ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«Ғ.ДАУКЕЕВ АТЫНДАҒЫ АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
Коммерциялық емес акционерлік қоғамы
Телекоммуникациялық желілер және жүйелер кафедрасы

«ҚОРҒАУҒА ЖІБЕРІЛДІ»
Кафедра меңгерушісі PhD доктор Темырканова Э.К.
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «____» _____2020ж.
(қолы)

ДИПЛОМДЫҚ ЖОБА

Тақырыбы: Желілік трафикті талдау үшін терең нейрондық желі құру
Мамандығы 5В071900 Радиотехника, электроника және телекоммуникациялар
Орындаған Дулат Адлен Рахманұлы_____Тобы РЭТ(ИКТ)-16-1
(Т.А.Ж.)
Ғылыми жетекшісі PhD доктор Темырканова Э.К._____
(ғылыми дәрежесі, атағы, Т.А.Ж.)
Кеңесшілер:
экономикалық бөлім бойынша:
доцент Тузелбаев Бакберген Ибадиллаевич
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «__02__» ____06_____2020ж.
(қолы)
өміртіршілігі қауіпсіздігі бойынша:
профессор Жандаулетова Фарида Рустембековна
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «___15_» ____05_____2020ж.
(қолы)
есептеу техникасын қолдану бойынша:
PhD доктор Темырканова Эльвира Кадылбековна
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «_12___» _____06_____2020ж.
(қолы)
Нормобақылаушы: доцент Мухамеджанова Альмира Далелханкызы_____
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «__16__» ___06_____2020ж.
(қолы)
Пікір беруші: _____
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «____» _____2020ж.
(қолы)

Алматы 2020

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
Коммерциялық емес акционерлік қоғамы
Ғарыш инженериясы және телекоммуникациялар институты
Телекоммуникациялық желілер және жүйелер кафедрасы

Дипломдық жобаны орындауға берілген

ТАПСЫРМА

Студент Дулат Адлен Рахманұлы_____
(Т.А.Ж.)

Жобаның тақырыбы_____
Желілік трафикті талдау үшін терең нейрондық желі құру

2020 ж «30» Сәуір № 56 университет бұйрығымен бекітілді.

Аяқталған жобаны тапсыру мерзімі «25» 05 2020ж.

Жобаға алғашқы деректер (талап етілетін зерттеу (жоба) нәтижелерінің параметрлері және зерттеу нысанының алғашқы деректері):_____
Python бағдарламалау тілі, Keras кітапханасы, Tensorflow кітапханасы, pandas кітапханасы, Matplotlib кітапханасы, Google Colab сервисі, KDD Cup Data Data Set дерекқоры

Диплом жобасындағы әзірленуі тиіс мәселелер тізімі немесе диплом жобасының қысқаша мазмұны:_____
1 Кіріспе
2 Желілік шабуылдар
3 Нейрондық желі
4 Python бағдарламалау тілі және кітапханалары
5 Нейрондық желіні құру және оқыту
6 Өміртіршілігі қауіпсіздігі бөлімі
7 Экономикалық бөлім
8 Қорытынды
9 Әдебиеттер тізімі
10 А қосымшасы

Графикалық материалдардың (міндетті түрде дайындалатын сызбаларды көрсету) тізімі:_____

1 Желілік шабуылдардың түрлері мен сипаттамасы

2 Биологиялық және жасанды нейрондардың құрылымы

3 Жасанды нейрондардың белсендіру функциялары

4 Жасанды нейрондық желілердің түрлері

5 Жасанды нейрондық желілерді оқыту

6 Python бағдарламалау тілі

7 Нейрондық желілерді құруға арналған Python кітапханалары

8 Дерекқорлармен жұмыс жасауға арналған Python кітапханалары

9 Google Colaboratory сервисі

10 “KDD Cup 1999 Data Data Set” дерекқоры

11 Жұмыстың орындалуы

Негізгі ұсынылатын әдебиеттер:_____

1 Swaroop C. H. A Byte of Python [Электрондық ресурс]. — URL: https://python.swaroopch.com

2 Жандаулетова, Ф. Р. Охрана труда: учебник для вузов / Ф.Р. Жандаулетова, Т.Е. Хакимжанов, Т.С. Санатова; МОН РК, НАО АУЭС. - Алматы : АУЭС, 2019. - 399 с.

3 Базылов К.Б., Алибаева С.А., Нурмагамбетова С. С. Бітіруші жұмысының экономикалық бөлімі үшін әдістемелік нұсқаулар. 050719 – Радиотехника, электроника және телекоммуникация мамандығының барлық оқу түрінің студенттеріне арналған. Алматы. АУЭС.2009.

Жоба бойынша жобаның бөлімдеріне қатысты белгіленген кеңесшілер

Бөлімдері	Кеңесшілері	Мерзімі	Қолы
Негізгі бөлім	Темырканова Э.К.	12.06.2020	
Өміртіршілік қауіпсіздігі	Жандаулетова Ф.Р.	15.05.2020	
Экономика	Тузелбаев Б.И.	02.06.2020	
Норма бақылау	Мухамеджанова А.Д.	16.06.2020	

Диплом жобасын дайындау
КЕСТЕСІ

№	Бөлімдердің атауы, әзірленетін мәселелердің тізімі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
1.	Желілік шабуылдар туралы мәлімет жинау	17.02.2020 – 25.02.2020	Орындалды
2.	Жасанды нейрон мен желілер туралы мәліметтер жинау	25.02.2020 – 03.03.2020	Орындалды
3.	Python бағдарламалау тілі	03.03.2020 – 15.03.2020	Орындалды
4.	Нейрондық желіні құруға арналған Python кітапханалары	15.03.2020 – 20.03.2020	Орындалды
5.	Деректер жиынымен жұмыс жасауға арналған Python кітапханалары	20.03.2020 – 25.03.2020	Орындалды
6.	Нейрондық желіні оқытуға арналған дерекқормен танысу	25.03.2020 – 27.03.2020	Орындалды
7.	Дерекқормен жұмыс	27.03.2020 – 17.04.2020	Орындалды
8.	Нейрожеліні құру және оқыту	17.04.2020 – 07.05.2020	Орындалды
9.	Нейрожеліні тестілеу	07.05.2020 – 09.05.2020	Орындалды
10.	Өмір тіршілігінің қауіпсіздігі	09.05.2020 – 16.05.2020	Орындалды
11.	Техникалық-экономикалық бөлім	03.05.2020 – 03.06.2020	Орындалды

Тапсырманың берілген уақыты «17» 02 2020ж.

Кафедра меңгерушісі _____(__Темырканова Э.К.__)
(қолы) (Т.А.Ж.)

Жобаның
ғылыми жетекшісі _____(__Темырканова Э.К.__)
(қолы) (Т.А.Ж.)

Орындалатын тапсырманы
қабылдаған студент _____(____Дулат А. Р._____)
(қолы) (Т.А.Ж.)

Аңдатпа

Бұл дипломдық жұмыста желілік трафикті жіктейтін нейрондық желі әзірленеді. Нейрожеліні құру Python бағдарламалау тілі және оның кітапханалары арқылы жүзеге асырылады. Желіні оқыту үшін "KDD Cup 1999 Data Data Set" деректер қоры қолданылады.

Аннотация

В данной дипломной работе разрабатывается нейронная сеть, классифицирующая сетевой трафик. Построение нейросети ведется посредством языка программирования Python и его библиотеками. Для обучения сети использовалась база данных “KDD Cup 1999 Data Data Set”.

Abstract

In this thesis, a neural network is developed that classifies network traffic. The construction of the neural network is carried out through the programming language Python and its libraries. For training the network, the database “KDD Cup 1999 Data Data Set” was used.

Мазмұны

Кіріспе

Қазіргі қоғамды ақпараттық технологияларсыз елестету мүмкін емес. Ақпараттық технологиялар адам өмірінің барлық салаларына еніп отыр. Олардың ажырамас бөлігі жаһандық Internet желісі болып табылады. Әрине, басты міндеттердің бірі желі ішінде ақпарат айналымының қауіпсіздігін қамтамасыз ету болып табылады.

Әртүрлі желілік ортада байланысты ұйымдастыру үшін әр түрлі типтегі компьютерлердің үйлесімділігін қамтамасыз ететін TCP / IP хаттамаларының жиынтығы қолданылады. Бұл протокол жиынтығы үйлесімділігі мен ғаламтор ресурстарға қол жетімділігі арқасында танымалдылыққа ие болды және жұмыс істеудің стандартына айналды. Дегенмен, TCP / IP протокол стегінің жаппай таралуы оның әлсіз жақтарын ашты. Атап айтқанда, осыған байланысты таратылған жүйелер қашықтықтан шабуылға ұшырайды, өйткені олардың құрамдас бөліктері әдетте мәліметтерді берудің ашық арналарын пайдаланады, ал зиянкестер жіберілетін ақпаратты тек пассивті тыңдап қана қоймай, сонымен бірге берілетін трафикті де өзгерте алады.

Қашықтан шабуылды анықтаудың қиындығы және оны жүргізудің салыстырмалы қарапайымдылығы (қазіргі жүйелердің шамадан тыс функционалдығына байланысты) мұндай заңсыз әрекеттерді қауіптілік тұрғысынан бірінші орынға қояды және қауіпке дер кезінде жауап қайтаруға жол бермейді, нәтижесінде қылмыскер сәтті шабуыл жасау мүмкіндігіне ие болады.

Статистика бойынша есептеуіш жүйелердегі деректерді ашу көбінесе баукеспе ұрылардың қызметіне, бағдарламалардағы қателіктерге, вирустардың әрекеттеріне (17%) немесе техникалық істен шығуларға (16%) байланысты емес, ал пайдаланушылардың қателіктеріне және санкцияланбаған әрекеттеріне (67%) байланысты болады.

Қауіпсіздік қатерінің бірі - қашықтағы желілік шабуылдар. Желілік трафикті бақылау және талдау бұл мәселелерді тиімді анықтауға және шешуге көмектеседі.

Киберқауіпсіздік саласында Машиналық оқыту алгоритмдері маңызды рөл атқарды. Терең оқыту-бұл адам миының функцияларын имитациялайтын Машиналық оқыту бөлімі. Терең оқыту тұжырымдамасы жоғары деңгейдегі мәселелерді шешу үшін қарапайым құрылыс блоктарын құруды қамтитын күрделі иерархиялық түсініктерді құрудан тұрады. Соңғы күндері терең оқыту әдістерін қолдану киберқауіпсіздікті пайдаланудың түрлі жағдайлары үшін қолданылады.

Бұл дипломдық жұмыстың негізгі мақсаты терең нейрондық желілер негізіндегі, желілік трафикті классификациялау әдісін әзірлеу болып табылады. Жасанды нейронды желіні құру үшін Python бағдарламалау тілі және оның кітапханалары қолданылды. Нейрондық желіні оқыту үшін Ирвайндағы Калифорния университеті дайындаған "KDD Cup 1999 Data Set" мәліметтер базасы пайдаланылды.

1 Теориялық бөлім

1.1 Желілік шабуылдар

Желілік шабуыл - бұл қашықтағы/жергілікті есептеу жүйесінің үстінен бақылауды басып алу (құқықтарды арттыру), не оны тұрақсыздандыру, не қызмет көрсетуден бас тартқызу, сондай-ақ осы қашықтағы/жергілікті есептеу жүйесін пайдаланушылардың деректерін алу болып табылатын әрекет [6].

1.1.1 Шабуылдардың жіктелуі. Әсер сипаты бойынша:

- пассивті;
- активті.

Бөлінген есептеу жүйесіне(БЕЖ) пассивті әсер - бұл жүйенің жұмысына тікелей әсер етпейтін, бірақ сонымен бірге оның қауіпсіздік саясатын бұзуы мүмкін белгілі бір әсер. БЕЖ жұмысына тікелей ықпал етудің болмауы пассивті қашықтықтағы әсерді (ПҚӘ) анықтау қиынға соғатындығына әкеледі. БЕЖ-дегі типтік ПКӘ-нің мысалы - желідегі байланыс арнасын тыңдау.

БЕЖ-ге белсенді әсер ету - жүйенің жұмысына тікелей ықпал ететін (жұмыс қабілеттілігін бұзу, БЕЖ конфигурациясының өзгерту және т.б.), ондағы қабылданған қауіпсіздік саясатын бұзатын әсер ету. Алыстағы шабуылдардың барлық дерлік түрлері белсенді әсер ету болып табылады. Белсенді әсердің пассивтен айқын айырмашылығы — оны табудың принципті мүмкіндігі, өйткені оны жүзеге асыру нәтижесінде жүйеде кейбір өзгерістер болады. Пассивті әсерде ешқандай із жоқ.

Ықпал ету мақсаты бойынша:

- қызмет көрсету жүйесін бұзу (жүйеге кіру);
- ақпараттық ресурстардың (АР) тұтастығын бұзу;
- АР құпиялығын бұзу.

Бұл белгілер бойынша жүзеге асырылатын жіктеу, негізінен, қауіптің үш негізгі түрінің - қызмет көрсетуден бас тарту, тұтастықты жария ету және бұзудың тікелей проекциясы болып табылады.

Кез келген шабуыл кезінде көзделетін басты мақсат-ақпаратқа рұқсатсыз қол жеткізу. Ақпаратты алудың екі принципті нұсқасы бар: бұрмалау және ұстап қалу. Ақпаратты ұстап қалу нұсқасы оны өзгерту мүмкіндігінсіз оған қол жеткізуді білдіреді. Сондықтан, ақпаратты ұстап қалу оның құпиялылығын бұзуға әкеледі. Желідегі арнаны тыңдау - ақпаратты ұстап қалу мысалы. Ақпараттың құпиялылығын бұзу пассивті әсерлерге жатады.

Ақпараттарды алмастыру мүмкіндігі жүйелік нысандар арасындағы ақпарат ағынын толық бақылау немесе басқа біреудің атынан түрлі хабарлама жіберу мүмкіндігі деп түсіну керек. Сондықтан ақпаратты алмастыру оның тұтастығын бұзуға әкелетіні түсінікті. Мұндай ақпаратты зақымдаушы әсер - бұл белсенді әсердің типтік мысалы. Ақпараттың тұтастығын бұзуға арналған қашықтықтан жасалған шабуыл мысалы ретінде «Жалған БЕЖ нысаны» бола алады

Шабуыл жасалған объектімен кері байланыс болған жағдайда:

- кері байланыс арқылы;
- кері байланыссыз (біржақты шабуыл).

Шабуылшы шабуыл жасалған нысанға бірнеше жауап жібереді және ол нысаннан жауап алуын күтеді. Сондықтан шабуылшы мен шабуылдаушы объекті арасында кері байланыс пайда болады, өз кезекте шабуылшының шабуылдаушы объектідегі барлық өзгерістерге жауап қайтаруға мүмкіндік береді. Бұл қашықтықтағы шабуылдың мәні, шабуылдаушы объектінің кері байланысы болған кезде жүзеге асырылады. Мұндай шабуылдар БЕЖ-де жиі кездеседі.

Кері байланыссыз шабуылдар шабуылдаушы объектінің өзгеруіне жауап қайтарудың қажеті жоқтығымен сипатталады. Мұндай шабуылдар, әдетте, шабуыл жасалған объектіге бірлік сұраныстар жіберу арқылы жүзеге асырылады. Зиянкестерге бұл сұрауларға жауап қажет емес. Ұқсас ҚШ-ны біржақты ҚШ деп те атауға болады. Бір бағытты шабуылдардың мысалы - әдеттегі «DoS шабуылы».

Әсердің басталу шарты бойынша.

Қашықтан әсер ету, кез-келген басқа әсер сияқты, белгілі бір жағдайларда ғана басталуы мүмкін. БЕЖ-де мұндай шартты шабуылдардың үш түрі бар:

- шабуылға ұшыраған объектінің талабы бойынша шабуыл;
- шабуыл жасалған объектіде күтілетін оқиғаның басталуы бойынша шабуыл;
- шартсыз шабуыл.

Шабуыл жасаушының ықтимал нысаны белгілі бір түрдегі сұранысты жіберген жағдайда, оның тарапынан әсер басталады. Мұндай шабуылды шабуылданған объектінің сұратымнан болған шабуылы деп атауға болады. Бұндай ҚШ типі БЕЖ-ге тән Интернет желісіндегі осындай сұраныстардың мысалы ретінде DNS - және ARP-сұраныстар, ал Novell NetWare-де — SAP-сұраныс бола алады.

Шабуыл жасалатын объектіде күтілетін оқиғаның басталуы бойынша шабуыл. Шабуылшы қашықтағы шабуыл нысанасының ОЖ күйін үнемі бақылап отырады және осы жүйеде белгілі бір оқиға болған кезде әсер ете бастайды. Шабуылдаушы нысан шабуыл бастаудың бастамашысы болып табылады. Мұндай оқиғаның мысалы ретінде, Novell NetWare-де LOGOUT командасын бермей, пайдаланушының сервермен жұмыс сеансының үзуі, болуы мүмкін.

Шартсыз шабуыл дереу және операциялық жүйенің күйіне және шабуыл жасалатын объектінің жағдайына қарамай жүзеге асырылады. Сондықтан шабуылдаушы бұл жағдайда шабуылдың бастамашысы болып табылады.

Жүйенің қалыпты жұмыс қабілеттілігін бұзу кезінде басқа мақсаттар қойылады және шабуылдаушы деректерге заңсыз қол жеткізуі болжанбайды. Оның мақсаты - шабуылданған объектідегі ОЖ-ні істен шығару және жүйенің қалған объектілері үшін осы объектінің ресурстарына қол жеткізу

мүмкіндігінен айыру болып табылады. Мұндай шабуылдың мысалы - DoS шабуылы.

