increase the depth of the network to increase its efficiency, and the observed not an increase, but a drop in this efficiency with an increase in the number of layers. This problem is due to the presence of the so-called disappearing/radiant. When deeper networks can begin to converge, a degradation problem is identified: with the network depth increases, accuracy becomes saturated and then degrades rapidly. Moreover, such a deterioration, by its nature, is not caused by retraining - the addition a larger number of layers in a deep model leads to a higher learning error, as seen in Fig. 2.2.1. The fact is that on deep neural networks, the sequence of image processing ultimately leads to the fact that important, but insufficiently highlighted details and features of the image are removed by filters as they are processed, thus ultimately degrading the recognition accuracy.

The way out of this situation is to use the so-called. residual neural networks, whose feature is a partial "pass" of the signal past a part of the network

filters, thus providing less attenuation of the gradient during the passage and computation along the neural network. An example of such a block can be seen in Fig. 2.3 [10]

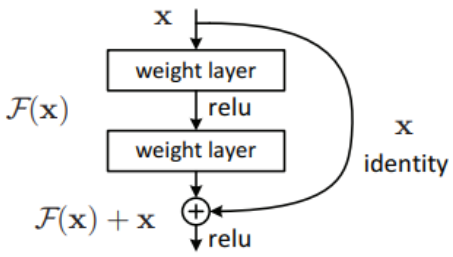

Fig. 2.3. Residual CNN basic subblock

For a shallow network with multiple layers using these activations, this is not a problem. However, when more layers are used it can cause the gradient to be too small for training to work effectively. Neural network gradients are found using error backpropagation. Simply put, backpropagation finds derived networks, moving layer by layer from the last layer to the initial one. According to the chaining rule, the derivatives of each layer are multiplied down the network (from the last level to the initial) to calculate the derivatives of the initial layers.

However, when the n hidden layers use sigmoid-like activation, the n small derivatives are multiplied together. Thus, the gradient decreases exponentially as you move down to the initial layers. A small gradient means the starting layer weights and offsets will not be effectively updated with every workout. Since these initial levels are often critical to recognizing the basic elements of the input, this can lead to general inaccuracy throughout the network.

The simplest solution is to use other activation functions such as ReLU, which do not cause a small derivative. Residual networks are another solution as they provide residual connections straight to earlier levels. As seen in Figure 2, the residual join directly adds the value at the start of the block, x, to the end of the block (F (x) + x). This residual linkage does not go through activation functions that crush the derivatives, resulting in a higher total block derivative.

Finally, the problem can be solved by using batch normalization levels. As stated earlier, the problem occurs when a large input space is mapped to a small one, causing the derivatives to disappear.

2.5. Learning of neural network

The way to train our model is called as backpropagation. The overall algorithm can be simplified to following steps:

- Calculate the error – how far is your model output from the actual output.
- Minimum Error – Check whether the error is minimized or not.
- Update the parameters – If the error is huge then, update the parameters (weights and biases). After that again check the error. Repeat the process until the error becomes minimum.
- Model is ready to make a prediction – Once the error becomes minimum, you can feed some inputs to your model and it will produce the output.

The backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem. In training the hybrid neural network, it was proposed to use two alogorhythms of backpropagation implementation - SGDM and ADAM.

The first is a gradient descent algorithm that updates the network parameters (weights and biases) to minimize the loss function by taking small steps at a predetermined rate at each iteration in the direction of a negative loss gradient. Stochastic gradient descent can oscillate along the steepest descent path to the optimum. Adding an impulse term to the parameter update is one way to reduce this fluctuation. Stochastic gradient descent with momentum update (SGDM) is calculated as (2.7.1.):

$$\theta_{\ell+1} = \theta_\ell - \alpha \nabla E(\theta_\ell) + \gamma(\theta_\ell - \theta_{\ell-1}), \quad (2.2)$$

Where γ determines the contribution of the previous gradient step to the current iteration.

Adam uses a parameter update similar to the other common RMSProp algorithm, but with an added twist. If the gradients are similar across many iterations, then using a moving average gradient allows the parameter updates to gain momentum in a specific direction. If the gradients contain mostly noise, then the moving average of the gradient becomes smaller, and therefore the parameter updates also become smaller (see 2.6.2.)

$$v_\ell = \beta_2 v_{\ell-1} + (1 - \beta_2)[\nabla E(\theta_\ell)]^2 \quad (2.)$$

Comparison of these learning algorithms on a basic neural network clearly showed (sees fig. 2.4 and fig. 2.5) the advantage of the ADAM algorithm (see fig. 2.6) and made it possible to choose an initial value of the learning rate of 0.001. T

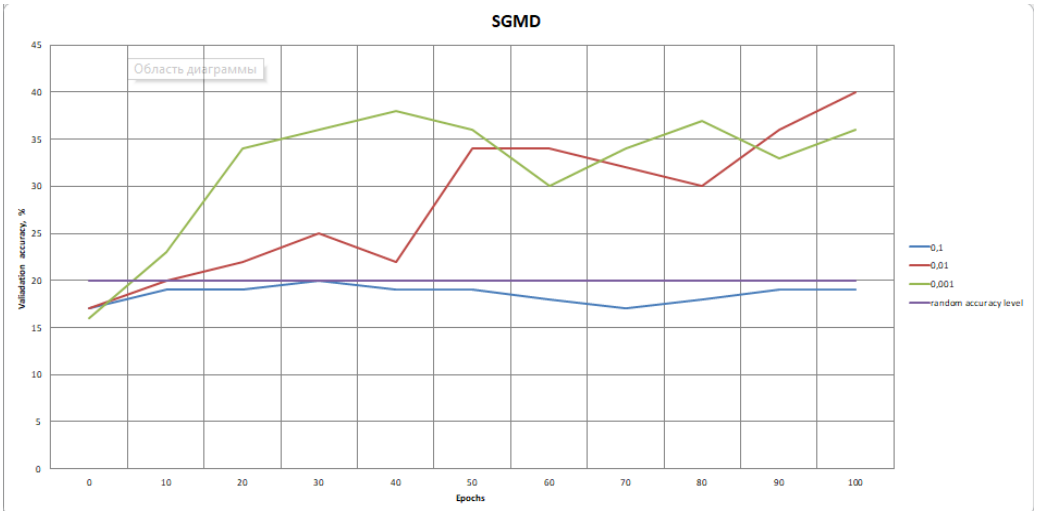

Fig. 2.4. Example of learning rate output, where the black line is the validation rate, the blue line is the training rate, and the orange line is the average value of the network error.

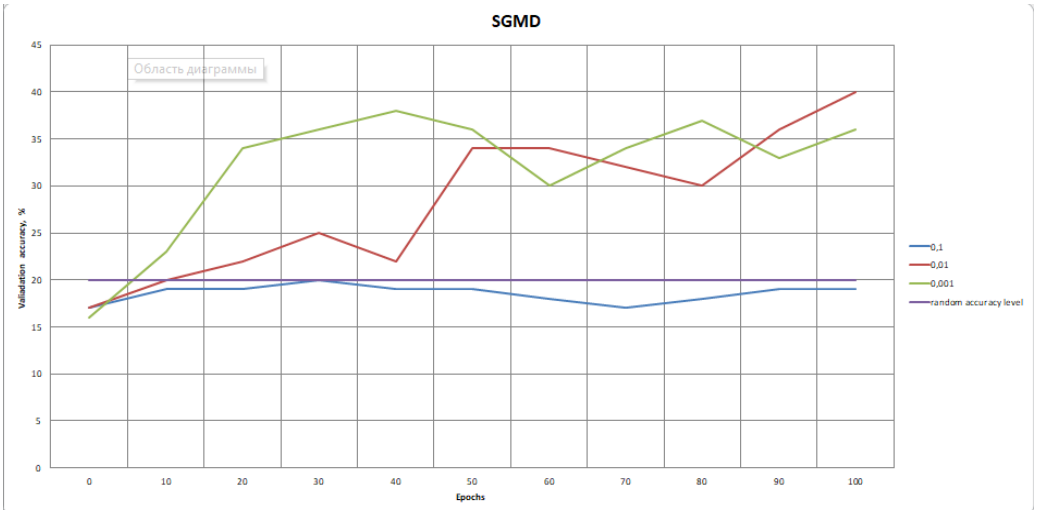

Fig. 2.5. SGMT efficiency depending on learning rate graph

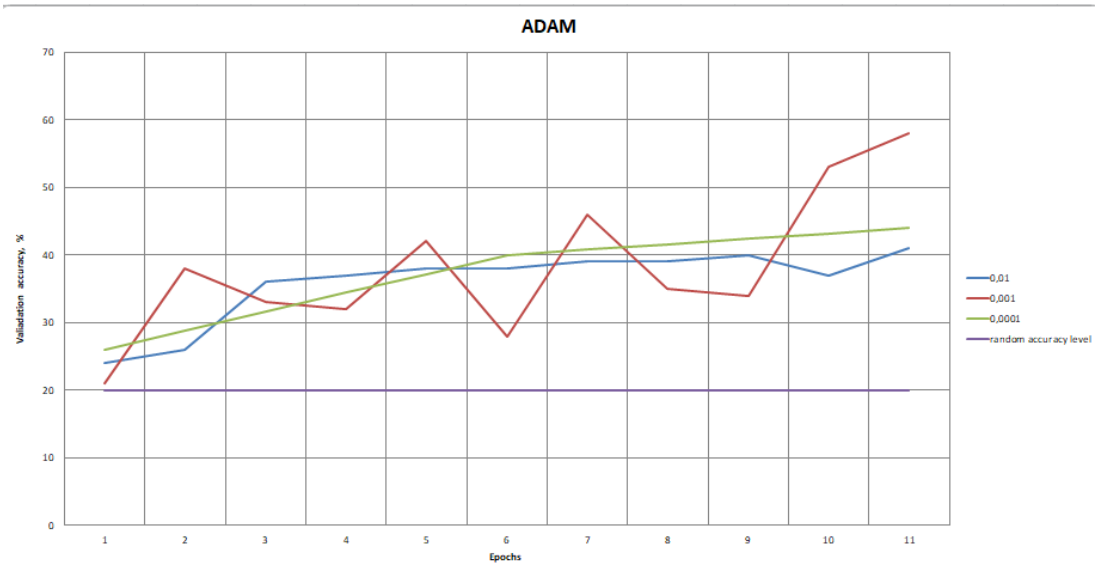

Fig. 2.6. ADAM efficiency depending on learning rate graph

2.6. Hybrid network with attention module

CNN uses their convolutional filters to extract information from images. The lower layers find elements of a low level of context, such as edges or gradients, while the upper layers, due to the multilayer aggregation of signals of lower levels, allow the detection of more complex features, such as faces, text or

any other complex and complex shapes. The structure of such block shown in Fig. 2.7:

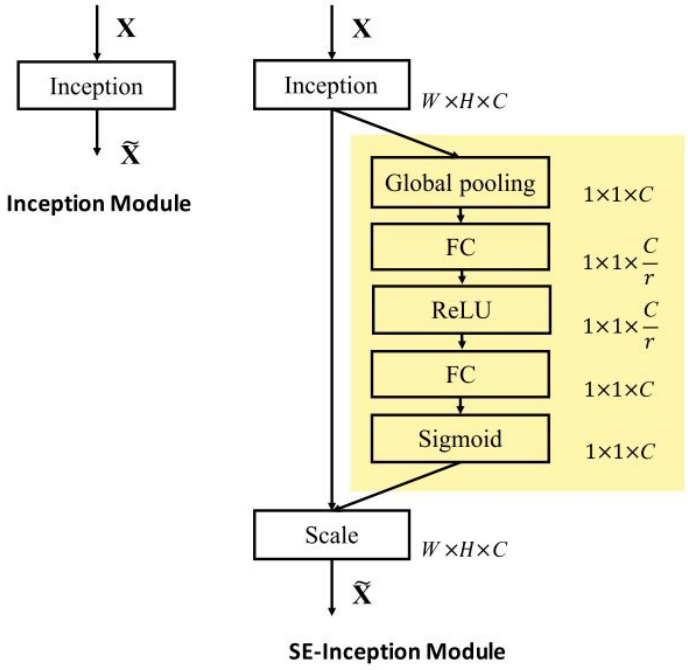

Fig. 2.7. Squeeze-and-excitation module

It is used by combining spatial and channel information of the image, propagated through the neural network. In the general case, neural network filters first find the spatial characteristics in each input channel, and then distribute the information to the outputs. An important nuance is that the network evaluates the weight of each of its channels in the same way when creating output function maps. Scheme of such attention module implementation are shown in Fig. 2.8. below:

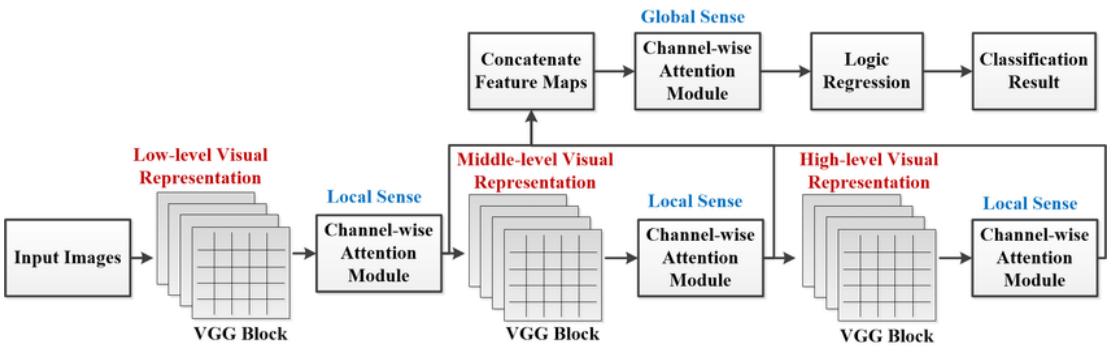

Fig. 2.8, Attention module structure

The essence of the neural network augmentation method using the CEB attention module (Fig. 2.8) Is to get a general idea of each channel, passing the signal through a two-level neural network, which outputs a vector of the same size. These n values can now be used as weights on the original feature maps, scaling each channel based on its importance. Thus, we essentially get a nonlinear filter that ensures the concentration of the network's attention in each individual case on the parameters that are important for identifying features - using a miniature neural network. The effect of this application on a "normal" neural network can be seen in Fig. 2.8. When we think of the English word "attention", we understand that it means to direct our attention to something and pay more attention. Deep learning's attention engine builds on this concept of directing your attention and pays more attention to certain factors when processing data.

Broadly speaking, attention is one of the components of a network architecture responsible for managing and quantifying interdependencies: Between iinput and output items (general attention) and inside input elements (self-attention) Let's say we have a sentence “How was your day?” Which we would like to translate into French - “Comment se pas ta journée”. The Attention component of the network will map important and relevant words from the input sentence for each word in the output sentence and assign higher weights to these words, increasing the accuracy of the output prediction, as shown in fig. 2.9: [11]

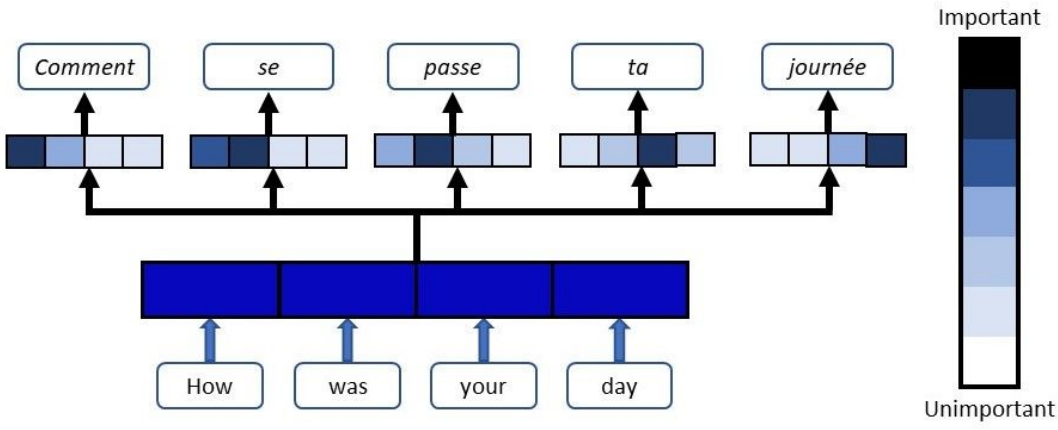

Fig. 2.9. Attention mechanism in translation tasks

Weights are assigned to the input words at each stage of translation. The above explanation (fig 2.9) of attention is very broad and vague due to the different types of attention mechanisms available. But don't worry, this article will give you a clearer understanding of how Attention works and how it achieves its goals. Since the attention mechanism has undergone many adaptations over the years to solve different problems, there are many different versions that apply. We will only consider here the most popular adaptations, namely its use in sequence-to-sequence models and the later version of Self-Attention.

While Attention does find applications in other areas of deep learning such as computer vision, its major breakthrough and success has come from its use in natural language processing (NLP) problems. This is because attention was introduced to address the problem of long sequences in machine translation, which is also a problem for most other NLP problems. [12]

2.7. Hybrid network with squeeze-and-excitation and attention modules

The structure of proposed neural network is shown if fig. 2.8. and 2.9. below:

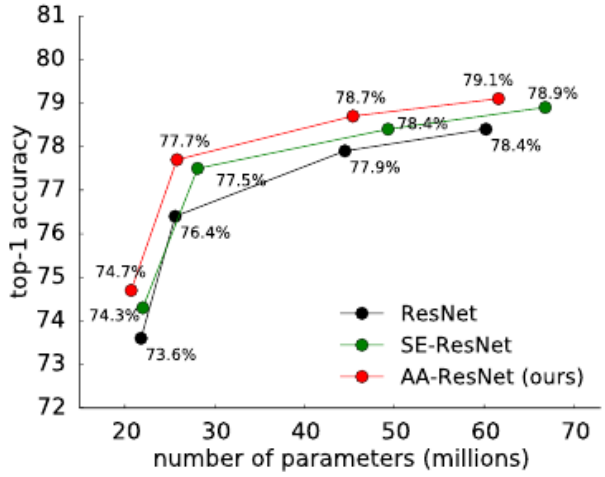

Fig. 2.8. Attention module augmentation effect on overall performance

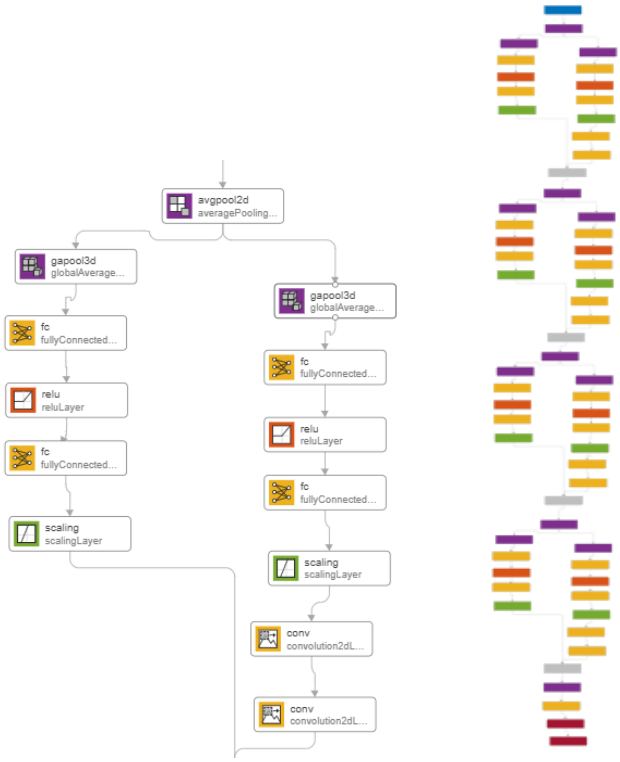

Fig. 2.9. Squeeze-and-excitation and attention module and improved neural network structure

