

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних систем та мереж

“ДОПУСТИТИ ДО ЗАХИСТУ”
Завідувач кафедри

_____ Жуков І.А.

“ _____ ” _____ 2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

випускника освітнього ступеня “МАГІСТР”
спеціальності 123 «Комп'ютерна інженерія»
освітньо-професійної програми «Комп'ютерні системи та мережі»

на тему: “ **Застосування машинного навчання для розпізнавання та ідентифікації
облич**”

Виконавець: _____ Аланно А.А.А.

Керівник: _____ Телешко І.В.

Нормоконтролер: _____ Надточій В.І.

Засвідчую, що у дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань

Аланно А.А.А

Київ 2020

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY
Faculty of Cybersecurity, Computer and Software Engineering
Computer Systems and Networks Department

“PERMISSION TO DEFEND GRANTED”

The Head of the Department

_____ Zhukov I.A.

“ _____ ” _____ 2020

MASTER’S DEGREE THESIS

(EXPLANATORY NOTE)

Specialty: 123 Computer Engineering

Educational-Professional Program: Computer Systems and Networks

Topic: “Machine learning implementation in face recognition and identification”

Completed by: _____ Alanno A.A.A.

Supervisor: _____ Teleshko I.V.

Standard’s Inspector: _____ Nadtochii V.I.

Kyiv 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних систем та мереж

Освітній ступінь: «Магістр»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерні системи та мережі»

“ЗАТВЕРДЖУЮ”

Завідувач кафедри

Zhukov I.A.

“ _____ ” _____ 2020 р.

ЗАВДАННЯ

на виконання дипломної роботи

Аланно Абдуллах Алі Абдулкарім

(прізвище, ім'я та по-батькові випускника в родовому відмінку)

1. Тема дипломної роботи: “Застосування машинного навчання для розпізнавання та ідентифікації облич” затверджена наказом ректора від 25.09.2020 р. № 1793/ст
2. Термін виконання роботи (проекту): з 1 жовтня 2020 р. до 25 грудня 2020 р.
3. Вихідні дані до роботи (проекту): Machine learning implementation in face recognition and identification
4. Зміст пояснювальної записки: Вступний огляд теми, огляд Штучної нейронної мережі в машинному навчанні та реалізація згорткової нейронної мережі у розпізнаванні обличчя та виявленні емоцій, висновок.
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: Матеріали представлені у вигляді презентації в Power Point.

NATIONAL AVIATION UNIVERSITY

faculty of Cybersecurity, Computer and Software Engineering

Department: Computer Systems and Networks

Educational Degree: “Master”

Specialty: 123 “Computer Engineering”

Educational-Professional Program: “Computer Systems and Networks”

“APPROVED BY”

The Head of the Department

“ _____ ” _____ Zhukov I.A.
_____ 2020 p.

Graduate Student’s Degree Thesis Assignment

Alanno Abdullah Ali Abdulkareem

1. Thesis topic: “Machine learning implementation in face recognition and identification” approved by the Rector’s order of 25.09.2020 p. № 1793/CT
2. Thesis to be completed between 01.10.2020 and 25.12.2020
3. Initial data for the project (thesis): *Machine learning implementation in face recognition and identification.*
4. The content of the explanatory note (the list of problems to be considered):
Introduction review of the topic, review of the Artificial neural network in machine learning filed and the implementation of Convolutional neural network in the face recognition and emotion detection, conclusion.
5. The list of mandatory graphic materials: *Graphic materials are given in MS Power Point presentation.*

6. Календарний план-графік

№ пор.	Завдання	Термін Виконання	Підпис керівника
1	Узгодити технічне завдання з керівником дипломної роботи	1.10.20- 8.10.20	
2	Виконати пошук та вивчення науково-технічної літератури за темою роботи	9.10.20- 15.10.20	
3	Опрацюйте теоретичний матеріал про штучну нейронну мережу в галузі машинного навчання та виберіть набір технологій	16.10.20- 18.10.20	
4	Розробити програму та провести тестування з метою переконання в тому, що вона працює так, як очікувалось	19.10.20- 03.11.20	
5	Виконати аналіз результатів програми, розробити рекомендації щодо її використання та оформити пояснювальну записку	04.11.20- 15.12.20	
6	Оформити графічну частину записки та подати матеріали роботи на антиплагіатну перевірку матеріалів	16.12.20- 05.12.20	
7	Отримати рецензію та відгук керівника. Надати матеріали роботи на кафедру	06.12.20- 12.12.20	

7. Дата видачі завдання: “1” жовтня 2020 р.

Керівник дипломної роботи _____ Телешко І.В.
(підпис керівника)

Завдання прийняв до виконання _____ Аланно А.А.А
(підпис випускника)

6. TIMETABLE

#	Completion stages of Degree Project (Thesis)	Stage Completion Dates	Signature of the supervisor
1	Technical task coordination with the supervisor	1.10.20- 8.10.20	
2	Selection and study scientific literature on the topic	9.10.20- 15.10.20	
3	Process theoretical material on Artificial Neural network in machine learning field and choose a stack of technologies	16.10.20- 18.10.20	
4	Develop a program and conduct testing to ensure that it works as expected	19.10.20- 03.11.20	
5	Analyze the results of the program, develop recommendations for its usage and make the explanatory notes	04.11.20- 15.12.20	
6	Preparation of the graphical materials and filing materials of work to antiplagiarism checking of materials	16.12.20- 05.12.20	
7	Receiving the reviews from reviewer and supervisor and providing the materials to the department	06.12.20- 12.12.20	

7. Assignment issue date: 1.10.2020

Diploma Thesis Supervisor _____ Teleshko I.V.
(Signature)

Assignment accepted for completion _____ Alanno A.A.A
(Student's Signature)

ABSTRACT

Explanatory note to the thesis " Machine learning implementation in face recognition and identification ": 111 pp., 69 figures,3 tables, 9 references.

Object of research: face recognition and emotion detection in image datasets and live webcam.

Purpose: to develop an application to recognize faces and emotion detection using machine learning.

Research methods: comparative analysis, processing of literature sources.

Obtained results: The emotion detection application was developed as a result of the project and it currently under development to reach high accuracy and minimum classification error. The main idea behind this model is to understand the result of tuning different components and layers of Convolutional Neural Network to reach the optimal result of the application. Further development could be for instance, to provide API to use in many sectors such as, public security and commercial interests to track human interests. A few examples like monitoring public CCTV cameras, immigration checks at airports can for be public security.

CONTENTS

LIST OF SYMBOLS, ABBREVEATIONS, TERMS	1
INTRODUCTION	2
1.1. Artificial Intelligence	3
1.1.1 Philosophy of AI	3
1.1.2 Goals of AI	3
1.2. Programming without and with AI.....	4
1.3 Applications of AI	5
1.4. Intelligence Definition	6
1.5. Difference between Human and Machine Intelligence.....	9
1.6. Research Areas	9
1.7. Advantages of using Artificial intelligence system	16
1.8. Disadvantages of using Artificial intelligence system.....	17
1.9. Challenges Faced by Artificial Intelligence	17
CONCLUSION ON THE FIRST PART	18
PART 2: WORKFLOW OF MACHINE LEARNING, MACHINE LEARNING TYPES, DATA SCIENCE, AND DEEP LEARNING	20
2.1. Machine Learning	20
2.2. Data Science	21
2.2.1. Branches of Data Science.....	22

2.2.2. Big Data.....	22
2.2.3. Data Analytics.....	23
2.3. Types of Data.....	24
2.4. Machine learning methods	25
2.4.1. Supervised Learning Algorithms.....	25
2.4.2. Unsupervised Learning Algorithms	28
2.4.3. Semi-supervised learning	31
2.4.4. Reinforcement learning	32
2.5. Categorizing on the basis of required output.....	33
2.6. Applications of Machine Learning	33
2.7. Origins of Deep Learning	34
2.7.1 Deep Learning.....	36
2.8. Difference Between Artificial Intelligence, Machine Learning, Deep Learning	36
2.9. Computer vision.....	37
CONCLUSION ON THE SECOND PART	41
PART 3: NEURAL NETWORK LAYERS, WORKING OF PERCEPTRON, CONVOLUTIONAL NEURAL NETWORK WORKFLOW	43
3.1. Neural Networks	43
3.2. Working of an Artificial Neuron (Perceptron)	45
3.2.1. ReLU	48

3.2.2. Sigmoid.....	49
3.2.3. Softmax.....	50
3.2.4. One Hot Encoding.....	50
3.3. Gradient Descent	51
3.4. Stochastic Gradient Descent	52
3.5. AdaDelta.....	53
3.6. Forward Propagation.....	54
3.7. Back Propagation.....	54
3.8. Types of Neural Networks	54
3.8.1. Single-Layer Neural Networks:.....	55
3.8.2. Multi-Layer Neural Networks	55
3.9. Recurrent Neural Networks	56
3.10. Modular Neural Networks	57
3.11. Convolutional Neural Networks.....	57
3.11.1. Implementation of Convolutional Neural Network	58
3.11.2. Convolution Layer	61
3.11.3. The objective of the Convolution Operation	62
3.11.4. Strides	63
3.11.5. Pooling Layer.....	63
3.11.6. Classification — Fully Connected Layer (FC Layer).....	65

3.12. Tune hyperparameter in Deep Neural Network.....	66
3.13. Important terms related to Convolutional Neural Network	68
3.13.1. Overfitting.....	68
3.13.2. Batch.....	69
3.13.3. Epoch.....	69
3.13.4. The Difference Between Batch and Epoch.....	70
3.13.5. Dense layer	71
3.13.6. Normalization	71
CONCLUSION ON THE THIRD PART	72
PART 4: PRACTICAL SIDE, IMPLEMENTING CNN FAMOUS MODEL TO DETECT FACE EMOTIONS	74
4.1. TensorFlow.....	74
4.2. Keras.....	75
4.3. Popular CNN Architecture.....	76
4.4. Introducing the concept of Transfer Learning and Fine-Tuning.....	82
4.5. Data Augmentation.....	84
4.6. Confusion matrix	84
4.7. Classification Report Analysis.....	88
4.8. Batch Normalization	88
4.9. The Dying ReLU Problem	90

4.10. Implementing CNN for detecting face and emotion using modified VGG16	91
4.10.1. Our Facial Expression Dataset	92
4.10.2. Our Approach	93
4.10.3. Building the model.....	95
4.10.4. Training the model.....	99
4.10.5. Testing section	102
4.10.6. Some right and wrong predictions for our model	105
CONCLUSION ON THE FOURTH PART	107
CONCLUSION	109
REFERENCE LIST	111

LIST OF SYMBOLS, ABBREVEATIONS, TERMS

AI -Artificial Intelligence

CV-Computer Vision

ML-Machine Learning

DL-Deep Learning

IT-Information Technology

FLS-Fuzzy Logic System

NLP-Natural Language Processing

NN-Neural Network

ANN-Artificial Neural Network

CNN-Convolutional Neural Network

GPU-Graphic Processing Unit

INTRODUCTION

In this graduation project there was covered a significant topic- Machine learning implementation in face detection and recognition. In the first part the concept of Artificial intelligence and research areas related to it. Moreover, the real-life implementation and advantages and disadvantages of using AI. In the second part it was the Machine learning and the workflow that have to followed from defining the problem to data collection, pre-processing data, and finally developing and the evaluating of the model. In addition, the type of collected data which are the labeled and unlabeled data, for that matter there are two main types of learning (supervised and unsupervised). It is also possible to find here the exact relationship and differences between AI, ML, DL. Computer vision and basic information about RGB color space by the end of this part.

In the third part it was considered the artificial neural network mechanism and the components of it, also the different operations that occur in the network such as the activation function that helps with providing nonlinearity to the perceptron, also include deep learning algorithms for instance, gradient descent furthermore, a deep explaining of CNN and the sequence and functions of the layers of it, although the objective of using CNN which to extract high-level features from images by using more than one ConvNets layer, typically the first ConvNets is responsible to extract the low-level features such as edges, color. With more layers the model adapts to high-level features as well

The main concept in training CNN is that more training data we have, the better our model will perform on unseen data.

PART 1: ARTIFICIAL INTELLIGENCE ADVANTAGES AND RESEARCH AREAS

1.1. Artificial Intelligence

Artificial Intelligence (AI) is a way of making a computer, a computer-controlled robot, or a software think intelligently, in the similar manner the intelligent humans think.

AI is accomplished by studying how the human mind thinks, and how humans learn, decide, and work while trying to solve a problem, and then use the results of this study as a basis for developing smart programs and systems.

1.1.1 Philosophy of AI

While harnessing the power of computer systems, human curiosity prompts him to ask, "Can a machine think and act like a human being?" Therefore, the development of artificial intelligence began with the intention of creating similar intelligence in machines that we find and consider high in humans.

1.1.2 Goals of AI

- Create Expert Systems: Systems that demonstrate intelligent behavior, learn, explain, explain and advise their users.
- Applying human intelligence to machines: creating systems that understand, think, learn, and act like humans.

Artificial Intelligence is a science and technology based on disciplines such as computer science, biology, psychology, linguistics, mathematics, and engineering. The main drive for artificial intelligence is the development of

computer functions associated with human intelligence, such as thinking, learning, and problem solving.

Among the following areas, one or more domains can contribute to building a smart system.

1.2. Programming without and with AI

The programming without and with AI is different in following ways

Programming Without AI	Programming with AI
A computer program without AI can answer the specific questions it is meant to solve.	A computer program with AI can answer the generic questions it is meant to solve.
Modification in the program leads to change in its structure.	AI programs can absorb new modifications by putting highly independent pieces of information together.
Modification is not quick and easy. It may lead to affecting the program adversely.	Quick and Easy program modification.

Table. 1.1. Programming with and without AI

In the real world, knowledge has some unwelcome properties such as the size is huge, besides what is unimaginable. In addition, it is not well-organized or well-coordinated. Moreover, it is constantly changing.

Artificial Intelligence Technology is a method for efficiently organizing and using knowledge in a way that it should be understood by the people providing it. It should be easily adjustable to correct errors. It should be useful in many situations although it is imperfect or imprecise.

1.3 Applications of AI

Games: AI plays an important role in strategy games like chess, poker, tic-tac-toe, etc., as the machine can think of a large number of possible situations based on the exploratory knowledge.

Natural language processing: It is possible to interact with a computer that understands the natural language spoken by humans.

Expert Systems: There are some applications that integrate machine, software and proprietary information to convey inference and counsel. It provides explanation and advice to users.

Vision systems: These systems understand, interpret and digest visual input on a computer. For example, the spy plane takes pictures that are used to find out spatial information or map areas.

Doctors use the clinical expert system to diagnose a patient. The police use computer programs that can recognize the face of the criminal with the stored image made by the forensic artist.

Speech recognition: Some smart systems are able to hear and understand language in terms of sentences and their meanings while a person speaks to it. It can deal with different accents, slang words, background noise, change in human noise due to cold, etc.

Handwriting recognition: Handwriting recognition software reads text written on paper with a pen or on the screen with a pen. It can recognize letter shapes and convert them into editable text.

Smart robots: Robots are able to perform human-given tasks. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, impact and pressure. They have efficient processors, multiple sensors and a huge memory to display intelligence. In addition, they are able to learn from their mistakes and can adapt to a new environment.

1.4. Intelligence Definition

The system's ability to compute, reason, perceive relationships and analogies, learn from experience, store and retrieve information from memory, solve problems, understand complex ideas, use natural language fluently, classify, generalize, and adapt new situations.

There are many types of intelligence such as linguistic intelligence, mathematical intelligence, interpersonal intelligence.

You can say a machine or a system is artificially intelligent when it is equipped with at least one and at most all intelligences in it.

The intelligence is intangible. It is consisting of

- Reasoning
- Learning
- Problem Solving
- Perception
- Linguistic Intelligence

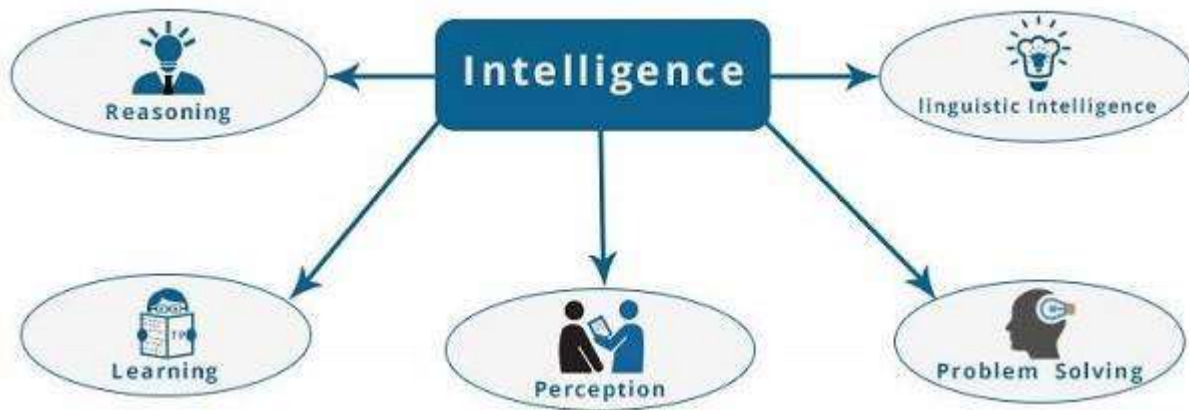


Fig. 1.1. Intelligent components

Let us go through all the components briefly:

Reasoning: It is the set of processes that enables us to provide basis for judgement, making decisions, and prediction.

Learning: is the activity of acquiring knowledge or a skill through study, practice, teaching or trying something. Learning enhances awareness of the subjects of study.

The ability to learn is possessed by humans, some animals, and systems that support artificial intelligence. Learning is categorized as:

Auditory learning: Is learning through listening and hearing. For example, students listen to recorded audio lectures.

Accidental learning: To learn by remembering the sequence of events one has witnessed or experienced. This is linear and organized.

Motor learning: Is learning by subtle movement of muscles. For example, choosing things, writing, etc.

Observational learning: Learning by watching and imitating others. For example, a child tries to learn by imitating his mother.

Perceptual learning: Learning to recognize stimuli that one has witnessed before. For example, identifying and classifying objects and situations.

Relational learning: Involves learning to distinguish different stimuli on the basis of relational properties, rather than absolute properties. For example, adding "slightly less" salt at the time of cooking potatoes that appeared salty the last time, when cooked with the addition of a tablespoon of salt.

Spatial learning: Is learning through visual stimuli such as images, colors, maps, etc. For example, a person can create a roadmap in mind before actually following the road.

Motivation and response learning: Are the learning to perform a specific behavior when a specific stimulus is present. For example, a dog raises its ear when hearing a doorbell.

Problem Solving: It is the process in which one perceives and tries to arrive at a desired solution from a present situation by taking some path, which is blocked by known or unknown hurdles.

Problem solving also includes decision making, which is the process of selecting the best suitable alternative out of multiple alternatives to reach the desired goal are available.

Perception: It is the process of acquiring, interpreting, selecting, and organizing sensory information.

Perception presumes sensing. In humans, perception is aided by sensory organs. In the domain of AI, perception mechanism puts the data acquired by the sensors together in a meaningful manner.

Linguistic Intelligence: It is one's ability to use, comprehend, speak, and write the verbal and written language. It is important in interpersonal communication.

1.5. Difference between Human and Machine Intelligence

- Humans perceive through patterns while machines perceive through a set of rules and data.
- Humans store and recall information by means of patterns, and machines do this by searching for algorithms for example, the number 40404040 is easy to remember, store and retrieve because its pattern is simple.
- Humans can know a whole being even if a part of it is missing or deformed; Whereas machines cannot do it properly.