Шабуылданатын объектіге қатысты шабуыл субъектісінің орналасуы бойынша:

- сегментаралық;
- ішкі сегменттік.

Кейбір анықтамалар:

- шабуыл көзі (шабуыл субъектісі) — шабуыл жүргізуші және тікелей ықпал етуді жүзеге асыратын бағдарлама (мүмкін оператор);
- хост (host) — желі элементі болып табылатын компьютер;
- маршрутизатор (router) — желідегі пакеттердің бағытталуын қамтамасыз ететін құрылғы;
- ішкі желі (subnetwork) деп — глобалды желінің бір бөлігі болып табылатын хосттар тобын атайды, олар үшін маршрутизатормен бірдей кіші желі нөмірі ерекшеленетін. Сондай-ақ, ішкі желі деп маршрутизатор арқылы хосттардың логикалық бірлестігі деп айтуға болады. Бір ішкі желідегі хосттар маршрутизаторды пайдаланбай бір-бірімен тікелей байланыса алады;
- желі сегменті — физикалық деңгейдегі хосттардың бірігуі.

Алыстатылған шабуыл тұрғысынан әсер ету субъектісі мен объектісінің өзара орналасуы, яғни олардың әртүрлі немесе бірдей сегменттерде болуы аса маңызды болып табылады. Сегмент ішіндегі шабуыл кезінде, шабуыл субъектісі мен объектісі бір сегментте орналасады. Сегментаралық шабуыл болған жағдайда әсер ету субъектісі мен объектісі әртүрлі желілік сегменттерде болады. Бұл жіктеу ерекшелігі шабуылдың «алыстық дәрежесін» бағалауға мүмкіндік береді.

Сегментаралық шабуылға қарағанда, сегмент ішіндегі шабуылды жүзеге асыру әлдеқайда оңай. Сонымен қатар, қашықтықтағы сегментаралық шабуыл ішкі сегменттік шабуылға қарағанда әлдеқайда үлкен қауіп тудырады. Бұл сегментаралық шабуыл жағдайында оның нысаны мен шабуылдаушының өздері бір-бірінен мыңдаған шақырым қашықтықта болуы мүмкін, бұл шабуылға тойтарыс беру шараларына айтарлықтай кедергі келтіруі мүмкін.

Әсер ететін OSI анықтамалық моделінің деңгейі бойынша:

- физикалық;
- арналық;
- желілік;
- көліктік;
- сеанстық;
- көрсетімдік;
- қолданбалы.

Стандарттау жөніндегі халықаралық ұйым (ISO) ISO 7498 стандартын қабылдады, ол ашық жүйелердің (OSI) өзара әрекеттесуін сипаттайды, оларға БЕЖ-де тиесілі. Әрбір желілік алмасу протоколы, сондай-ақ, әрбір желілік бағдарлама сияқты, OSI эталондық 7-деңгейлік моделіне сәйкес келеді. Мұндай көп деңгейлі проекция желілік хаттамада немесе бағдарламада

пайдаланылатын OSI моделінің терминдерінде сипаттауға мүмкіндік береді. ҚШ - желілік бағдарлама және оны ISO/OSI эталондық моделіне проекция тұрғысынан қарастыру қисынды.

1.1.2 Желілік шабуылдардың сипаттамасы. Деректерді фрагменттеу.

IP хаттамасының деректер пакетін желі арқылы жіберу кезінде бұл пакетті бірнеше фрагментке бөлу жүзеге асырылуы мүмкін. Кейіннен, адресатқа жеткенде, пакет осы фрагменттерден қалпына келтіріледі. Зиянкес шабуылдаушы көптеген фрагменттерді жіберуді бастай алады, бұл қабылдау жағында бағдарламалық буферлердің толып кетуіне және кейбір жағдайларда жүйенің бұзылуына әкеледі.

Ping flooding шабуыл.

Бұл шабуылшыдан Интернетке жылдам арналарға қол жеткізуді талап етеді.

Ping бағдарламасы Echo REQUEST типті ICMP пакетін, оған уақыт пен оның идентификаторын қойып, жібереді. Алушы машинаның ядросы ICMP ECHO REPLY пакетімен сол сұранысқа жауап береді. Оны алғаннан кейін ping пакеттің өту жылдамдығын береді.

Стандартты жұмыс режимінде пакеттер бірнеше уақыт аралығынан кейін, желіні жүктемей жіберіледі. Бірақ ICMP echo request/reply-пакеттердің "агрессивті" режимінде ағыны пайдалы ақпаратты тарату қабілеттілігінен айырып, шағын желінің жүктелуін тудыруы мүмкін.

IP-де инкапсуляцияланған стандартты емес хаттамалар.

IP пакетінде инкапсуляцияланған пакеттің протоколын (TCP, UDP, ICMP) анықтайтын аймағы бар. Зиянкестер ақпараттық ағындарды бақылаудың стандартты құралдарымен бекітілмейтін деректерді беру үшін осы аймақтың стандартты емес мәнін пайдалана алады.

Smurf шабуылы.

Smurf шабуылы – шабуыл объектінің компьютері атынан, кең тарату желісіне, ICMP сұраныс жіберуінде негізделген. Нәтижесінде мұндай кең тарату пакеттерін қабылдаған компьютерлер зардап шеккен компьютерге жауап береді, бұл байланыс арнасының өткізу қабілетінің айтарлықтай төмендеуіне және бірқатар жағдайларда шабуылдаушы желінің толық оқшауланыуна әкеледі. Smurf шабуылы өте тиімді және кең таралған.

DNS spoofing шабуылы.

Бұл шабуылдың нәтижесі DNS серверінің кэшіне IP мекенжайы мен домендік атау арасындағы сәйкестікті енгізу болып табылады. Сәтті шабуыл нәтижесінде DNS серверінің барлық пайдаланушылары домен атаулары мен IP мекенжайлары туралы қате ақпарат алады. Осы шабуыл бірдей домендік аты бар DNS пакеттерінің көптігімен сипатталады. Бұл DNS алмасудың кейбір параметрлерін таңдау қажеттілігімен байланысты.

IP spoofing шабуылы.

Интернет желісіндегі шабуылдардың көп саны бастапқы IP-мекенжайдың ауыстыруымен байланысты. Мұндай шабуылдарға syslog spoofing да жатады, ол шабуылдаушы компьютерге ішкі желінің басқа

компьютерінен хабарлама жіберуден тұрады. Syslog протоколы жүйелік журналдарды жүргізу үшін пайдаланылғандықтан, шабуылдаушы компьютерге жалған хабарлама жіберу арқылы ақпаратты немесе рұқсат етілмеген кіру іздерін жоюға болады.

Пакеттерді таңу.

Шабуылшы желіге жалған қайтару мекенжайы бар пакеттерді жібереді. Осы шабуылмен шабуылдаушы басқа компьютерлер арасында орнатылған қосылымдарды өз компьютеріне ауыстыра алады. Бұл жағдайда шабуылдаушының қол жеткізу құқығы серверге қосылысынан зиянкестің компьютеріне қосылған пайдаланушының құқықтарына тең болады.

Sniffing - арнаны тыңдау.

Тек жергілікті желі сегментінде ғана мүмкін. Барлық желілік карталар жергілікті желінің жалпы арнасы бойынша берілетін пакеттерді ұстап қалу мүмкіндігін қолдайды. Бұл жағдайда жұмыс станциясы желінің сол сегментінің басқа компьютерлеріне арналған пакеттерді қабылдай алады. Осылайша, желі сегментіндегі барлық ақпарат алмасу қаскүнемге қолжетімді болады. Бұл шабуылды сәтті жүзеге асыру үшін шабуылдаушы компьютер шабуыл жасалған компьютермен бірдей желінің сегментінде орналасуы керек.

Маршрутизаторда пакеттерді ұстап қалу.

Маршрутизатордың желілік бағдарламалық жасақтамасы осы маршрутизатор арқылы берілетін барлық желілік пакеттерге қол жеткізуге болады, бұл пакеттерді ұстап тұруға мүмкіндік береді. Осы шабуылды іске асыру үшін зиянкестер ең болмағанда желінің бір маршрутизаторына артықшылықты қолжеткізімі болуы тиіс. Маршрутизатор арқылы әдетте көптеген пакеттер беріледі, оларды жаппай ұстап қалу мүмкін емес. Алайда, жекелеген пакеттер зиянкестермен кейіннен талдау үшін ұстап қалуы және сақталуы мүмкін. Пайдаланушылардың құпия сөздерін қамтитын FTP пакеттерін, сондай-ақ электрондық поштаны ұстап қалу неғұрлым тиімді.

ICMP протоколының көмегімен жалған маршрутты хостқа таңу.

Интернет желісінде ICMP (Internet Control Message Protocol) деп аталатын арнайы хаттама бар, оның функцияларының бірі маршрутизатордың өзгеруі туралы хосттарға хабарлау болып табылады. Бұл басқару хабарламасы redirect деп аталады. Желілік сегменттің кез-келген хостынан маршрутизатордың атынан шабуылға ұшыраған хостқа жалған бағыттау хабарламасын жіберу мүмкіндігі бар. Нәтижесінде хосттың бағыттау кестесі өзгертеді, содан кейін осы хосттың барлық желілік трафигі, мысалы, қайта бағыттау туралы жалған хабарлама жіберген хост арқылы өтеді. Осылайша, Интернеттің бір сегментіне жалған бағытты белсенді түрде енгізуге болады.

WinNuke.

TCP қосылымы арқылы жіберілетін тұрақты мәліметтермен қатар, стандарт жедел (Out of Band) деректерді жіберуді де қарастырады. TCP пакеттері деңгейінде бұл нольдік емес urgent pointer түрінде көрінеді. Windows орнатылған компьютерлердің көпшілігінде NetBIOS желілік протоколы бар, ол оның қажеттіліктері үшін үш IP портын қолданады: 137,

138, 139. Егер сіз Windows машинасына 139 порты арқылы қосылып, OutOfBand деректерін бірнеше байт жіберетін болсаңыз, онда NetBIOS бұл деректермен не істеу керектігін білмей машинаны тоқтады немесе қайта жүктейді. Windows 95 үшін бұл әдетте TCP / IP драйверінде қате туралы және ОЖ қайта жүктелгенге дейін желіде жұмыс істей алмайтындығы туралы көгілдір мәтіндік экранға ұқсайды. Сервис бумалары жоқ NT 4.0 қайта жүктеледі, ServicePack 2 жиынтығымен NT 4.0 көк экранға түседі. Желідегі ақпарат бойынша, Windows NT 3.51 және Windows 3.11 for Workgroups де мұндай шабуылға бейім.

139 портына деректерді жіберу NT 4.0 қайта қосылуына немесе Service Pack 2 орнатылғанға «өлімнің көгілдір экранына» әкеледі. Дәл осындай 135 және басқа порттарға деректерді жіберу RPCSS.EXE процесінің айтарлықтай жүктелуіне әкеледі. Windows NT WorkStation-да бұл айтарлықтай баяулауға әкеледі, Windows NT Server іс жүзінде қатып қалады.

Сенімді хостты ауыстыру.

Осы түрдегі қашықтықтан шабуылдарды сәтті жүзеге асыру шабуылдаушыға сенімді хост атынан сервермен сеанс өткізуге мүмкіндік береді. (Сенімді хост - бұл серверге заңды түрде қосылған станция). Шабуылдың бұл түрін жүзеге асыру әдетте шабуылдаушы станциядан оның бақылауындағы сенімді станция атынан алмасу пакеттерін жіберуден тұрады.

Шабуылдарды анықтаудың заманауи әдістерінің көпшілігінде ережелерге немесе статистикалық тәсілге негізделген басқарылатын кеңістікті талдаудың қандай да бір нысаны қолданылады. Бақыланатын кеңістік журналдар немесе желілік трафик болуы мүмкін. Талдау әкімші немесе шабуылдарды анықтау жүйесінің өзі құратын алдын ала анықталған ережелер жиынтығына сүйенеді.

Кез-келген шабуылды уақыт бойынша немесе бірнеше шабуылдаушылар арасында бөлуді сараптамалық жүйелер көмегімен анықтау қиын. Шабуылдар мен хакерлердің алуан түрлілігіне байланысты, сараптамалық жүйенің ережелер базасындағы арнайы тұрақты жаңартулар ешқашан шабуылдардың барлық спектрін дәл анықтауға кепілдік бере алмайды.

Нейрондық желілерді пайдалану - сараптамалық жүйелердегі осы проблемаларды шешудің бір әдісі. Пайдаланушыға қарастырылған сипаттамалардың дерекқорда жазылған ережелерге сәйкестігі туралы нақты жауап бере алатын сараптамалық жүйелерден айырмашылығы, нейрондық желі ақпаратты талдайды және мәліметтерді тануға үйретілген сипаттамаларға сәйкес келетіндігін бағалауға мүмкіндік береді. Нейрондық желі ұсынуының сәйкестік деңгейі 100% жетуі мүмкін, бірақ таңдау сенімділігі қолдағы тапсырма мысалдарын талдауда жүйенің сапасына байланысты болады.

Алдымен нейрожелі алдын ала таңдалған пәндік аймақтың мысалдарын дұрыс сәйкестендіруге үйретеді. Нейрожелінің реакциясы талданады және жүйе қанағаттанарлық нәтижелерге қол жеткізу үшін теңшейді. Оқытудың

бастапқы кезеңіне қосымша, нейрожелі пән саласымен байланысты деректерге талдау жүргізуіне қарай уақыт өте келе тәжірибе жинақтайды.

Құқық бұзушылықты анықтаудағы нейрондық желілердің маңызды артықшылығы, шабуылдардың сипаттамаларын «зерттеу» және бұрын желіде байқалмаған элементтерді анықтау қабілеті болып табылады [7].

1.2 Нейрондық желі

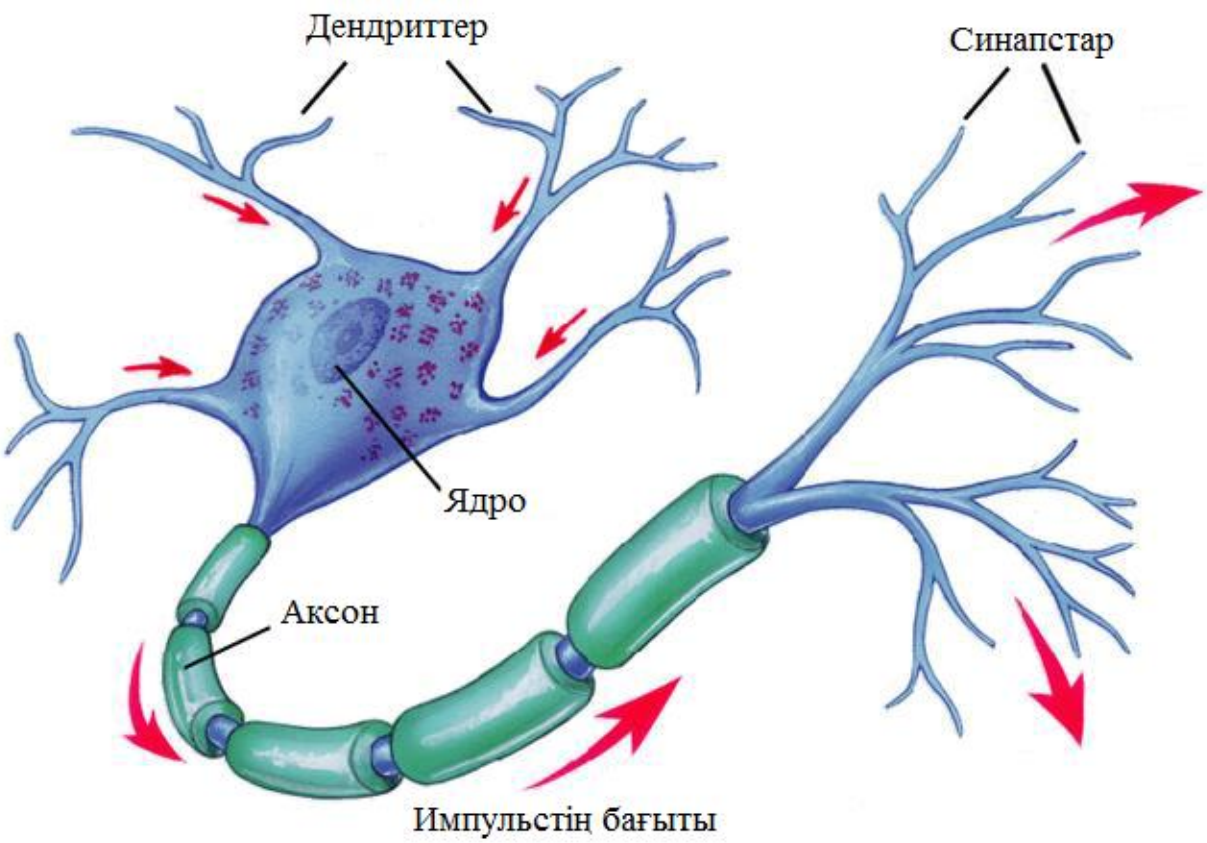
Жасанды нейрондық желі (ЖНЖ) (жасанды нейрондық желі (ANN)) - бір-бірімен өзара әрекеттесетін жасанды нейрондар жиынтығы болып табылатын биологиялық нейрондық жүйенің жеңілдетілген моделі.

Нейрондық желілердің жұмысының негізгі принциптерін 1943 жылы Уоррен Мак-Каллок пен Уолтер Питтс сипаттаған. 1957 жылы нейрофизиолог Франк Розенблатт алғашқы нейрондық желіні ойлап тапты, ал 2010 жылы оқытуға арналған деректердің үлкен көлемі машиналық оқыту үшін нейрондық желілерді пайдалануға мүмкіндік берді.