Evaluation of the efficiency of training the basic and augmented by the attention module of the network was carried out using the control of the parameter validation accuracy, which is an estimate after checking on a randomly selected 30 percent of the initial training sample. This training is necessary to check the correctness of building the architecture and the result of training the network, due to the possibility of retraining the network on the same sample due to the iterative learning process, as well as comparing the training of the finished ResNet-18 network on the same sample that was used to train the created hybrid neural networks. The results can be seen in the graph shown in Figure 2.10:

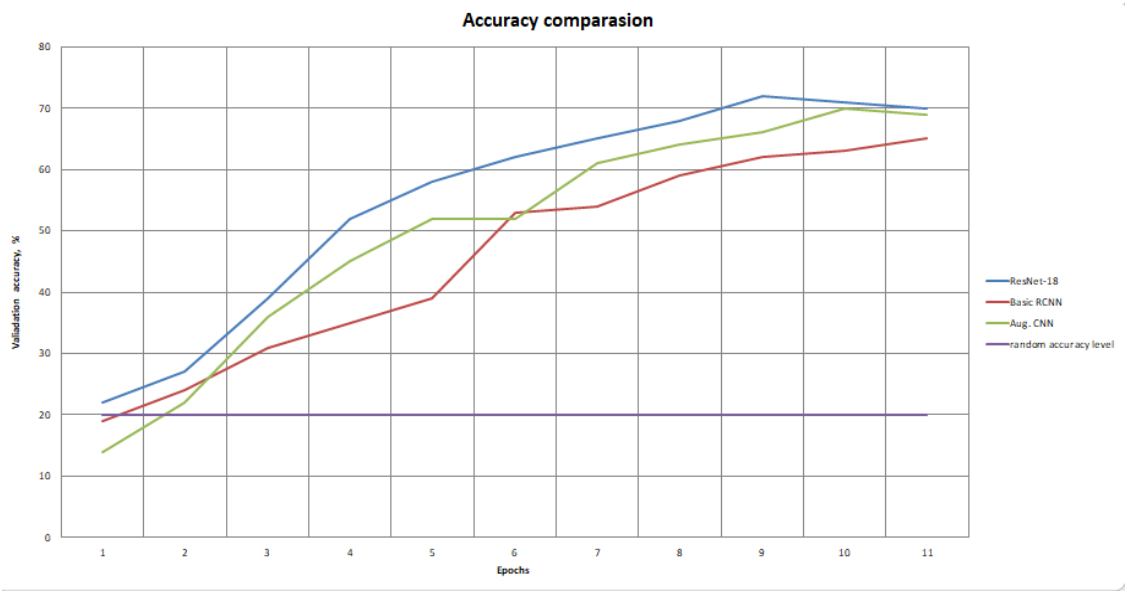

Fig. 2.10. CNNs efficiency classification efficiency comparison

As can be seen from the results of the study, the use of CEN modules in a residual neural network can significantly increase the efficiency of its tasks, with a minimum increase in the resources required for computing - for example, one fully connected layer from the attention module stores itself about 9 times less parameters than the convolutional layer of a comparable dimension (see fig. 2.11. below)

RESNET-18			Proposed hybrid RCNN		
Layer №	Size	kernel size	Layer №	Size	kernel size
Poling	112x112	5x5	Poling	120x120	3x3
Conv2	64	3x3	Conv2	64	3x3
Conv3	128	3x3	Conv3	128	3x3
Conv4	256	3x3	Conv4	256	3x3
Conv5	512	3x3	Conv5	256	1x1

Fig. 2.11. RESNET-18 and created hybrid residual convolutional neural network, number of neurons in convolutional layers comparison

Chapter 3

Architecture features

3.1. Batch normalization.

The problem of learning deep networks. Learning deep neural networks, such as networks with dozens of hidden layers, is challenging.

One aspect of this problem is that the model is updated layer by layer in the opposite direction from output to input using an error estimate that assumes that the weights in the layers prior to the current layer are fixed. Very deep models involve the composition of several functions or layers. The gradient tells how to update each setting, assuming no other layers are changing. In practice, we update all layers at the same time.

Since all layers change during the update, the update procedure always has a moving target. For example, the layer weights are updated to reflect the expectation that the previous layer will output values at a given distribution. This distribution is likely to change after updating the weights of the previous layer.

Training deep neural networks is complicated by the fact that the distribution of the input data of each layer changes during training, as the parameters of the previous layers change. This slows down learning, requiring lower learning rates and careful parameter initialization, and is known to make it difficult to train models with saturating nonlinearities. Batch normalization, on the other hand, can be defined as "accelerating deep learning of the network by reducing the internal covariate shift." The authors of the article introducing this definition of batch normalization call the change in the distribution of input data during training "internal covariate shift". [13]

Batch normalization, or batchnorm for short, is proposed as a method to help coordinate the updating of multiple layers in a model. Batch normalization provides an elegant way to reparameterize virtually any deep network. Re-

parameterization greatly reduces the problem of coordinating updates at many levels.

It accomplishes this scaling of the layer output, in part by standardizing the activations of each input variable for each mini-batch, for example, node activations from the previous layer. Recall that standardization refers to scaling the data so that the mean is zero and the standard deviation is one, such as standard Gaussian.

Batch normalization modifies the parameters of the model so that some units are always standardized by definition. This process is also called "whitening" when applied to computer vision images.

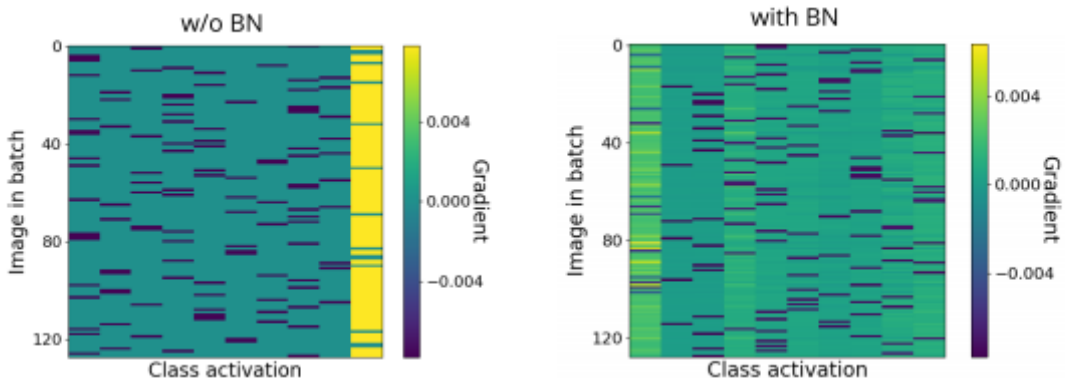

Fig. 3.1. Batch normalization effect vizualization

By lightening the inputs for each level, we would take a step towards achieving fixed distributions of the inputs that would eliminate the deleterious effects of internal covariate shift. Standardizing the activations of the previous level means that the assumptions that the next level makes about the variance and distribution of inputs during the weight update will not change, at least not dramatically. This stabilizes and speeds up the learning process for deep neural networks.

Batch normalization acts to standardize only the mean and variance of each unit to stabilize training, but allows the relationship between units and the non-

linear statistics of an individual unit to be altered. Normalizing the input for the layer affects the training of the model, dramatically reducing the number of epochs required. It can also have a regularizing effect, reducing generalization error in much the same way as using activation regularization.

Batch normalization can significantly affect optimization performance, especially for convolutional networks and networks with sigmoidal nonlinearity. Although “internal covariance shift” was the motivation for developing the method, there are some suggestions that batch normalization is effective instead, since it smooths out and in turn simplifies the optimization function that is solved when training the network. It fundamentally affects network training: it greatly simplifies the solution of the corresponding optimization problem. This ensures, in particular, that the gradients are more predictable and thus allow for a wider range of learning rates and faster network convergence. [14]

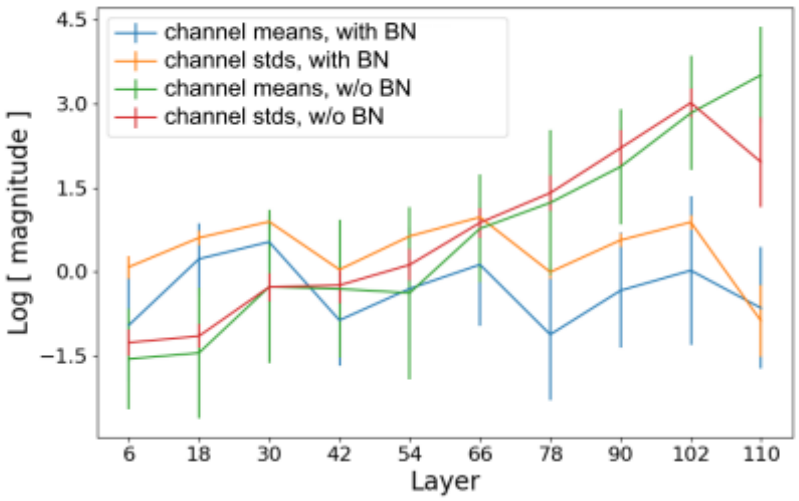

Fig. 3.2. Batch normalization effect on channel magnitude

There are various reasons why Batch Norm is believed to affect all of this. Here we will reveal the intuition of the most important reasons.

First, we can see how normalizing the input to produce a similar range of values can speed up learning. One simple intuition is that the Batch Norm does the same with the values in the mesh layers, not just the inputs.

Second, in his original paper that introduces the concept of batch normalization, it is argued that it reduces the internal covariance shift of the network. A covariance shift is a change in the distribution of data. Internal covariance shift is a change in the input distribution of the inner layer of the neural network. For neurons in the inner layer, the input received (from the previous layer) is constantly changing. This is due to the many calculations done before him and the weight of the training process.