1.6. Research Areas

The field of artificial intelligence is huge in breadth and width. Moving forward, we consider popular and widely flourishing areas of research in the field of artificial intelligence

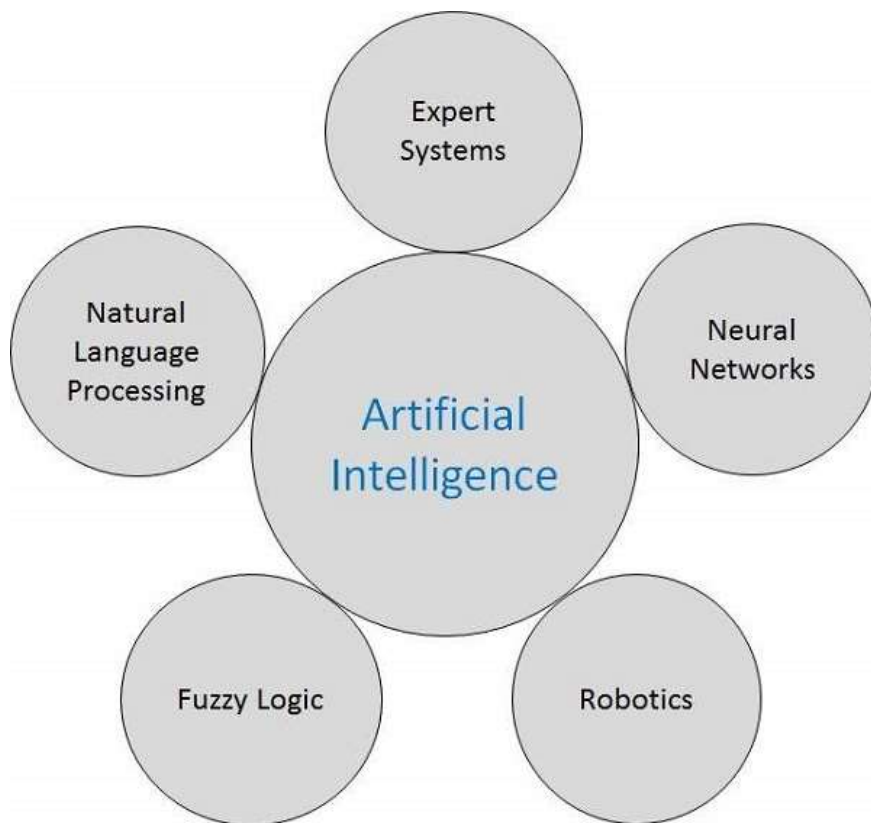


Fig. 1.2. AI research areas

There is a large array of applications where AI is serving common people in their day-to-day lives.

The domain of AI is classified into Formal tasks, Mundane tasks, and Expert tasks. Humans learn normal (normal) tasks from birth. They learn by perception, speaking, and using language and locomotives. They learn formal and expert duties later, in that order.

For humans, normal tasks are easier to learn. The same was considered true before attempting to perform normal tasks in hardware. Earlier, all the work of AI was concentrated in the field of ordinary tasks.

Later on, it turns out that the machine requires more knowledge, more sophisticated cognitive representation, and complex algorithms to handle ordinary

tasks. This is the reason why AI work is more thriving in the field of expert tasks now, as the expert task field needs expert knowledge without common sense, which can be easier to act and manipulate.

An AI system is composed of an agent and its environment. The agents act in their environment. The environment may contain other agents.

An agent is anything that can perceive its environment through sensors and act on that environment through effects.

The human agent has sensory organs such as the eyes, ears, nose, tongue and skin parallel to the sensors and other organs such as hands, legs and mouth of the responders. A robotic agent that replaces cameras and infrared rangefinders for various sensors, actuators and transponders. The software agent encodes the bit strings as their programs and actions.






Sr.No.	Research Areas	Real Life Application
1	Expert Systems Examples – Flight-tracking systems, Clinical systems.	
2	Natural Language Processing Examples: Google Now feature, speech recognition, Automatic voice output.	
3	Neural Networks Examples – Pattern recognition systems such as face recognition, character recognition, handwriting recognition.	
4	Robotics Examples – Industrial robots for moving, spraying, painting, precision checking, drilling, cleaning, coating, carving, etc.	
5	Fuzzy Logic Systems Examples – Consumer electronics, automobiles, etc.	

Fig. 1.3. Real life application of AI

Agent performance measure: They are the criteria that determine how successful an agent will be.

Agent behavior: it is the action that an agent performs after any given sequence of perception.

Perception: it is the input to the sensory perception of an agent in a particular situation.

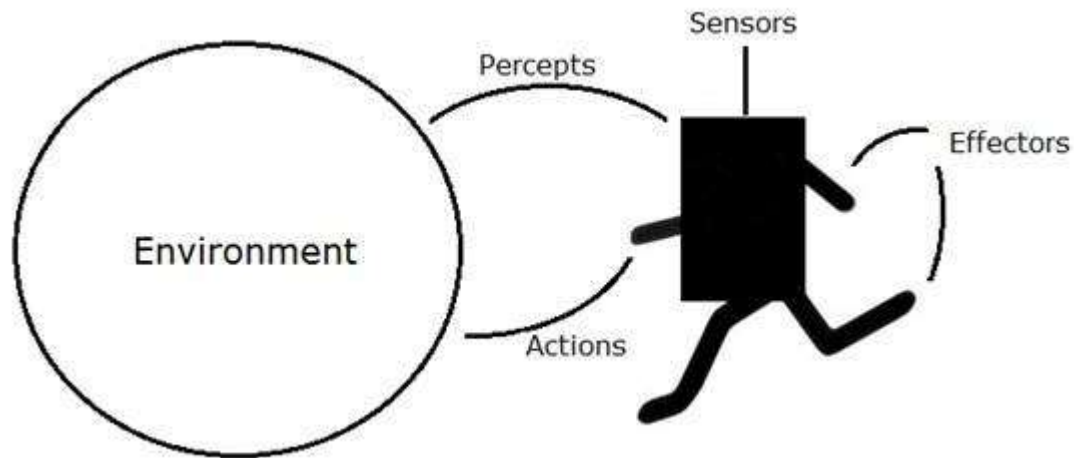


Fig. 1.4. Agent and environment

The sequence of perception: it's the history of everything the agent has realized so far.

Agent function: It is a map from the sequence of principle to action.

Rationality: is nothing but a state of being rational, rational, and having a good sense of judgment. Rationality is concerned with the actions and expected outcomes depending on what the agent perceives. Performing actions in order to obtain useful information is an important part of rationality.

The ideal rational factor is the factor that is capable of performing the expected actions to maximize the scale of his performance, on the basis of realizing sequence, the knowledge base included in it the rationality of the agent depends on the following:

- Performance measures that determine the degree of success.
- Realizing agent sequence to date.
- The agent's previous knowledge of the environment.

- Actions that can be taken by the agent.

The rational agent always performs the correct action, as the correct action means the action which makes the agent more successful in the specified sequence of perception. The problem the dealer solves is characterized by performance metric, environment, actuators, and sensors.

Research is the universal method for problem solving in artificial intelligence. There are some individual games like square games, sudoku, crossword puzzles, etc. Search algorithms help you to search for a specific location in such games. Games such as 3x3 eight tiles, 4x4 fifteen pieces, and 5x5 24 pieces of puzzles are challenges to find a single agent path. It consists of a matrix of tiles with a blank slab. The player is required to arrange the tiles by moving the tiles vertically or horizontally in an empty space with the goal of achieving certain objectives. Other examples of single agent path determination problems are the traveling salesman problem, the Rubik's Cube and Proof of Theory.

Search Terminology

The problem area: is the environment in which to search. (A group of states and a set of factors that change those states)

Problem Status: it is the initial case + target status.

Problem area diagram: Represents the problem state. Cases are shown by nodes and factors are shown by edges.

Depth of problem: the length of the shortest path or the shortest chain of operators from initial state to target state.

Space complexity: The maximum number of nodes stored in memory.

Time complexity: The maximum number of nodes created.

Acceptance: an algorithm property to always find the perfect solution.

Branching factor: average number of sub-nodes in the problem area graph.

Depth: the length of the shortest path from the initial case to the target state.

Fuzzy Logic Systems

Fuzzy Logic Systems (FLS) produce acceptable but definite output in response to incomplete, ambiguous, distorted, or inaccurate (fuzzy) input. Fuzzy Logic (FL) is a way of thinking similar to human thinking. The FL approach simulates a human decision-making method that includes all intermediate possibilities between numerical values yes and no.

The traditional logic block that a computer can understand takes precise inputs and produces specific outputs like TRUE or FALSE, which is equivalent to human's YES or NO.

The inventor of fuzzy logic, Lotfi Zadeh, noted that unlike computers, human decision-making involves a range of possibilities between YES and NO, such as Definitely yes, maybe yes, maybe no, definitely no.

Fuzzy logic operates on the potential levels of the inputs to achieve the specified output. It can be implemented in systems of various sizes and capabilities ranging from small micro-controllers to large network-based workstation-based control systems. It can be implemented in hardware, software, or both.

Fuzzy logic is useful for both commercial and practical purposes. It can control machines and consumer products. It may not give an accurate explanation,

but it may give an acceptable explanation. Fuzzy logic helps deal with uncertainty in engineering.

Fuzzy Logic Systems Architecture

- Fuzzification Module: It transforms the system inputs, which are crisp numbers, into fuzzy sets. It splits the input signal into five steps
- Knowledge Base: stores IF-THEN rules provided by experts.
- Inference Engine: Simulates the process of human reasoning by making fuzzy inferences on inputs and IF-THEN rules.
- Defuzzification Module: Converts the fuzzy set obtained by the inference engine into a clear value.

Membership functions allow you to define the linguistic term and represent a fuzzy group graphically. The membership function of a fuzzy group A in the discourse realm X is defined as $\mu_A: X \rightarrow [0,1]$.

Here, each X element is assigned to a value between 0 and 1. The value is called membership or degree of membership. The degree of membership of element in X defines to the vague group A.

- x axis represents the universe of discourse.
- y axis represents the degrees of membership in the [0, 1] interval.

Multiple organic functions can be applicable to distort the numerical value. Simple membership functions are used because using complex functions does not add more precision to the output.

1.7. Advantages of using Artificial intelligence system

1. It allowed machines to replace manpower in performing certain tasks, especially mundane and tiring ones.
2. It allowed machines to become much more efficient, giving them the ability to solve problems on their own and requiring less work on the part of developers.
3. It can be accessed and utilized at any time.
4. It can also perform tasks that would generally be difficult or dangerous for human beings to do.
5. When developed well, there is less scope of errors on their part.

1.8. Disadvantages of using Artificial intelligence system

Over the years, as artificial intelligence continued to garner people's interest, it also began to display certain drawbacks such as:

1. It was expensive to develop and maintain.
2. It resulted in unemployment, since machines began to take over tasks that people used to do.
3. Many people misused the technology for personal benefit and unethical gain.
4. At times it was difficult to find people with enough experience to develop the programs needed for the problem.
5. It took a lot of time and computational power to develop the various AI models.

1.9. Challenges Faced by Artificial Intelligence

Keeping the pros and cons in mind, we can now understand that although artificial intelligence has deeply interested many people, it also struggles in many ways to reach its full potential. At present, the challenges faced by AI include the following:

1. The scope of artificial intelligence is somewhat limited. This is mainly because of the number of resources, technology, funds, and manpower available for it.
2. Real-world implementation is still not easy. Many AI machines exist only theoretically or as prototypes, but have not yet been put into practical applications.
3. Security is a big issue when it comes to artificial intelligence. This is because AI requires loads of data in order to be trained, and this data can be taken either ethically or unethically from people. As mentioned before, AI ethics is a growing area of importance in the field of AI, but it still has a long way to go before it can really have any kind of significant impact.

CONCLUSION ON THE FIRST PART

Artificial Intelligence (AI), which is the ability of a digital computer or computer-controlled robot to perform tasks usually associated with intelligent objects. The term is frequently applied to project development of systems that enjoy distinct intellectual processes, such as the ability to think, discover meaning, generalize, or learn from past experiences. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to perform very complex tasks, for example discovering evidence of mathematical theories or playing chess with great mastery. However, despite continuous advances in computer processing speed and memory capacity, there are still no programs that can match human flexibility in broader fields or in tasks that require a lot of daily knowledge. On the other hand, some programs have achieved the

performance levels of experts and human professionals in performing certain specific tasks, so that artificial intelligence in this limited sense exists in various applications such as medical diagnostics, computer search engines, face recognition, voice recognition or handwriting.

The intelligence consists of many components but the most important feature in artificial intelligence is the ability to absorb new modification by putting highly independent pieces of information together.

PART 2: WORKFLOW OF MACHINE LEARNING, MACHINE LEARNING TYPES, DATA SCIENCE, AND DEEP LEARNING

2.1. Machine Learning

Machine learning can be defined as the process of teaching a machine to think like a human being in order to perform a particular task, without being explicitly programmed.

Think about the first time you learned to read. You began by learning the alphabet, then you formed words by joining these letters together.

Machine learning works in the same way. The machine learns, understands, and thinks like a human being, and then uses this thought process to do some work. With machine learning, machines can be taught to perform higher-level tasks. Just like human beings need to learn to increase their natural intelligence, machines need to learn to increase their artificial intelligence. This is why machine learning is so important when it comes to developing artificially intelligent systems.

The Machine Learning Workflow

A typical machine learning problem follows these steps:

1. Defining the problem: We first need to determine what exactly our problem is before we can begin solving it. We need to figure out what the problem is, whether it is feasible to use machine learning to solve it, and so on.
2. Collecting the data: We then need to gather our data based on our problem definition. The data is extremely important, and must thus be collected with care.

We need to make sure that we have data corresponding to all the necessary factors required for our analysis.

3. Pre-processing the data: We need to clean up the data to make it more usable.

This includes removing outliers, handling missing information, and so on.

This is done to decrease the possibility of obtaining errors from our analysis.

4. Developing the model: We can now create our machine learning model, which will be used in solving the problem. This model takes the data as input, performs computations on it, then produces some output from it.

5. Evaluating the model: The model needs to be evaluated to verify its accuracy and to make sure that it can work on any new data that may be provided to it.

As we have seen in this workflow, machine learning is done with the help of data loads and loads of data. Machines take this data, analyze it, and develop conclusions from it. This is how the idea of data science evolved to be an integral part of machine learning.

2.2. Data Science

Data science allows us to obtain knowledge from data. The entire process of collecting, manipulating, analyzing, and developing inferences from data is known as data science.

Data science is the combination of statistics, mathematics, computer programming, complex problem solving, data capturing, and working with data to cleanse it, prepare it, and use it. The data that is obtained from the process of data science can, once it is prepared, be fed into a machine in order to help it learn.

2.2.1. Branches of Data Science

Data science consists of the following main areas:

- Data collection: Gathering the data
- Data storage: Keeping the data for later access
- Data wrangling/munging: Cleaning up the raw data for easier utilization
- Data visualization: Viewing the data graphically

There are two other areas, namely, big data and data analytics, that although treated as separate entities, also deal with data and thus come under the data science umbrella.

2.2.2. Big Data

Big data refers to the storage of huge volumes of data. This data is mainly characterized in the following three ways:

High volume: This refers to the quantity of data that is generated and stored. The amount of this data is immense as it finds its sources in images, videos, audios, and text.

High velocity: This refers to the speed at which the data is generated and processed. Usually, this data is available in real time, which means it is continuously produced and handled.

High veracity: This refers to the quality of the data. The data produced here can greatly vary, and this can affect the overall analysis. Big data can be applied in the following areas: Communication, Finance, Retail, Education, Media.

Some of the challenges that big data faces include the following:

Gathering data: Since the amount of data is so huge, it is not an easy task to collect it.

Storing data: Very powerful storage units are required to store such massive amounts of data.

Transferring and sharing data: In order to successfully transfer and share large quantities of data, advanced techniques and tools are required.

2.2.3. Data Analytics

Raw data can be analyzed to observe trends and to come up with conclusions based on these trends. Thus, data analytics refers to the inspection of data in order to derive insights and develop inferences and conclusions based on it.

It follows several steps and consists of various methods that help in making the process more effective and in obtaining the desired results. It makes use of a variety of statistical and mathematical techniques.

There are four main types of data analytics, as follows:

- 1.Descriptive analytics: This is used to explain what has occurred. For example, it can be used to describe the present performance of a company.
- 2.Prescriptive analytics: This is used to predict what will occur in the future. For example, it can be used to determine the profits of a company based on its previous performance.
- 3.Diagnostic analytics: This is used to determine why something has occurred. For example, it can be used to understand why a company might be seeing losses.
- 4.Prescriptive analytics: This is used to figure out what needs to occur,

For example, it can be used to come up with better strategies and ideas to help a company get back on track and make profits again. Data analytics can be applied in the following areas: healthcare, energy management, travel, finance.

Big data and data analytics are each very intricate parts of data science, and thus there is a huge demand for them, especially when it comes to employment. The data obtained as a result of these methods can then be used in machine learning and related fields.

There are various sources from which data can be obtained, depending on the use case. In the next section, we will go through some of the important methods that are used for data collection.

2.3. Types of Data

The data that is collected and used can be either of the following:

Labeled: Each class/type is labeled based on certain characteristics so that the machine can easily identify and separate the data into its respective groups. For example, if you have a collection of pictures that are separated and tagged as “cat” or “fish” accordingly.

Unlabeled: Each class/type is not labeled, and so the machine needs to figure out how many classes are there and which item belongs where, and then it must separate the data on its own. For example, if you have a set of pictures, but they are not separated and tagged as “cat” or “fish” accordingly. In this case, the machine would need to identify some particular features that differentiate one animal from the other (like a cat’s whiskers or a fish’s fins).

2.4. Machine learning methods

Based on the kind of data being used, there are two main types of machine learning method.

2.4.1. Supervised Learning Algorithms

Supervised learning is one among the foremost basic sorts of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the info must be labeled accurately for this method to figure, supervised learning is extremely powerful when utilized in the proper circumstances.

In supervised learning, the ML algorithm is given a little training dataset to figure with. This training dataset may be a smaller a part of the larger dataset and serves to offer the algorithm a basic idea of the matter, solution, and data points to be addressed. The training dataset is additionally very almost like the ultimate dataset in its characteristics and provides the algorithm with the labeled parameters required for the matter.

The algorithm then finds relationships between the parameters given, essentially establishing a cause-and-effect relationship between the variables within the dataset. At the top of the training, the algorithm has a thought of how the info works and therefore the refire the relationship between the input and the output.

This solution is then deployed to be used with the ultimate dataset, which it learns from within the same way because the training dataset. this suggests that supervised machine learning algorithms will still improve even after being

deployed, discovering new patterns and relationships because it trains itself on new data.

The goal of every supervised learning algorithm is to map the input to the output, as shown in the following equation: $y = f(x)$

There are several algorithms that can be used to solve a machine learning problem with the help of supervised learning. These algorithms can be segregated into the following categories:

1. Regression algorithms: These algorithms contain outputs that are real or countable. For example, height (4 feet, 5 feet, 6 feet), age (27, 31, 65), or price (100 rupees, 20 pounds, 10 dollars)

2. Classification algorithms: These algorithms contain outputs that are abstract or categorical. For example, colors (orange, purple, turquoise), emotions (happy, sad, angry), or gender (girl, boy). To give some idea of what these algorithms are, let's go through three common types of algorithms that are used:

- Linear regression
- Logistic regression
- K-Nearest neighbors

Linear Regression: As the name suggests, linear regression is a type of regression algorithm. It models the relationship between a dependent variable and an independent variable. Graphically, it looks something like

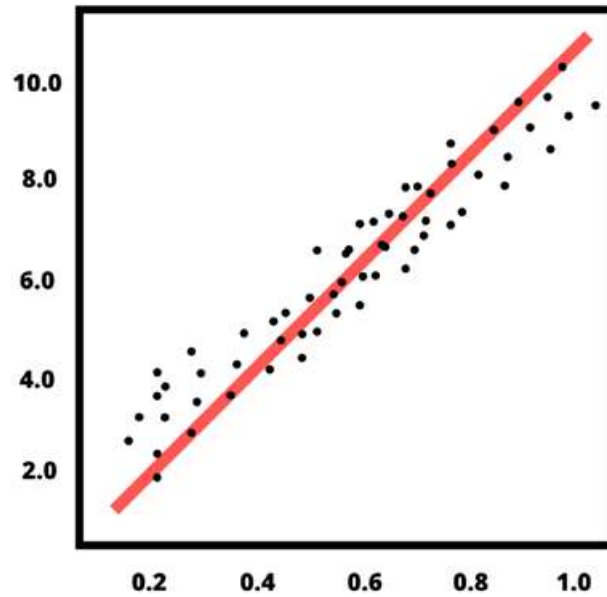


Fig. 2.1. Linear regression

Logistic Regression

Although the name says regression, this is generally used as a classification algorithm. It is usually the first choice for programmers who wish to conduct binary classification.