Қазіргі уақытта нейрондық желілер машиналық оқытудың көптеген салаларында қолданылады және күрделілігі әртүрлі мәселелерді шешеді.

Жасанды нейрондық желілер теориясы адам миының бейнесі мен ұқсастығында құрылған интеллектуалды құрылғыны жасау идеясына негізделген. Адам миының ақпаратты өңдеу әдісі қарапайым Нейман компьютерлерінде қолданылатын әдістерден түбегейлі ерекшеленеді. Биологиялық мидың басты артықшылығы - оның есептеулерінің параллелділігі [8].

1.2.1 Нейрондардың құрылымы. Адам миы нейрондардан және оларды байланыстыратын талшықтардан тұрады. Әр нейрон жасушалық денеден, дендриттерден және аксондардан тұрады және электрлік белсенділікке ие. Нейронның әрекет етудің жеңілдетілген принципі келесідей: дендриттер арқылы нейрон басқа нейрондардан жүйке импульстарын алады; нейронның денесінде сигнал өңделеді; өңделген сигнал аксон арқылы көптеген жүйке талшықтарына - синапстарға еніп, басқа нейрондарға жіберіледі. 1.1-суретте биологиялық нейронның құрылымы көрсетеді. Қарапайым нейронның шамамен 10^3:10^4 дендриттері болуы мүмкін, яғни. 10,000 басқа нейрондарға қосыла алады. Адамның миында шамамен 10^{11} нейрон бар [9].



1.1 сурет – Биологиялық нейронның құрылымы

Атап кетсек, биологиялық ми өз нейрондарын және олардың арасындағы байланыстарын белгілі бір мәселені шешу үшін ұйымдастыру қабілеттілігі арқасында, бейнелерді тану, сезім мүшелерінің сигналдарын өңдеу, моторлық функциялар және т. б. сияқты мәселелерді ең заманауи компьютерлерден тез орындайды. Аксондарда тарайтын жүйке импульстерінің жылдамдығы шамамен 100м/с құрағанымен, өз кезекте бұл электр импульстерінің таралу жылдамдығынан миллион есе аз, нейрондар арасындағы байланыстардың көптігіне байланысты, яғни, ақпаратты параллель өңдеудің арқасында, адам миы «қарапайым» компьютерлерге, салыстырмалы түрде, ұзақ уақытты қажет ететін міндеттерді бірден шешеді.

Жасанды нейрон (Маккалок-Питтстің математикалық нейроны, Формальды нейрон) - табиғи нейронның жеңілдетілген моделі болып табылатын жасанды нейрондық желінің торабы. Математикалық тұрғыдан алғанда, жасанды нейрон барлық кіріс сигналдарының сызықтық комбинациясы ретіндегі жалғыз аргументінің, қандайда бір, сызықты емес функцияны білдіреді. Бұл функция белсендіру функциясы немесе іске қосу функциясы, сонымен қатар, табыстама функциясы деп аталады. Алынған нәтиже бір шығысқа жіберіледі. Мұндай жасанды нейрондар желілерге біріктіріледі - бір нейрондардың шығыстарын басқаларының кірістерімен байланыстырады. Жасанды нейрондар мен желілер идеалды нейрокомпьютердің негізгі элементтері болып табылады.

Математикалық нейрон - өлшенген сумматор, оның жалғыз шығысы оның кірісі және өлшеу матрицасы арқылы анықталады:

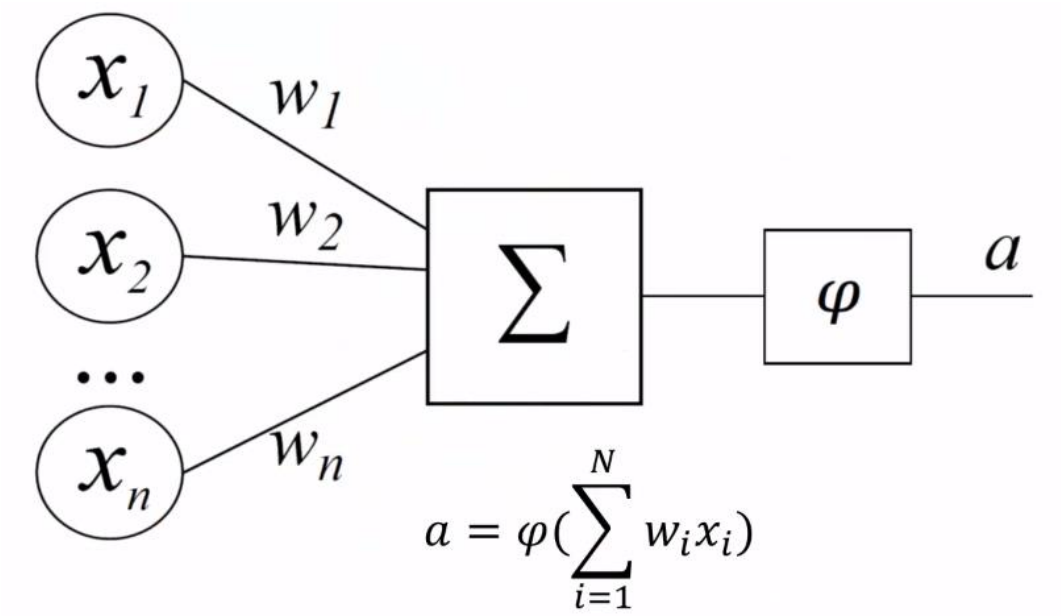
$$OUT = f\left(\sum_{i=1}^{n} w_i x_i + w_0 x_0\right) \tag{1.1}$$

Мұндағы xi және wi, сәйкесінше, нейронның кірістеріндегі сигналдар және кірістердің салмақтары. Нейронның кірісіндегі сигналдардың мүмкін мәндері әрдайым [0,1] аралықта болады, олар дискретті (нөл немесе бір) немесе аналогты болуы мүмкін. x0 қосымша кіріс және оған сәйкес келетін салмақ нейронның инициализациясы үшін қолданылады. Инициализация ретінде, нейронның белсендіру функциясының көлденең өсі бойымен ығысуы, яғни нейронның сезімталдығының шегін қалыптастыруы түсіндіріледі. Сонымен қатар, кейде жылжыту деп аталатын кездейсоқ шамалар нейронның шығысына арнайы қосылады. Жылжыту үнемі жүктелген қосымша синапстардың сигналы ретінде қарастыруға болады.

Негізінен, нейрондар желілік топологиядағы орналасуына байланысты жіктеледі:

- кіріс нейрондары - кіріс сигналын кодтайтын бастапқы векторды қабылдайды. Әдетте, бұл нейрондар есептеу операцияларын жасамайды, бірақ қабылданған кіріс сигналын шығысқа, мүмкін оны күшейтіп немесе әлсіретіп, жібереді;
- шығыс нейрондары - бұл желінің шығыстары. Шығыс нейрондарында қандайда бір есептеу операциялары іске асыра алады;
- аралық нейрондар - негізгі есептеу операцияларын орындайды [10].

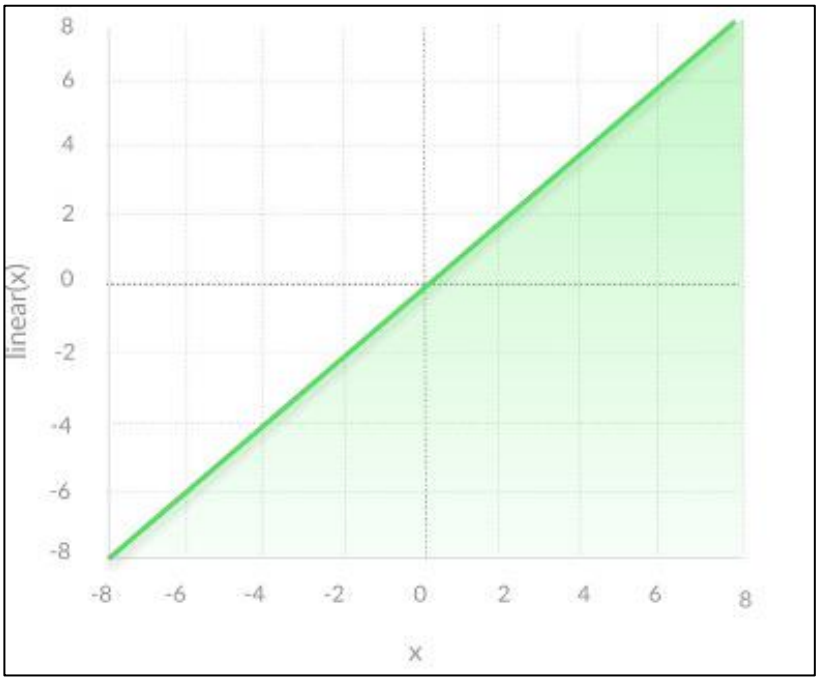
Математикалық нейронның графикалық түрі 1.2 суретте келтірілген.



1.2 сурет – Жасанды нейронның құрылымы

1.2 суретке сәйкес, нейронның n кірісі бар(x1, x2, …, xn), олардың әрқайсысының w салмағы, байланыс арқылы өтетін сигналға көбейтіледі. Осыдан кейін өлшенген сигналдар барлық сигналдарды өлшенген сомаға агрегациялайтын сумматорға жіберіледі. Өлшенген соманы шығысқа жібермес бұрын, нейрон оны өңдеп, тиісті шығыс сигналын қалыптастыруы керек. Осы мақсаттар үшін нейронның шығуы болатын өлшенген соманы кейбір санға түрлендіретін белсендіру функциясын қолданады [8].

1.2.2 Табыстама функцияларының негізгі түрлері. Сызықтық табыстама функциясы (1.3 сурет).



1.3 сурет - Сызықтық табыстама функциясы

Сызықтық функция (ағылш. linear function) түзу сызық болып табылады, яғни $f(x) = \sum_i t_i x_i$ (мұндағы t – функцияның параметрі), бұл белсендіру функциясының нәтижесі берілген аргументке пропорционалды. Бірақ сызықтық функцияда екі негізгі мәселе бар:

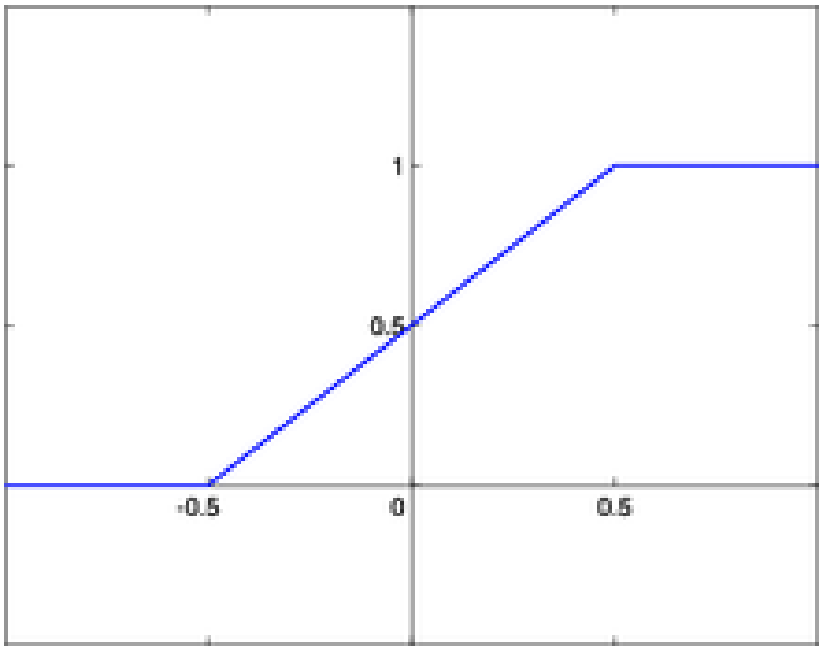
Қатені кері тарату әдісін пайдалану мүмкін еместігі. Осы оқыту әдісінің негізінде градиенттік түсу жатыр және оны табу үшін туынды қажет, ал осы белсендіру функциясы үшін - бұл константа және кіріс мәндеріне байланысты емес. Яғни салмақтарды жаңарту кезінде осы қадамда эмпирикалық тәуекел жақсарады ма немесе жоқ па деп айтуға болмайды.

Осы іске қосу функциясы бар бірнеше қабатты нейрондық желіні қарастырайық. Әрбір қабат үшін шығыс мәні сызықтық болғандықтан, олар сызықтық функция нәтижесі болып табылатын сызықтық комбинацияны құрайды. Яғни соңғы қабатта іске қосу функциясы тек бірінші қабаттағы кіріс мәндеріне байланысты. Бұл дегеніміз, қабаттардың кез келген саны тек бір

қабатпен алмастырылуы мүмкін, демек, көп қабатты желіні құрудың мағынасы жоқ.

Қабатты құрылымы бар жасанды нейрондық желілерде, осы типтегі табыстама функциялары бар нейрондар, әдетте, кіріс қабатын құрайды. Қарапайым сызықтық функциядан басқа, оның модификацияларын қолдануға болады. Мысалы, жартылай сызықты функция (егер оның аргументі нөлден аз болса, онда ол нөлге тең болады, ал басқа жағдайларда ол сызықты функция ретінде әрекет етеді) немесе қадамдық (қанықтылықты сызықтық функция (1.4 сурет)), оны келесі формуламен көрсетуге болады:

$$f(x) = \begin{cases} 0 \; if \; x \; \leq 0 \\ 1 \; if \; x \; \geq 1 \\ x \; else \end{cases} \tag{1.2}$$

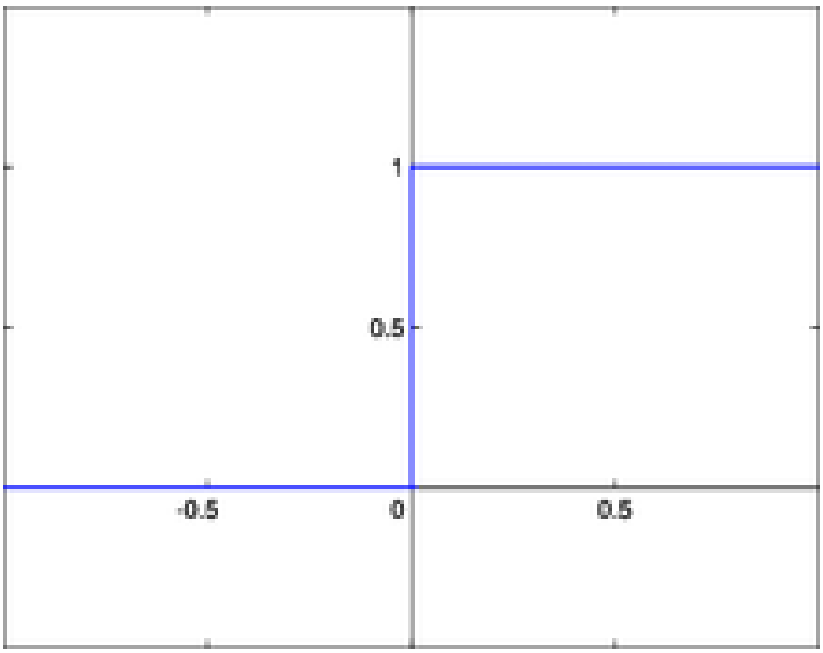


1.4 сурет - Қанықтылықты сызықтық функция

Бұл жағдайда функцияның екі ось бойымен ығысуы мүмкін (1.4 сурет).

Сызықтық функцияларға қатысты қадамдық және жартылай сызықтық белсендіру функциялардың кемшіліктері, олар барлық сандық осьте дифференцияланбайды, яғни оқыту кезінде кейбір алгоритмдерді пайдалану мүмкін емес.

Шекті табыстама функциясы (1.5 сурет).



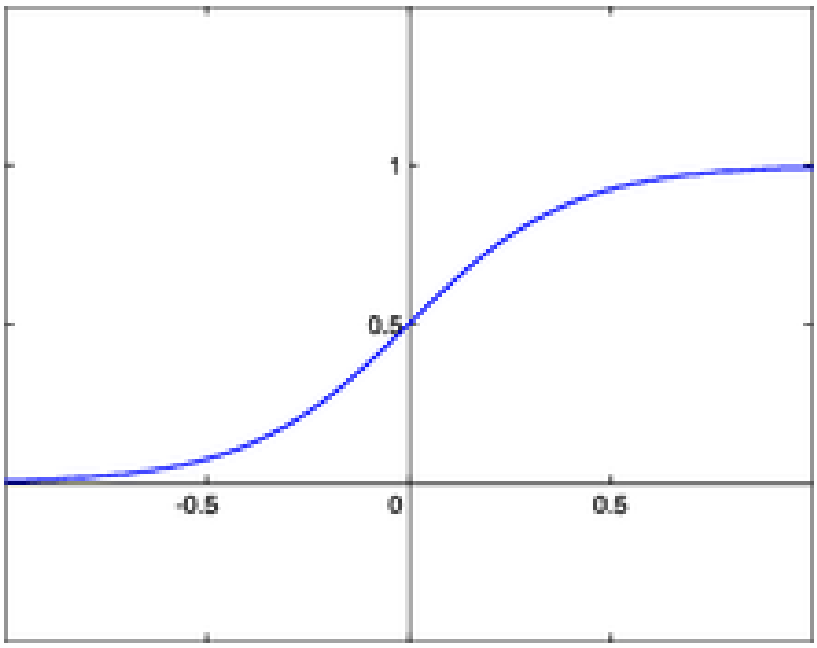
1.5 сурет - Шекті табыстама функциясы.