Applying a batch norm ensures that the mean and standard deviation of the layer's inputs always remain the same; \ beta and \ gamma respectively. Thus, the number of changes in the distribution of the input layers is reduced. Deeper layers have a more reliable basis for what the input values will be, which helps in the learning process.

Finally, it looks like Batch Norm has a regularizing effect. Since it is computed in minibatch rather than across the entire dataset, there is some noise in the distribution of the model's data every time. It can act as a regularizer that can help overcome overfitting and help you learn better. However, the added noise is pretty low. Thus, it is usually not sufficient for proper regularization on its own and is usually used in conjunction with Dropout.

Burst rate works in a very similar way in convolutional neural networks. While we could do it the same way as before, we must respect the convolution property. In convolutions, we have generic filters that traverse the feature maps of the input data (in images, the feature map usually represents the height and width). These filters are the same on every feature map. Then it is wise to normalize the output in the same way by sharing it with function maps. In other words, this means that the parameters used for normalization are calculated along with each feature map as a whole. In a typical Batch Norm, each function will have a different mean and standard deviation. Here, each feature map will have a single mean and standard deviation used for all the features it contains. [15]

3.2. Convolution kernel

Combination can be used to reduce the selection of the contents of feature maps, reduce their width and height, while maintaining their specific features. The problem with deep convolutional neural networks is that the number of feature maps often increases with the depth of the network. This problem can dramatically increase the number of parameters and calculations required when using larger filters, such as 5 × 5 and 7 × 7.

To solve this problem, a 1 × 1 convolutional layer can be used, which offers channel merging, often referred to as feature map merge or projection layer. This simple technique can be used to reduce dimensionality by reducing the number of feature maps while maintaining their signature. It can also be used directly to create a unambiguous projection of feature maps to merge features by channels or to increase the number of feature maps, for example, after traditional merging of layers.

With the astonishing success of AlexNet in 2012, the revolution of the convolutional neural network (CNN) began. CNN-based frameworks in deep learning, such as GoogleNet, ResNet, and several variants, have shown impressive results in object detection and semantic segmentation in computer vision. When you start looking at most of today's successful CNN architectures, such as GoogleNet, ResNet, and SqueezeNet, you'll find that the 1X1 convolution layer plays an important role. At first glance, it seems pointless to use a single digit to convert the input image (after all, wider filters, such as 3X3, 5X5, can work with a fragment of the image, rather than a single pixel in this case). However, the 1X1 convolution has proven to be an extremely useful tool, and when used properly, it will help create surprisingly deep architectures.[16]

Recall that a convolutional operation is the linear application of a smaller filter to a larger input, resulting in an output feature map. A filter applied to an input image or input function map always gives one number. Systematic filtering

of the input data from left to right and top to bottom results in a 2D feature map. One filter creates one matching function map.

The filter must have the same depth or number of channels as the input, but regardless of the depth of the input and filter, the resulting output is one number, and one filter creates a feature map with one channel. Let's make it concrete with a few examples:

- If the input has one channel, such as a grayscale image, then the 3x3 filter will be applied in 3x3x1 blocks.

- If the input image has three channels for red, green and blue, then a 3x3 filter in 3x3x3 blocks will be applied.

- If the input is a block of feature maps from another convolutional or merging layer and has a depth of 64, then a 3x3 filter will be applied in 3x3x64 blocks to generate single values to create a single output feature map.

The output depth of one convolutional layer is determined only by the number of parallel filters applied to the input.

The entry depth, or the number of filters used in convolutional layers, often increases with the depth of the network, leading to an increase in the number of resulting feature maps. This is a common design pattern for models. In addition, some network architectures, such as the initial architecture, can also combine output feature maps from multiple convolutional layers, which can also greatly increase the insertion depth for subsequent convolutional layers.

A large number of feature maps in a convolutional neural network can cause a problem, since the convolutional operation must be performed at the entry depth. This is a particular problem if the convolutional operation being performed is relatively large, for example 5 × 5 or 7 × 7 pixels, as this can lead to significantly more parameters (weights) and, in turn, computations to perform convolutional operations (large space-time complexity). Merge layers are designed to scale down feature maps and systematically halve the width and height of feature maps on the network. However, merge levels do not change the

number of filters in the model, the depth, or the number of channels. Deep convolutional neural networks require a layer of an appropriate type of union that can downsample or reduce the depth or number of function maps.

The solution is to use a 1 × 1 filter to reduce the depth or number of feature maps. A 1 × 1 filter will only have one parameter or weight for each input channel, and like any filter, it produces one output value. This structure allows the 1 × 1 filter to act as a separate neuron with input from the same position on each of the input feature maps. This single neuron can then be applied systematically in one step, from left to right and top to bottom, without the need for padding, resulting in a feature map with the same width and height as the input.

The 1 × 1 filter is so simple that it doesn't use adjacent pixels in the input; it cannot be considered a convolutional operation. Instead, it is linear weighting or input projection. In addition, non-linearity is used, as for other convolutional layers, which allows the projection to perform non-trivial computations on the input property maps. This simple 1 × 1 filter provides a useful way to generalize the input function maps. Using multiple 1 × 1 filters, in turn, allows you to customize the number of input function map summaries generated, effectively allowing you to increase or decrease the depth of function maps as needed.

Therefore, a 1 × 1 convolutional layer with a filter can be used at any point in the convolutional neural network to control the number of feature maps. As such, it is often referred to as a projection operation or projection layer, or even a feature map or channel merge layer. [17]

3.3. Channel learning features

A convolutional neural network, or CNN for short, is a specialized type of neural network model designed to work with 2D image data, although they can be used with 1D and 3D data. The centerpiece of a convolutional neural network is

the convolutional layer, which gives the network its name. This layer performs an operation called convolution.

In the context of a convolutional neural network, convolution is a linear operation that involves multiplying a set of weights with the input, as in a traditional neural network. Given that the method was designed for two-dimensional input, the multiplication is performed between the input data array and a two-dimensional array of weights called a filter or kernel. The filter is smaller than the input, and the type of multiplication applied between the slice of the input size with the filter size and the filter is a dot product. A dot product is an element-wise multiplication between a chunk of the input signal and a filter-sized filter, which is then added, always resulting in a single value. Since the result is a single value, the operation is often referred to as a "dot product".

Using a filter that is smaller than the input filter is intentional because it allows the same filter (set of weights) to be multiplied multiple times by the input array at different entry points. In particular, the filter is applied systematically to each overlapping portion or chunk with the filter size of the input data from left to right, top to bottom. This systematic application of the same filter to an image is a powerful idea. If the filter is designed to detect a certain type of function in the input data, then applying this filter systematically throughout the input image allows the filter to detect this function anywhere in the image. This capability is commonly referred to as translational invariance, for example the general interest in whether a feature is present rather than where it is present.

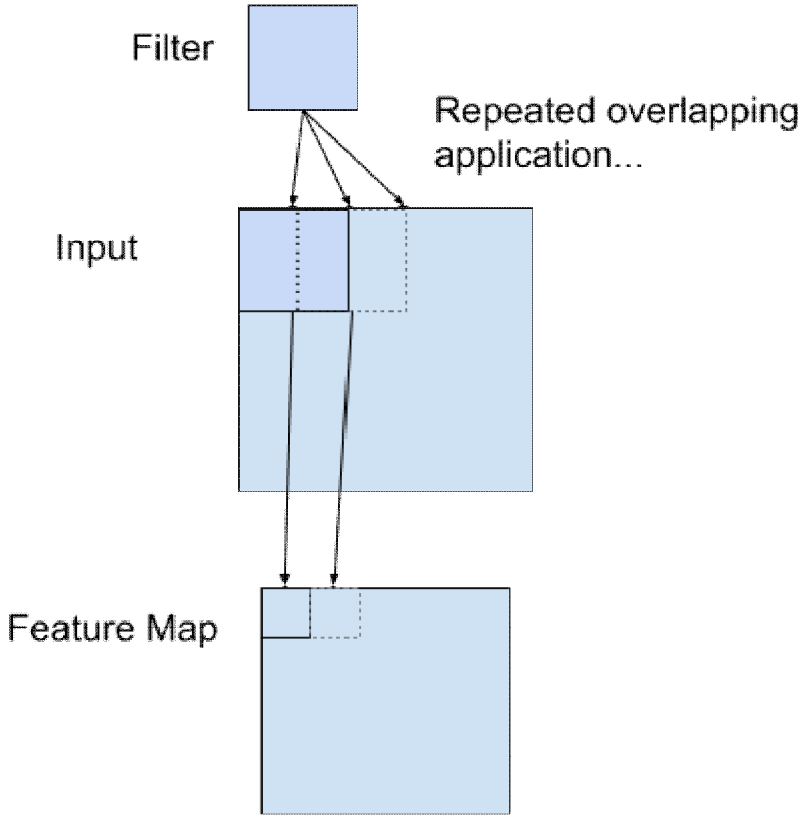

Fig. 3.3. Filter application in process of creation of feature map

Local translation invariance can be a very useful property if we care more about whether a feature is present than where it is located. For example, when determining if an image contains a face, we don't need to know the location of the eyes to the pixel, we just need to know that the eye is on the left side of the face and the eye is on the right. side of the face. The result of a single multiplication of the filter by the input array is one value. Because the filter is applied multiple times to the input array, the result is a two-dimensional array of outputs that represent the filtering of the input. Thus, the two-dimensional output array of this operation is called a "feature map". Once the feature map has been created, we can pass each value in the feature map through a nonlinearity like ReLU, in much the same way we do for the output of a fully connected layer.

If you are working in digital signal processing or a related field of mathematics, you can understand the convolution of a matrix as something different. In particular, the filter (kernel) is flipped before being applied to the

input. Technically, the convolution described using convolutional neural networks is actually "cross-correlation". However, in deep learning this is called a "convolution" operation. Many machine learning libraries implement cross-correlation, but they call it convolution. So we have an input such as an image of pixel values, and we have a filter, which is a set of weights, and the filter is systematically applied to the input to create a feature map.