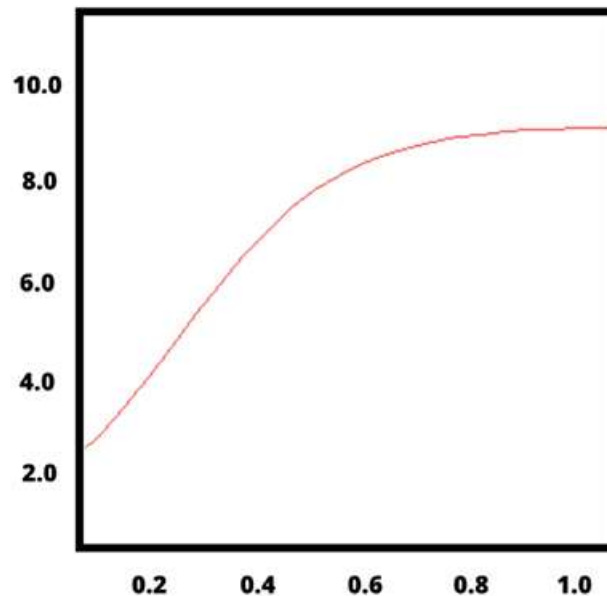


Fig. 2.2. Logistic regression

K-Nearest Neighbors

This algorithm can be used for both regression and classification. It assumes that similar units exist in close proximity to one another. It uses this idea to come up with a solution. It looks something like Figure 2-3.

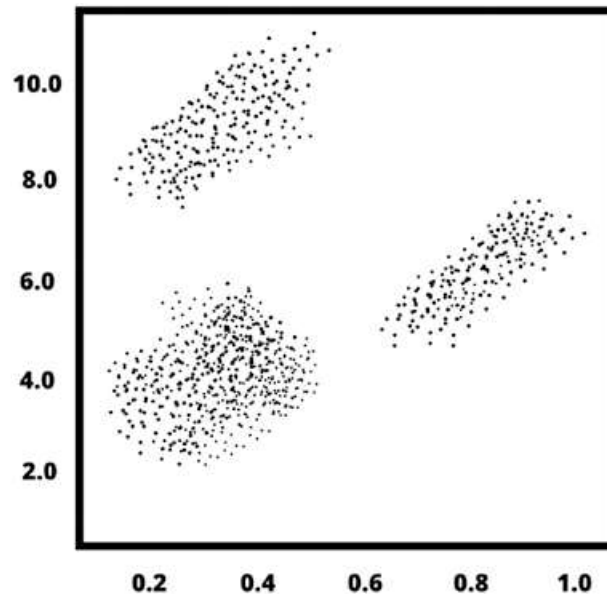


Fig. 2.3. K-Nearest Neighbors

Some applications of supervised learning algorithms

1. Spam detection
2. Bioinformatics: This is the method of keeping a record of a person's biological information for later use. One of the most common examples of this is the security system on our cell phones, which can scan our fingerprint and grant us access accordingly.

2.4.2. Unsupervised Learning Algorithms

The goal of unsupervised learning algorithms is to discover possible patterns from the set of data that is provided. The algorithm has no prior information about the patterns and labels presents in the data.

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.

In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.

The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

There are several algorithms that can be used to solve a machine learning problem with the help of unsupervised learning. These algorithms can be segregated into the following categories:

Cluster analysis: This approach finds similarities among the data and then groups the common data together in clusters.

Dimensionality reduction: This approach attempts to reduce the complexity of data while still keeping the data relevant.

Let us now have a look at two common algorithms that are used for unsupervised learning: K-means clustering and principal component analysis.

K-Means Clustering

The K-means clustering method forms k clusters out of a total of n observations. Each observation will be a part of the cluster with the closest mean.

Principal Component Analysis

Principal component analysis uses orthogonal transformation to statistically transform a set of potentially correlated variables to a set of linearly uncorrelated variables, known as principal components. It is used to reduce the dimensionality of the data.

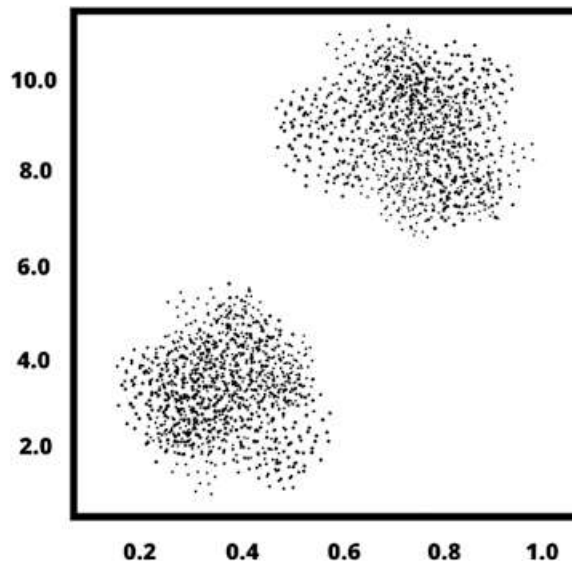


Fig. 2.4. K-Means clustering

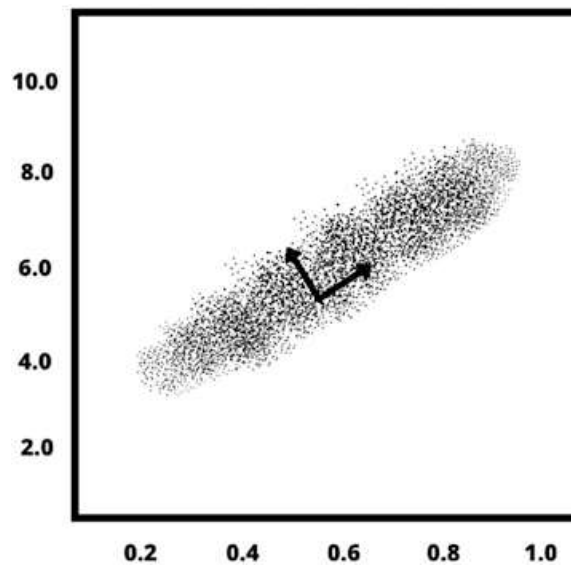


Fig. 2.5. Principal component analysis

Applications of Unsupervised Machine Learning Algorithms

Anomaly detection is the identification of certain anomalies or observations that are different from the rest of the observations. These anomalies are also called outliers. For example, credit card fraud can be discovered by detecting unusual transactions made with the credit card.

Association is the process of identifying associations between different observations with the help of provided data. For example, in e-commerce it is easy to figure out the type of products a customer might be interested in by analyzing previous purchases.

Apart from supervised and unsupervised machine learning, there are also two lesser-known methods of machine learning, as follows:

2.4.3. Semi-supervised learning

This method uses some labeled data and a larger proportion of unlabeled data for training. Semi-supervised learning where an incomplete training signal is given: a training set with some (often many) of the target outputs missing. There is a

special case of this principle known as Transduction where the entire set of problem instances is known at learning time, except that part of the targets are missing.

2.4.4. Reinforcement learning

This method is similar to training a pet. It sends positive signals to the machine when it gives the desired output, to let it know that it is right and to help it learn better. Similarly, it sends negative signals to a machine if it provides an incorrect output.

Reinforcement learning directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favorable outputs are encouraged or ‘reinforced’, and non-favorable outputs are discouraged or ‘punished’.

Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not.

In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result.

In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this

percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.

2.5. Categorizing on the basis of required output

Another categorization of machine learning task arises when one considers the desired output of a machine-learned system:

Classification: When inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are “spam” and “not spam”.

Regression: Which is also a supervised problem, A case when the outputs are continuous rather than discrete.

Clustering: When a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

2.6. Applications of Machine Learning

Over the years, as machine learning began to grow in popularity, enthusiasts began to do more research into different ways of solving machine learning problems. They soon came up with different types of algorithms that prove to be the most efficient, depending on the data and parameters that you are using. Thus, these algorithms became available for worldwide use. Developers can easily choose which method they want to implement and follow the algorithm accordingly.

Machine learning models are structured differently, based on what is required of them.

E-Commerce: Machine learning can help to boost online sales with the help of recommendation systems (that recommend relevant products to potential customers), analytics and predictions (to learn from past sales and improve future sales), etc.

Autonomous cars: Cars that drive themselves will no longer be a thing of the future. They already exist, and in a few years will be available on the market for anyone and everyone to access.

Manufacturing: Many companies use robots to develop their products in a quicker and more convenient manner. This is because robots can be programmed to work tirelessly, with more accuracy, and at less cost.

Healthcare: AI technology has improved immensely, and a major piece of evidence of that is the fact that it is now being implemented in healthcare. One very interesting application is robotic arm surgery, where, as the name suggests, a robotic arm is used to conduct surgery. Starting with the use of the Arthrobot in 1985, this type of surgery has been conducted several times to facilitate increased precision and decreased incision.

2.7. Origins of Deep Learning

Deep learning is a branch of machine learning that uses artificial neural networks to help the machine to think about and respond to a particular problem.

The important thing to remember here is that, although it is very tempting to think of deep learning as an independent area under artificial intelligence, it is definitely not. It is very much a part of artificial intelligence, and is a subset of machine learning

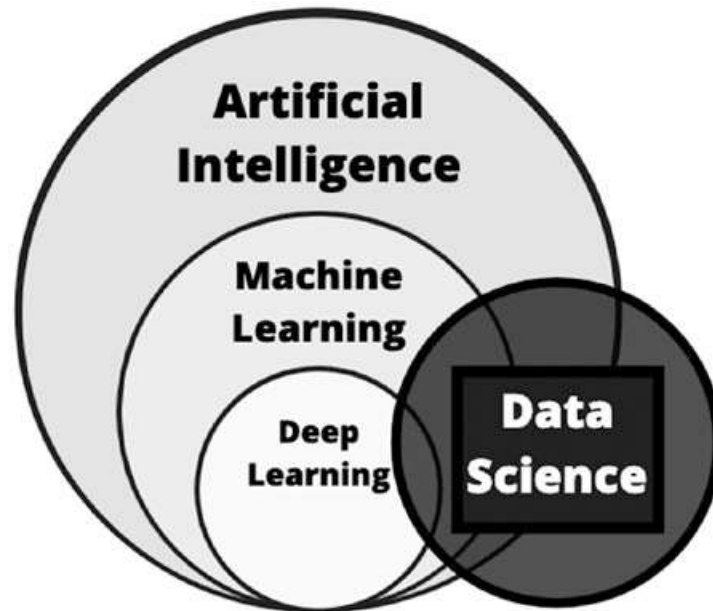


Fig. 2.6. origins of deep learning

Deep learning is a subset of machine learning within the artificial intelligence sphere. Thus, we can say that deep learning is a subset of machine learning, which is a subset of artificial intelligence. Data science is like a common denominator here, as it is a necessary part of all three areas. The origins of deep learning can be credited to Walter Pitts and Warren McCulloch.

Walter Pitts was a logician in computational neuroscience, while Warren McCulloch was a neurophysiologist and cybernetician. In the year 1943, they created a computer model that was inspired by the neural networks present in the human brain. They developed something called threshold logic, which was a combination of mathematics and algorithms that compared the total input with a

certain threshold. This enabled them to recreate the process of thinking, just as it happens in the brain. This was a breakthrough that led to many more deep learning innovations.

2.7.1 Deep Learning

Deep Learning is basically a sub-part of the broader family of Machine Learning which makes use of Neural Networks (similar to the neurons working in our brain) to mimic human brain-like behavior. DL algorithms focus on information processing patterns mechanism to possibly identify the patterns just like our human brain does and classifies the information accordingly. DL works on larger sets of data when compared to ML and prediction mechanism is self-administered by machines.

2.8. Difference Between Artificial Intelligence, Machine Learning, Deep Learning

Artificial intelligence	Machine learning	Deep learning
AI stands for Artificial Intelligence, and is basically the study/process which enables machines to mimic human behavior through particular algorithm.	ML stands for Machine Learning, and is the study that uses statistical methods enabling machines to improve with experience.	DL stands for Deep Learning, and is the study that makes use of Neural Networks(similar to neurons present in human brain) to imitate functionality just like a human brain.
AI is the broader family consisting of ML and DL as it's components.	ML is the subset of AI.	DL is subset of ML
AI is a computer algorithm which exhibits intelligence through decision making.	ML is an AI algorithm which allows system to learn from data.	DL is a ML algorithm that uses deep(more than one layer) neural networks to analyze data and provide output accordingly.
Search Trees and much complex math is involved in AI.	If you have a clear idea about the logic(math) involved in behind and you can visualize the complex functionalities like K-	If you are clear about the math involved in it but don't have idea about the features, so you break the complex

	Mean, Support Vector Machines, etc., then it defines the ML aspect.	functionalities into linear/lower dimension features by adding more layers, then it defines the DL aspect.
The aim is to basically increase chances of success and not accuracy.	The aim is to increase accuracy not caring much about the success ratio.	It attains the highest rank in terms of accuracy when it is trained with large amount of data.
Three broad categories/types Of AI are: Artificial Narrow Intelligence (ANI), Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI)	Three broad categories/types Of ML are: Supervised Learning, Unsupervised Learning and Reinforcement Learning	DL can be considered as neural networks with a large number of parameters layers lying in one of the four fundamental network architectures: Unsupervised Pre-trained Networks, Convolutional Neural Networks, Recurrent Neural Networks and Recursive Neural Networks
The efficiency Of AI is basically the efficiency provided by ML and DL respectively.	Less efficient than DL as it can't work for longer dimensions or higher amount of data.	More powerful than ML as it can easily work for larger sets of data.
Examples of AI applications include: Google's AI-Powered Predictions, Ridesharing Apps Like Uber and Lyft, Commercial Flights Use an AI Autopilot, etc.	Examples of ML applications include: Virtual Personal Assistants: Siri, Alexa, Google, etc., Email Spam and Malware Filtering.	Examples of DL applications include: Sentiment based news aggregation, Image analysis and caption generation, etc.

Table. 2.1. Difference between AI and ML and DL

2.9. Computer vision

Computer vision is a discipline (artificial intelligence section) that extracts information from images, and the images can be of different types. There can be photos, videos, sets of photos or an image for medical purposes. Recently, algorithms that work with images and combine color information about points and their spatial position have become very popular, because sensors that receive such

information have become much more accessible and powerful, and it has become much easier for robots or computers to understand the world.

The information that different algorithms extract from images in computer vision can also have a different nature, a different type. Some algorithms simply split the image into parts that correspond to individual objects or different parts of objects.

A simple definition of an image is that Visual representation of a real-life object (a person or any other object) in a two-dimensional form is called an image. An image is nothing but a collection of pixels in different color spaces. On the other hand, pixels are subsamples of an image that, when get combined, give us the complete image. You might think of a complete image as a set that consists of small samples.

Image resolution is the number of pixels present in an image. The greater the number of pixels, the better quality. Image resolutions are described, for example, as 320×240 , 640×480 , 800×600 , 1024×768 , and so on. This means, for example, that there are 1024-pixel columns and 768-pixel rows. The total number of pixels is obtained by multiplying both numbers, which gives us 786,432 pixels.

PPI and DPI

PPI means “pixels per inch” whereas DPI means “dots per inch.” They are the units for measuring image resolution. If we consider an inch of an image, the number of square pixels we are able to see inside it is represented by PPI. DPI on the other hand, is related to printing. When we print an image and look at an inch of the print, the number of dots of ink used is represented by DPI.

Bitmap Images

In general, when we look at pixel values, they are a range of integers. But, when we convert the range of integers into bytes, we then have a bitmap image. One kind of bitmap is a binary image in which each pixel has one of two numbers: either a zero or a one. They represent black or white and are often used for storing images efficiently.

Lossless Compression

When we want to reduce the size of a file (which can be an image), but we don't want to compromise quality, this kind of compression is called a lossless compression. The compressed file can be saved, but when we require it, during the decompression process, all the information is restored and we get the actual image.

This first type of compression gives priority to the information contained in the file, especially when compressing text, where we cannot afford to lose even a single piece of information.

Lossy Compression

With lossy compression, on the other hand, some of the data may be lost.

Lossy compression prioritizes saving space, rather than the accuracy of the retrieved file. Some files, such as those that contain music or images, can be trimmed and still be unaffected by the compression.

Image File Formats

The following are some of the most widely used image formats,

- JPEG: Joint Photographic Experts Group
- JPEG2000: New JPEG format developed in 2000
- TIFF: Tagged Image File Format
- GIF: Graphics Interchange Format
- BMP: Bitmap
- PNG: Portable Network Graphics
- SVG: Scalable Vector Graphics

Color Spaces

The organization of the colors of in an image in a specific format is called color space. The way in which a color is represented is called a color model. Each and every image uses one of the following color spaces for effective picture representation:

RGB: red, green, blue

XYZ: color in the x, y, and z dimensions

HSV/HSL: hue, saturation, and value/hue, saturation, and lightness

Using the RGB color space, red, green, and blue are mixed in different ways to make different color combinations. We use RGB Because the eyes have color receptors that can perceive these three colors and their combinations quite effectively.

We can form any color, theoretically, from these three colors. Each color's intensity is defined within a range of 0 to 255. This range is called color depth.

CONCLUSION ON THE SECOND PART

Supervised learning where the name indicates a supervisor as a teacher. Basically, supervised learning is learning in which we teach or train a machine using well-classified data which means that some of the data is already marked with the correct answer. After that, the machine is provided with a new set of examples (data) so that the supervised learning algorithm analyzes the training data (Collection of Training Examples) and produces a valid result from the labeled data.

On the other hand, during training of ANN under unsupervised learning, input vectors of similar type are combined to form clusters. When a new input pattern is applied, the neural network gives an output response that indicates which class the input pattern belongs to. There is no feedback from the environment on what the required output should be and whether it is correct or incorrect. Hence, in this type of learning, the network itself must discover patterns and features from the input data, and the relationship to the input data on the output.

Deep learning is a subset of machine learning within the artificial intelligence sphere. Data science is like a common denominator here, as it is a necessary part of all three areas. Deep learning uses the Artificial neural network (similar to the neurons working in our brain) to mimic human brain-like behavior.

DL algorithms focus on the mechanism of information processing patterns to identify patterns just as our human brain does and classify information accordingly.

DL works on larger sets of data when compared to ML and prediction mechanism is self-administered by machines.

PART 3: NEURAL NETWORK LAYERS, WORKING OF PERCEPTRON, CONVOLUTIONAL NEURAL NETWORK WORKFLOW

3.1. Neural Networks

The neural network, or artificial neural network, was inspired by and modeled after the biological neural network. These networks, like the human brain, learn to perform specific tasks without being explicitly programmed.

A neural network is composed of a series of neurons that are connected together to form a type of network, hence the name neural network. A neuron, or an artificial neuron, is the fundamental unit of a neural network. It is a mathematical function that replicates the neurons in the human brain, perceptron is nothing but an artificial neuron. In deep learning, the terms perceptron and neuron are used interchangeably.

Mathematically speaking, Artificial Neural Networks are highly complex composite functions providing the ability of computing non-linear problems to the computers.