Басқа атауы - Хевисайд функциясы. Нейронның кірісіндегі өлшенген сигнал белгілі бір деңгейге жеткенше, шығысындағы сигнал нөлге тең болады. Нейронның кірісіндегі сигнал көрсетілген деңгейден асқан кезде, шығыс сигналы секірмелі 1-ге өзгереді. Қабатталған жасанды нейрондық желілердің алғашқы өкілі - перцептрон тек осы түрдегі нейрондардан тұрды. Бұл функцияның математикалық жазбасы келесідей:

$$f(x) = \begin{cases} 1 & if\ x \geq T \\ 0 & else \end{cases} \tag{1.3}$$

Бұл функция бинарлық классификация үшін өте жақсы жұмыс істейді. Бірақ жіктеуге нейрондардың көбірек саны қажет болғанда және мүмкін болатын класстардың саны екіден көп болған кезде жұмыс істемейді.

Сигмоидальды табыстама функциясы (1.6 сурет).



1.6 сурет - Сигмоидальды белсендіру функциясы

Қазіргі уақытта ең жиі қолданылатын табыстама функцияларының бірі. Сигмоидальды белсендіру функциясының енгізілуі шекті табыстама функциясының шектеулігімен шартталған, яғни шекті белсендіру функцияның шығысында тек қана бір немесе нөл деген мәндері классификациялаудан басқа мәселелерді шеше алмауы негіз болды. Сигмоидальды функцияларды қолдану бізге нейронның бинарлы шығыстардан аналогты шығыс сигналдарына ауысуға мүмкіндік берді. Мұндай типті беру функциялары, әдетте, нейрондық желінің ішкі қабаттарында орналасқан нейрондарға тән.

Математикалық логистикалық функцияны былайша көрсетуге болады:

$$\sigma(x) = \frac{1}{(1 + \exp(-tx))} \tag{1.4}$$

мұндағы t - функцияның тіктігін анықтайтын параметр. t шексіздікке ұмтылған кезде функция шекті табыстама функциясын көрсетеді. t = 0 болған жағдайда сигмоида 0,5 мәнді тұрақты функцияға түседі. Бұл функцияның мәндерінің диапазоны (0,1) аралықта орналасқан. Осы функцияның маңызды артықшылығы оның туындысының қарапайымдылығы:

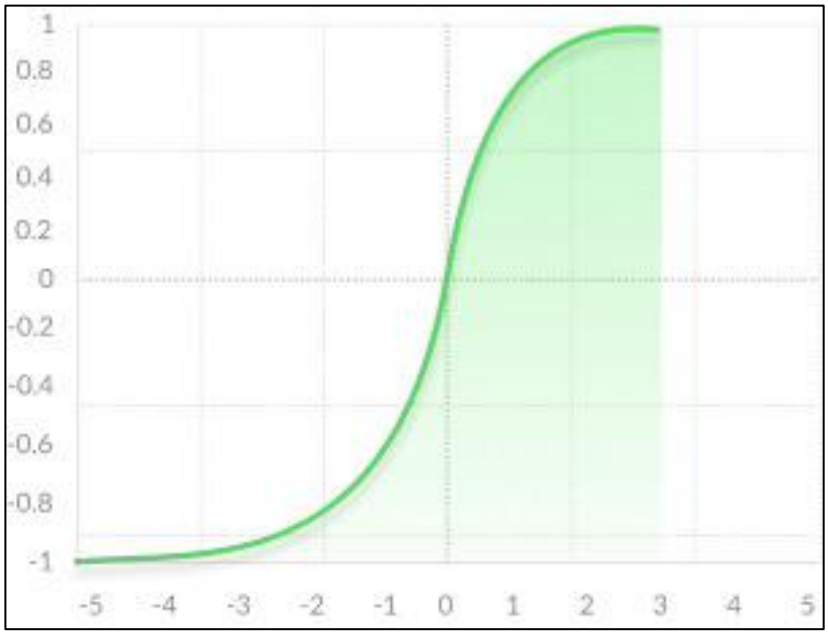
$$\frac{d\sigma(x)}{dx} = t\sigma(x)\big(1 - \sigma(x)\big) \tag{1.5}$$

Бұл функцияның туындысы оның мәні арқылы көрсетілуі мүмкіндігі, кері тарату алгоритмі бойынша желіні оқыту кезінде осы функцияны пайдалануды жеңілдетеді. Мұндай беріліс сипаттамасы бар нейрондардың

ерекшелігі олар күшті сигналдарды әлсіз сигналдарға қарағанда айтарлықтай аз күшейтеді, өйткені күшті сигналдардың аймақтары сипаттаманың қиғаш учаскелеріне сәйкес келеді. Бұл үлкен сигналдардан қанығудың алдын алуға мүмкіндік береді.

Сигмоидальды функциясының көптеген күшті жақтарына қарамастан, оның айтарлықтай кемшілігі бар. Мұндай функцияның туындысы салыстырмалы аз аралықтан басқа барлық нүктелерде өте аз. Бұл градиентті түсу арқылы салмақтарды жақсарту процесін қиындатады. Сонымен қатар, бұл мәселе модельде көп қабаттар болған жағдайда қиындайды. Осы мәселе жоғалып бара жатқан градиент мәселесі деп аталады.

Гиперболалық тангенс (1.7 сурет).



1.7 сурет - Гиперболалық тангенс

Гиперболалық тангенс функциясы келесі формада болады:

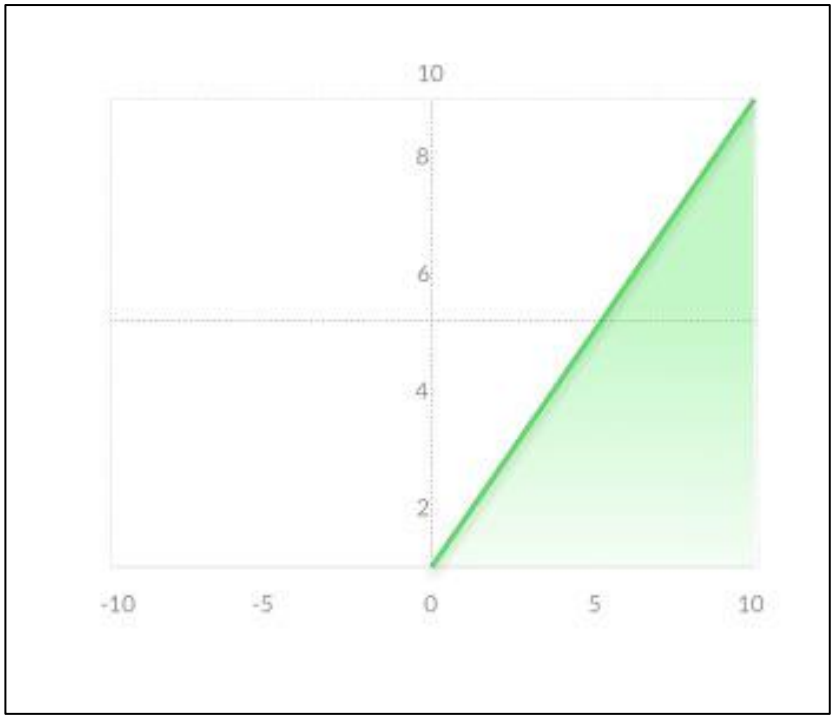
$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \qquad (1.6)$$

Бұл функция $tanh(z) = 2 \cdot \sigma(2z) - 1$ түзетілген сигмоидтық функциясы болып табылады, яғни ол сол артықшылықтар мен кемшіліктерді сақтайды, бірақ (-1; 1) мәндердің диапазоны үшін.

Егер қалыптандыру қажет емес жағдай болса, әдетте, гиперболалық тангенс функциясын сигмоидадан қарағанда қолданған ұтымды. Бұл белсендіру функциясын анықтау аймағы нөлге қатысты орталықтандырылғандықтан, градиентті есептеу кезінде белгілі бір бағытта жылжу үшін шектеуді алып тастайды. Сонымен қатар, гиперболалық тангенстің туындысы нөлге жақын мәндерде жоғары болады, соның

нәтижесінде, бұл градиенттік түсудің үлкен амплитудасын қамтамасыз етеді, бұл өз кезекте, жылдам жинақтылыққа әкеледі [10].

ReLU функциясы (1.8 сурет).



1.8 сурет – ReLU функциясы

Rectified Linear Unit - терең оқыту кезінде жиі қолданылатын белсендіру функциясы. Бұл функция теріс аргумент алса, 0 мәнін береді, оң аргумент болған жағдайда, функция санның өзін қайтарады. Яғни, оны былай жазуға болады:

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

Бір қарағанда, ол сызықты және сызықтық функциямен бірдей проблемалар бар деп көрінуі мүмкін, бірақ бұл олай емес және оны көп қабаттары бар нейрондық желілерде қолдануға болады. ReLU функциясы сигмоидальдық және гиперболалық тангенс функцияларына қарағанда бірнеше артықшылықтарға ие:

- туынды өте тез және қарапайым есептеледі. Теріс мәндер үшін - 0, оң мәндер үшін – 1;
- белсендірудің сиректілігі. Нейрондардың саны өте көп желілерде сигмоид тәрізді функцияны немесе гиперболалық тангенсті белсендіру функциясы ретінде пайдалану барлық дерлік нейрондардың іске қосылуына әкеледі, бұл модельді оқыту өнімділігіне әсер етуі мүмкін. Егер ReLU қолданса, онда қосылатын нейрондар саны функцияның сипаттамаларына байланысты аз болады, және желінің өзі оңай болады [11].

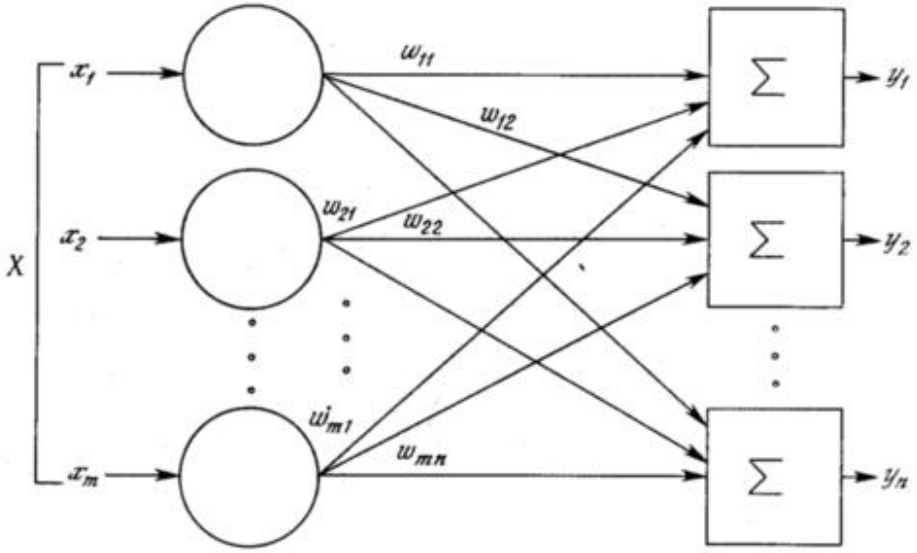
Гиперболалық тангенс пен сигмоидты есептеу ресурстарды көп қажет ететін операцияларды орындаумен қатар жүреді. Алайда, ReLU есептеу белсендіру матрицасының нөлдік деңгейіне қарапайым шекті түрлендіруді қолдану арқылы жүзеге асырылады. ReLU-дың тағыда бір артықшылығы, оның қанықтыруға ұшырамауы.

ReLU-ды қолдану сигмоидты және гиперболалық тангентке қарағанда стохастикалық градиенттің түсуінің жинақталу жылдамдығын едәуір арттырады. Бұл сызықтық сипатқа және осы функцияның қанықтырулықтың болмауына байланысты деп саналады.

1.2.3 Нейрондық желілердің түрлері. Әдетте, нейрондық желілердің көпшілігінде тек бір ғана тапсырманы орындайтын кіріс қабаты бар - ол тек қана, кіріс сигналдарын қалған нейрондарға таратады. Бұл қабаттың нейрондары ешқандай есептеулер жасамайды. Нейрондық желілер төменде көрсетілген негізгі түрлерге бөлінеді.

Бір қабатты нейрондық желілер.

Бір қабатты нейрон желісі (ағыл. Single-layer neural network) - бірден кіріс қабатының сигналдарын, түрлендіретін және бірден жауап беретін, шығыс қабатына табыстайтын желі (1.9 сурет).

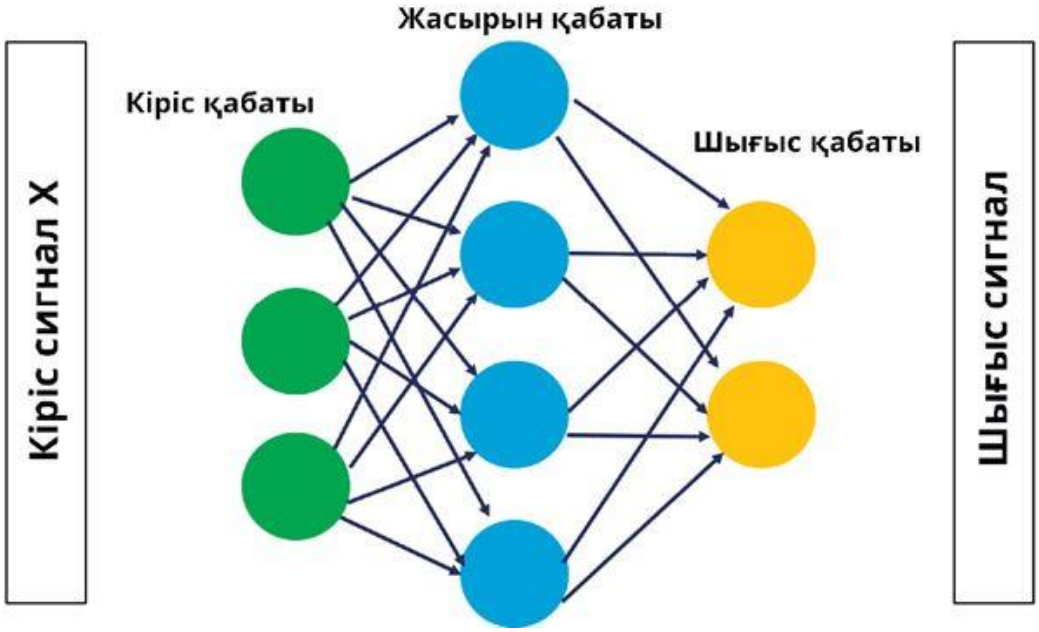


1.9 сурет - Бір қабатты нейрондық желінің схемасы

Бір қабатты нейрондық желі схемасынан көріп тұрғанымыздай (1.9 сурет), x1, x2, xn сигналдары кіріс қабатына түседі (ол нейрондық желі қабаты ретінде қарастырылмайды), содан кейін сигналдар қарапайым нейрондардың шығыс қабатына таратылады. Кіріс қабатының нейронынан шығыс қабаттың нейронына дейінгі әр қабырғада тиісті байланыс салмағы жазылған.

Көп қабатты нейрондық желілер.

Көп қабатты нейрондық желі, терең нейрондық желі (ағылш. Multilayer neural network) - бұл кіріс, шығыс және олардың арасында бір (бірнеше) нейрондардың жасырын қабаттарынан тұратын нейрондық желі (1.10 сурет).



1.10 сурет – Көп қабатты нейрондық желінің схемасы

Кіріс және шығыс қабаттарынан басқа бұл нейрондық желілерде аралық, жасырын қабаттар бар. Мұндай желілер бір қабатты нейрондық желілерге қарағанда әлдеқайда үлкен мүмкіндіктерге ие, алайда жасырын қабаттың нейрондарын оқыту әдістері салыстырмалы түрде жақында құрастырылған.

Жасанды нейрондық желілер циклдері жоқ, сигналды тікелей тарату желілеріне(feedforward networks) және циклдер рұқсат етілген рекурренттік желілерге(recurrent networks) бөлінеді.

Тікелей тарату желілері.

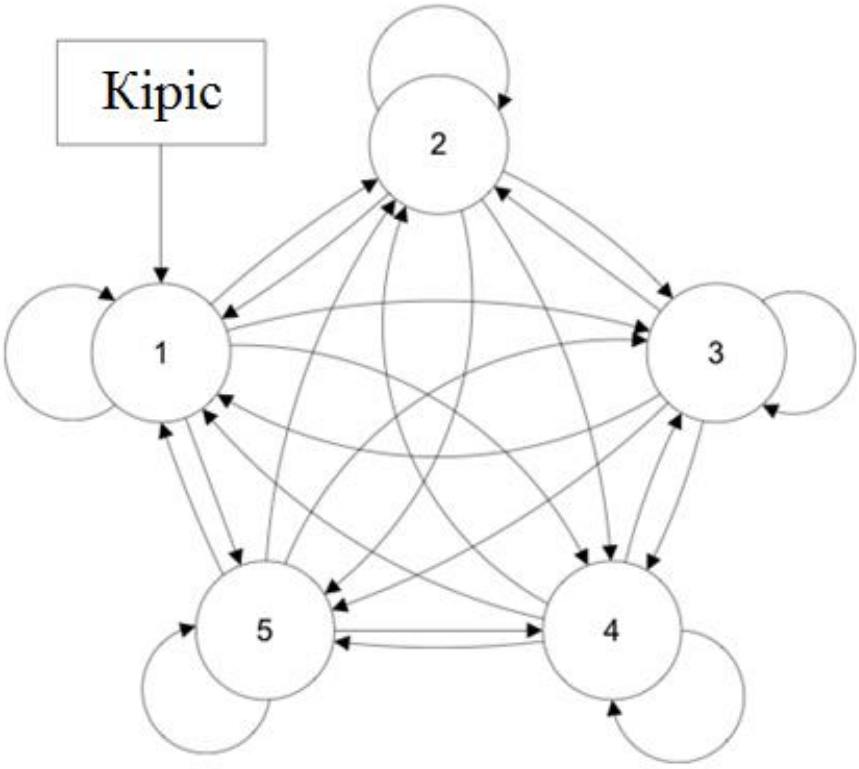
Тікелей тарату желілері (ағылш. Feedforward neural network) — сигналдардың кіріс қабатынан шығыс қабатына қатаң таратылатын жасанды нейрондық желілер. Кері бағытта сигнал таратылмайды. Мұндай желілер кең қолданыста және белгілі бір мәселелерді шешеді: болжау, кластерлеу және бейнелерді тану.