Convolution in computer vision. The idea of applying a convolutional operation to image data is neither new nor unique to convolutional neural networks; it is a common technique used in computer vision. Historically, filters were designed by hand by computer vision experts who were then applied to an image to produce a feature map or filter result, which somewhat simplifies image analysis.

For example, the following is a hand-crafted 3 × 3 element filter for vertical line detection:

0.0, 1.0, 0.0

0.0, 1.0, 0.0

0.0, 1.0, 0.0

Applying this filter to an image will result in a feature map containing only vertical lines. This is a vertical line detector. You can see this by looking at the weights in the filter; any pixel values in the center vertical line will be positively activated, and any on either side will be negatively activated. If you systematically drag this filter through the pixel values in the image, only the pixels of the vertical line can be selected. You can also create a diagonal line detector and apply it to an image, for example:

0.0, 0.0, 1.0

0.0, 1.0, 0.0

1.0, 0.0, 0.0

And another one:

0.1, 0.0, 0.0

0.0, 1.0, 0.0

0.0, 0.0, 1.0

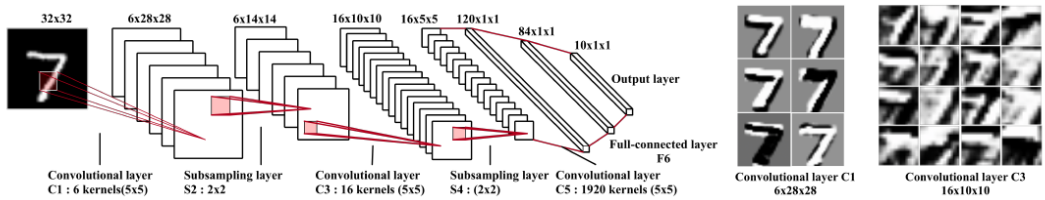

Fig. 3.4. Example of feature map in CNN application

Combining the results of both filters, for example combining both feature maps, will select all graphic lines. A collection of dozens or even hundreds of other small filters can be designed to detect other features of the image. The novelty of using convolution in a neural network is that the filter values are weights that need to be learned while training the network. The network learns what types of functions to extract from the input. In particular, when learning with stochastic gradient descent, the network is forced to learn to extract functions from the image that minimize losses for a specific task that the network is trained to solve, for example, extract functions that are most useful for classifying images.

Convolutional neural networks do not learn a single filter; in fact they are learning several functions in parallel for a given input. For example, a convolutional layer typically learns 32 to 512 filters in parallel for a given input. This gives the model 32 or even 512 different ways to extract features from the input, or many different ways to both "learn to see" and after training, many different ways to "see" the input. This variety allows for specialization, for example, not just strings, but specific strings that are visible in your specific training data.

Color images have multiple channels, usually one for each color channel, such as red, green, and blue. In terms of data, this means that one image provided as input to the model is actually three images. The filter should always have the

same number of channels as the input, often referred to as "depth". If the input image has 3 channels (for example, depth 3), then the filter applied to that image must also have 3 channels (for example, depth 3). In this case, a 3x3 filter would actually be 3x3x3 or [3, 3, 3] for rows, columns, and depth. Regardless of the depth of the input and the depth of the filter, the filter is applied to the input using a dot product operation that results in a single value.

This means that if the convolutional layer has 32 filters, these 32 filters will not only be 2D for the input 2D image, but also 3D, having specific filter weights for each of the three channels. However, each filter results in a single feature map. This means that the depth of inference of applying a convolutional layer with 32 filters is 32 for the generated 32 function maps.

Convolutional layers are applied to more than just the input data, for example. raw pixel values, but they can also be applied to the output of other layers. Overlaying convolutional layers allows hierarchical decomposition of input data. Filters that work directly with raw pixel values will learn to extract low-level features such as lines.

Filters that work with the output of the first line layers can extract items that are combinations of lower-level items, such as items that contain multiple lines to express shapes. This process continues until faces, animals, houses, and so on are removed from very deep layers. Thus, we observe the abstraction of functions to a higher and higher order as the depth of the network increases [18].

3.4. Architecture synthesis

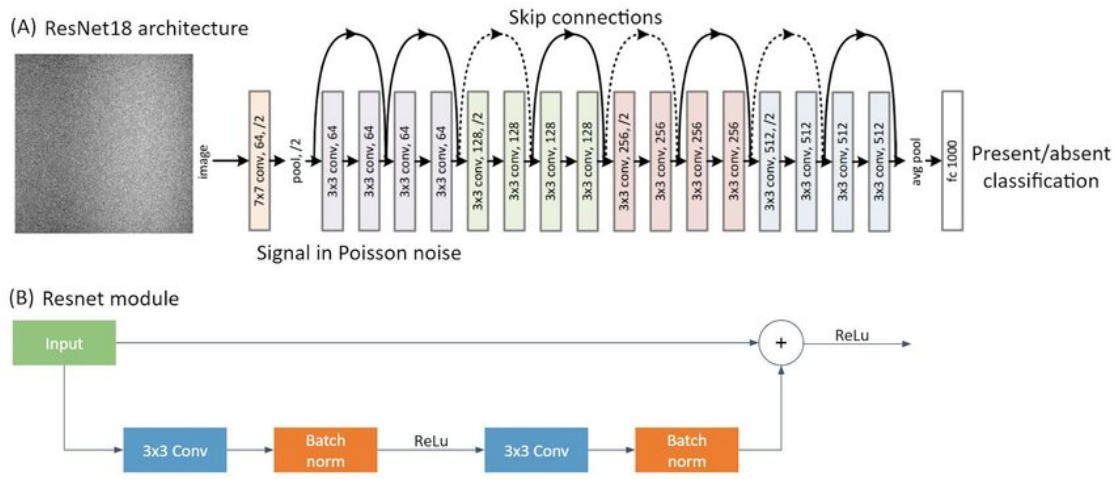

Fig. 3.5. Example of RCCNs architectures

The residual neural network (ResNet), built from the corresponding blocks, is a creation of the team of creators of the Pytorch library, and is a collection of several networks (named by the number of layers, for example, ResNet-18), united by a common architecture topology. An example of such a network can be seen in Fig. 3.4.1.

layer	layer parameters
input	220x220x3 resolution
pooling	3x3 mask
convlolutional	2x2, [32] 2x
dropout	0,1 prob.
convlolutional	2x2, [64] 2x
pooling	3x3 mask
convlolutional	2x2, [128] 2x
pooling	3x3 mask
convlolutional	2x2, [256] 2x
pooling	3x3 mask
convlolutional	2x2, [512] 2x
obal pooling	3x3
ly-connected	5 outputs (5 classes)
sofmax classifcation	5 outputs

Fig. 3.6. Typical residual network architecture and characteristics

A neural network synthesized on the basis of residual blocks, used as a baseline for evaluating performance, is shown in Fig. 3.4.2. Its difference from the standard ResNet-18 is the smaller number of convolutional layers and the presence of additional convolutional filters at the output of each of the sub-

blocks, which provide additional aggregation of features with further compression of the feature map, the ultimate goal of these improvements is to improve performance by reducing the number of calculated parameters.

Further improvement of the network is carried out by including attention blocks in its structure, which are two fully connected layers, between which there is an amplifying filter with a tangential function of activation, as well as a mastic filter at the output. The number of layers in each fully connected layer corresponds to the number of feature maps on the corresponding convolutional layers of the PCNN sub-block

ResNeXt is a variant of ResNet codenamed ResNeXt with the following building block:

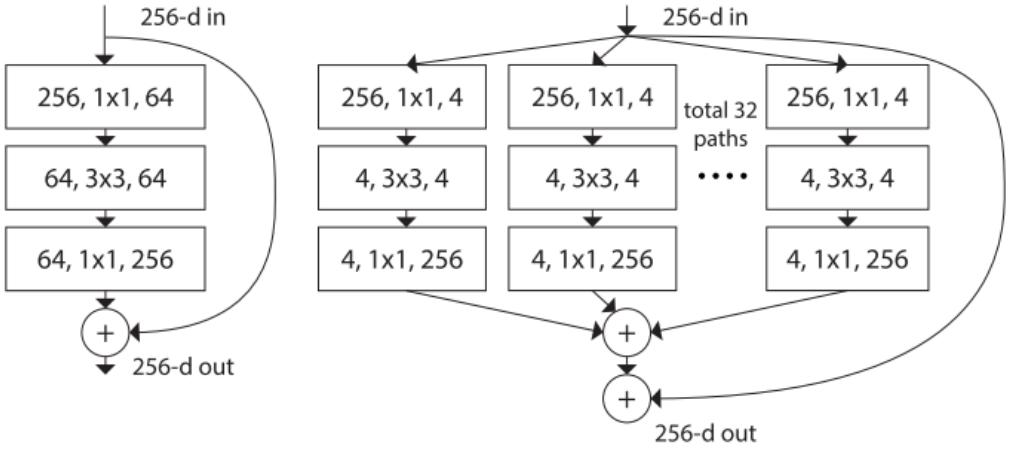

Fig. 3.7. ResNeXt principal structure

It is very similar to the Inception module, they both follow the split-transform-merge paradigm, except this option, the outputs of different paths are combined by adding them. Another difference is that in the inception, each path is different (1x1, 3x3, and 5x5 convolution) from each other, whereas in this architecture all paths use the same topology. The authors introduced a hyperparameter called cardinality - the number of independent paths - to provide a new way to tune the capacity of the model. Experiments show that accuracy can be achieved more efficiently by increasing power than by deepening or expanding. The authors state that compared to Inception, this new architecture is easier to adapt to new datasets / tasks as it has a simple paradigm and only one

hyperparameter is tunable, while Inception has many hyperparameters (like the kernel size of the convolutional layer of each path) for settings.

Tightly connected CNN is a proposed new architecture called DenseNet that further exploits the effects of fast connections - it connects all layers directly to each other. In this new architecture, each layer's input is composed of feature maps of all earlier layers, and its output is passed to each subsequent layer. Function maps are aggregated with depth concatenation.