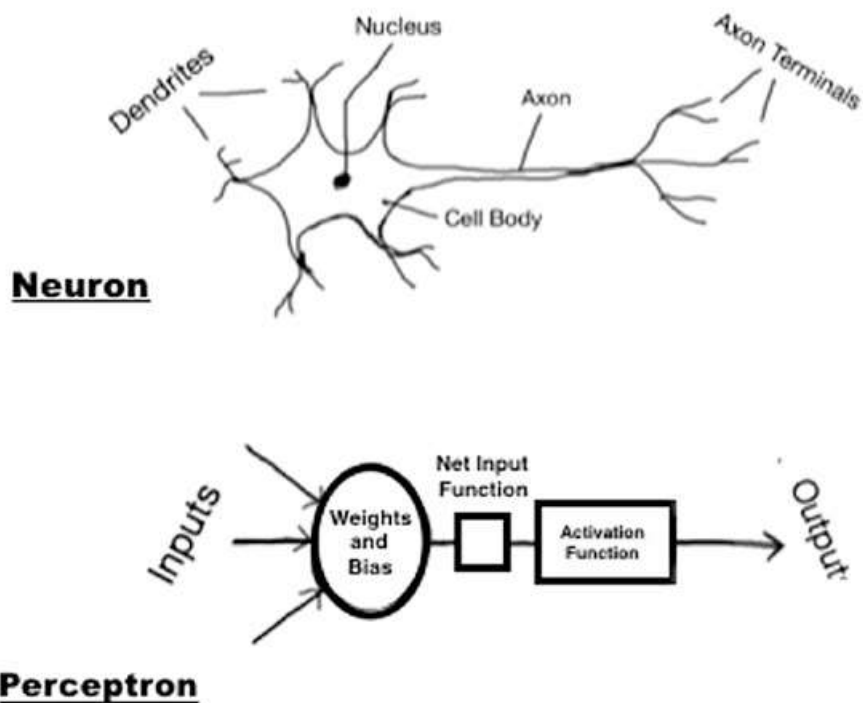


Fig. 3.1. Neural and biological perceptron

The neural network consists of the layer of input signals or neurons where the information enters the neural network, a layer of output signal neurons where we can get the result out of the Network, and a number of various hidden layers in between.

Every neuron is connected with other neuron through a connection link. Each connection link is associated with a weight that has information about the input signal.

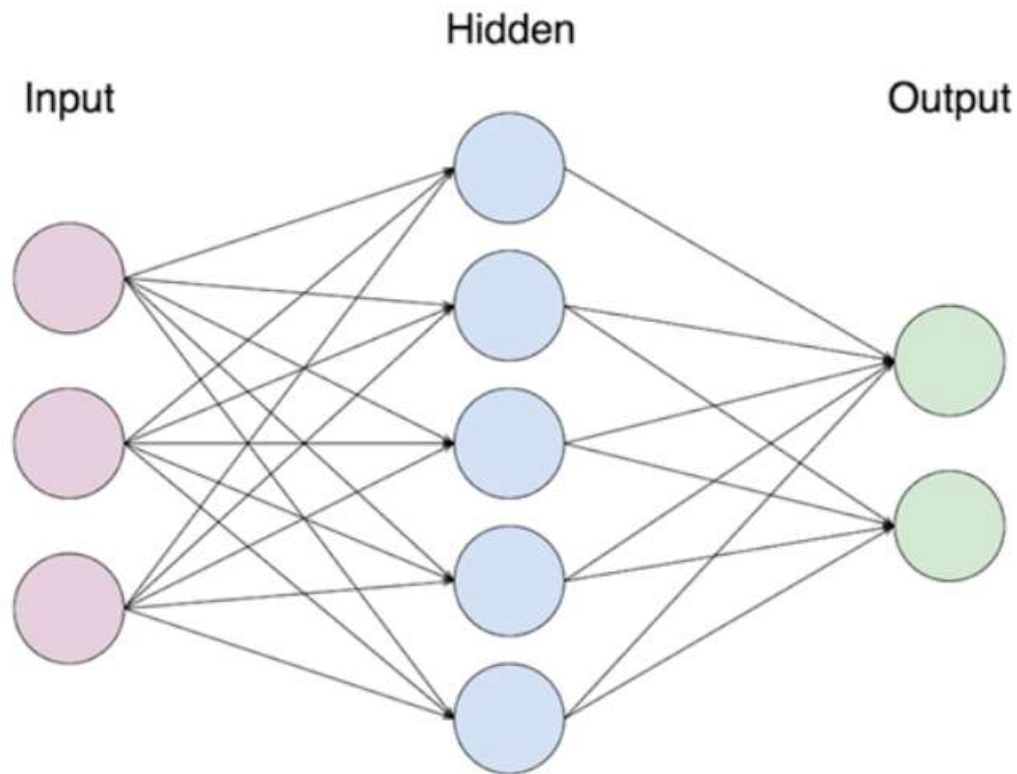


Fig. 3.2. Basic Neural Network

This is the most useful information for neurons to solve a particular problem because the weight usually excites or inhibits the signal that is being communicated. Each neuron has an internal state, which is called an activation signal. Output signals, which are produced after combining the input signals and activation rule, may be sent to other units.

3.2. Working of an Artificial Neuron (Perceptron)

The perceptron follows a particular flow of steps in order to achieve its desired output. Let's go through these steps one by one to understand how a perceptron works.

Step 1: Accepting Inputs

The perceptron accepts inputs from the user in the form of digital signals

provided to it. These inputs are the “features” that will be used for training the model. They are represented by $x(n)$, where n is the number of the feature. These inputs are then fed to the first layer of the neural network through a process called forward propagation.

Step 2: Setting the Weights and Bias

Weights: The weights are calculated and set while training the model. They are represented by $w(n)$, where n is the number of the weight. For example, the first weight will be w_1 , the second weight will be w_2 , and so on.

Bias: The bias is used to train a model with higher speed and accuracy. We generally represent it with b .

The activation function in Neural Networks takes an input ' x ' multiplied by a weight ' w '. Bias allows you to shift the activation function by adding a constant (i.e. the given bias) to the input. Bias in Neural Networks can be thought of as analogous to the role of a constant in a linear function, whereby the line is effectively transposed by the constant value.

Step 3: Calculating the Net Input Function thus, each input feature is multiplied by its corresponding weight, and the sum of all these products is taken. Then, the bias is added to this result.

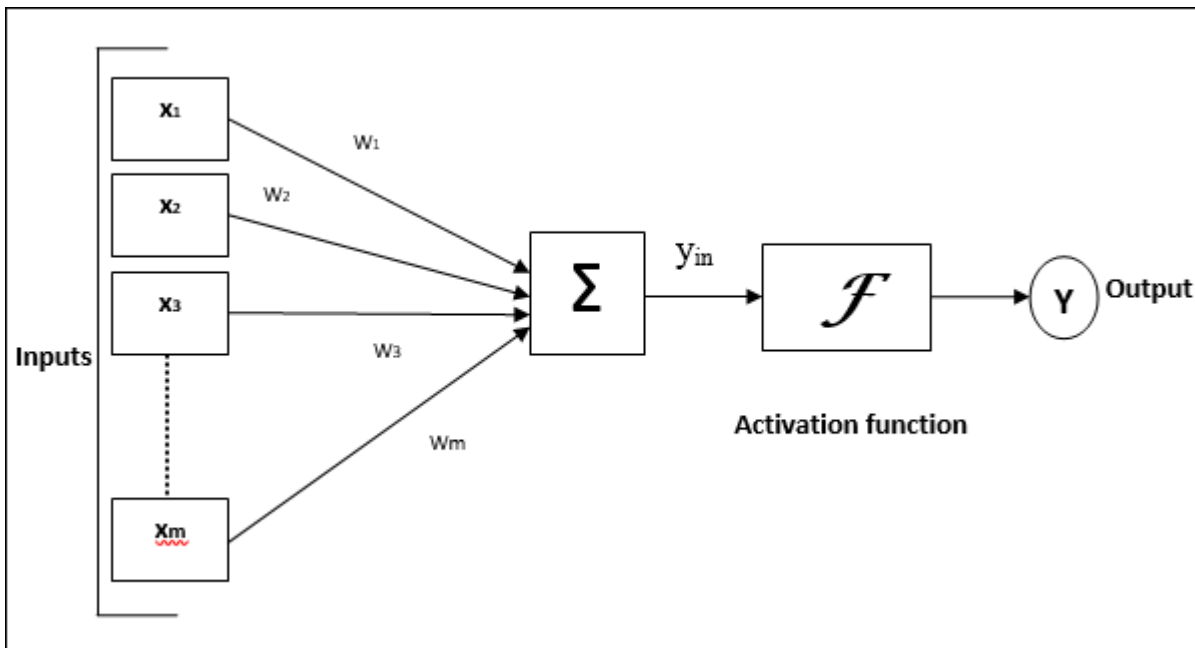


Fig. 3.3. Model of Artificial Neural Network

The Perceptron Learning Rule: According to this rule, the algorithm automatically determines the optimum values for the weights. The input features are then multiplied by these weights in order to determine if the perceptron should forward the signal or not. The perceptron is fed with several signals, and if the resultant sum of these signals exceeds a particular threshold, it either returns an output signal or doesn't.

Step 4: Passing the Values Through the Activation Function

The activation function helps with providing nonlinearity to the perceptron. There are three types of activation functions that can be used:

ReLU, Sigmoid, and Softmax.

3.2.1. ReLU

The Rectified Linear Unit is used to eliminate negative values from our outputs. If the output is positive, it will leave it as it is. If the output is negative, it will display a zero.

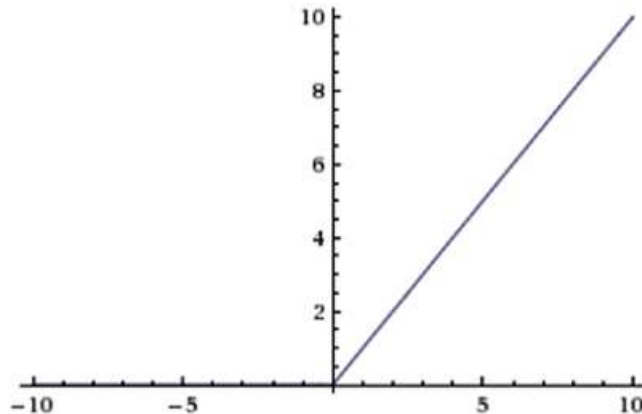


Fig. 3.4. ReLU function graph

Advantages

1. It is scalable.
2. It provides efficient computation.
3. It works well for neural networks with complex datasets.

Disadvantages

1. The output value is not restricted, which means it can cause issues if large values are passed through it.
2. The neurons can become inactive and “die” when the learning rate is large.
3. There is asymmetric handling of data, and results can end up inconsistent.

3.2.2. Sigmoid

It is a special mathematical function that produces an output with a probability of either 1 or 0. Also to make the output behave more smoothly for given inputs, x_1, x_2, \dots, x_N , and weights. w_1, w_2, \dots, w_N , and bias b .

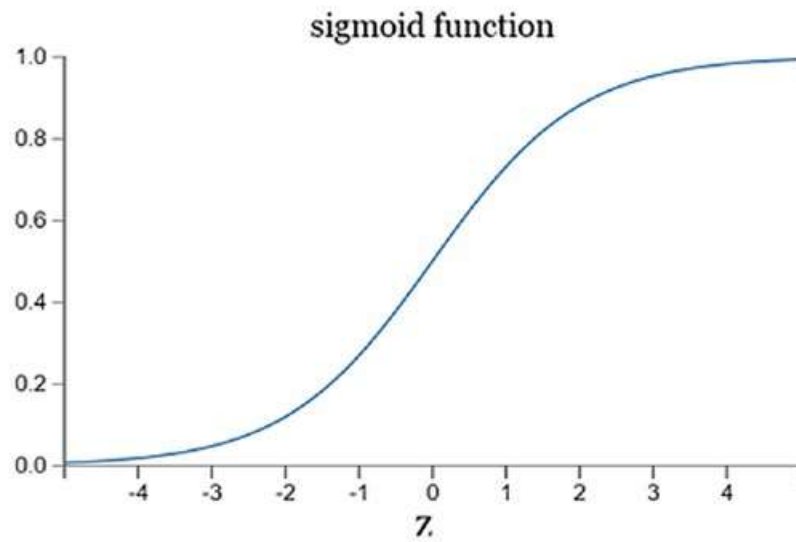


Fig. 3.5. Sigmoid function

Advantages

1. It is differentiable and monotonic.
2. It can be used for binary classification.
3. It is useful when we need to find only the probability.

Disadvantages

1. It does not give precise values.
2. There is the issue of a vanishing gradient, which prevents the sigmoid function from being used in multi-layered networks.
3. The model can get stuck in a local minimum during its training.

3.2.3. Softmax

It is generally used in the final layer of a neural network. It is generally used to convert the outputs to values that, when summed up, result in 1. Thus, these values will lie between 0 and 1.

Advantages

1. It can be used for multi-class classification.
2. The range is only between 0 and 1, thus simplifying our work.

Disadvantages

1. It does not support a null class.
2. It does not work for linearly separable data. The most common practice is to use a ReLU activation function in all the hidden layers, and then to use either a Softmax activation function (for multi-class classification) or Sigmoid activation function (for binary classification).

3.2.4. One Hot Encoding

One Hot Encoding is a tweak that can be used while producing the outputs. It is used to round off the highest value to 1, while making the other values 0. This makes it easier to figure out which is the necessary class, as it is easier to spot a 1 from a list of 0s, rather than finding the highest value from a random list of numbers.

For example, say we have a set of inputs like 0.11, 0.71, 0.03, 0.15. Here, it is obviously not too difficult to identify the highest value since there are only four values. Now imagine if the list had about 1,000 values.

But, with the help of One Hot Encoding, we can easily identify the one from the zeroes. That is why it is a popular technique used in neural networks.

Step 5: Producing the Output

The final output is then passed from the last hidden layer to the output layer, which is then displayed to the user.

Now that we know how a perceptron works, let's go a little more in depth as to how a neural network performs a deep learning task.

3.3. Gradient Descent

Gradient descent is a deep learning algorithm that is used for optimization.

It determines the values of the parameters of a function in order to ensure that the value of the cost function is minimized.

It minimizes the function by iteratively moving in such a way that it follows the path of steepest descent, depending on the negative of the gradient.

Gradient descent is one of the best approaches used for optimization. When we talk about machine learning, there's always some errors in prediction. This error is denoted by the cost function. If the cost function's value is zero, our accuracy of prediction is 100%. To keep the value of the cost function low, we use a gradient descent approach.

For example, let's define the cost function as $f(x) = ax + b$. a and b are the parameters. Imagine that the curve of this function is similar to a bowl. First, we give some random values to a and b , which puts our cost function at a particular position on a curve. Our aim is to change the values of a and b in such a way that the cost function reaches the bottom of the bowl/curve. To do this, we use a learning

rate. At the end of gradient descent, we get values for a and b that are either zero or very, very close to it.

The gradient of the error function with respect to the weights of the neural network is calculated. Afterward, the output is compared with the labels in order to calculate the error.

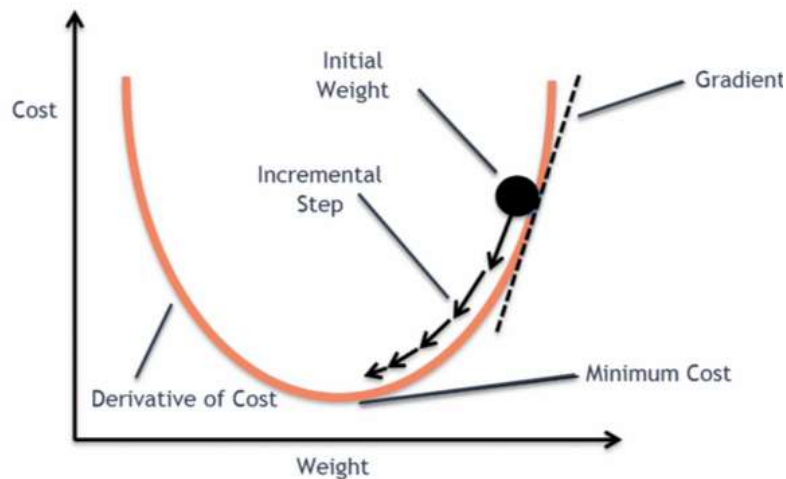


Fig. 3.6. Gradient descent

3.4. Stochastic Gradient Descent

Stochastic gradient descent is the faster version of the normal gradient descent algorithm. Using the normal gradient descent, a and b are updated for every record present in the dataset, which makes the process very time-consuming if the dataset is large. Stochastic gradient descent, on the other hand, updates the data after one complete training instance is completed, not in individual training records. This makes the process faster.

With this algorithm, you may find that the cost function jumps around

on the curve, but it finally does reach the bottom of the curve.

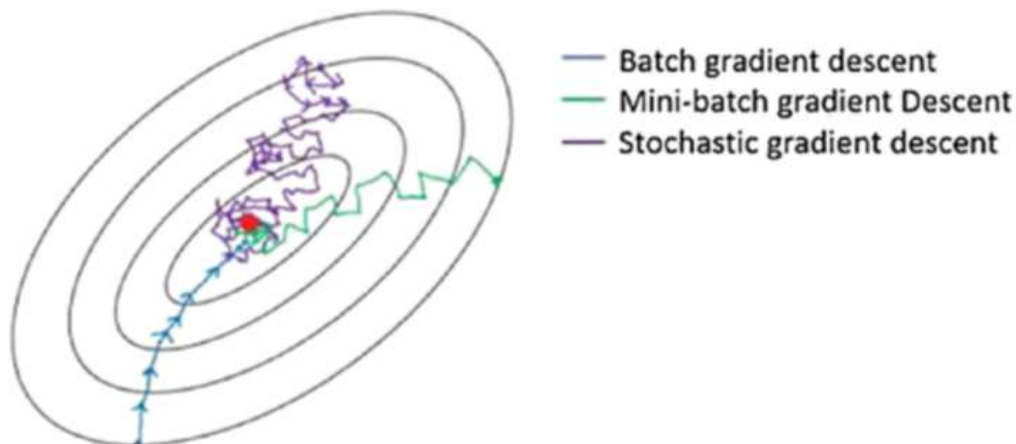


Fig. 3.7. Stochastic Gradient Descent

3.5. AdaDelta

AdaDelta belongs to the family of stochastic gradient descent approaches. Using this method, apart from its stochastic gradient descent features, special importance is given to the value of coefficients. This is called parameter tuning. Aside from this, the learning rate is not initialized with any kind of default value; it is updated automatically. Therefore, it involves the least amount of manual intervention, and hence better accuracy.

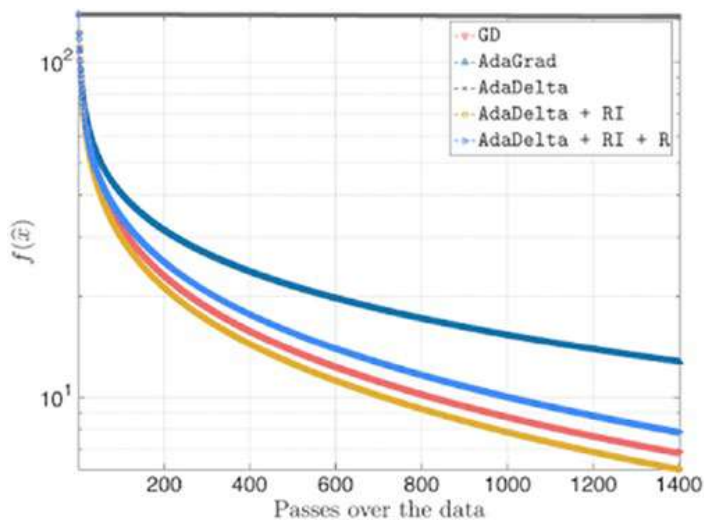


Fig. 3.8. AdaDelta

3.6. Forward Propagation

A perceptron accepts inputs or “features,” processes them, and then predicts the output. This output is then compared with the labels to measure the error. This is known as the forward propagation.

The data is fed into a layer of the neural network, passed through the activation function, and then fed to the next layer. The data must move forward to ensure that an output is achieved.