Алайда, нейрондық желілерде сигнал кері бағытта да жүре алады.

Рекуррентті нейрондық желілер.

Рекуррентті нейрондық желілер (РНС, ағылш. Recurrent neural network; RNN) - элементтердің арасындағы байланыс бағытталған тізбекті құрайтын нейрондық желілердің түрі. Осының арқасында оқиғалар серияларын уақыт немесе кеңістіктік тізбектерді өңдеу мүмкіндігі пайда болады. Тікелей тарату желілеріне қарағанда рекурренттік желілер өз ішкі жадын әртүрлі ұзындықтағы тізбектерді өңдеу үшін пайдалана алады. Сондықтан RNN желілері, белгілі бір тұтас нәрсенің бөлшектерге бөлінген кездегі есептерде қолданылады, мысалы: қолжазба мәтінін тану немесе сөйлеуді тану. Осы желілер үшін көптеген әртүрлі қарапайым-күрделі архитектуралық шешімдер ұсынылған. Соңғы уақытта ұзақ мерзімді және қысқа мерзімді жадысы бар

желі (LSTM) және басқарылатын рекурренттік блок (GRU) кеңінен таралған (1.11 сурет).



1.11 сурет - Кері байланысы бар желінің схемасы

Тікелей тарату желілерінде желінің шығысы кіріс сигналымен және жасанды нейронның салмағымен анықталады. Кері байланыс желілерінде нейронның шығысы кіріске оралуы мүмкін. Бұл нейронның шығысы оның салмағы мен кіріс сигналымен ғана емес, сонымен қатар алдыңғы шығыс сигналдарымен де анықталады (өйткені олар қайтадан кіріске оралды) [8].

1.2.4 Нейрондық желілерді оқыту. Нейрондық желіні оқыту - бұл желі берілген дәлдікпен қажетті функцияны жақындататындай нейрондар арасындағы қосылыстардың салмақтарын анықтау процесі. Нейрондық желілерді оқытудың үш тәсіл бар: мұғаліммен оқыту(supervised learning), мұғалімсіз оқыту(unsupervised learning) және нығайтумен оқыту(reinforcement learning). Мұғаліммен бірге оқыған кезде желіге кіру сигналдары (объектілер) беріледі және олар үшін алдын ала дұрыс жауап белгілі (оқыту жиыны). Желінің салмақтары, шығыс сигналдарының мәні қаншалықты дұрыс берілгеніне байланысты, белгілі бір ережелер бойынша өзгереді. Мұғалімсіз оқу кезінде, алдын-ала дұрыс шығыс сигналы анықталмаған объектілер желінің кірісіне беріледі. Бекітумен оқыту кезінде, желімен өзара әрекеттесетін сыртқы ортаның болуын талап етеді. Оқыту осы ортадан алынған сигналдардың негізінде жүргізіледі.

Маккаллок-Питтстің нейрондық желілері оқытылмаған болатын және нейрондардың барлық кірістері үшін салмақтар алдын ала берілуі тиіс болған.

Нейрондық желілерді оқыту идеясын алғаш рет 1949 жылы Дональд Хебб ұсынды. Хэббке сәйкес, бірге белсендірілген нейрондардың байланыстары күшейтіліп, бір-бірінен бөлек жұмыс жасайтын нейрондардың байланыстары әлсіреуі керек. Хэбб нейрондардың кіріс сигналдарының салмағын желінің дұрыс жауап бергеніне немесе бермеуіне (мұғаліммен жаттығу) сәйкес өзгерту ережелерін ұсынды. А.В. Новиков объектілердің іріктемесі сызықты бөлінетін болған жағдайда, Хэбб ережелеріне негізделген нейронды оқыту әдісінің дұрыстығын дәлелдеді. Кейіннен мұғаліммен де, мұғалімсіз де оқуға арналған бірнеше ұқсас ережелер ұсынылды.

1970 жылы А. Г. Ивахненко нейрондар арасындағы байланыс салмағын есептеп қана қоймай, сонымен қатар қолданбалы есептің қажеттіліктеріне байланысты желідегі қабаттар мен нейрондардың санын анықтауға мүмкіндік беретін аргументтерді есептеудің топтық әдісін (group method of data handling) әзірледі. Мұғаліммен оқыту тәсілін қолдана отырып, желі деңгейлері кезекпен құрылады және оқыту жиынының негізінде регрессиялық талдаумен оқытылады. Содан кейін желіні жеңілдету кезеңі, белгілі дұрыс жауаптары бар, жаттығуда қолданылмаған объектілердің жеке жиынтығын (валидация жиыны, validation set) пайдалана отырып жүзеге асырылады. Желідегі қажет емес нейрондарды жою үшін жүйелеу(регуляризация) қолданылады.

Қазіргі уақытта, нейрондық желілерді, соның ішінде терең желілерді оқыту үшін, градиентті түсіру әдісіне негізделген қателіктерді кері қайтару алгоритмі қолданылады. Алгоритм 1970 жылы магистрлік диссертацияда нейрондық желілермен байланыссыз ұсынылды. Бұл алгоритмді нейрондық желілерді оқытуда алғашқы қолдану 1981 жылы жарияланған жұмыста сипатталған. Осыдан кейін тағы бірнеше жұмыстар осы тақырыпта жарық көрді. Қателерді кері қайтарудың алгоритмі мұғаліммен бірге оқытуды қолданады, яғни оған алдын ала белгілі дұрыс жауаптары бар оқыту жиынтығы қажет. Желі шығысының мәндері дұрыс жауаптардан қаншалықты ерекшеленетінін анықтайтын қате өлшемі енгізілді. Содан кейін қателік өлшемі градиенттік түсіру әдісін қолдана отырып, салмақтарды өзгерту арқылы азайтылады. Әрбір салмақтың шығыс мәніне қаншалықты әсер ететінін бағалау үшін салмағы бойынша қателердің дербес туындысы есептеледі. Содан кейін салмағы градиентті ескере отырып, кіші мәндерге өзгертіледі. Бұл шығудағы қате қолайлы мәндерге дейін азайғанша қайталанады. Желідегі нейрондардың салмақтарының бастапқы мәндері кездейсоқ түрде орнатылады.

Бірнеше жасырын қабаттары бар терең нейрондық желіде қате есептеліп, бір қабаттан екінші қабатқа жіберіледі. Бірінші кезеңде нейрондық желінің шығысындағы мәні дұрыс жауаптармен салыстырылып, қателіктің мәні есептеледі. Содан кейін, шығыс қабатының кірісіндегі қателік есептеледі және бұл мән жасырын қабаттың шығысындағы қателікті есептеу үшін қолданылады. Осылайша, есептеу кіріс қабатындағы қате белгілі болғанға дейін жалғасады. Сондықтан бұл алгоритм қателіктердің кері таралуы деп аталады.

Қатенің кері таралуы алгоритмін қолдану арқасында, нейрондық желілерді оқытуды жүзеге асырудың бірнеше нұсқасы болуы мүмкін. Толық оқыту кезінде, градиент оқу жиынының барлық объектілері үшін есептеледі. Алайда, оқыту жиынтығы үлкен болғанда бұл тәсіл тиімді емес, себебі оның барлық элементтерін өңдеуге көп уақыт кетеді. Альтернативті нұсқа — стохастикалық градиентті түсіру әдісін пайдалану, онда оқыту жиынының бір элементін (онлайн-оқыту) немесе бірнеше элементтерін (пакеттердегі немесе шағын іріктемелердегі оқыту) өңдеу кезінде салмақтар өзгертіледі. Іс жүзінде нейрондық желілерді оқыту үшін стохастикалық градиентті түсіру әдісі немесе оның модификациясы жиірек қолданылады.

XX ғасырдың 80-жылдарының соңында тиімді терең оқыту үшін, жалғыз ғана қателіктерді кері таратудың алгоритмі жеткіліксіз екені белгілі болды. Кейбір сәтті мысалдарға қарамастан, бірнеше жасырын қабаттарды қолданудың практикалық жүзіндегі артықшылықтары аз болды. Мұның себебі, 1991 жылы - жойылып бара жатқан градиент мәселесі жұмысында тұжырымдалды. Белсендірудің дәстүрлі функцияларын қолданған кезде кері таралатын қателіктердің сигналдары тез азаяды (немесе керісінше, шамадан тыс үлкен болады). Практикалық мәселелерде олар желідегі қабаттар санына байланысты экспоненциалды түрде азаяды. Бұл мәселе ұзақ кідіріс мәселесі(long time lag problem) ретінде белгілі.

Жойылып бара жатқан градиент мәселесін шешудің бір тәсілі - оқыту үшін градиентті қолданудан толық бас тарту. Кейбір тапсырмалар үшін салмақты кездейсоқ белгілеу арқылы жақсы нәтижеге қол жеткізуге болады. Эволино оқыту әдісі шығыс қабаттың оңтайлы салмағын сызықтық әдістерін және жасырын қабаттардың салмағын анықтау үшін эволюцияны қолданады. Сонымен қатар әмбебап іздеу әдістерін қолдануға болады.

Нейрондық желілердің дамуына $ReLU(x) = max(0, x)$ деп анықталатын жартылай сызықты белсендіру функциясын қолдану ұсынысы елеулі үлесті қосты. Мұндай активтендіру функциясы жоғалып кететін градиент проблемасынан құтылуға мүмкіндік береді, өйткені сигналдың оң мәнінде, сигмоидальды активтендіру функцияларынан қарағанда, оның өзгерісі болмайды. Сонымен қатар, жартылай сызықты іске қосу функциясы нейрондық желінің оқыту уақытын қысқартады. Шығыс сигналы теріс болған нейрондар есептеулерге қатыспайды, ал қалғаны үшін тек сызықты есептеулер қажет.

2010 жылы Ксавье Глоро(Xavier Glorot) және Иошуа Бенджио(Yoshua Bengio) өз жұмысында, салмақтарды бастапқы инициализациялау әдістерінің және желідегі сигналды тікелей және кері бағытта таратуға белсендіру функцияларының әсерін зерттеу жүргізді. Зерттеуге сәйкес, салмақтарды бастапқы инициациялаумен бірге логистикалық сигмоидальдық функцияны пайдалану терең нейрондық желілерді құру үшін нашар жарамды, себебі тез қанығуға әкеледі. Белсендірудің гиперболалық тангенс функциясы симметриялық болғандықтан(функцияның орташа мәні 0, мәні — (- 1, 1)) бұндай кемшіліктері жоқ. Глоро мен Бенджио нейрондық желі салмақтарын

инициализациялаудың жаңа әдісін ұсынды. W желісі салмақтарының бастапқы мәндері мынадай формула бойынша анықталады:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \qquad \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right] \tag{1.8}$$

мұндағы U — кесінді бойынша біркелкі таралуы, n_j — желінің ағымдағы қабатындағы нейрондар саны, n_{j+1} — желінің келесі қабатындағы нейрондар саны. Нормаланған инициализацияны пайдалану нейрондардың қанығуын төмендетуге әкеледі және қате туралы сигнал айтарлықтай жақсы таралады.

2015 жылы салмақтардың нормаланған инициализациясы әдісі белсендірудің жартылай сызықтық функциясы үшін бейімделген. W бастапқы салмағын келесідей анықтау ұсынылады:

$$W \sim U\left[-\frac{\sqrt{2}}{n_j}, \qquad \frac{\sqrt{2}}{n_j}\right] \tag{1.9}$$

мұндағы U — кесінді бойынша біркелкі таралуы, nj — желінің ағымдағы қабатындағы нейрондар саны. Салмақтардың нормаланған инициализациясы әдісі терең нейрондық желілерді, сапалы оқытуға, алдын ала мұғалімсіз оқытудың қажеттілігінсіз қол жеткізуге мүмкіндік берді.

Сергей Йоффе (Sergey Ioffe) және Кристиан Жегеды (Christian Szegedy) 2015 жылы нейрондық желілерде терең нейрондық желіні оқыту сапасын арттыруға мүмкіндік беретін пакеттік нормалаудың(batch normalization) арнайы қабаттарын пайдалануды ұсынды. Жұмыста қателікті кері тарату алгоритмі, егер кіріс деректері нормаланған болса (нөлдік мат. күтімі және бірлік дисперсиясы бар) тезірек жинақталатыны түсіндіріледі. Бірақ нейрондық желі бойынша сигналдарды тарату кезінде оның математикалық күтімі мен дисперсиясы өзгеріп отырады, бұл оқыту процесіне теріс әсер етеді. Сергей Йоффе мен Кристиан Жегеда тек нейрондық желінің кірісінде ғана емес, желінің әрбір қабатының алдында да нормалауды орындауды ұсынды. Нормалау стохастикалық градиентті түсіру (немесе оның модификациясы) әдісінің әрбір деректер пакеті (mini-batch) үшін жеке орындалады. B пакетінде xi кіріс деректерінің m элементтерін қамтиды: B = {x1, ..., xm}. Нормаланған $\hat{x}_i$ мәндері мынадай формула бойынша анықталады:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{1.10}$$

мұндағы μ_B - пакеттегі мәліметтердің орташа мәні, σ_B^2 - дисперсия, є - әдіс тұрақтылығы үшін енгізілетін тұрақты мән. Содан кейін деректердің айқындығын сақтау үшін жылжыту және масштабтау жүргізіледі:

$$y_i = \gamma \hat{x_i} + \beta \quad (1.11)$$

мұндағы уі – нәтижелік мән, γ және β - оқу процесінде анықталатын параметрлер.

Пакеттік нормалау пакеттік нормалаудың қабаттары түрінде жүзеге асырылады және нейрондық желіде бірнеше қажетті жерлерге қоюға болады. Пакеттік нормалаудың қосымша артықшылығы - жаттығу уақытының қысқаруы және шамадан тыс оқытылудың азаюы.

Салмақтарды нормаланған инициализациясы және топтаманы нормалау әдістері тәжірибеде жойылып бара жатқан градиент проблемасын шешуге және бірнеше ондаған қабаттардан тұратын терең нейрондық желілерді оқытуға көмектеседі. Бұл кейбір авторларға жоғалып бара жатқан градиент мәселесі қазіргі уақытта шешілгенін айтуға мүмкіндік берді [12].

2 Нейрондық желіні құруға және оқытуға арналған инструменттер

Қазіргі уақытта терең нейрондық желілерді оқытуға арналған көптеген бағдарламалық жүйелер жасалды. Олардың ішінде ең танымалдары - Caffe, Theano, TensorFlow, Torch және CNTK. Олардың негізгі сипаттамалары 2.1 кестеде келтірілген.

2.1 кесте – Терең нейрондық желілерді оқытуға арналған бағдарламалық жүйелер

Сипаттамасы	Caffe	Theano	TensorFlow	Torch	CNTK
Негізгі тіл	C++	Python	C++	Lua	C++
API	C++, Python	Python	C++, Python	Lua, Python	C++, C#, Python
Көпядролық CPU	+	+	+	+	+
GPU	+	+	+	+	+
Xeon Phi	+	+	-	-	-
Үлестірмелі оқыту	+	-	+	+	+
Жасақтаушы	Беркли компьютерлік көру және оқыту орталығы	Монреаль университеті	Google	Ронан Коллаберт	Microsoft
Ашық код	+	+	+	+	+
Үйренген желілер	+	-	+	+	+

Caffe кітапханасы - алғашқы танымал терең оқыту жүйелерінің бірі. Беркли компьютерлік көру және оқу орталығында жасалынған, бастапқы коды 2014 жылы ашылған. Caffe құрамында алдын-ала оқытылған ең көп дайын модельдер бар. Theano Монреаль университетінде жасалған Канадалық жүйе.

Theano Python-да жасалынған, бірақ Python-да жасалған бағдарлама автоматты түрде C ++ бағдарламасына өзгертіліп, компиляцияланып, орындалатындықтан жоғары өнімділікті қамтамасыз етеді.

TensorFlow 2015 жылы Google компаниясымен құрылған және тензорлармен тиімді жұмыс істеу және бағанда деректерді ағынды өңдеу жүйесін қамтиды.

Torch кітапханасы Lua тілінде жасалған және MATLAB сияқты Машиналық оқыту бағдарламаларын жасау үшін ыңғайлы жоғары деңгейлі интерфейсті ұсынады. Theano-дағыдай, C тілін интеграциялау арқылы жоғары өнімділік қамтамасыз етіледі. Torch авторлары C және Lua интеграциясының қарапайымдылығына байланысты Python орнына Lua-ны пайдалануға таңдады.

Microsoft компаниясы CNTK (Cognitive Toolkit) жүйесін жасап, 2016 жылы бастапқы кодтарын ашты.