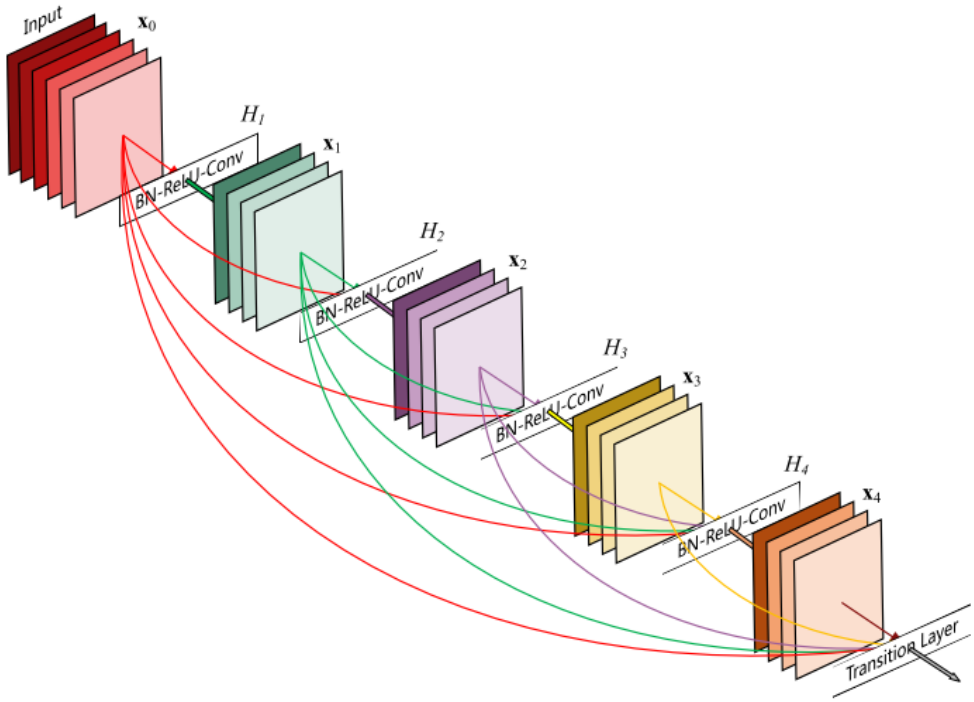

Fig. 3.8. DenseNet topology

In addition to solving the vanishing gradient problem, the authors of [8] argue that this architecture also encourages reuse of functions, which makes the network very efficient in terms of parameters. One simple interpretation of this is that the output of the identity mapping has been added to the next block, which could hinder the flow of information if the feature maps of the two layers have very different distributions. Consequently, combining feature maps can preserve them all and increase the variability of the output, encouraging reuse of features.

Deep network with stochastic depth. While ResNet has proven to be effective in many applications, one of the major drawbacks is that a deeper network usually takes weeks to learn, making it nearly impossible in real-world

applications. To solve this problem, an illogical method of randomly dropping layers during training and using the entire network for testing was introduced. The authors used a residual block as the building block of their network, so during training, when a particular residual block is turned on, its input goes through both the ID tag and the weight layers, otherwise the input only goes through the ID tag. During training, each layer has a “probability of survival” and is randomly discarded. During testing, all blocks remain active and are recalibrated according to the probability of survival during training.

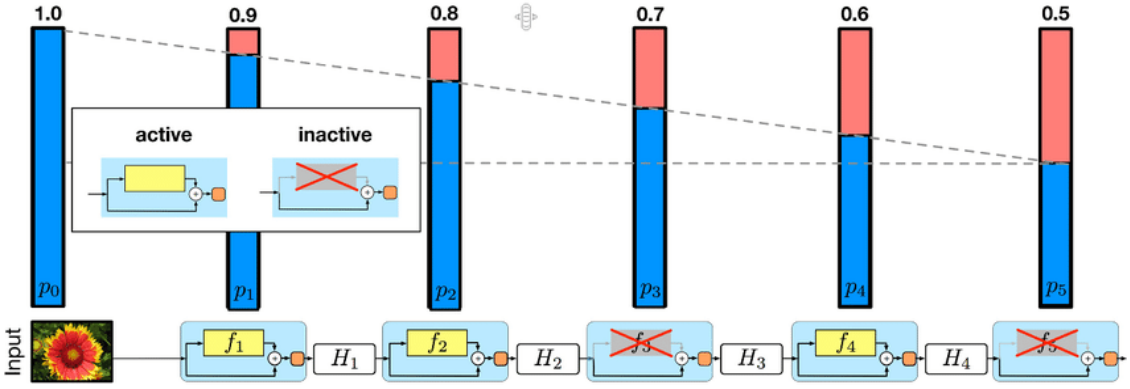

Fig. 3.9. Deep network gradient fading

Like Dropout blocks, training a deep network with stochastic depth can be thought of as training many smaller ResNet's. The difference is that this method randomly removes the entire layer, while Dropout only removes some of the hidden units in one layer during training. Experiments show that training ResNet with 110 layers of stochastic depth results in better performance than training ResNet with a constant depth of 110 layers, while training time is significantly reduced. This suggests that some levels (paths) in ResNet may be redundant.

Chapter 4

Neural network

4.1. Structure of network and program

Structurally, the program is a set of blocks and links in the Matlab environment, which can be transferred to the working environment in the form of code, along with all the parameters, using the built-in translator. The example of such scheme are shown in Fig. 4.1. below:

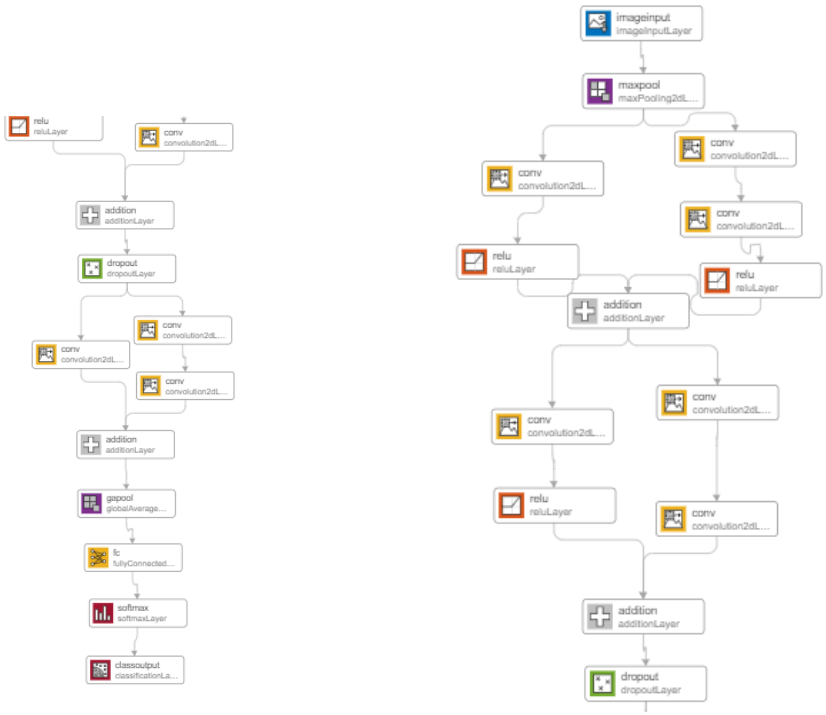

Fig. 4.1 Example of basic CNN structure implementation

In form of MATLAB code it will looks like (fragment below):

```
tempLayers = [
    imageInputLayer([227 227 3],"Name","imageinput")
    averagePooling2dLayer([5
5],"Name","avgpool2d_1","Padding","same")];
lgraph = addLayers(lgraph,tempLayers);
tempLayers = [
    globalAveragePooling3dLayer("Name","gapool3d_2")
```

```
    fullyConnectedLayer(10,"Name","fc_3")
    reluLayer("Name","relu_2")
    fullyConnectedLayer(10,"Name","fc_4")
    scalingLayer("Name","scaling_2")
    convolution2dLayer([3 3],64,"Name","conv_1","Padding","same")
    convolution2dLayer([3 3],64,"Name","conv_2","Padding","same")];
lgraph = addLayers(lgraph,tempLayers);
```

Thus, we can change its parameters after translation by writing new instead of already set.

4.2. Input and output data

Suppose you are working with the MNIST dataset and you know that each image in MNIST is 28 x 28 x 1 (black and white image contains only 1 channel). The total number of neurons in the input layer will be 28 x 28 = 784, this can be controlled. What if the image size is 1000 x 1000, which means you need 10□ neurons in the input layer. Oh! It seems to require a huge number of neurons to work. This is computationally correct. So here comes the Convolutional Neural Network or CNN. Simply put, CNN extracts a feature of an image and transforms it into a lower dimension without losing its characteristics. In the following example, you can see that the original image size is 224 x 224 x 3. If you continue without convolution, you need 224 x 224 x 3 = 100, 352 number of neurons in the input layer, but after applying convolution you enter the tensor dimension reduced up to 1 x 1 x 1000. This means you only need 1000 neurons in the first layer of the feedforward neural network. Example of dataset classification organization shown in fig. 4.2 below:

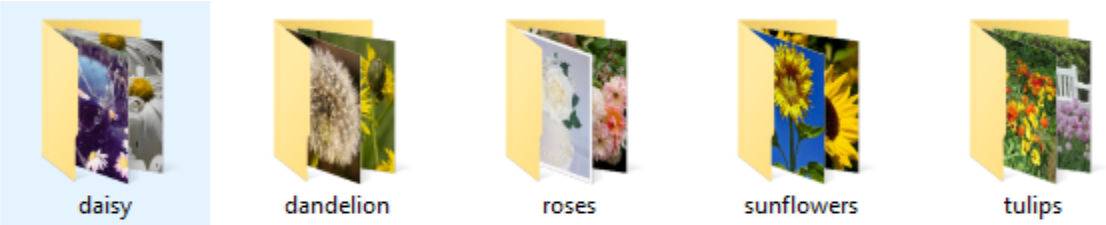

Fig. 4.2. Dataset example

Thinking about images it is easy to understand that they have height and width, so it would be wise to represent the information they contain in a two-dimensional structure (matrix), until you remember that images have colors and add color information, we need another dimension, and that's when tensors become especially useful. Images are encoded into color channels, the image data is represented in each color intensity in a color channel at a given point, the most common of which is RGB, which stands for red, blue and green. The evolution of signals through CNN shown below in fig. 4.3:

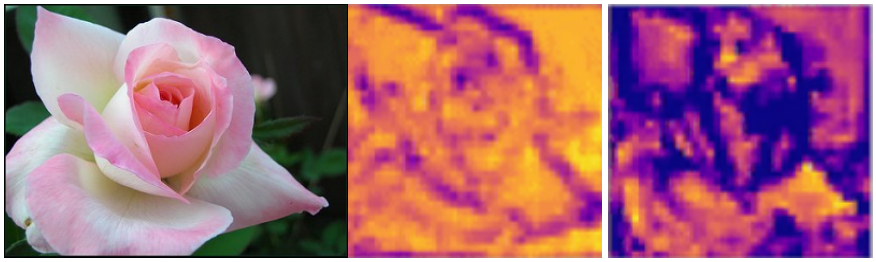

Fig. 4.3. Original image and feature map output on convolutional layer example

If you look at different images from the convolution layer filters, it's pretty clear how different filters in different layers try to highlight or activate different parts of the image. Some filters act as edge detectors, others detect a specific area of the flower, such as the center of the flower, and still others act as background detectors. This behavior of convolutional layers is easier to see in the initial layers, because as you go deeper, the pattern captured by the convolution kernel becomes more and more sparse, so it could be that such patterns may not even exist in your image, and therefore it will not be captured.

After taking a closer look at these filter images from different convolution layers, it becomes very clear that the different layers are actually trying to learn

from the image data provided to them. The patterns found in the filters in the initial layers seem to be very simple, composed of lines and other basic shapes that tell us that earlier layers learn about the basic functions of images like edges, colors, etc., templates become more complex, assuming that the deeper layers are actually learning much more abstract information that helps those layers generalize classes rather than a specific image. And that is why, in the previous section, we saw several activations of empty filters in deeper layers, because this particular filter was not activated for this image, in other words, the image has no information that the filter was interested in.

4.3. Interface description

Deep learning toolbox are extension module for the standard MATLAB modeling environment. It is a separate subroutine that runs in the "applications" section, called “deep network desinger”. It allows you to both load ready-made networks and create your own based on blocks. See fig. 4.4:

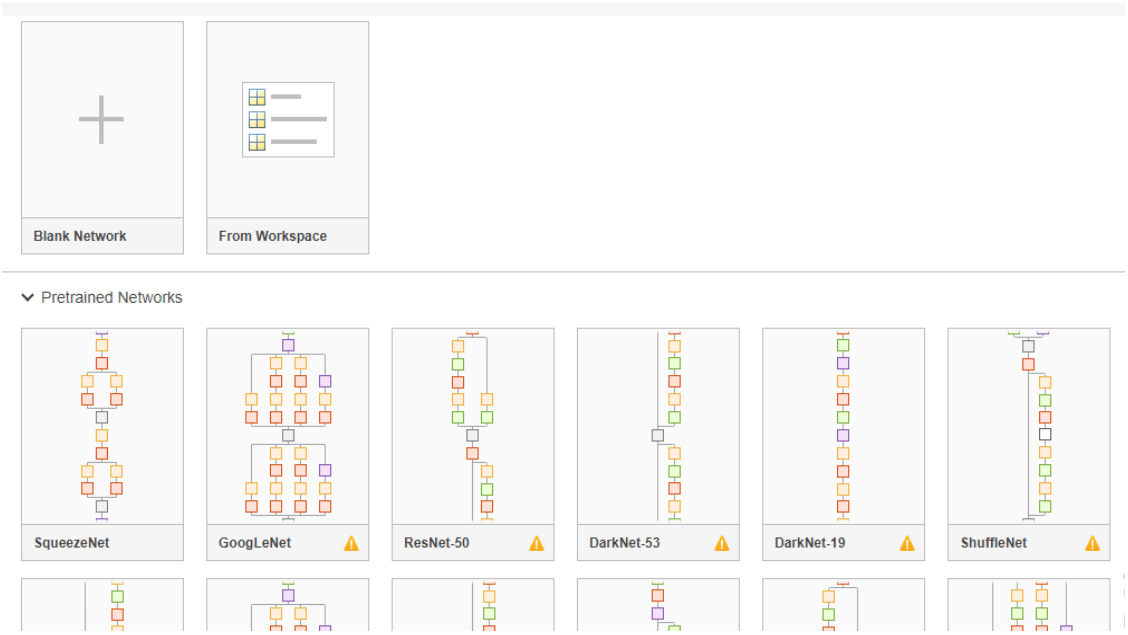

Fig. 4.4. Example of pre-trained neural network list

The already prepared blocks can be gragged from left panel, as shown in fig. 4.5:

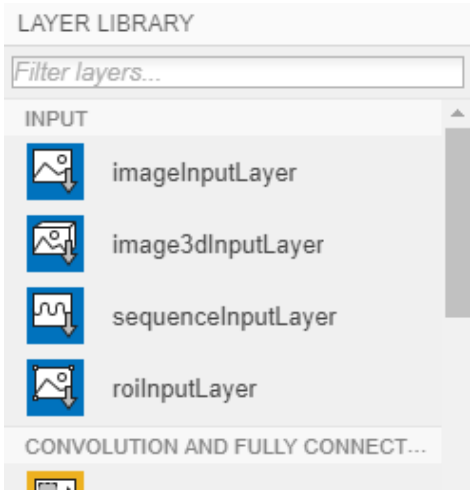

Fig. 4.5.. A part of building blocks list

And then they are individually configured in accordance with the required parameters on the panel on the right in the workspace, which appears when th desired block is selected, example on fig. 4.6:

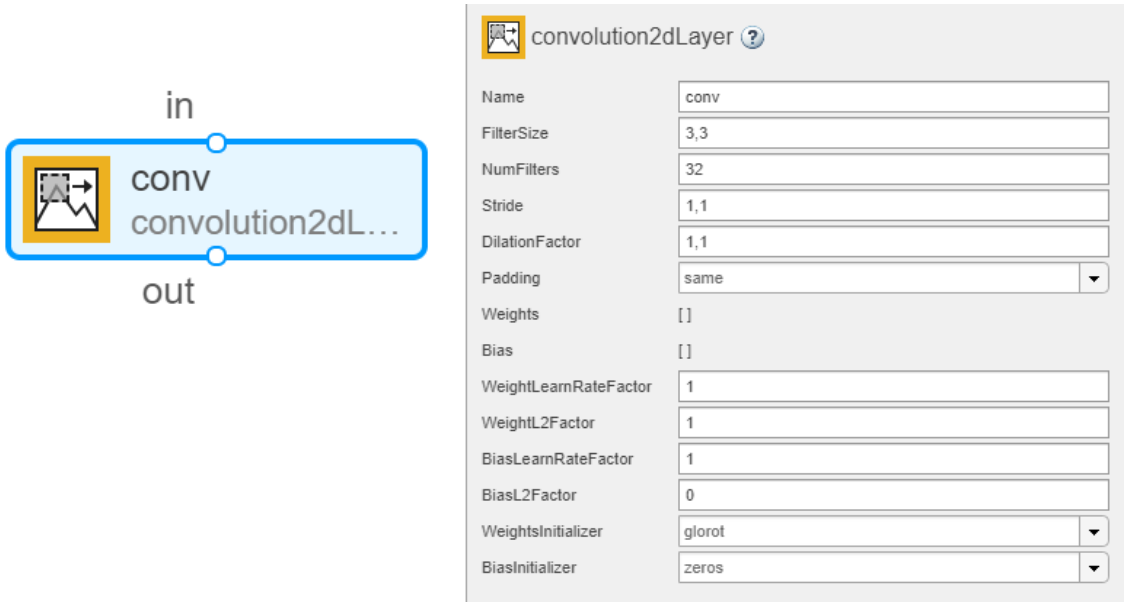

Fig. 4.6. Layer properties

A separate tab allows you to immediately select the path to the desired dataset and determine the percentage of the training sample that will be randomly selected from it, see fig. 4.7:

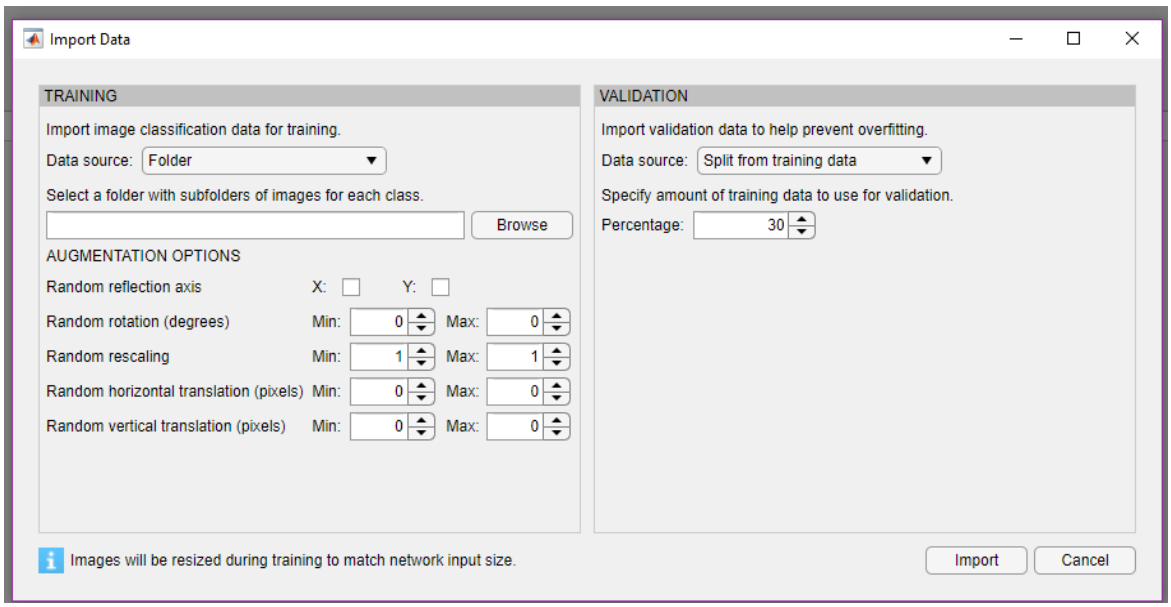

Fig. 4.7. Dataset configuration

The learning window allows you to select the required learning algorithm and its parameters. See fig. 4.4:

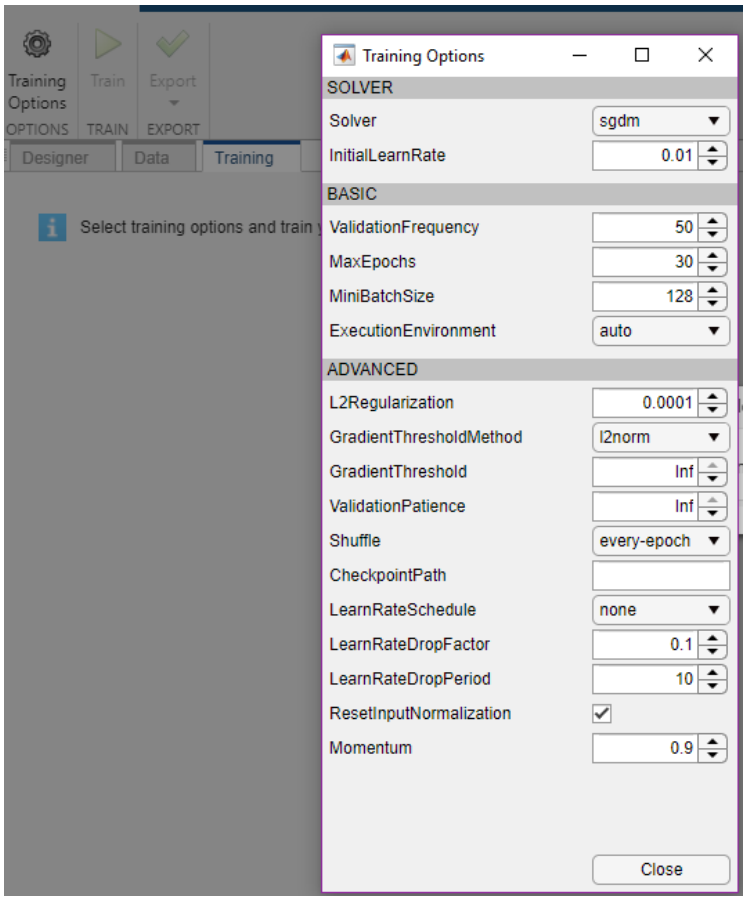

Fig. 4.8. Training data configuration

After verification and training, the code along with the network parameters can be exported to the main work area of MATLAB, as shown in fig. 4.9:

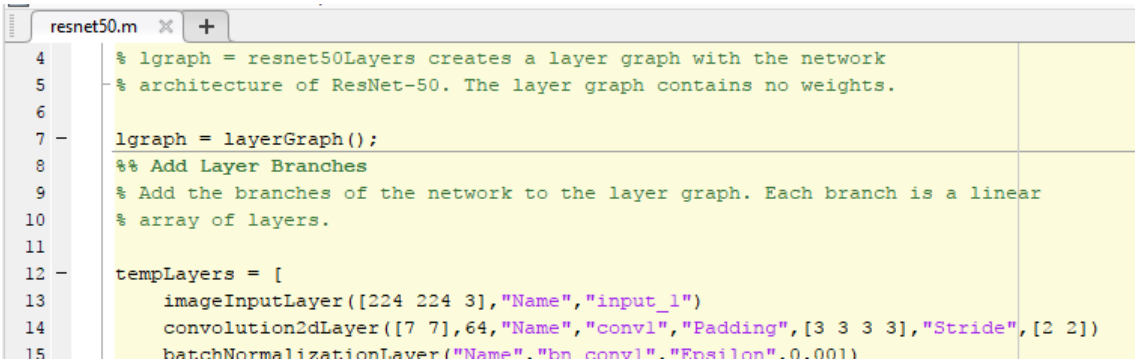

Fig. 4.9. Example of exported code in MATLAB workspace.

4.4. Network training result example

The assessment of the classes of the incoming image is carried out by the built-in tools of the MATLAB program - which allow both to independently carry out verification statistical processing, and to display individual results of this in the form of a table of probability of belonging. An example of such a result can be seen in the figure 4.10 below;

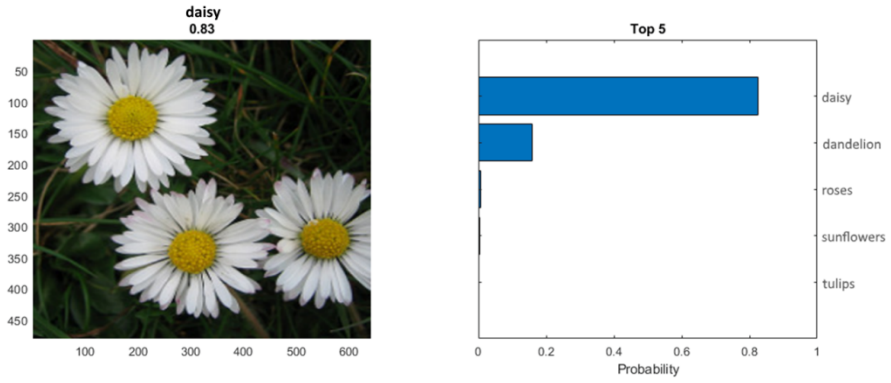

Fig. 4.10. Squeeze-and-excitation and attention module and improved neural network structure

CONCLUSIONS

1. In such work using the modern technical approaches for data processing there was chosen the CPU-base method for neural network processing.
2. Also was chosen for NN programming graphical interface of MATLAB Deep Learning Toolbox
3. The task set at the beginning of the work was to optimize the structure of the neural network, the numerical performance indicators of which would be comparable to existing solutions. The task as a whole has been completed, at the same time, a further need to study the influence of the structure of attention modules on the efficiency of the network as a whole is outlined, which requires a separate study.
4. An analysis of the accuracy showed that the results of hybrid neural network results quite close to existed solution with less sophisticated structure, but more expensive but with a large total number of calculated parameters.

REFERENCES

1. Dan C. Cireësan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, "Flexible, High Performance Convolutional Neural Networks for Image Classificatio", JuËrgen Schmidhuber. IDSIA, USI and SUPSI Galleria 2, 6928 Manno-Lugano, Switzerland, pp 1237-1238.
2. Zenghui Wang, Waseem Rawat, "Deep Convolutional Neural Networks for Image Classifcation: A Comprehensive Review", URL: https://www.researchgate.net/publication/317496930_Deep_Convolutional_Neural_Networks_for_Image_Classification_A_Comprehensive_Review , (date of access: 10.06.2021).
3. Yu Han, "Feature Extraction and Image Recognition with Convolutional Neural Networks", URL: https://iopscience.iop.org/article/10.1088/1742-6596/1087/6/062032/pdf (date of access: 10.06.2021).
4. Jianxin Wu, "Introduction to Convolutional Neural Networks", 2017, LAMDA Group National Key Lab for Novel Software Technology, Nanjing University, pp 5-24.
5. Aleix M.Matine, Avinash C.Kak, "PCA versus LDA", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.23, No. 2, February 2001.
6. Intro to optimization in deep learning: Momentum, RMSProp and Adam, URL: https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/ (date of access: 10.06.2021).
7. Dan C. Cireësan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, JuËrgen Schmidhuber, "High-performance neural networks for visual object classification", 2011
8. Ian Goodfellow, Yoshua Bengio, Aaron Courville, "Deep Learning", The MIT Press, 2016, pp. 74-82
9. M. Seetha, I.V. Muralikrishna, B.L. Deekshatulu B..L. Malleswari, P. Hegde, "Artificial neural networks and other methods of image classification" URL: http://www.jatit.org/volumes/research-papers/Vol4No11/5Vol4No11.pdf (date of access: 10.06.2021).
10. Sakali Mustafa, Lam Kin-Man, Yan Hong, "A faster converging snake algorithm to locate object boundaries, IEEEtransactions on imageprocessing" ,Vol.15, No. 5, 2006.
11. Jiuxiang Gua, Zhenhua Wangb, Jason Kuenb, Lianyang Mab, Amir Shahroudyb, Bing Shuaib, Ting Liub, Xingxing Wangb, Li Wangb, Gang Wangb, Jianfei Caic, Tsuhan Chenc, "Recent Advances in Convolutional Neural Networks" URL: https://arxiv.org/pdf/1512.07108.pdf%C3%A3%E2%82%AC%E2%80%9A (date of access: 10.06.2021).

12. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep Residual Learning for Image Recognition”, URL: https://arxiv.org/abs/1512.03385 (date of access: 10.06.2021).
13. Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, Quoc V. Le, “Attention Augmented Convolutional Networks” URL: https://arxiv.org/abs/1904.09925 (date of access: 10.06.2021).
14. Options for training deep learning neural network, URL: https://www.mathworks.com/help/deeplearning/ref/trainingoptions.html , (date of access: 10.06.2021).
15. Kevin Gurney, “An introduction to neural networks”, 1997, CRC Press, pp. 17-20.
16. Charu C. Aggarwal, “Neural Networks and Deep Learning: A Textbook”, 2018
17. Ian Goodfellow, Yoshua Bengio, Aaron Courville, “Deep Learning”, The MIT Press, 2016
18. Amer Zayegh, Nizar Al Bassam, “Neural Network Principles and Applications”, URL: https://www.researchgate.net/publication/329264107_Neural_Network_Principles_and_Applications (date of access: 10.06.2021).