Thus, for any hidden layer in a neural network, after the first layer, the input is nothing but the output that is generated from the previous layer.

3.7. Back Propagation

Back propagation of the error is a deep learning algorithm that is used in training a supervised learning model. It calculates the gradient of the loss function corresponding to the weights generated by the network for a single input and output pair. It modifies the values of the weights in order to minimize the loss. It is an efficient method of calculation, and thus makes it feasible to use gradient methods to train multi-layer networks. The algorithm computes the gradient of the loss function with respect to each weight by the chain rule, by proceeding one layer at a time. It iterates backward from the final layer. This is done to avoid redundant calculations during the process.

3.8. Types of Neural Networks

There are several types of neural networks, all based on their structure, composition, and flow. Let’s go ahead and discuss a few of the common and most important ones that are used by deep learning developers.

3.8.1. Single-Layer Neural Networks:

The perceptron is the oldest single-layer neural network. As you have seen before, it takes the input from the user, multiplies it by the corresponding weight, adds that to the bias to get the net input function, and then passes the result through the activation function to get the final output. Every perceptron produces only a single output.

This type of neural network is not very efficient due to its extremely limited complexity. Thus, researchers came up with a model that contained more than one layer of perceptron's.

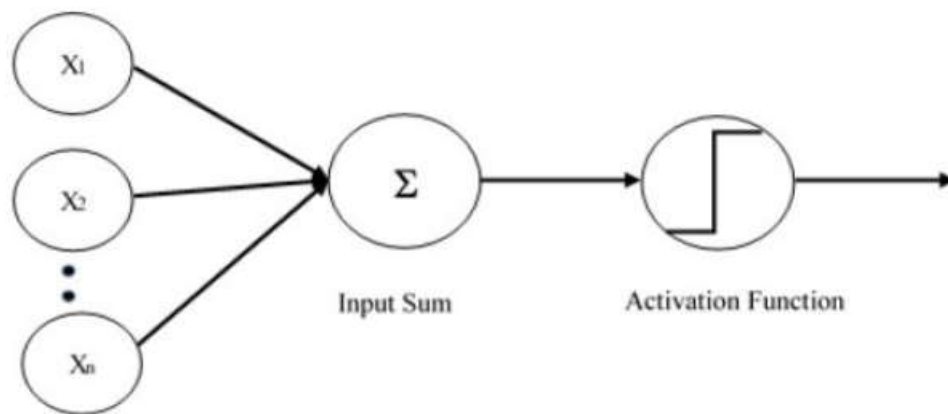


Fig. 3.9. Single-layer perceptron

3.8.2. Multi-Layer Neural Networks

This type of neural network is used mainly for natural language processing, speech recognition, image recognition, etc. It consists of two or more layers of perceptron's, as follows:

- **Input layer:** This is all the available numerical data that is fed into the system and then transferred to the rest of the neural network.

Hidden layers: This is where all the neurons are located. Every layer can have any number of neurons. They are known as “hidden” layers because they remain hidden within the neural network as they perform the necessary computations.

- Output layer: This is the final result of all the calculations that happened in the hidden layers.

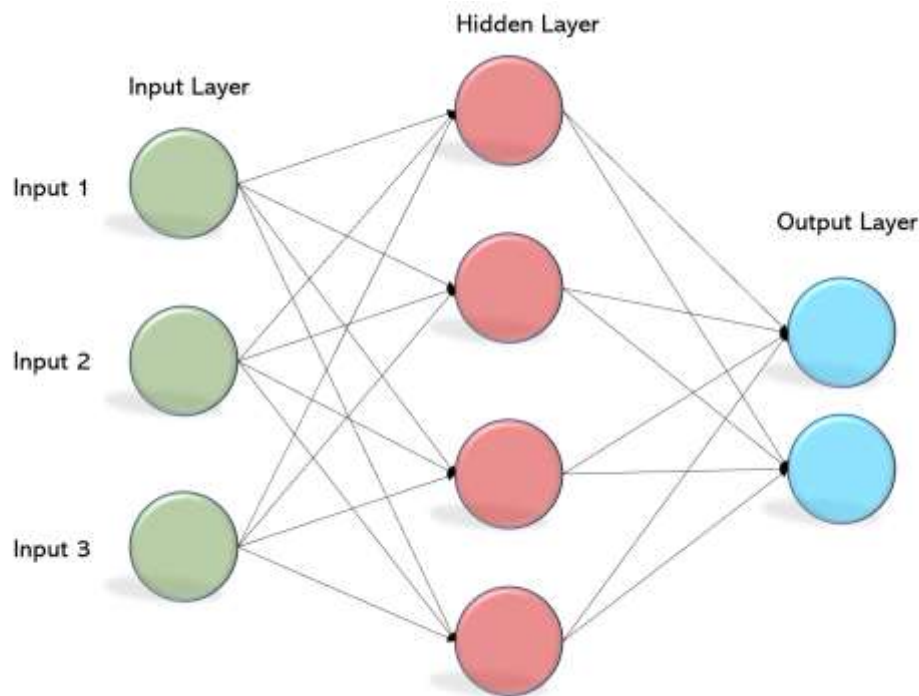


Fig. 3.10. Multi-layer perceptron

3.9. Recurrent Neural Networks

Recurrent neural networks (RNNs) are used for temporal data; i.e., data that requires past experiences to predict future outcomes. State matrices remember previous states of data by storing the last output, and then use this data to calculate the new output.

Long short-term memory (LSTM) networks save the state matrices in two states: long term and short term. RNNs begin in the layers after the first layer. Here, each

node acts as a memory cell during the computation, which allows it to compare previous values with new values during back propagation. These neural networks can be used for stock market predictions, natural language processing, and price determination.

Sequence-to-Sequence Models: A sequence-to-sequence model is mainly used when the lengths of the input data and output data are unequal.

It makes use of two recurrent neural networks, along with an encoder and a decoder. The encoder processes the input data, while the decoder processes the output data. These models are usually used for chatbots and machine translation.

3.10. Modular Neural Networks

Modular neural networks have several different networks that each work independently to complete a part of the entire task. These networks are not connected to each other, and so do not interact with each other during this process. This helps in reducing the amount of time taken to perform the computation by distributing the work done by each network. Each sub-Task would require only a portion of the total time, power, and resources needed to complete the work.

3.11. Convolutional Neural Networks

Convolutional neural networks follow the same principle as multi-layer neural networks, the only difference being that they include “convolutional layers,” which make use of filters.

A filter is a grid of size $A \times B$ that is moved across the image and gets multiplied several times by it to produce a new value. Each value represents a line or an edge in the image.

Once the filters have been used on the image, its important characteristics can be extracted. This is done with the help of a pooling layer. These layers pool or collect the main features of each image. One popular technique of doing this is known as max pooling, which takes the largest number of each image and stores it in a separate grid. It thus compresses the main features into a single image and then passes it on to a regular multi-layer neural network for further processing.

These neural networks are mainly used for image classification. They can also be used in search engines and recommender systems.

3.11.1. Implementation of Convolutional Neural Network

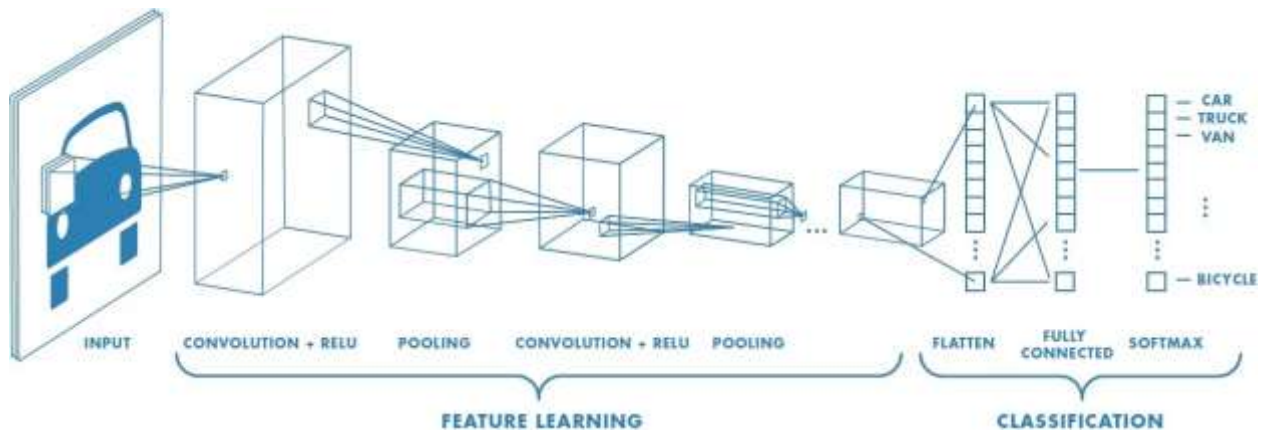


Fig. 3.11. CNN layers

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.

Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters.

The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights.

In the figure, we have an RGB image which has been separated by its three-color planes — Red, Green, and Blue.

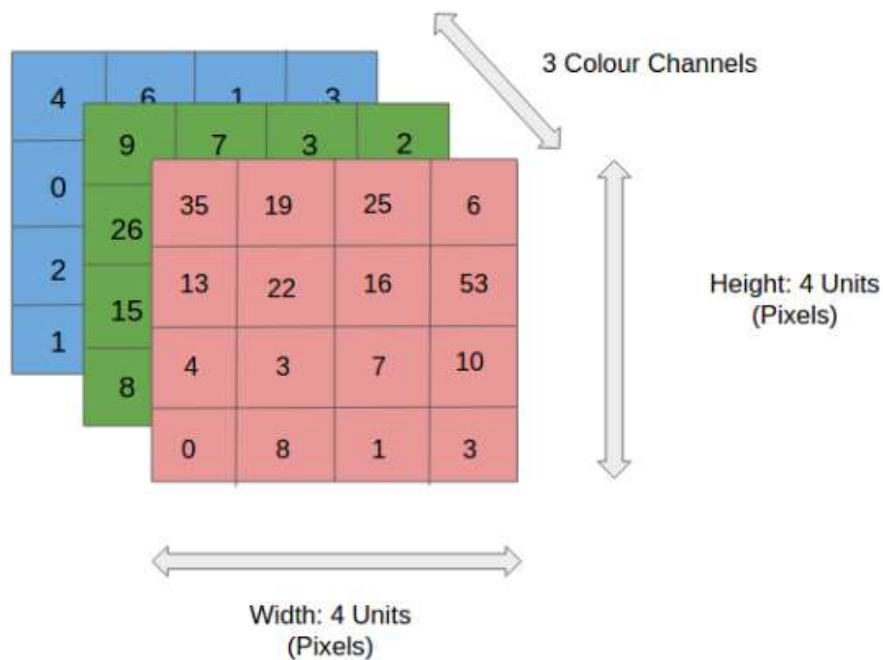


Fig. 3.12. 4x4x3 RGB image

You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.

This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

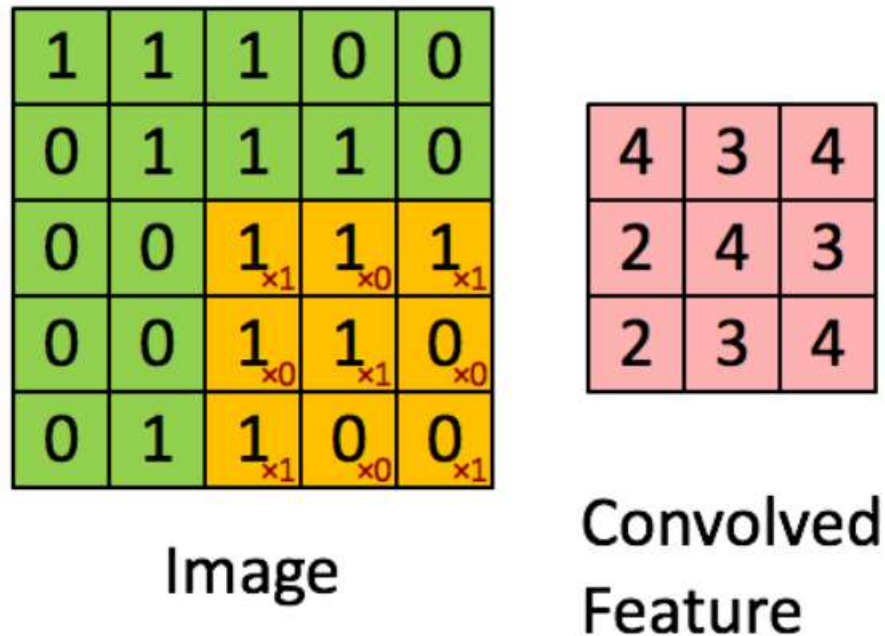


Fig. 3.13. Convoluting a $5 \times 5 \times 1$ image with a $3 \times 3 \times 1$ convolved feature

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB). In the above demonstration, the green section resembles our $5 \times 5 \times 1$ input image, I. The element involved in carrying out the convolution operation in the first part of

a Convolutional Layer is called the Kernel/Filter, K , represented in the color yellow. We have selected K as a $3 \times 3 \times 1$ matrix.

The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering.

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between K_n and I_n stack ($[K_1, I_1]; [K_2, I_2]; [K_3, I_3]$) and all the results are summed with the bias to give us a squashed one-depth channel Convolved Feature Output.

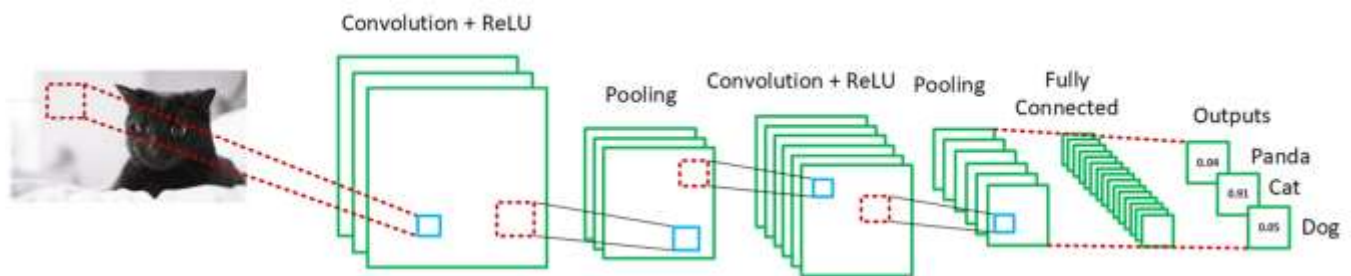


Fig. 3.14. Workflow of CNN

3.11.2. Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution image after applying different types of filters (Kernels).

3.11.3. The objective of the Convolution Operation

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would.








Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Fig. 3.15. Common filters

3.11.4. Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

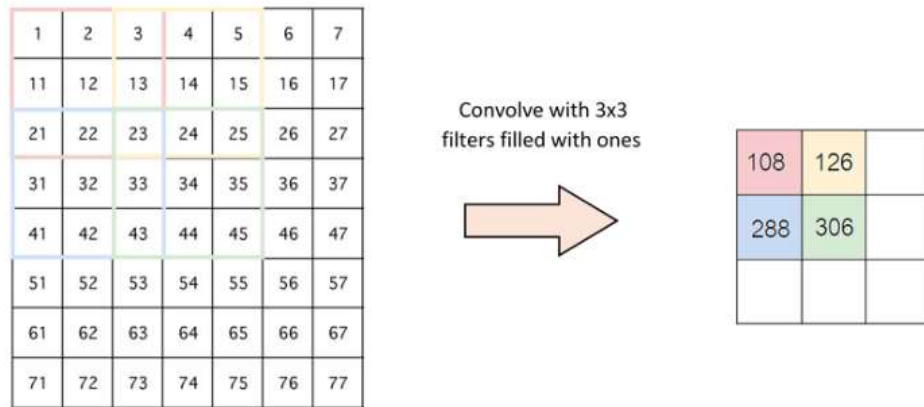


Fig. 3.16. Stride 2 movement

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits. We add a border of 0s around our input Basically, this is equivalent of adding a black border around an image.
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

3.11.5. Pooling Layer

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are

rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling.

Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism.

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

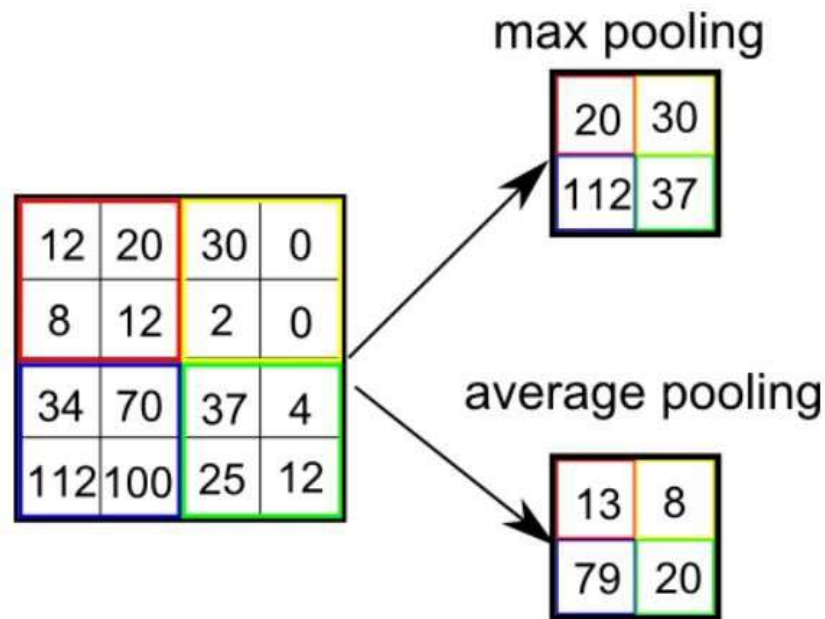


Fig. 3.17. Types of pooling

After going through the above process, we have successfully enabled the model to understand the features. we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

3.11.6. Classification — Fully Connected Layer (FC Layer)

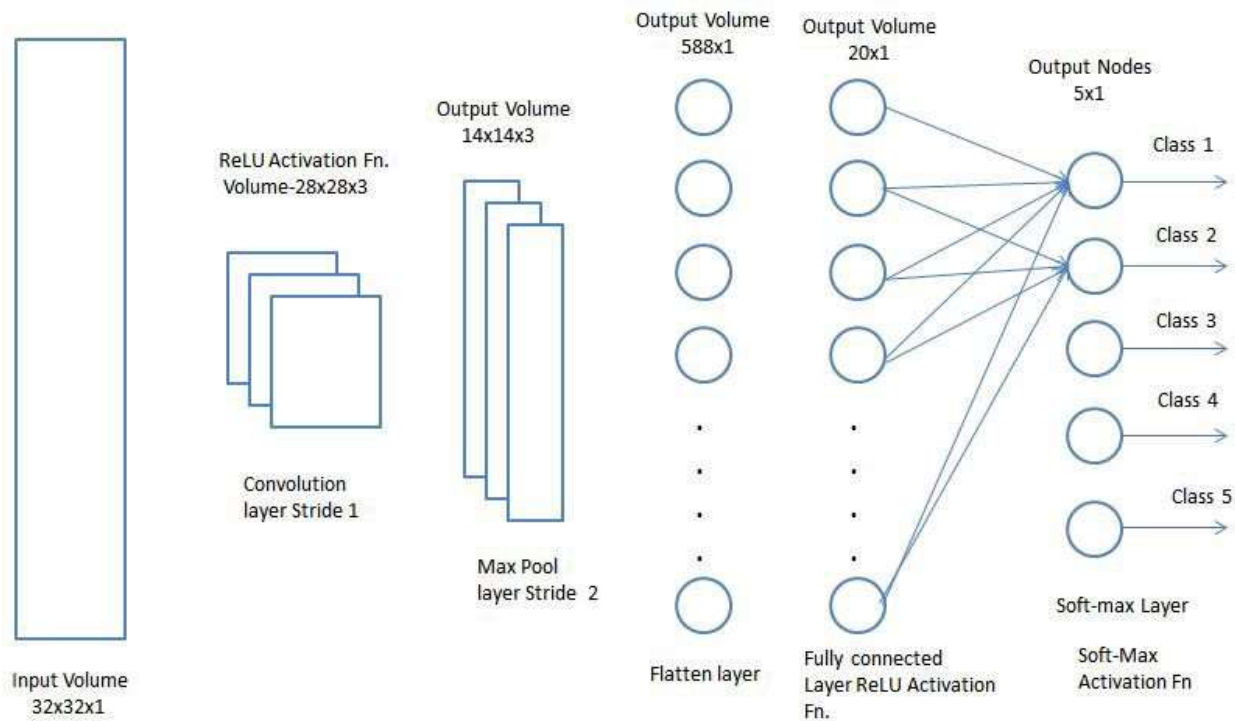


Fig. 3.18. Fully connected layer(FC layer)

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

Now input image been converted into a suitable form for the Multi- Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between

dominating and certain low-level features in images and classify them using the Softmax classification technique.