Барлық осы нейрондық желілерді тереңдетіп оқыту жүйелері оқыту процесін жеделдету үшін көп ядролы процессорларды және GPU есептеу үдеткіштерін (соның ішінде оңтайландырылған cuDNN кітапханасын) қолдана алады. Сонымен қатар, маңызды артықшылығы — бағдарламаны қайта өзгертудің қажеті жоқ, процессор мен GPU-ға параллельдеу автоматты түрде жүзеге асырылады. Caffe және Theano жүйелері қосымша Intel Xeon Phi үдеткіштерін қолдайды, бұл терең нейрондық желілер үшін оқыту уақытын едәуір қысқартады. Theano-дан басқа барлық жүйелерде, нейрондық желілердің үлестірілген оқытуын есептеу кластерлерінде қолдануға болады.

Жоғарыда сипатталған жүйелермен қатар, терең нейрондық желілерді оқыту үшін ыңғайлы және қарапайым интерфейсті ұсынатын Keras кітапханасын да атап өтуге болады. Keras — бұл жеке жүйе емес, ол Theano, TensorFlow немесе CNTK-дің үстінен ғана жұмыс істейді. 2016 жылы Keras TensorFlow құрамына кірді.

Жақында құрылған, бірақ танымалдылыққа ие жаңа терең оқыту кітапханалары да назар аударарлық. PaddlePaddle (Baidu жасаған) және MXNet жүйелері үлестірілген кластерлерде терең нейрондық желілерді оқыту үшін арнайы әзірленген. Neon кітапханасын Nervana компаниясымен әзірленген. Intel Nervana-ны сатып алғаннан кейін Neon GPU және Intel Xeon Phi үдеткіштерін, сондай-ақ алдын-ала дайындалған көптеген нейрондық желілерді қолдайтын ең жылдам дамып келе жатқан кітапханалардың біріне айналды. MXNet және Neon өнімділік сынақтарында жақсы нәтижелер көрсетеді [12].

2.1 Python бағдарламалау тілі

Python - әзірлеушілердің өнімділігі мен кодтардың оқылуын жақсартуға бағытталған жоғары деңгейлі жалпы мақсаттағы бағдарламалау тілі. Python ядросының синтаксисі минималды, сонымен қатар, стандартты кітапханасы көптеген пайдалы функцияларды қамтиды.

Python құрылымдық, объектіге-бағытталған, функционалды, императивті және аспектіге-бағытталған бағдарламалауды қолдайды. Негізгі архитектуралық ерекшеліктер - динамикалық типтендіру, жадыны автоматты басқару, толық интроспекция, біріңғай жағдайлардың өңдеу механизмі, көп ағынды есептеулерді қолдау және жоғары деңгейлі мәліметтер құрылымы. Бағдарламаларды модульдерге бөлудің мүмкіндігі бар және оларды өз кезегінде пакеттерге біріктіруге болады.

Платформалардың көпшілігін қолдайтын CPython интерпретаторы Python-ның эталондық іске асырылуы болып табылады. Ол Python Software Foundation лицензиясы бойынша таратылады және оны кез-келген қолданбада, соның ішінде жеке меншікте шектеусіз пайдалануға мүмкіндік береді. JVM үшін әзірленген интерпретаторда компиляцины, CLR, LLVM және басқа тәуелсіз іске асыруларды қолдайтын мүмкіндік бар. PyPy жобасы JIT компиляциясын қолданады, бұл Python бағдарламаларының жылдамдығын едәуір арттырады.

Python - белсенді дамып келе жатқан бағдарламалау тілі, тілдік қасиеттерді қосатын/өзгертетін жаңа нұсқалар шамамен екі жарым жылда шығады. Тіл ресми түрде стандартталған жоқ, авторының бақылауымен әзірленген CPython - де-факто стандарттың рөлін атқарады. Қазіргі уақытта Python 8,5% көрсеткішімен TIOBE рейтингінде үшінші орында тұр. Сарапшылар бұл Python-ның рейтингте болған уақытындағы ең жоғары балл екенін айтады.

Python жасаушылары «The Zen of Python» («Питонның дзені» немесе «Пайтонның дзені») деп аталатын белгілі бір бағдарламалау философиясын ұстанады. Оның мәтінін import this командасы бойынша Python интерпретаторы береді(сессияда бір рет жұмыс істейді). Тим Петерс осы философияның авторы болып саналады.

Философия мәтіні.

Ұсқынсыздан гөрі әдемі жақсы.

Айқын еместен гөрі айқын жақсы.

Күрделіден гөрі жеңіл жақсы.

Шиеленіскеннен гөрі күрделі жақсы.

Тіркемеленгеннен гөрі тегіс жақсы.

Тығыздан гөрі сейілген жақсы.

Оқылу жеңілдігі маңызды.

Ерекше жағдайлар ережелерді бұзатындай ерекше емес.

Дегенімен тиімділік тазалықтықтан басым.

Қателер ешқашан үнсіз өтпеуі керек.

Тек айқын үнсіздік болмаса.

Екі мәндік алдында, болжалдау азғыруынан арылыңыз.

Мұны шешуде бір – және тек қана бір айқын тәсілі болғаны дұрыс.

Дегенімен бұл жолы, алғашында, айқын болмауы мүмкін, егер сіз голландтық болмасаңыз.

Ешқашаннан гөрі қазір жақсы.

Дегенімен көп жағдайда «дәл» қазірден гөрі ешқашан жақсы.

Егер іске асыруды түсіндіру қиын болса, онда бұл жаман идея.

Егер іске асыруды түсіндіру оңай болса, онда бұл, мүмкін, жақсы идея.

Атаулар кеңістігі - бұл керемет нәрсе! Оларды көбірек жасайық!

Тарихы.

Python тілін әзірлеуін Нидерландтық CWI институтының қызметкері Гуидо ван Россум 1980 жылдардың соңында бастады. Үлестірілген Amoeba ОЖ-і кеңейтілетін скрипттік тілін талап етілгендіктен, Гуидо бос уақытында, ABC тілі үшін негізделген нобайлы жұмыстарын қолдана отырып(Гуидо бағдарламалауды үйретуге бағытталған осы тілді әзірлеге қатысты), Python-ды жазуды бастады. 1991 жылдың ақпан айында Гуидо бастапқы мәтінді alt.sources жаңалықтар тобында жариялады. Басынан бастап Python объектіге бағытталған тіл ретінде жасалынған.

Мейірімді, елгезек пайдаланушылар қоғамдастығы, Гуидоның дизайнерлік түйсігі, Python-ның сәттілік факторларының бірі болып саналады. Тілдің дамуы — талқылау, таңдау, іске асырудың және PEP(Python Enhancement Proposal; Python-ды жетілдіру бойынша ұсыныстар) құжаттарын құрудың нақты реттелген процесіне сәйкес жүреді.

2008 жылғы 3 желтоқсанда ұзақ сынақтардан кейін Python 3000 бірінші нұсқасы шығарылды (немесе Python 3.0, Py3k аббревиатурасы да қолданылады). Python 3000 Python-ның ескі нұсқаларымен үйлесімділіктің максималды (бірақ толық емес) сақталуымен көптеген архитектуралық кемшіліктерді шешеді.

Портталуы.

Python барлық танымал платформаларға портталған және барлығында жұмыс істейді — қалталы дербес компьютерлерден бастап мейнфреймдерге дейін. Microsoft Windows, UNIX-тың барлық дерлік нұсқалары(FreeBSD және Linux қоса алғанда), Plan 9, Mac OS және macOS, iPhone OS (iOS) 2.0 және одан жоғары, iPadOS, Palm OS, OS/2, Amiga, HaikuOS, AS/400 және тіпті OS/390, Windows Mobile, Symbian және Android-қа арналған порттары бар.

Платформаның ескіруіне қарай оны тілдің негізгі тармағында қолдау тоқтатылады. Мысалы, 2.6 нұсқасынан Windows 95, Windows 98 және Windows ME қолдауы тоқтатылды. Алайда, бұл платформаларда алдыңғы Python нұсқаларын қолдануға болады-қазіргі уақытта қауымдастық 2.3 бастап Python нұсқаларын белсенді қолдайды (олар үшін түзетулер шығады).

Көптеген портталатын жүйелерге қарағанда, Python барлық негізгі платформалары үшін осы платформаға тән технологиялардың қолдауына ие (мысалы, Microsoft COM/DCOM). Сонымен қатар, Java виртуалды машинасына арналған Python-ның арнайы — Jython нұсқасы бар, бұл интерпретаторға Java қолдайтын кез келген жүйеде орындалуға мүмкіндік

береді, бұл ретте Java кластары тікелей Python-нан қолданыла алады және тіпті Python-да жазыла алады. Сондай-ақ, бірнеше жобалар Microsoft.NET платформамен интеграцияны қамтамасыз етеді, олардың негізгілері — IronPython және Python.Net.

Мәліметтердің түрлері мен құрылымдары.

Python динамикалық типтендіруді қолдайды, яғни айнымалы түрі тек жұмыс уақытында анықталады. Сондықтан, «айнымалы мәнін тағайындау» орнына «айнымалыны қандай да бір атаумен байланыстыру» туралы айтқан жөн. Python-да енгізілген типтер: бульдық, жолдық, Unicode-жол, еркін дәлдікті бүтін сан, қалқыма нүктелі сан, комплексті сан және басқалар. Python-дағы коллекциялар ішінен: тізім, кортеж(өзгермейтін тізім), сөздік, жиындар және басқалар кіреді. Барлық мәндер объектілер болып табылады, соның ішінде функциялар, әдістер, модульдер, кластар.

Жаңа типті класс(class) жазу арқылы немесе кеңейту модуліндегі жаңа типті анықтай отырып(мысалы, C түрінде жазылған) қосуға болады. Класс жүйесі мұрагерлікті (жалғыз және көп) және метапрограммалауды қолдайды. Кірістірілген көптеген түрлерден және кеңейту түрлерінен мұрагерлік мүмкін.

Барлық объектілер сілтемелік және атомарлық болып бөлінеді. Атомарлыққа int, long (3-нұсқада int-тің кез-келген саны кіреді, өйткені 3-нұсқада өлшем шегі жоқ), complex және басқалар жатады. Атомарлық объектілерді тағайындау кезінде олардың мәні көшіріледі, ал сілтемелік үшін объектіге сілтеме ғана көшіріледі, сондықтан тағайындаудан кейінгі екі айнымалы мән де бірдей мәнді қолданады. Сілтемелік объектілер өзгермелі және өзгермейтін болып бөлінеді. Мысалы, жолдар мен кортеждер өзгермейтін болып табылады, ал тізімдер, сөздіктер және көптеген басқа объектілер өзгеріске ұшырайды. Python-дағы кортеж - бұл өзгермейтін тізім болып саналады. Көптеген жағдайларда кортеждер тізімдерге қарағанда тезірек жұмыс істейді, сондықтан, егер реттілікті өзгертуді жоспарламаған жағдайда оларды қолданған дұрыс.

Синтаксис және семантика.

Тіл нақты және дәйекті синтаксиске, ойластырылған модульдікке және масштабтылыққа ие, соның арқасында Python-да жазылған бағдарламалардың бастапқы коды оңай оқылады. Python функциясына аргументтерді берген кезінде бірлесіп пайдалану (call-by-sharing) шақыруын қолданады.

2018 жылы Гуидо ван Россум, тілдің құрастырушысы, master және slave терминдерін parent және child-қа, саяси дұрыстылық үшін тілдің терминологиясын өзгерту туралы шешім қабылдады.

Операторлар.

Операторлар жиынтығы дәстүрлі болып келеді.

- if(егер болса) шартты операторы. else(әйтпесе) кейінгі алтернативті блок. Егер бірнеше шарттар мен баламалар болса, elif(else if қысқ.) қолдануға болады;

- while және for цикл операторлары. Цикл ішінде break және continue, циклді үзу және сәйкесінше келесі итерацияға өту үшін қолдануға болады;

- class классты анықтау операторы;
- def функцияны, әдісті немесе генераторды анықтайтын оператор. Ішінде функциядан немесе әдістен қайтару үшін return(қайтару) қолдануға болады, ал генератор жағдайында yield(беру) мүмкін;
- Try — except — else немесе try — finally ерекшеліктерді өңдеу операторы (2.5 нұсқасынан бастап, бір блокта finally, except және else пайдалануға болады);
- pass операторы ештеңе жасамайды. Кодтың бос блоктары үшін қолданылады.

Тілдің қызықты синтаксистік ерекшеліктерінің бірі - шегіністі қолдана отырып кодтар блоктарын бөлу (бос орындар немесе табуляция), сондықтан Python-да Паскаль тілінде сияқты begin/end операторлық жақшалар немесе Си сияқты фигуралық жақшалар жоқ. Бұл «трюк» бағдарламадағы жолдар мен таңбалардың санын азайтуға және "жақсы" бағдарламалау стиліне үйретеді.

Өрнектер.

Өрнектер Python-да толық құқылы оператор. Құрамы, синтаксис, ассоциативтілік және операциялардың басымдылығы бағдарламалау тілдері үшін өте үйреншікті және жақшаларды аз пайдалануды пайымдайды.

Жолдар үшін пішімдеу операциясын бөлек атап өту керек (si-ден printf () аналогиясы бойынша жұмыс істейді), ол бөлуден қалдықты алу сияқты таңбаны пайдаланады:

```
>>> str_var = "world"
>>> print("Hello, %s" % str_var)
Hello, world
```

Python-да ыңғайлы тізбекті салыстырулар бар. Бағдарламалардағы мұндай жағдайлар сирек емес

```
1 <= a < 10 and 1 <= b < 20
```

Сонымен қатар, логикалық (and және or) операциялары жалқау(lazy evaluation) болып табылады: егер бірінші операнд операцияның мәнін есептеу үшін жеткілікті болса, онда бұл операнд нәтиже ретінде қолданылады, әйтпесе логикалық операцияның екінші операнды есептеледі. Бұл логика алгебрасының қасиеттеріне негізделген: мысалы, егер «НЕМЕСЕ» (or) операциясының бір аргументі ақиқат болса, онда бұл операцияның нәтижесі де әрдайым ақиқат болады. Егер екінші операнд күрделі өрнек болса, бұл оны есептеуге арналған шығындарды қысқартуға мүмкіндік береді. Бұл факт 2.5 нұсқасына дейін шартты конструкцияның орнына кеңінен қолданылды:

```
a < b and "кіші" or "үлкен немесе тең"
```

Енгізілген деректер түрлері, әдетте, өз литералдары үшін ерекше синтаксиске ие (константаның бастапқы кодында жазылған):

"бір мезгілде жол және Юникод-жол"
'бір мезгілде жол және Юникод-жол'
"""бір мезгілде жол және Юникод-жол"""
True or False # бульдық литералдар
3.14 # қалқыма нүктелі сан
0b1010 + 0o12 + 0xA # екілік, сегіздік және он алтылық есептеу жүйелеріндегі сандар
1 + 2j # комплексті сан
[1, 2, "a"] # тізім
(1, 2, "a") # кортеж
{'a': 1, 'b': 'B'} # сөздік
{'a', 6, 8.8} # жиын
lambda x: x**2 # анонимдық функция

Тізімдер (және басқа да тізбектер) үшін Python кесу(срез) операцияларының жиынтығын ұсынады. Тізім элементтерінің индексі нөлден басталады. s[N:M] кесіндінің жазбасы N-нен (қоса алғанда) M-ге (қоса алмағанда) дейінгі барлық элементтерді білдіреді. Алайда индексті елемеуге болады. Мысалы, s[: M] жазбасы барлық элементтер кескінге басынан бастап түсетіндігін білдіреді; s[N:] жазбасы барлық элементтер кескінге соңына дейін түсетіндігін білдіреді; s[:] белгісі барлық элементтер кескінге басынан аяғына дейін түсетіндігін білдіреді.

Мүмкіндіктер. Объектілі-бағытталған бағдарламалау(ОББ).

Python тілінің дизайны объектілі-бағытталған бағдарламалау моделінің айналасында жасалынған. Python-да ОББ-ның жүзеге асырылуы талғампаз, қуатты және жақсы ойластырылған болып табылады, сонымен бірге басқа объектілі-бағытталған тілдермен салыстырғанда өте ерекше болып табылады.

Мүмкіндіктері мен ерекшеліктері:

- класстар, төмендегі келтірілген барлық мүмкіндіктермен, бір мезгілде, объектілер болып табылады;
- мұрагерлік, соның ішінде көптік(множественное);
- полиморфизм (барлық функциялар виртуалды);
- инкапсуляция (екі деңгей — жалпыға қолжетімді және жасырын әдістер мен өрістер). ерекшелігі — жасырын мүшелер пайдалану үшін қол жетімді және тек ерекше аттармен жасырын ретінде белгіленед;
- объектінің өмірлік циклін басқаратын арнайы әдістер: конструкторлар, деструкторлар, жад таратқыштары;
- операторларды шамадан тыс жүктеу(overloading)(is, '.', '=' және символдық логикалықтан басқасының барлығы);
- қасиеттері (функциялар арқылы өрісті имитациялау);

- өрістерге қатынауды басқару (өрістер мен әдістерді эмуляциялау, ішінара қатынау және т. б.);
- ең көп таралған операцияларды басқару әдістері (ақиқат мән, len(), терең көшіру, сериализация, объект бойынша итерация, …);
- метабағдарламалау (класстарды құруды басқару, класстарды құру триггерлері және т. б.);
- толық интроспекция;
- классикалық және статикалық әдістер, класстық өрістер;

функциялар мен кластарға салынған класстар.

Функционалды бағдарламалау.

Python функционалдық бағдарламалау парадигмасын қолдайды, атап айтқанда:

- функция объект болып табылады;
- жоғары ретті функциялар;
- рекурсия;
- дамыған тізімді өңдеу (тізімдік өрнектер, тізбектермен операциялар, итераторлар);
- тұйықталу аналогы;
- функцияны ішінара қолдану;
- тілдің ішінде басқа құралдарды жүзеге асыру мүмкіндігі (мысалы, карринг).

Модульдер мен пакеттер.

Python-да бағдарламалық қамтамасыздандыру (қосымша немесе кітапхана) модульдер түрінде орындалады, олар өз кезегінде пакеттерге жинала алады. Модульдер каталогтарда да, ZIP архивтерінде де орналасуы мүмкін. Модульдер шығу тегіне байланысты екі түрі болуы мүмкін: «таза» Python-да жазылған модульдер және басқа бағдарламалау тілдерінде жазылған кеңейту модульдері(extension modules). Мысалы, стандартты кітапханада «таза» pickle модулі және оның C-дағы: cPickle аналогы бар. Модуль бөлек файл түрінде, ал пакет бөлек каталог ретінде орындалады. Модульдің бағдарламаға қосылуы import операторы арқылы жүзеге асырылады. Модуль импорттағаннан кейін модульдің аттар кеңістігіне қол жетімділік беретін жеке нысан ретінде ұсынылады. Бағдарламаны орындау кезінде модульді reload() функциясымен қайта жүктеуге болады.

Интроспекция.

Python орындау уақытының толық интроспекциясын қолдайды. Бұл дегеніміз, кез келген объект үшін оның ішкі құрылымы туралы барлық ақпаратты алуға болады.

Интроспекцияны қолдану pythonic style деп аталатынның маңызды бөлігі болып табылады және PyRO, PLY, Cherry, Django және т. б. сияқты, бағдарламалаушының айтарлықтай уақытын үнемдейтін, Python кітапханалары мен фреймворктарында кеңінен қолданылады.

Итераторлар.

Python бағдарламалары итераторларды кеңінен қолданады. for циклі тізімдермен де итератормен де жұмыс істей алады. Көптеген коллекциялар итераторларды қамтамасыз етеді, итераторларды пайдаланушылар өздерінің жеке нысандары үшін де анықтай алады. Стандартты кітапхананың itertools модулінде итераторлармен жұмыс істеуге арналған құралдар бар.

Генераторлар.

Тілдің қызықты мүмкіндіктерінің бірі генераторлар болып табылады — ішкі күйін(жергілікті айнымалы мәндер және ағымдағы нұсқаулар) сақтайтын функциялар. Генераторлар деректер құрылымына және жалқау есептеулерге арналған итераторлар ретінде пайдаланылуы мүмкін. Мысал: Фибоначчи сандар генераторы.

Генераторды шақырғанда, функция ағымдағы орындау нүктесін және жергілікті айнымалы функцияның күйін сақтайтын объект-итераторын дереу қайтарады. Келесі мәнді шақырған кезінде (айқын емес шақырылатын for циклында, next() әдісі арқылы) генератор функцияны алдыңғы тоқтау нүктесінен келесі yield немесе return операторына дейін орындауды жалғастырады.

Python 2.4 генераторлық өрнектер — генератор нәтижесінде беретін өрнектер пайда болды. Генераторлық өрнектер аралық нәтижелері бар тізімді пайдалануды талап ететін жерде жады үнемдеуге мүмкіндік береді:

```
>>> sum(i for i in xrange(1, 100) if i % 2 != 0)
2500
```

Бұл мысалда 1-ден 99-ға дейінгі тақ сандар қосылады.

Сондай-ақ, Python тіркемеленген генераторларды қолдайды. Мысалы, екі өлшемді массив құру үшін, барлық жолдардың генераторы ішінде жол болып табылатын тізім генераторын орналастыру қажет: [[0 for j in range(m)] for i in range (n)]

Орындау мәнмәтінін басқару.

Python 2.5 код блогын орындау контекстін басқару құралы — with операторы және contextlib модулі пайда болды.

Оператор кейбір әрекеттерге дейін және одан кейін блокта қозғалған ерекшеліктерге немесе return операторларына қарамастан, кейбір басқа әрекеттер міндетті түрде орындалуға тиіс жағдайларда қолданылуы мүмкін: файлдар жабылуы тиіс, ресурстар босатылуы тиіс, стандартты кіріс енгізу қайта жіберілуі аяқталған және т.б. Оператор кодтың оқылуын жақсартады, демек, қателерді болдырмауға көмектеседі.

Стандартты кітапхана.

Бай стандартты кітапхана Python-ның тартымды жақтардың бірі болып табылады. Бұл жерде көптеген желілік хаттамалармен және интернет форматтарымен жұмыс істеуге арналған құралдар бар, мысалы, HTTP-серверлер мен клиенттерді жазуға арналған модульдер, пошта хабарламаларын талдау және жасауға, XML-мен жұмыс істеуге және т.б.

арналған модульдер бар. Операциялық жүйемен жұмыс істеуге арналған модульдер жиынтығы кроссплатформалық қосымшаларды жазуға мүмкіндік береді. Тұрақты өрнектермен, мәтіндік кодтармен, мультимедиялық форматтармен, криптографиялық хаттамалармен, мұрағаттармен, деректерді сериализациялау, юнит-тестілеуді қолдау және т. б. жұмыс істеуге арналған модульдер бар.

Басқа тілдермен салыстыру.

Салыстырмалы түрде кеш пайда болған Python көптеген бағдарламалау тілдерінің әсерінен құрылды:

- ABC-операторларды топтастыру үшін шегіністер, жоғары деңгейлі деректер құрылымы (map) (Python нақты ABC жобалау кезінде жіберілген қателерді түзету әрекеті ретінде құрылды);
- Modula-3-пакеттер, Модульдер, try және except-пен бірге else-ты пайдалану, функция аргументтері (сондай-ақ Common Lisp әсер етті);
- C, C++ — кейбір синтаксистік конструкциялары (Гуидо ван Россум өзі жазуы бойынша – Python, C-бағдарламалаушыларда жақтырмаушылықты тудырмау үшін C — дың ең қарама-қайшы емес конструкциясын пайдаланды);
- Smalltalk — объектілі-бағытталған бағдарламалау;
- Lisp, атап айтқанда, Scheme — функционалдық бағдарламалаудың жеке ерекшеліктері (lambda, map, reduce, filter және басқалар));
- Fortran — массивтердің кесіктері, кешенді арифметика;
- Miranda — тізімдік өрнектер;
- Java — logging модулдері, unittest, threading (түпнұсқа модульдің мүмкіндіктерінің бөлігі іске асырылмаған), стандартты кітапхананың xml.sax, ерекшеліктерді өңдеу кезінде finally және except ортақ пайдалану, декораторлар үшін @ пайдалану;
- Icon — генераторлар;
- Python басқа мүмкіндіктерінің көп бөлігі (мысалы, бастапқы кодтың байт-компиляциясы) бұрын басқа тілдерде іске асырылған.

Python-ды ең жиі Perl және Ruby-мен салыстырады. Бұл тілдер де интерпретацияланады және бағдарламаларды орындаудың жылдамдығы шамамен бірдей. Perl сияқты, Python скриптерді(сценарийлерді) жазу үшін сәтті қолданылуы мүмкін.

Ruby сияқты, Python-ОББ үшін жақсы ойластырылған жүйе. Бірақта Python-да ОББ жүзеге асырылуы басқа да объектілі-бағытталған тілдерден ерекшеленеді. Атап айтқанда:

Ruby-ге қарағанда, Python "барлығы — объект" идеологиясын ұстамайды және кластар иерархиясына кірмейтін енгізілген примитивті типтерді қолдайды. Мұндай шешімді объекті көзқарастың жанкүйерлері ыңғайсыз деп санауы мүмкін болса да, тіларалық өзара іс-әректті жеңілдетеді және техникалық жағынан тиімді етеді.

Python-да кейбір ОББТ(Java, Object Pascal, Ruby,...) сияқты барлық объектілер жалпы әдістерді иеленетін нақты ортақ базалық класс жоқ. Python-

да формальды жаңа класс object типін (тікелей немесе жанама) иеленсе де, бұл тек синтаксистік тәсіл болып табылады, өйткені барлық объектілер үшін ортақ әдістер — id, type, isinstance, issubclass, str, repr, getattr, ... object-тен мұра етілмеген, ал тек жаһандық функциялар түрінде іске асырылған. Мұндай шешім осы әдістердің мінез-құлқын өзгерту артық жүктемемен емес, класстың арнайы әдістерін анықтаумен жүргізілуіледі.

Коммерциялық қосымшалар ортасында Python бағдарламасын орындау жылдамдығын Java қосымшаларымен жиі салыстырады.

Қолдануы.

Python — тұрақты және кең таралған тіл. Ол көптеген жобаларда және әртүрлі сапаларда негізгі бағдарламалау тілі ретінде немесе кеңейтулер мен қосымшаларды интеграциялау үшін қолданылады. Python-да көптеген жобалар іске асырылған, сондай-ақ ол болашақ бағдарламалардың прототипін құру үшін белсенді қолданылады. Dropbox, Google (мысалы, Youtube және Youtube API кейбір бөліктері Python жазылған), Facebook, Instagram сияқты көптеген ірі компанияларда Python кеңінен қолданылады.

Python озінің NumPy, SciPy және MatPlotLib пакеттері арқасында ғылыми есептеулер үшін әмбебап орта ретінде Matlab, IDL және басқалардың кең таралған мамандандырылған коммерциялық пакеттердің аналогы ретінде белсенді қолданылады. Astropy кітапханасы — астрономиялық есептеулерге арналған танымал құрал.

Қарапайымдылық және ықшамдылық үйлесімімен бірге күрделі абстракциялар мен қуатты әр түрлі құралдарды пайдалану мүмкіндігі Python-ды скрипті тіл ретінде ыңғайлы етеді. Оның енгізілу мүмкіндігі интерпретатордың көлемімен шектеледі, бірақ ірі жүйелерде бұл шектеу маңызды емес. Autodesk Maya, Blender, Houdini және Nuke сияқты үш өлшемді кәсіби графикалық бағдарламаларында Python бағдарламалардың стандартты мүмкіндіктерін кеңейту үшін қолданылады. Python Microsoft Power BI Desktop -те, енгізілген сұратым тілдерімен және R бағдарламалау тілімен қатар, деректерді ETL-процестерде, есептеулерде және деректерді графикалық визуалдауда жүктеу кезеңінде пайдаланылуы мүмкін.

Сондай-ақ, Python жобаларды құрастыру жүйелерінде стандартты емес немесе күрделі міндеттерді орындау үшін жарамды, бұл бастапқы файлдарды алдын ала компиляциясы қажеттілігінің жоқтығына байланысты. Google Test жобасында ол C++ тілі кластары үшін бастапқы mock кодын генерациялау үшін қолданылады.

Python интерпретаторы ОЖ командалық файлдарын жазу үшін қуатты командалық қабықша және скрипт тілі ретінде пайдаланылуы мүмкін. Python-скрипттерден сыртқы бағдарламаларға жүгінудің жеңілдігі және жүйені басқаруға рұқсат беретін кітапханалардың болуы Python-ды жүйелік әкімшілендіру үшін ыңғайлы құрал етіп жасайды. Ол Linux платформасында осы мақсат үшін кеңінен қолданылады: әдетте Python жүйемен бірге қамтамасыздандырады, көптеген дистрибутивтерде инсталляторлар және жүйелік утилиталардың визуалды интерфейсі дәл Python-да жазылған. Ол

басқа Unix-жүйелерді, атап айтқанда, Solaris және macOS-та әкімшілендіруде де қолданылады. Тіл мен кітапханалардың кроссплатформалығы оны әр түрлі операциялық жүйелі компьютерлерді қолданатын гетерогенді орталарда жүйелік әкімшілендіру міндеттерін біріздендірілген автоматтандыру үшін тартымды етеді [13].

2.1.1 Нейрондық желілерді құруға арналған кітапханалар. TensorFlow (2.1 сурет).

2.1 сурет — TensorFolow

TensorFlow — адам қабылдауының сапасына жететіндей, бейнелерді автоматты түрде табу және жіктеу мақсатында, нейрондық желіні құру және оқыту мәселелерін шешу үшін, Google компаниясымен әзірленген машиналық оқытуға арналған ашық бағдарламалық кітапхана. Зерттеу үшін де, Google өз өнімдерін әзірлеу үшін де қолданылады. Кітапханамен жұмыс істеу үшін негізгі API Python үшін іске асырылды, сондай-ақ C Sharp, C++, Haskell, Java, Go және Swift үшін іске асырулары бар.

Бастапқыда TensorFlow Google ішкі пайдалану үшін Google Brain командасымен әзірленген болатын, 2015 жылы жүйе Apache 2.0 ашық лицензиясымен еркін қол жеткізуге ауыстырылды.

DistBelief машиналық оқытудың жабық жүйесі 2011 жылдан бастап, Google Brain-мен оның ішкі жобалар үшін, терең оқытылатын нейрондық желілермен жұмыс істеу мақсатында әзірленді. Ол alphabet холдингінің фирмалар тобтарының көптеген зерттеулерінде және коммерциялық жобаларында қолданыла бастады. DistBelief жетістігінен кейін, Google фирмасы жобаны жаңа деңгейге шығаруды шешті және рефакторинг үшін Джефф Дин кірген бірнеше әзірлеушіден тұратын топты бөлді; топтың мақсаты кітапхана кодтарын оңайлату және оңтайландыру, пайдаланудың сенімділігі мен ыңғайлылығын арттыру болды. Жаңа кітапхана TensorFlow деп аталды. 2013 жылы жобаға Джеффри Хинтон — ғалым қосылды, оның басшылығымен 2009 жылы нейрондық желілердің дәлдігін айтарлықтай

жақсартуға мүмкіндік берген қатенің жалпыланған кері таралу әдісі және басқа да бірқатар жақсартулар жасалды (бұл, атап айтқанда, сөйлеуді тану қателігінің 25% - ға төмендеуіне алып келді).

TensorFlow 2015 жылдың 9 қарашасында еркін қол жеткізу үшін ашылды. TensorFlow - Google Brain машиналық оқытудың екінші буыны. Эталондық іске асырылуы кейбір ғана құрылғыларда жұмыс істеген кезде, TensorFlow CPU-да да GPU-да да графикалық процессорларда жалпы мақсаттағы есептеулерді қолдау үшін CUDA архитектурасына сүйене отырып көптеген параллельді процессорларда жұмыс істей алады . TensorFlow 64 биттік Linux, macOS, Windows, сонымен қатар Android және iOS мобильді есептеу платформалары үшін де қол жетімді.

TensorFlow есептеулері мәліметтер ағындары түрінде күй бағандары арқылы көрсетіледі. TensorFlow атауы "тензор" деп аталатын көп өлшемді деректер массивтерін алу амалынан шыққан. 2016 жылдың маусым айында Google-дағы Джефф Дин GitHub-та TensorFlow-ға 1500 репозиторий жүгінгенін және соның ішінде тек 5-і Google-дан болғанын атап өтті.

2016 жылдың мамыр айында Google өз әзірлемесіндегі терең оқыту мақсаттарына арналған, TensorFlow міндеттеріне бейімделген және төмен дәлдіктегі арифметикадағы(мысалы, 8 биттік архитектураға арналған) жоғары өнімділікті қамтамасыз ететін, модельдерді оқытуға қарағанда, жылдам қолдануға бағытталған аппараттық үдеткішті — тензорлық процессорды (TPU) қолданғаны туралы хабарлады.

Google деректерді өңдеудің жеке тапсырмаларында TPU-ны қолданғаннан кейін шығындалатын энергияның әр ваттына тиімділік көрсеткіштерін жақсарғаны туралы айтылды.

TensorFlow DeepDream сияқты жүйелерде бейнелерді автоматты түрде аннотациялау үшін қолдану өте қолайлы. 2015 жылдың 26 қазанынан бастап Google іздеу жүйелерінің рейтингісінің өзектілігін арттыру үшін Google RankBrain бағдарламасын қолданып келеді. RankBrain TensorFlow бағдарламасына негізделген.

TensorFlow генеративті-жарыс желілерін (GAN) оқытуға мүмкіндік береді [14].

Keras (2.2 сурет).

2.2 сурет — Keras

Keras — бұл Python-да жазылған бастапқы коды ашық нейрондық желілілер кітапханасы. Ол TensorFlow, Microsoft Cognitive Toolkit, R, Theano немесе PlaidML-тің үстінен жұмыс істей алады. Терең нейрондық желілермен жылдам тәжірибе жасауға әзірленген, ол қолдану ыңғайына, модульдікке және кеңейтілуге бағытталған. Keras ONEIROS (Open Neuro-Elektron Intelligent Robot Operating System) жобасының ғылыми-зерттеу жұмыстарының бір бөлігі ретінде әзірленді, оның негізгі авторы және жүргізушісі Google инженері Франсуа Шолле болып табылады. Шолле сонымен қатар XCeption терең нейрондық желі моделінің авторы.

2017 жылы Google-нің TensorFlow тобы TensorFlow негізгі кітапханасында Keras-ты қолдауға шешім қабылдады. Шолле Керастың машиналық оқытудың жеке фреймворктан гөрі интерфейс болып табылатынын түсіндірді. Бұл кітапхана пайдаланылатын бэкэндке қарамастан терең оқыту модельдерін жасауды жеңілдететін жоғары деңгейлі, интуитивті абстракциялар жиынтығын ұсынады. Microsoft-та CNTK серверін Keras-қа қосты, CNTK v2.0 нұсқасында қол жетімді.

Keras құрамында терең нейрондық желі кодын жазуға қажетті кодтауды жеңілдететін, нейрондық желінің құрылымдық блоктарының көптеген түрлері бар, мысалы, қабаттар, мақсаттар, белсендіру функциялары, оптимизаторлар және сурет пен мәтіндік деректермен жұмыс істеуді жеңілдететін көптеген құралдар. Код GitHub-та орналастырылған, ал қауымдастыққа қолдау көрсету форумдары GitHub мәселелер парақшасы мен Slack арнасын қамтиды.