3.12. Tune hyperparameter in Deep Neural Network

Hyperparameters are the variables which determines the network structure (Number of Hidden Units) and the variables which determine how the network is trained (Learning Rate). Hyperparameters are set before training (before optimizing the weights and bias).

Hyperparameters related to Network structure are:

Number of Hidden Layers and units: Hidden layers are the layers between input layer and output layer. Many hidden units within a layer with regularization techniques can increase accuracy. Smaller number of units may cause underfitting.

Drop out: The term “dropout” refers to dropping out units (both hidden and visible) in a neural network. Simply put, dropout refers to ignoring units (i.e., neurons) during the training phase of certain set of neurons which is chosen at random. By “ignoring” More technically, at each training stage, individual nodes are either dropped out of the net with probability $1-p$ or kept with probability p , so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. The reason why we need drop out is to avoid overfitting. A fully connected layer occupies most of the parameters, and hence, neurons develop co-dependency amongst each other during training which curbs the individual power of each neuron leading to over-fitting of training data.

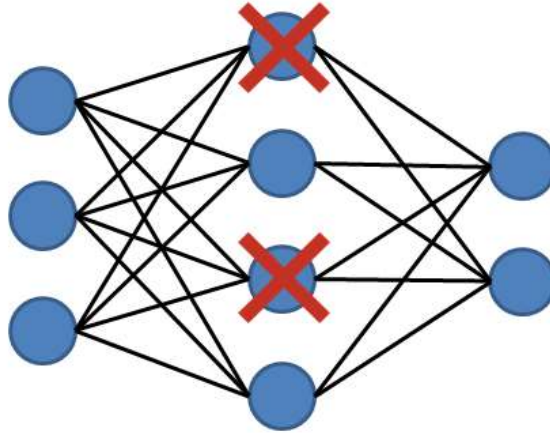


Fig. 3.19. Dropout

Network Weight Initialization: Ideally, it may be better to use different weight initialization schemes according to the activation function used on each layer.

Activation function: Activation functions are used to introduce nonlinearity to models, which allows deep learning models to learn nonlinear prediction boundaries.

Generally, the rectifier activation function is the most popular. Sigmoid is used in the output layer while making binary predictions. Softmax is used in the output layer while making multi-class predictions.

Hyperparameters related to Training Algorithm:

Learning Rate: The learning rate defines how quickly a network updates its parameters.

Low learning rate slows down the learning process but converges smoothly. Larger learning rate speeds up the learning but may not converge.

Momentum: Momentum helps to know the direction of the next step with the knowledge of the previous steps. It helps to prevent oscillations. A typical choice of momentum is between 0.5 to 0.9.

Number of epochs: Number of epochs is the number of times the whole training data is shown to the network while training. Increase the number of epochs until the validation accuracy starts decreasing even when training accuracy is increasing(overfitting).

Batch size: Batch size is the number of sub samples given to the network after which parameter update happens.

3.13. Important terms related to Convolutional Neural Network

3.13.1. Overfitting

Overfitting is a statistical concept. It occurs when an analysis is said to be too accurate with respect to the data provided to it, and thus it can result in an improperly trained model. When we train our model, we may get very high accuracy. However, when testing the model, we may find a drastic difference in the accuracy. This is the result of overfitting.

It can happen when we train a model too many times, or with too little data. The model ends up getting very familiar with the training data and can thus achieve a very high accuracy with it. However, it messes up anyway when it comes to making predictions on new data, because it has still not been trained in the right way.

3.13.2. Batch

The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters. Training dataset can be divided into one or more batches.

When all training samples are used to create one batch, the learning algorithm is called batch gradient descent. When the batch is the size of one sample, the learning algorithm is called stochastic gradient descent. When the batch size is more than one sample and less than the size of the training dataset, the learning algorithm is called mini-batch gradient descent. Batch Gradient Descent.

Batch Size = Size of Training Set Stochastic Gradient Descent. Batch Size = 1.
Mini-Batch Gradient Descent. $1 < \text{Batch Size} < \text{Size of Training Set}$.

In the case of mini-batch gradient descent, popular batch sizes include 32, 64, and 128 samples.

3.13.3. Epoch

The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset.

One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches. For example, as above, an epoch that has one batch is called the batch gradient descent learning algorithm.

The number of epochs is traditionally large, often hundreds or thousands, allowing the learning algorithm to run until the error from the model has been sufficiently

minimized. You may see examples of the number of epochs in the literature and in tutorials set to 10, 100, 500, 1000, and larger.

3.13.4. The Difference Between Batch and Epoch

The batch size is a number of samples processed before the model is updated. The number of epochs is the number of complete passes through the training dataset. The size of a batch must be more than or equal to one and less than or equal to the number of samples in the training dataset.

The number of epochs can be set to an integer value between one and infinity. You can run the algorithm for as long as you like and even stop it using other criteria besides a fixed number of epochs, such as a change (or lack of change) in model error over time.

They are both integer values and they are both hyperparameters for the learning algorithm, e.g. parameters for the learning process, not internal model parameters found by the learning process. Finally, let's make this concrete with a small example.

Assume you have a dataset with 200 samples (rows of data) and you choose a batch size of 5 and 1,000 epochs.

This means that the dataset will be divided into 40 batches, each with five samples. The model weights will be updated after each batch of five samples.

This also means that one epoch will involve 40 batches or 40 updates to the model. With 1,000 epochs, the model will be exposed to or pass through the whole dataset 1,000 times. That is a total of 40,000 batches during the entire training process.

3.13.5. Dense layer

Dense layer is the regular deeply connected neural network layer. It is most common and frequently used layer. Dense layer does the below operation on the input and return the output.

output = activation (dot (input, kernel) + bias)

where, input represent the input data and kernel represent the weight data

dot represent NumPy dot product of all input and its corresponding weights.

Bias represents a biased value used in machine learning to optimize the model activation represent the activation function

3.13.6. Normalization

Normalization is a procedure to change the value of the numeric variable in the dataset to a typical scale, without misshaping contrasts in the range of value. This has the impact of settling the learning process and drastically decreasing the number of training epochs required to train deep neural networks.

And that technique often used as part of data preparation for Machine learning and the goal is to change the values of numeric columns in dataset to a common scale without distorting differences in the range of values

It is not required for all dataset, its required only when features have different values. For example, age is from 0-100, on the other hand, the salary is from 1000-2000, Therefor the salary will have more influence on the result.

CONCLUSION ON THE THIRD PART

A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer.

CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as $[32 \times 32 \times 12]$ if we decided to use 12 filters.

RELU layer will apply an elementwise activation function, such as the $\max(0, x)$ thresholding at zero. This leaves the size of the volume unchanged ($[32 \times 32 \times 12]$).

POOL layer will perform a down sampling operation along the spatial dimensions (width, height), resulting in volume such as $[16 \times 16 \times 12]$.

FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size $[1 \times 1 \times 10]$, where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

In summary:

- A ConvNet architecture is in the simplest case a list of Layers that transform the image volume into an output volume (e.g. holding the class scores)

- There are a few distinct types of Layers (e.g. conv/fc/relu/pool are by far the most popular)
- Each Layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function
- Each Layer may or may not have parameters (e.g. CONV/FC do, RELU/POOL don't)
- Each Layer may or may not have additional hyperparameters (e.g. CONV/FC/POOL do, RELU doesn't)

PART 4: PRACTICAL SIDE, IMPLEMENTING CNN FAMOUS MODEL TO DETECT FACE EMOTIONS

4.1. TensorFlow

TensorFlow is an open-source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy-to-understand framework.

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.

It includes a programming support of deep neural networks and machine learning techniques.

It includes a high scalable feature of computation with various data sets.

TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

TensorFlow is well-documented and includes plenty of machine learning libraries.

It offers a few important functionalities and methods for the same.

TensorFlow is also called a “Google” product. It includes a variety of machine learning and deep learning algorithms. TensorFlow can train and run deep neural

networks for handwritten digit classification, image recognition, word embedding and creation of various sequence models.

4.2. Keras

Keras is compact, easy to learn, high-level Python library run on top of TensorFlow framework. It is made with focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details. The creation of framework can be of the following two types

- Sequential API
- Functional API

Consider the following eight steps to create deep learning model in Keras:

Loading the data, Preprocess the loaded data, Definition of model, Compiling the model, Fit the specified model, evaluate it, Make the required predictions, Save the model.

4.3. Popular CNN Architecture

The reason why we use a pre-trained model is to build a intuition for good model architecture, and to use these pre-trained models without retraining for our problem.

The ImageNet Large Scale Visual Recognition Challenge or ILSVRC for short is an annual competition held between 2010 and 2017 in which challenge tasks use subsets of the ImageNet dataset.

The goal of the challenge was to both:

- 1.Promote the development of better computer vision techniques
- 2.To benchmark the state of the art

Year	CNN	Developed By	Error rates	No. of parameters
1998	LeNet	Yann LeCun et al		60 thousand
2012	AlexNet	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	15.3%	60 million
2013	ZFNet	Matthew Zeiler, Rob Fergus	14.8%	
2014	GoogLeNet	Google	6.67%	4 million
2014	VGGNet	Simonyan, Zisserman	7.3%	138 million
2015	ResNet	Kaiming He	3.6%	

Fig. 4.1. ImageNet winner summary

VGG16 & VGG19

This architecture, which was one of the first to appear, was introduced by Simonyan and Zisserman in 2014 with their paper entitled Very Deep Convolutional Networks for Large Scale Image Recognition.

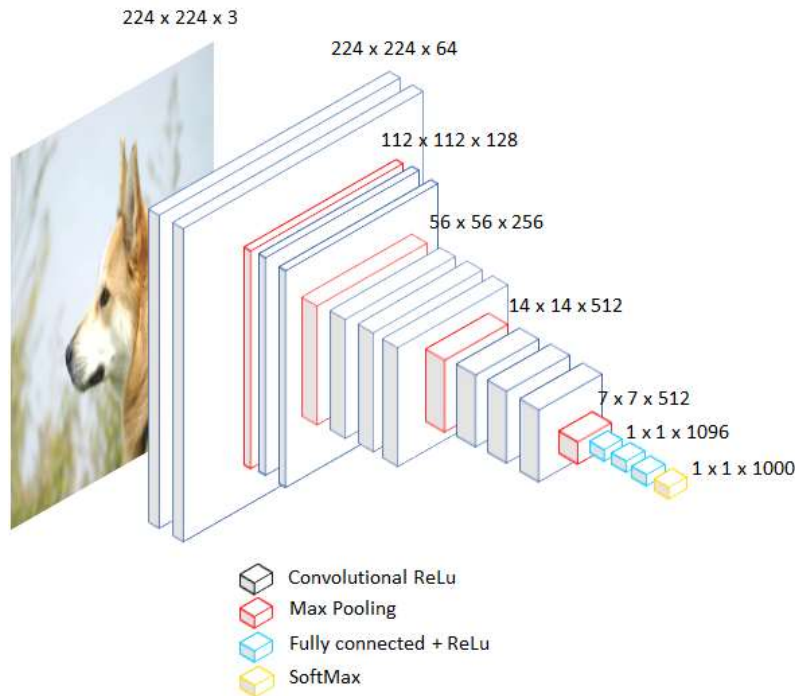


Fig. 4.2. VGG16

It is a simple architecture, using only blocks composed of an incremental number of convolutional layers with 3x3 size filters. Besides, to reduce the size of the activation maps obtained, max-pooling blocks are interspersed between the convolutional ones, reducing the size of these activation maps by half. Finally, a classification block is used, consisting of two dense layers of 4096 neurons each, and the last layer, which is the output layer, of 1000 neurons.

The 16 and 19 refer to the number of weighted layers that each network has (convolutional and dense layers, pooling layers are not counted). They correspond to columns D and E in the table below.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig. 4.3. VGG16 and VGG19

The rest of the architectures in the table are there because, at that time, Simonyan and Zisserman had a hard time training their architecture to converge. Since they couldn't do it, what they came up with was to train networks with simpler

architectures first, and once these converged and were trained, they took advantage of their weights to initialize the next network, which was a little more complex, and so on until they got to the VGG19. This process is known as “pre-training”.

This network, however, has a couple of disadvantages: It takes too long to train, it has a very high number of parameters.

ResNet

The ResNet architecture, developed by He et al. in 2015, was a milestone in introducing an exotic type of architecture based on “modules”, or as it is now known, “networks within networks”. These networks introduced the concept of “residual connections”.

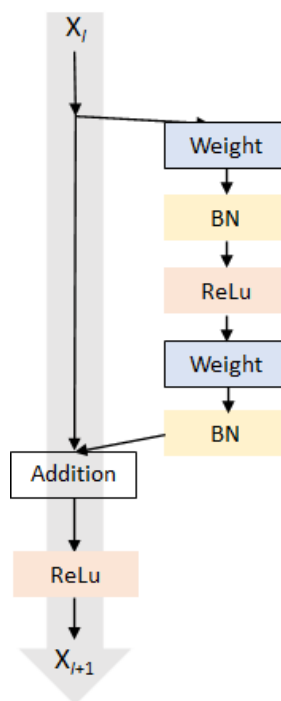


Fig. 4.4. ResNet

These blocks allow to reach the layer $l+1l+1$ part of the previous activation map without modification, and partly modified by the block belonging to the layer l , as you can see in the image above.

In 2016 they improved this architecture by including more layers in these residual blocks.

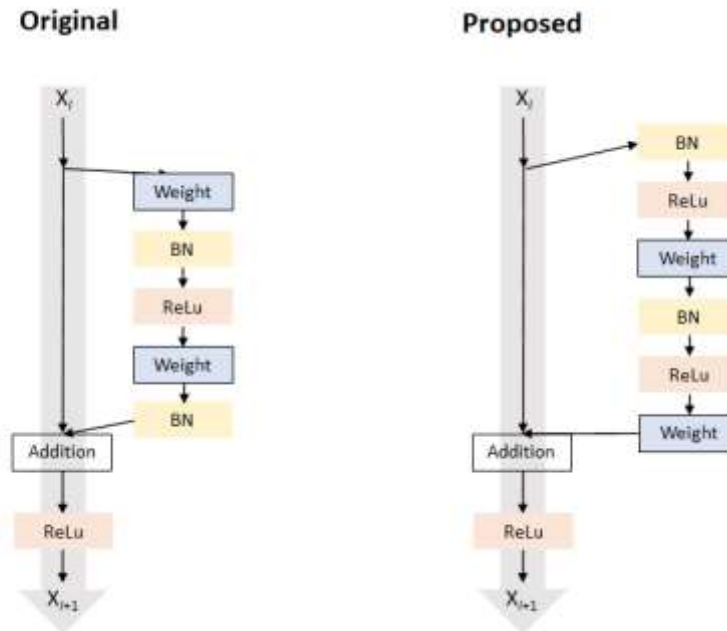


Fig. 4.5. Improvement of ResNet

There are variations of ResNet with a different number of layers, but the most used is ResNet50, which consists of 50 layers with weights. It is remarkable that although it has many more layers than the VGG, it needs much less memory, almost 5 times less. This is because this network, instead of dense layers in the classification stage, uses a type of layer called Global Average Pooling, which converts the 2D activity maps of the last layer in the feature extraction stage to an n-classes vector that is used to calculate the probability of belonging to each class.

Inception V3

This type of architecture, which was introduced in 2014 by Szegedy et al. in their paper “Going Deeper with Convolutions”, uses blocks with filters of different sizes that are then concatenated to extract features at different scales.

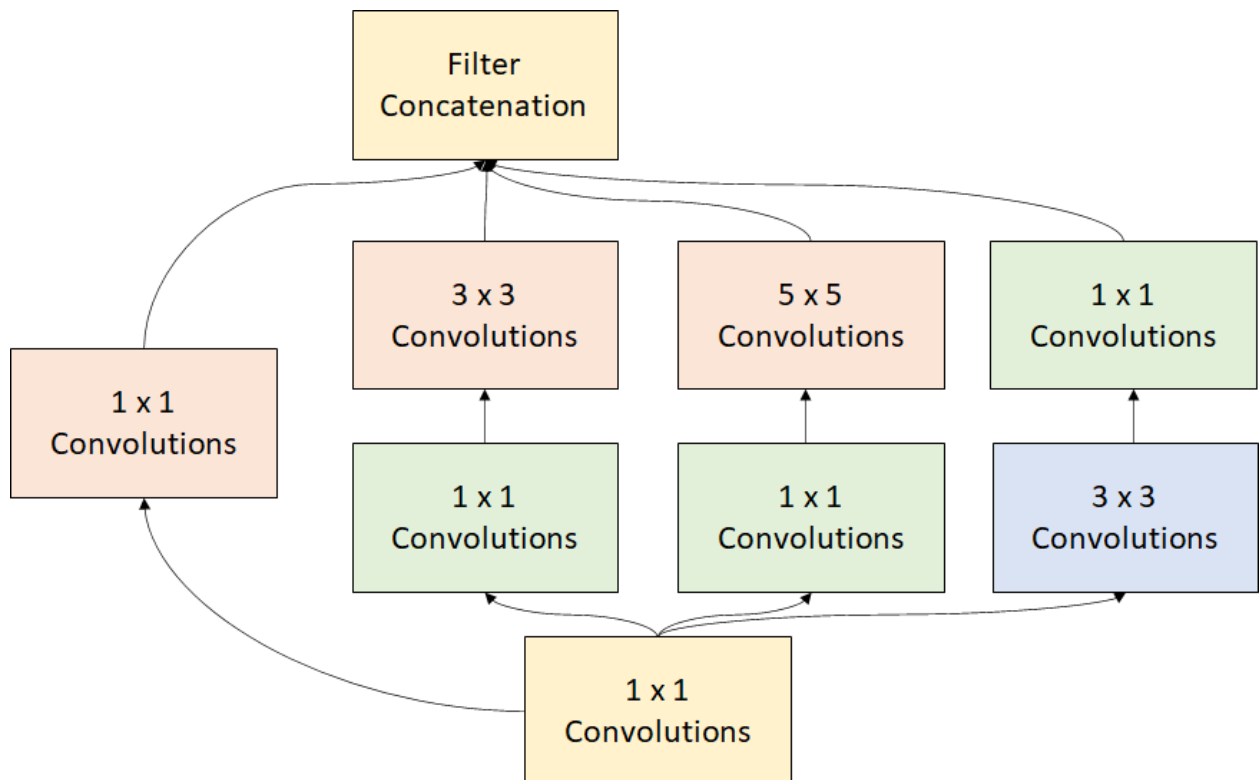


Fig. 4.6. Inception V3

To help you understand this, the goal of the inception block is to calculate activation maps with 1x1, 3x3 and 5x5 convolutions to extract features at different scales. Then you simply concatenate all these activation maps into one. This architecture requires even less memory than the VGG and ResNet.

Xception

This architecture was proposed by François Chollet (the creator of Keras) and the only thing he brings to Inception is that he optimally makes the convolutions so

that they take less time. This is achieved by separating the 2D convolutions into 2 1D convolutions.

In terms of memory, it is very similar to Xception, and this is the outline of its architecture:

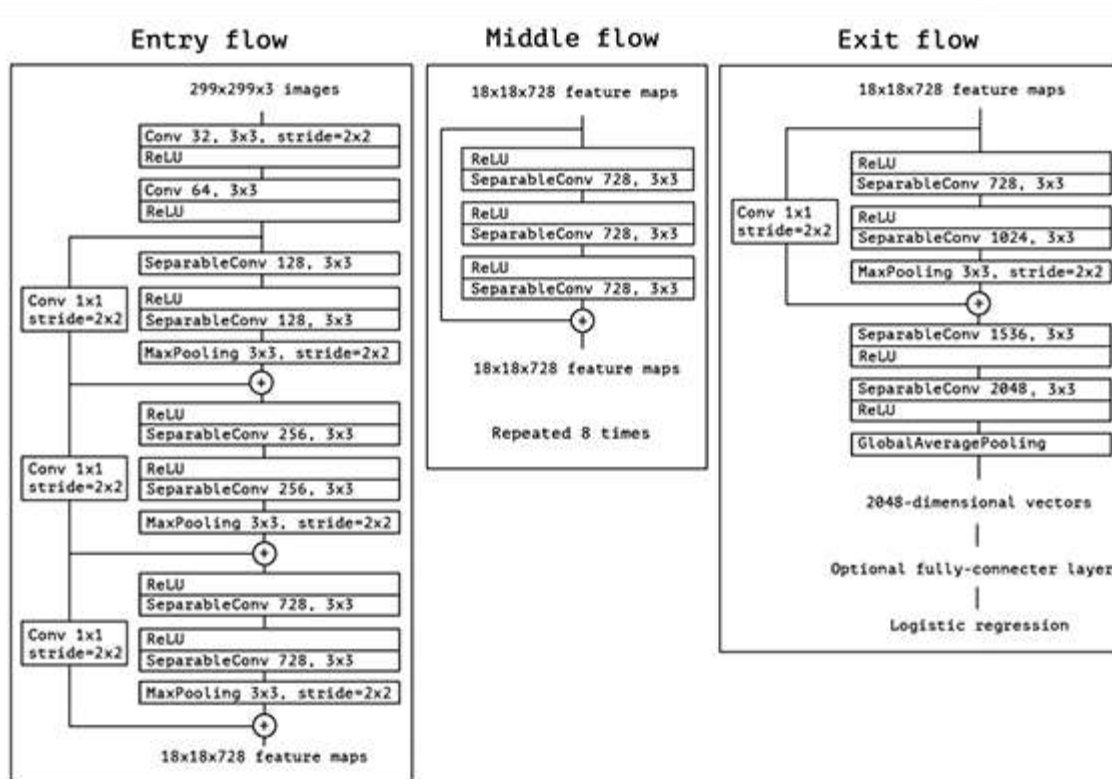


Fig. 4.7. Xception

4.4. Introducing the concept of Transfer Learning and Fine-Tuning

Training a complicated and Deep CNN's is slow, AlexNet and VGG (in particular) are deep, parameter laden networks. VGG has over 138M parameters! ResNet50 has 50 hidden layers While these networks, attain relatively excellent performance on ImageNet, training them on a CPU is an exercise in futility. These CNNs are often trained for a couple weeks or more using arrays of GPUs.

The concept of Fine Tuning is often and justifiably confused with Transfer Learning. However, it is merely a type of Transfer Learning.

Fine Tuning is where we take a pre-trained Deep CNN (ResNet50, Inception or VGG) and use the already trained (typically on ImageNet) Convolutional Layers to aid our new image classification task. Typically, in fine tuning we are taking an already trained CNN and training it on our new dataset. We then either Freeze these lower layers, and train the top or FC layers only. Sometimes we do go back and train the lower layers to improve our performance.

In most deep CNN's the first few CONV layers learn low level features and as we progress through the network, it learns more mid/high level features or patterns.

In Fine Tuning we keep or Freeze these trained low-level features, and only train our high-level features needed for our new image classification problem.

We freeze the already pre-trained lower CONV layers as they well trained to capture universal type features (curves, edges, blobs). We replace the top FC layer with our own, mainly because we maybe training to fit a different number of classes e.g., going from 1000 to 2.

Using a small learning rate helps us to avoid distorting the newly initialized weights too quickly. Transfer Learning. As we have done in Fine Tuning, we have taken a pre-trained network and trained it (or segments/layers of it) on some new data for a new image classification task. (it does not have to be new data or a new goal, Fine Tuning simply implies we further train an already trained network).

Transfer Learning, is almost the same thing and is often used interchangeably with Fine Tuning. It implies we are taking the ‘knowledge’ learned from a pre-trained network

4.5. Data Augmentation

In deep learning, the more training data/examples we have the better our model will perform on unseen data (test data), but in some cases we don’t have enough data, to solve this problem we use data augmentation which has many benefits, first of all it takes small dataset and make it much larger therefore less efforts in creating the dataset.

By adding different variations of data such as zooming, shifts, rotations, etc. make our classifier much more invariant to changes in our images. Thus, making it far more robust.

Moreover, reduces the overfitting due to the increased variety in the training dataset. Kera’ s built-in Data Augmentation API, performs a just-in-time augmented image dataset. This means images aren’ t created and dumped to a directory (which will be wasteful storage). Instead, it generates this dataset during the training process

Keras provides many types of augmentations such as Rotations, Horizontal and vertical shifts, shearing, zooming.

4.6. Confusion matrix

Model accuracy is not a reliable metric of performance, because it will yield misleading results if the validation data set is unbalanced. For example, if there were 90 cats and only 10 dogs in the validation data set and if the model predicts

all the images as cats. The overall accuracy would be 90%. The confusion matrix allows us to visualize the performance of the trained model. It makes it easy to see if the system is confusing two classes. It also summarizes the results of testing the model for further inspection.

```
[[ 977    0    0    0    0    0    1    1    1    0]
 [    0 1130    3    1    0    0    1    0    0    0]
 [    1    0 1029    0    1    0    0    1    0    0]
 [    0    0    4 1003    0    1    0    1    1    0]
 [    0    0    0    0  976    0    0    0    2    4]
 [    3    0    0    5    0  881    3    0    0    0]
 [    6    2    0    0    1    1  948    0    0    0]
 [    1    2   11    2    0    0    0 1010    1    1]
 [    4    0    3    0    0    0    0    2  963    2]
 [    1    2    0    1    5    3    0    2    5  990]]
```

Fig. 4.8. Confusion matrix

Recall

Actual true positives over how many times the classifier predicted that class.

Let's look at the number 7:

TP = 1010 and FN = 18

$1010 / 1028 = 98.24\%$

$$Recall = \frac{TP}{TP + FN}$$

	0	1	2	3	4	5	6	7	8	9	
[[977	0	0	0	0	0	0	1	1	1	0]	0
[[0	1130	3	1	0	0	0	1	0	0	0]	1
[[1	0	1029	0	1	0	0	0	1	0	0]	2
[[0	0	4	1003	0	1	0	0	1	1	0]	3
[[0	0	0	0	976	0	0	0	0	2	4]	4
[[3	0	0	5	0	881	3	0	0	0	0]	5
[[6	2	0	0	1	1	948	0	0	0	0]	6
[[1	2	11	2	0	0	0	1010	1	1]	7	
[[4	0	3	0	0	0	0	0	2	963	2]	8
[[1	2	0	1	5	3	0	2	5	990]]	9	

Fig. 4.9. Recall

Precision

Number of correct predictions over how many occurrences of that class were in the test dataset. Let's look at the number 7:

$$TP = 1010 \text{ and } FP = 7$$

$$1010 / 1017 = 99.31\%$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

	0	1	2	3	4	5	6	7	8	9	
[977	0	0	0	0	0	1	1	1	0]	0
[0	1130	3	1	0	0	1	0	0	0]	1
[1	0	1029	0	1	0	0	1	0	0]	2
[0	0	4	1003	0	1	0	1	1	0]	3
[0	0	0	0	976	0	0	0	2	4]	4
[3	0	0	5	0	881	3	0	0	0]	5
[6	2	0	0	1	1	948	0	0	0]	6
[1	2	11	2	0	0	0	1010	1	1]	7
[4	0	3	0	0	0	0	2	963	2]	8
[1	2	0	1	5	3	0	2	5	990]]	9

Fig. 4.10. Recall

F-measure

It is difficult to compare two models with low precision and high recall or vice versa. So, to make them comparable, we use F-Score. F-score helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more.

Using scikit-learn we can automatically generate a classification Report that gives us Recall, Precision, F1 and Support

	precision	recall	f1-score	support
0	0.98	1.00	0.99	980
1	0.99	1.00	1.00	1135
2	0.98	1.00	0.99	1032
3	0.99	0.99	0.99	1010
4	0.99	0.99	0.99	982
5	0.99	0.99	0.99	892
6	0.99	0.99	0.99	958
7	0.99	0.98	0.99	1028
8	0.99	0.99	0.99	974
9	0.99	0.98	0.99	1009

Fig. 4.11. F-measure

4.7. Classification Report Analysis

High recall with low precision. This tells us that most of the positive examples are correctly recognized (low False Negatives) but there are a lot of false positives i.e. other classes being predicted as our class in question.

Low recall with high precision. Our classifier is missing a lot of positive examples (high FN) but those we predict as positive are indeed positive (low False Positives)

4.8. Batch Normalization

Previously, we applied a generic form of Normalization scaling our image data from values between 0-255 to values between 0 to 1 (this was done by dividing it by 255). We do this to reduce the influence of larger data points

Batch Norm however is used to normalize the activations of a input tensor before passing it into the next layer in the network.

Before we begin training our NN, we randomize our weights, but what if one our weights becomes extremely large, it's corresponding Neuron's output will be very

large! Cascading throughout our NN causing instability. Moreover, Batch Normalization is applied to a layer we select in our NN.

Batch Norm normalizes the output from the activation functions of a selected layer. It then normalizes the output by multiplying it by a parameter and then adding another parameter to this result. The result is that all the activations leaving a batch normalization layer will have approximately zero mean, the weights now don't become imbalanced with extreme values since normalization is now included in the gradient process.

Batch Normalization reduces the number of Epochs it takes our NN to converge, it aids in regularization (reducing overfitting), it allows us to improve stability of our training, thus allowing us to use large learning rates, the downside is that batch Normalization slows down training

4.9. The Dying ReLU Problem

ReLU's quite often face the issue of dying, especially when the learning rate is set to a higher value, as this triggers weight updating that doesn't allow the activation of the specific neurons, thereby making the gradient of that neuron forever zero. Another risk offered by ReLU is the explosion of the activation function, as the input value, x_j , is itself the output here. Although ReLU offers other benefits as well, such as the introduction of sparsity in cases where x_j is below 0, leading to sparse representations, and as the gradient returned in cases where ReLU is constant, it results in faster learning, accompanied by the reduced likelihood of the gradient vanishing.

Fixing the Dying ReLU problem

- LReLU's (Leaky ReLU's): These mitigate the issue of dying ReLU's by introducing a marginally reduced slope (~ 0.01) for values of x less than 0. LReLU's do offer successful scenarios.
- ELU (Exponential Linear Unit): These offer negative values that push the mean unit activations closer to zero, thereby speeding the learning process, by moving the nearby gradient to the unit natural gradient.
- Leaky ReLU (small negative slope)
- Parametric or PReLU
- Exponential or ELU (sometimes called Scaled ELU or SELU)

All of the above activation functions fix the dying ReLU problem by having no zero slope parts. ELU tends to be a good mix of the good parts of ReLU and Leaky ReLU, however it can saturate on large negative values.

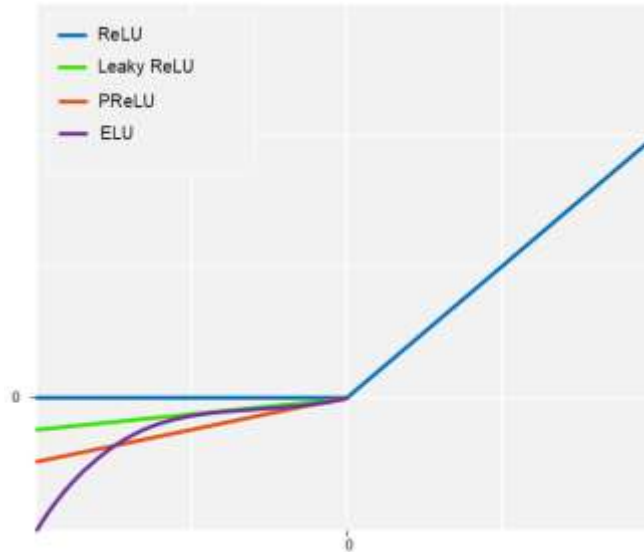


Fig. 4.12. ReLU variants

4.10. Implementing CNN for detecting face and emotion using modified VGG16

In the modified VGG16 (VGG9) we are going to use nine weight layers, each layer has a series of 3x3 kernel size. Convolution to Relu blocks where the number of these filters increases the deeper, we go.

Little VGG or VGG9
9 weight layers
Input(224x224 RGB images)
1.Conv3-64 (kernel size 3x3 and 64 filters) 2.Conv3-64(kernel size 3x3 and 64 filters)

Maxpool
3.Conv3-128 (kernel size 3x3 and 128 filters)
4.Conv3-128(kernel size 3x3 and 128 filters)
Maxpool
5.Conv3-256 (kernel size 3x3 and 256 filters)
6.Conv3-256(kernel size 3x3 and 256 filters)
Maxpool
7.FC-256(Fully-connected layer with 256 Dense)
Maxpool
8.FC-20(Fully-connected layer with 256 Dense)
Maxpool
9.FC-20(Fully-connected layer with 256 Dense)
Soft-max
2,270,804 Parameters

Table.4.1 Modified VGG 16

4.10.1. Our Facial Expression Dataset

Our dataset is hosted on Kaggle and has been around since 2013. It consists of 28,709 examples of faces showing the following Expressions (Angry, fear, Happy, sad, surprise, Neutral)

Our images are grayscale and 48 x 48 pixels



Fig. 4.13. Examples of our dataset

4.10.2. Our Approach

Use our LittleVGG model with the following changes Swapping ReLU activations for ELU, changing our initializations from Kera's default Glorat Normal to HeNormal (tends to work better for VGG type Networks).

We then use OpenCV with a HAAR Cascade Classifier trained on faces, to extract our face from our webcams and classifier our emotions in real time!

```
In [1]: from __future__ import print_function
import tensorflow
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
```

Fig. 4.14. Import libraries

We are importing the required models and layers from keras such as sequential, Dropout, Activation, Batch Normalization, and Conv2D.

```

num_classes = 6
img_rows, img_cols = 48, 48
batch_size = 16

train_data_dir = './fer2013/train'
validation_data_dir = './fer2013/validation'

# Let's use some data augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    shear_range=0.3,
    zoom_range=0.3,
    width_shift_range=0.4,
    height_shift_range=0.4,
    horizontal_flip=True,
    fill_mode='nearest')

```

Fig. 4.15. Data augmentation

After that we are going to define number of classes in our dataset which is 6, in addition. We define the dimensions of our pictures and the batch size for each epoch. Moreover, we add the data augmentation to provide more variants to the model, by adding rotate, shear, flip and rescale features.

```

validation_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    color_mode = 'grayscale',
    target_size=(img_rows, img_cols),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True)

validation_generator = validation_datagen.flow_from_directory(
    validation_data_dir,
    color_mode = 'grayscale',
    target_size=(img_rows, img_cols),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True)

```

Fig. 4.16. Augmented data

We will provide the new pictures of the augmentation features to the objects `train_generator`, `validation_generator`.

Keras LittleVGG Model

```
In [3]: model = Sequential()

model.add(Conv2D(32, (3, 3), padding = 'same', kernel_initializer="he_normal",
                input_shape = (img_rows, img_cols, 1)))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(32, (3, 3), padding = "same", kernel_initializer="he_normal",
                input_shape = (img_rows, img_cols, 1)))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
```

Fig. 4.17. Model building

4.10.3. Building the model

And by now we ready to design our classifier, we build the model using keras sequential API. In this model we provide the following:

9 weight layers and these layers includes: 7 Convolutional neural network, and 2 Fully connected network. The kernel size or filter for each convolutional is 3x3, and the kernel initialization for them is He Normal (tends to work better for VGG type Networks), in addition. The activation function in this model is “elu” and the padding size is 2x2 with Max pooling type is activated for all convolution’s blocks, we keep the Drop out at the range of 20%.

Note that the number of filters in each block is increasing, in the first block we started with 32, in the second block is 64 and in the third and fourth block is 128, 256 respectively.

```

# Block #3: third CONV => RELU => CONV => RELU => POOL
# layer set
model.add(Conv2D(128, (3, 3), padding="same", kernel_initializer="he_normal"))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same", kernel_initializer="he_normal"))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

# Block #4: third CONV => RELU => CONV => RELU => POOL
# layer set
model.add(Conv2D(256, (3, 3), padding="same", kernel_initializer="he_normal"))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), padding="same", kernel_initializer="he_normal"))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

```

Fig. 4.18. Convolution blocks

After 4 blocks of CNN, is the time to prepare the data to Fully connected layers by flatten the data in one vector and having the “elu” activation function and the same initializer for the CNN, the important thing to notice that we add dropout of 50%

In both FC blocks. At the end we provide our model with softmax that used to convert the outputs to values that, when summed up, result in 1. Thus, these values will lie between 0 and 1.

```

# Block #5: first set of FC => RELU Layers
model.add(Flatten())
model.add(Dense(64, kernel_initializer="he_normal"))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Block #6: second set of FC => RELU Layers
model.add(Dense(64, kernel_initializer="he_normal"))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Block #7: softmax classifier
model.add(Dense(num_classes, kernel_initializer="he_normal"))
model.add(Activation("softmax"))

```

Fig. 4.19. Fully connected layers

Softmax is used because we do have a multi-classification problem, the model goal is to predict the emotion of the given pictures.

```
print(model.summary())
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 32)	320
activation (Activation)	(None, 48, 48, 32)	0
batch_normalization (Batch Normalization)	(None, 48, 48, 32)	128
conv2d_1 (Conv2D)	(None, 48, 48, 32)	9248
activation_1 (Activation)	(None, 48, 48, 32)	0
batch_normalization_1 (Batch Normalization)	(None, 48, 48, 32)	128
max_pooling2d (MaxPooling2D)	(None, 24, 24, 32)	0
dropout (Dropout)	(None, 24, 24, 32)	0
conv2d_2 (Conv2D)	(None, 24, 24, 64)	18496
activation_2 (Activation)	(None, 24, 24, 64)	0
batch_normalization_2 (Batch Normalization)	(None, 24, 24, 64)	256
conv2d_3 (Conv2D)	(None, 24, 24, 64)	36928
activation_3 (Activation)	(None, 24, 24, 64)	0
batch_normalization_3 (Batch Normalization)	(None, 24, 24, 64)	256

Fig. 4.20. Summary of model

By using `summary ()` command, it's obvious to see the model layers with all the details and the number of trainable parameters.

So far, the program fetches the data from the dataset and provide the augmentations to enhance the data and make more variant, in addition. We built the model by adding the convolution layers, pooling layers, activation function, FC and SoftMax function at the end of our model.

4.10.4. Training the model

Training our model

```
In [4]: from tensorflow.keras.optimizers import RMSprop, SGD, Adam
        from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLRonPlateau

        checkpoint = ModelCheckpoint("/Users/High-Tech/DeepLearningCV/Trained Models/emotion_little_vgg_3.h5",
                                    monitor="val_loss",
                                    mode="min",
                                    save_best_only = True,
                                    verbose=1)

        earlystop = EarlyStopping(monitor = 'val_loss',
                                  min_delta = 0,
                                  patience = 3,
                                  verbose = 1,
                                  restore_best_weights = True)

        reduce_lr = ReduceLRonPlateau(monitor = 'val_loss', factor = 0.2, patience = 3, verbose = 1, min_delta = 0.0001)

        # we put our call backs into a callback list
        callbacks = [earlystop, checkpoint, reduce_lr]

        # we use a very small learning rate
        model.compile(loss = 'categorical_crossentropy',
                      optimizer = Adam(lr=0.001),
                      metrics = ['accuracy'])

        nb_train_samples = 28273
        nb_validation_samples = 3534
        epochs = 10
```

Fig. 4.21. Model training

Now we shall start training our model by importing keras API. From the TensorFlow library and adding the optimization functions, and callback like the checkpoint and Early Stopping.

The number of epochs is not that significant. More important is the validation and training error. As long as it keeps dropping training should continue. For instance, if the validation loss starts increasing that might be an indication of overfitting and needs to be addressed using some regularization techniques (like data augmentation, dropout, batch normalization). The model should set the number of epochs as high as possible and terminate training based on the error rates. We provide a checkpoint to monitor the validation loss of the model and only save the best model in the training process. Moreover, Early Stopping will have the same parameter for monitor argument. We are going to use a small Learning rate,

however it's important to split the dataset to train and validation samples before start training.

```
nb_train_samples = 28273
nb_validation_samples = 3534
epochs = 10

history = model.fit_generator(
    train_generator,
    steps_per_epoch = nb_train_samples // batch_size,
    epochs = epochs,
    callbacks = callbacks,
    validation_data = validation_generator,
    validation_steps = nb_validation_samples // batch_size)

Epoch 1/10
1767/1767 [=====] - 820s 464ms/step - loss: 1.9752 - acc: 0.2095 - val_loss: 1.7341 - val_acc: 0.2504
Epoch 00001: val_loss improved from inf to 1.73408, saving model to /home/deeplearningcv/DeepLearningCV/Trained Models/emotion_little_vgg_3.h5
Epoch 2/10
1767/1767 [=====] - 755s 427ms/step - loss: 1.7395 - acc: 0.2504 - val_loss: 1.6814 - val_acc: 0.2965
Epoch 00002: val_loss improved from 1.73408 to 1.68137, saving model to /home/deeplearningcv/DeepLearningCV/Trained Models/emotion_little_vgg_3.h5
Epoch 3/10
1767/1767 [=====] - 796s 451ms/step - loss: 1.6683 - acc: 0.3025 - val_loss: 1.5962 - val_acc: 0.3545
Epoch 00003: val_loss improved from 1.68137 to 1.59624, saving model to /home/deeplearningcv/DeepLearningCV/Trained Models/emotion_little_vgg_3.h5
Epoch 4/10
1767/1767 [=====] - 626s 354ms/step - loss: 1.5520 - acc: 0.3702 - val_loss: 1.4812 - val_acc: 0.4369
Epoch 00004: val_loss improved from 1.59624 to 1.48116, saving model to /home/deeplearningcv/DeepLearningCV/Trained Models/emotion_little_vgg_3.h5
Epoch 5/10
1767/1767 [=====] - 621s 351ms/step - loss: 1.4572 - acc: 0.4189 - val_loss: 1.4714 - val_acc: 0.4381
```

Fig. 4.22. Model training

When the validation loss is improving in the model, the checkpoint function will save the module, however when the validation loss is not improving for certain numbers of epochs

The training process will stop and keep the best validation in the storage.

Also, we do have to add class labels and data to our model, and to check the performance of our trained model it's necessary to visualize it by using confusion matrix to see if the system is confusing two classes. It also summarizes the results of testing the model for further inspection.

It's noticeable that scikit-learn library is imported to the model and the reason is that, we can automatically generate a classification Report that gives us Recall, Precision, F1 and Support


```

In [4]: import matplotlib.pyplot as plt
import sklearn
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np

nb_train_samples = 28273
nb_validation_samples = 3534

# We need to recreate our validation generator with shuffle = false
validation_generator = validation_datagen.flow_from_directory(
    validation_data_dir,
    color_mode = 'grayscale',
    target_size=(img_rows, img_cols),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)

class_labels = validation_generator.class_indices
class_labels = {v: k for k, v in class_labels.items()}
classes = list(class_labels.values())

#Confusion Matrix and Classification Report
Y_pred = model.predict_generator(validation_generator, nb_validation_samples // batch_size)
y_pred = np.argmax(Y_pred, axis=1)

print('Confusion Matrix')
print(confusion_matrix(validation_generator.classes, y_pred))
print('Classification Report')
target_names = list(class_labels.values())
print(classification_report(validation_generator.classes, y_pred, target_names=target_names))

plt.figure(figsize=(8,8))
cnf_matrix = confusion_matrix(validation_generator.classes, y_pred)

plt.imshow(cnf_matrix, interpolation='nearest')
plt.colorbar()
tick_marks = np.arange(len(classes))

```

Fig. 4.23. Creating confusion matrix and classification report

In the Fig. 4.24. We find the confusion matrix and classification report. In that report we can notice that f1-score is the highest for happy feature and that confirm that the model can predict the pictures with happy faces with high accuracy, in addition. The support is highest for happy class.

In the Fig. 4.25. We can see a presentation of confusion matrix and it's noticeable that the happy class has the highest value in the f1-score and that value is represented in the figure below.

Another example can be the neutral class which has the value of 270 according to f1-score which considered less than the happy class.

```

Found 3534 images belonging to 6 classes.
Confusion Matrix
[[216  3  39 130  86  17]
 [119 19  49 173  94  74]
 [ 30  1 743  61  33  11]
 [ 95  6 161 203  98  63]
 [ 60  2  47 240 238  7]
 [ 28  8  46  48  7 279]]
Classification Report

```

	precision	recall	f1-score	support
Angry	0.39	0.44	0.42	491
Fear	0.49	0.04	0.07	528
Happy	0.68	0.85	0.76	879
Neutral	0.24	0.32	0.27	626
Sad	0.43	0.40	0.41	594
Surprise	0.62	0.67	0.64	416
micro avg	0.48	0.48	0.48	3534
macro avg	0.48	0.45	0.43	3534
weighted avg	0.48	0.48	0.45	3534

Fig. 4.24. Confusion matrix and classification report

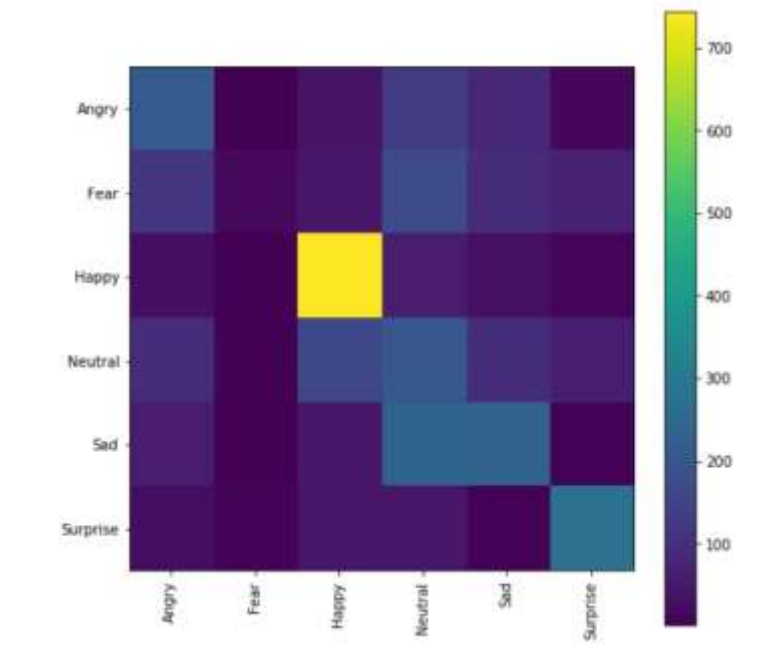


Fig. 4.25. Classification figure

4.10.5. Testing section

After we done with training, we load our model that we trained by using checkpoint model and we extract the class names from the dataset.

Loading our saved model

```
In [3]: from keras.models import load_model
classifier = load_model('/home/deeplearningcv/DeepLearningCV/Trained Models/emotion_little_vgg_3.h5')
```

Get our class labels

```
In [4]: validation_generator = validation_datagen.flow_from_directory(
    validation_data_dir,
    color_mode = 'grayscale',
    target_size=(img_rows, img_cols),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)

class_labels = validation_generator.class_indices
class_labels = {v: k for k, v in class_labels.items()}
classes = list(class_labels.values())
print(class_labels)

Found 3534 images belonging to 6 classes.
{0: 'Angry', 1: 'Fear', 2: 'Happy', 3: 'Neutral', 4: 'Sad', 5: 'Surprise'}
```

Fig. 4.26. Loading the model

The next stage is to test our model on our validation data. We create a window that has the predicted image, in addition. The true label for the image and the image itself.

`getRandomImage ()` is a function to get a random image from our validation dataset which could belong to any class, Moreover. We identify the required picture dimension for our model.

We run the model in the range 0 - 10 images to test our result, However. It's possible to change the value of the range.

Let's test on some of validation images

```
In [5]: from keras.models import load_model
        from keras.optimizers import RMSprop, SGD, Adam
        from keras.preprocessing import image
        import numpy as np
        import os
        import cv2
        import numpy as np
        from os import listdir
        from os.path import isfile, join
        import re

        def draw_test(name, pred, im, true_label):
            BLACK = [0,0,0]
            expanded_image = cv2.copyMakeBorder(im, 160, 0, 0, 300 ,cv2.BORDER_CONSTANT,value=BLACK)
            cv2.putText(expanded_image, "predicted - "+ pred, (20, 60) , cv2.FONT_HERSHEY_SIMPLEX,1, (0,0,255), 2)
            cv2.putText(expanded_image, "true - "+ true_label, (20, 120) , cv2.FONT_HERSHEY_SIMPLEX,1, (0,255,0), 2)
            cv2.imshow(name, expanded_image)

        def getRandomImage(path, img_width, img_height):
            """function loads a random images from a random folder in our test path """
            folders = list(filter(lambda x: os.path.isdir(os.path.join(path, x)), os.listdir(path)))
            random_directory = np.random.randint(0,len(folders))
            path_class = folders[random_directory]
            file_path = path + path_class
            file_names = [f for f in listdir(file_path) if isfile(join(file_path, f))]
            random_file_index = np.random.randint(0,len(file_names))
            image_name = file_names[random_file_index]
            final_path = file_path + "/" + image_name
            return image.load_img(final_path, target_size = (img_width, img_height),grayscale=True), final_path, path_class

        # dimensions of our images
        img_width, img_height = 48, 48
```

Fig. 4.27. Testing the model on validation dataset

The dimension of the image is important step in the validation phase of our model, the width and height should be defined as 48 pixels. In addition, attribute for grayscale should be true

4.10.6. Some right and wrong predictions for our model



Fig. 4.28. Correct prediction

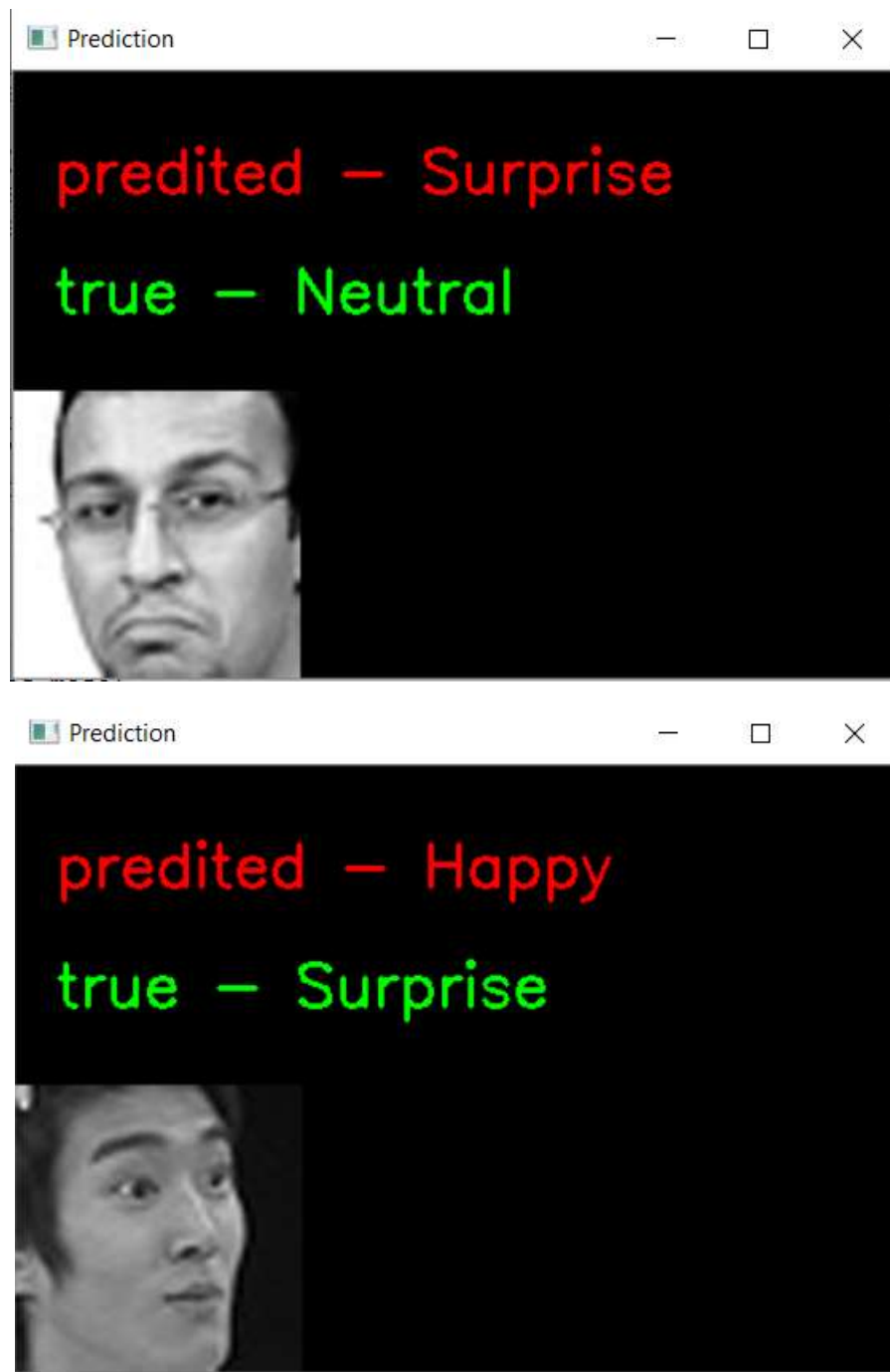


Fig. 4.29. Wrong prediction

It's possible to try the model on live webcam to predict the emotion of the person next to the camera, the model can also detect whether there is a face next to the camera or not.

Let's try this on our webcam

```
In [14]: import cv2
import numpy as np
from time import sleep
from keras.preprocessing.image import img_to_array

face_classifier = cv2.CascadeClassifier('./Haarcascades/haarcascade_frontalface_default.xml')

def face_detector(img):
    # Convert image to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)
    if faces is ():
        return (0,0,0,0), np.zeros((48,48), np.uint8), img

    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        roi_gray = gray[y:y+h, x:x+w]

        try:
            roi_gray = cv2.resize(roi_gray, (48, 48), interpolation = cv2.INTER_AREA)
        except:
            return (x,w,y,h), np.zeros((48,48), np.uint8), img
        return (x,w,y,h), roi_gray, img

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    rect, face, image = face_detector(frame)
    if np.sum([face]) != 0.0:
        roi = face.astype("float") / 255.0
        roi = img_to_array(roi)
        roi = np.expand_dims(roi, axis=0)
```

Fig. 4.30. Testing model on webcam

CONCLUSION ON THE FOURTH PART

The most important stage in building the model is to train the classifier on variant dataset to avoid misclassified classes in the future such as overfitting, the solution for this problem is to provide data augmentation features to our dataset for example, zoom, scale, and shear. There are crucial requirements before training phase such as preprocessing the data, for instance, normalization function to transform our entry values to 0 to 1 range, to avoid the influence of large value data.

The main concept in training CNN is that more training data we have, the better our model will perform on unseen data.

The best approach in training is to use callback functions such as checkpoints(), which allow the classifier to save the best model by monitoring validation loss or other argument, another useful callback is the earlystop() and it's used to stop the training at certain point by monitoring an argument stopped getting any improvement, for example monitoring the validation loss. Callback functions are important to save the time and processing power of the machine for the reason that training a completed and accurate classifier require a lot of time and computational power that's not available for many computers, using many GPU (graphic processing unit) is expensive and not available for everyone as well.

CONCLUSION

The primary goal is to allow computers to learn automatically without human intervention or assistance and to adjust procedures accordingly.

Deep learning is a subset of machine learning in which artificial neural networks are inspired by the human brain. These analyze and synthesize insights from that data, then learn from it later. Any deep learning algorithm will repeat and perform the task iteratively, tweaking and optimizing a little bit each time, in order to improve the outcome. The term "deep learning" has been coined because neural networks have different layers that allow learning, de-learning, and re-learning.

Convolutional Neural Networks are very similar to ordinary Neural Networks
Convolutional neural networks are very similar to normal neural networks.

It consists of nerve cells that have weights and biases that can be learned. Each neuron receives some input, performs a point product and optional follows it in a non-linear manner. The entire network still expresses a single, differential degree function: from pixels of the initial image at one end to degrees of separation at the other end. And they still have loss function (like SVM / Softmax) on the last layer (fully connected).

The ConvNet architectures explicitly assume that the inputs are images, which allows us to encode specific properties in the architecture. These then make the front-end function more efficient to implement and greatly reduce the number of parameters in the network.

Three-dimensional volumes of neurons. Convolutional neural networks take advantage of the fact that the input consists of images and it constrains the structure

in a more logical way. In particular, unlike a normal neural network, ConvNet layers contain neurons arranged in 3 dimensions: width, height and depth.

By modifying the model VGG16 which is a famous Convolutional Neural Network architecture, the application can identify face emotions in the given images that's a result of using many convolution layers and optimization functions to detect the face features in the given image, in addition to use many epochs on our large dataset to train the model and reduce the validation loss as much as possible. Moreover, the application can do the same feature on live webcam. The idea behind developed application is to implement machine learning technology in the field of face and emotion recognition, which is become popular in big smart cities to monitor people movements and behaviors. The common reason to use is for public security and commercial interests to track human interests. A few examples like monitoring public CCTV cameras, immigration checks at airports can for be public security. Also, regarding commercial interests, we can mention about retail stores, educational schools, medical clinics etc. which can use facial recognition to track shoppers, patients, students.

As the result of graduation project, there was gained understanding of the current state of machine learning specially regarding neural networks. There was also gained information about concepts and tools, necessary for building a model to detects emotions, which was also successfully developed. The project covers a lot of information about machine learning and technologies behind it, the application can be a good point for further and more deep implementations in the field of machine learning and the areas related to it.

REFERENCE LIST

1. Practical Machine Learning and Image Processing: For Facial Recognition, Object Detection, and Pattern Recognition Using Python; Book by Himanshu Singh
2. Neural Network for Pattern Recognition; Book by Christopher Bishop.
3. Thoughtful Machine Learning with Python; Book by Matthew Kirk.
4. Learning OpenCV 3 Computer Vision with Python, 2nd Edition; Book by Joe Minichino and Joseph Howse
5. Hands-On Machine Learning with Scikit-Learn and TensorFlow- Concepts, Tools, and Techniques to Build Intelligent Systems; Book by Aurelien Geron
6. Machine learning “TensorFlow and keras”[Internet Resources]; Access mode: <https://www.tensorflow.org/tutorials/keras/>
7. Image classification using keras[Internet Resources];Access mode: https://keras.io/examples/vision/image_classification_from_scratch
8. Real time object detection using TensorFlow and keras[Internet Resources]; Access mode: <https://www.mygreatlearning.com/blog/object-detection-using-tensorflow/>
9. Learn facial expression from an image using dataset [Internet Resources]; Access mode:<https://www.kaggle.com/msambare/fer2013>