Стандартты нейрондық желілерден басқа Keras үйіркілі және рекурентті нейрондық желілерді қолдайды. Ол dropout, пакеттік нормалау және pooling сияқты басқа жалпы қызметтік қабаттарын қолдайды.

Keras пайдаланушыларға смартфондарда (iOS және Android), интернетте немесе Java виртуалды машинасында терең модельдер жасауға мүмкіндік береді. Бұл сондай-ақ CUDA үйлесімімен бірге, графикалық процессорлар кластерлерінде (GPU) және тензорлық процессорлар кластерлерінде (TPU) терең оқыту модельдерінің таратылған оқытуын қолдануға мүмкіндік береді.

Keras 2018 жылдың ортасына қарай 250,000 жеке пайдаланушылардың бар екенін айтты. Keras KDnuggets 2018 бағдарламалық жасақтамасында жүргізілген сауалнамада 10-шы ең танымал құрал болды және 22% қолданушыларды тіркеді [15].

2.1.2 Дерекқормен жұмыс жасауға арналған кітапханалар. NumPy — бастапқы коды ашық Python бағдарламалау тіліне арналған кітапхана. Мүмкіндіктері:

- көп өлшемді массивтерді(оның ішінде матрицаларды) қолдау;
- көп өлшемді массивтермен жұмыс істеуге арналған жоғары деңгейлі математикалық функцияларды қолдау.

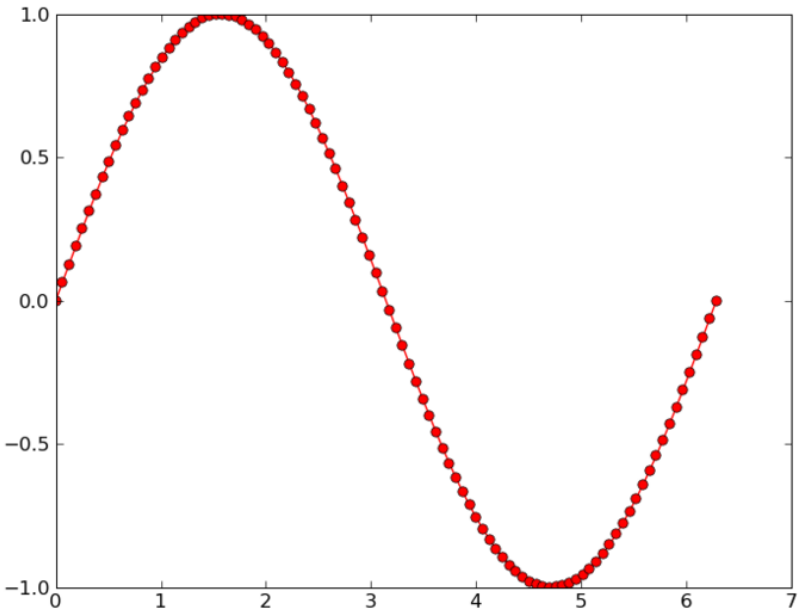
Интерпретацияланатын тілдерде (мысалы, Python) іске асырылған математикалық алгоритмдер компиляцияланатын тілдерде (мысалы, Фортран, Си, Java) іске асырылған сол алгоритмдерден көп жағдайда баяу жұмыс істейді. NumPy кітапханасы көп өлшемді массивтермен жұмыс істеу үшін оңтайландырылған есептеу алгоритмдерінің (функциялар мен операторлар түрінде) іске асыруларын ұсынады. Нәтижесінде NumPy қолданумен іске асырылған және массивтерге жасалатын (матрицаларға) операциялардың кезектілігі түрінде көрсетілуі мүмкін болатын кез келген алгоритм MATLAB-та орындалатын эквивалентті код сияқты тез жұмыс істейді.

NumPy MATLAB-тың тегін баламасы ретінде қарастыруға болады. MATLAB бағдарламалау тілі NumPy-ға сырттай ұқсас келеді: екеуі де интерпретацияланатын, екеуі де скалярларға емес, массивтерге (матрицаларға) операция жасауға мүмкіндік береді. MATLAB-тың артықшылығы көптеген пакеттердің («тулбокстардың») құрамында болуы, мысалы, Simulink. NumPy үшін де ұқсас "пакеттер" бар, мысалы, SciPy кітапханасы көптеген MATLAB-ұқсас функционалдылықты ұсынады, Matplotlib кітапханасы MATLAB стилінде графиктер құруға мүмкіндік береді. MATLAB және NumPy сызықтық алгебра негізгі есептерін шешу үшін LAPACK кітапхана кодына негізделген кодты қолданады.

NumPy жұмыс үлгісін қарастырайық.

```
x = linspace(0, 2*pi, 100)
y = sin(x)
plot(x, y, 'ro-')
show()
```

Нәтижесінде Matplotlib кітапханасы 2.3 суретте көрсетілген графикты құрады [16].



2.3 сурет — NumPy кодының графикалық іске асырылуы

pandas — бұл деректерді өңдеуге және талдауға арналған Python бағдарламалық тіліндегі кітапханасы. pandas-тың деректермен жұмысы төменгі деңгейдегі құрал болып табылатын NumPy кітапханасының үстінен құрылған. Сандық кестелер мен уақыт қатарларын басқаруға арналған арнайы құрылымдар мен операцияларды қамтамасыз етеді. Кітапхана атауы көп өлшемді құрылымдық ақпарат жиынтығын сипаттау үшін пайдаланылатын «панельдік деректер» (ағыл. panel data) эконометрикалық терминінен шыққан. pandas жаңа BSD лицензиясы бойынша таратылады.

Қолданудың негізгі саласы - Python ортасында мәліметтерді жинау және тазарту үшін ғана емес, сонымен бірге статистикалық өңдеудің нақты тілдеріне ауыспастан (R және Oktava сияқты) деректерді талдау және модельдеу тапсырмаларын орындау.

Сондай-ақ, деректердің «жергілікті» түрлерін енгізу бойынша жұмыстар жүргізілуде.

Пакет барлығынан бұрын мәліметтерді жалпы белгілері бойынша тазартуға және бастапқы бағалауға арналған, мысалы, орташа мән, квантилдер бойынша және т.б.; бұл толық мағынада статистикалық пакет емес, дегенімен, DataFrame және Series типтегі мәліметтер жиынтығы деректерді талдау мен машиналық оқытудың көптеген модулдеріне (SciPy, Scikit-Learn және басқалары) кірмелі жиынтық ретінде пайдаланылады.

Кітапхананың негізгі ерекшеліктері:

- екі өлшемді деректердің индекстелген массивтерін басқаруға арналған DataFrame объектісі;

- жадыдағы құрылымдар мен әртүрлі форматтағы файлдар арасында мәліметтер алмасу құралдары;
- деректерді біріктіруге арналған құралдар және жетіспейтін ақпаратты өңдеу әдістері;
- деректер жинағын қайта форматтау, оның ішінде құрама кестелер құру;
- деректерді индекс мәндері бойынша кесу, индекстеудің кеңейтілген мүмкіндіктері, үлкен деректер жиынтығынан іріктемелеу;
- деректер бағандарын кірістіру және жою;
- топтастыру мүмкіндіктері сізге «бөлу, өзгерту, біріктіру» (англ. split-apply-combine) сияқты үш сатылы әрекеттерді орындауға мүмкіндік береді;
- деректер жиынтығын қосу және біріктіру;
- иерархиялық индекстеу төменгі өлшемді құрылымдарда жоғары өлшемді мәліметтермен жұмыс істеуге мүмкіндік береді;
- уақыт қатарларымен жұмыс: уақыт кезеңдерін қалыптастыру және өзгеру аралықтары және т.б.

Кітапхана жоғары өнімділік үшін оңтайландырылған, кодтың маңызды бөліктері Cython және C-де жазылған [17].

Matplotlib — бұл Python бағдарламалау тіліндегі мәліметтерді екіөлшемді (2D) графикада визуализацияға арналған кітапхана (3D графикасы да қолданылады). Алынған суреттерді жарияланымдарда иллюстрация ретінде қолдануға болады.

Matplotlib негізінен Джон Хантермен жазылған және қызмет етеді сонымен қатар BSD тәрізді лицензия бойынша таратылады. Түрлі форматта жасалған суреттер интерактивті графикада, ғылыми жарияланымдарда, пайдаланушының графикалық интерфейсінде, диаграмма құру қажет ететін веб-қосымшаларда қолданыла алады. Құжаттамада автор Matplotlib MATLAB-тың графикалық командаларын имитациялаудан басталғанын, бірақ одан тәуелсіз жоба екенін айтады.

2.1.1 нұсқа - соңғы тұрақты нұсқасы - Python 2.7 немесе 3.4 нұсқасын және одан жоғары және NumPy 1.7.1 және одан жоғары нұсқасын қажет етеді.

Matplotlib кітапханасы ОББ қағидаттарына негізделген, бірақ MATLAB командаларының аналогтарын беретін pylab процедуралық интерфейсін қамтиды.

Matplotlib - бұл икемді, оңай конфигурацияланатын пакет, ол NumPy, SciPy және IPython-мен бірге MATLAB-қа ұқсас мүмкіндіктерді ұсынады. Қазіргі уақытта пакет бірнеше графикалық кітапханалармен жұмыс істейді, соның ішінде wxWindows және PyGTK.

Пакет көптеген графиктер мен диаграммаларды қолдайды:
- графиктер (line plot);
- шашырату диаграммалары (scatter plot);
- бағаналық диаграммалар (bar chart) мен гистограммалар (histogram);
- дөңгелек диаграммалар (pie chart);
- жапырақты диаграммалар (stem plot);
- контурлық графиктер (contour plot);

- градиент өрістері (quiver);
- спектрлік диаграммалар (spectrogram).

Пайдаланушы координаталар осін, торды көрсете алады, белгілер мен түсініктемелерді қоса алады, логарифмдік шкала немесе полярлық координаттар қолдана алады.

Қарапайым 3D графиканы mplot3d құралдар жиынтығының (toolkit) көмегімен жасауға болады. Басқа да құралдар жиынтығы бар: картография үшін, Excel-мен жұмыс істеуге арналған, GTK-тің утилиталары үшін және басқалары.

Matplotlib көмегімен анимациялық суреттер жасауға болады.

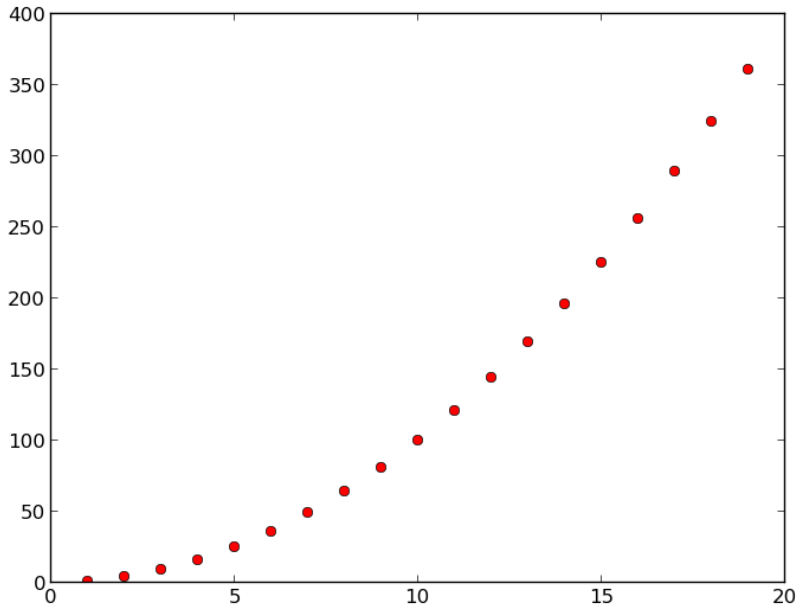
Қолдау көрсетілетін векторлық және растрлықтық кескін форматтарының жиынтығын FigureCanvasBase.filetypes сөздіктерінен алуға болады. Әдеттегі қолдау көрсетілетін форматтар:

- Encapsulated PostScript (EPS);
- Enhanced Metafile (EMF);
- JPEG;
- PDF;
- PNG;
- Postscript;
- RGBA (өңделмеген формат);
- SVG;
- SVGZ;
- TIFF.

Сонымен қатар, басқа модульдерді пакеттік класстар негізінде құруға болады [18].

Келесі мысал графиктер құруды көрсетеді (2.4 сурет):

```
from pylab import *
plot(range(1, 20),
    [i * i for i in range(1, 20)], 'ro')
savefig('example.png')
show()
```



2.4 сурет — Matplotlib графигы

2.1.3 Google Colaboratory. Google Colaboratory – бұл жақында пайда болған, машиналық және терең оқыту саласындағы зерттеулерді жеңілдетуге бағытталған сервис. Коллаборатордың көмегімен GPU бар құрылғыға, атап айтқанда атап айтқанда Tesla K80-ге қашықтан қол жеткізуге болады.

Colaboratory-да алдын ала орнатылған Tensorflow және барлық дерлік керекті Python кітапханалары бар. Егер қандай да бір пакет болмаса, оларды «pip» немесе «apt-get» арқылы орнатуға болады.

Colaboratory файлдары қарапайым .ipynb «ноутбуктер» болып табылады және Google дискінде сақталады. Сондай-ақ, файлдарды қашықтағы компьютерден Google дискісіне және керісінше жүктеуге мүмкіндік беретін функциялар жиынтығы бар. Бұл файлдармен басқалармен бөлісе аласыз, оларға Google құжаттарындағыдай пікірлер жаза аласыз.

2.2 Нейрондық желіні оқытуға арналған дерекқор

Желілік басып кіруді анықтауға арналған бағдарламалық қамтамасыз ету компьютерлік желіні рұқсатсыз пайдаланушылардан, соның ішінде инсайдерлерден қорғайды. Басып кіру детекторын оқыту мәселесі - басып кіру немесе шабуыл деп аталатын "жаман" байланысты және " жақсы " қалыпты байланысты ажыратуға қабілетті болжау моделін (яғни жіктеуіш) құру.

DARPA басып кіруді анықтауды бағалау бағдарламасы 1998 жылы МТИ Линкольн зертханасымен(MIT Lincoln Labs) дайындалды және басқарылады. Бұның мақсаты басып кіруді анықтау саласындағы зерттеулер нәтижелерін бақылау және бағалау болды. Әскери желі ортасында имитацияланатын шабуылдардың кең спектрін қамтитын аудитке арналған деректердің

стандартты жинағы ұсынылды. KDD 1999 жылғы басып кіруді анықтау бойынша конкурс осы деректер жиынтығының нұсқасын пайдаланады.

Lincoln Labs АҚШ-тың ӘӘК-нің (Әскери-Әуе Күштері) типтік жергілікті желісін имитациялайтын LAN үшін тоғыз апталық TCP дамптарының өңделмеген деректерін жинауға арналған орта құрды. Олар ӘӘК-нің жергілікті желсінің нақты ортасы сияқты басқарды, бірақ оны көптеген шабуылдарға ұшыратты.

Өңделмеген оқыту деректері жеті апталық желілік трафиктің төрт гигабайттық сығылған екілік TCP-дампты құрады. Бұл қосылыстардың бес миллион жазбасы түрінде өңделген. Осыған ұқсас, екі апталық тестілеу деректерінен, қосылым туралы екі миллион жазба алынды.

Қосылыс — белгілі бір уақытта басталатын және аяқталатын TCP-пакеттер тізбегі, олардың арасында белгілі бір хаттамаға сәйкес бастапқы IP-мекен-жайынан мақсатты IP-мекен-жайына және кері деректер ағынының жіберілуі жүзеге асырылады. Әрбір қосылыс қалыпты немесе шабуыл ретінде, нақты белгілі бір шабуыл түрі ретінде белгіленеді. Әрбір қосылым жазбасы шамамен 100 Байттан тұрады.

Шабуыл төрт негізгі санатқа бөлінеді:

- DOS – қызмет көрсетуден бас тарту, мысалы, SYN flood;
- R2L – қашықтан компьютерден рұқсат етілмеген кіру, мысалы, құпия сөзді табу;
- U2R – жергілікті Root артықшылықтарына рұқсат етілмеген қол жеткізу, мысалы, "буферді толтыруының" түрлі шабуылдары;
- зондтау – бақылау және басқа да зондтау, мысалы, порттарды сканерлеу.

Тестілеу мәліметтерінің ықтималдық үлестірімі оқыту мәліметтеріндегідей емес және оқыту мәліметтерінде жоқ шабуылдардың нақты түрлерін қамтитынын атап кеткен жөн. Бұл тапсырманы шынайы етеді. Кейбір сарапшылардың пікірінше, жаңа шабуылдардың көпшілігі белгілі шабуылдардың нұсқалары болып табылады және белгілі шабуылдардың "сигнатурасы" жаңа нұсқаларды ұстау үшін жеткілікті болуы мүмкін. Оқытуға арналған деректер жиынтығында шабуылдарының 24 түрі бар, сонымен қатар қосымша 14 түрі тек тестілік деректерде ғана қамтылған.

Туынды функциялар.

Деректер жиынтығында қалыпты қосылыстарды шабуылдардан ажыратуға көмектесетін жоғары деңгейлі функциялар қолданылады. Туынды функциялардың бірнеше санаттары бар.

"same host" функциясы тек соңғы екі секунд ішіндегі ағымдағы қосылыстағы сияқты тағайындалу хосттардың қосылыстарын ғана тексереді және протоколдың мінез-құлқымен, қызмет көрсетумен және т. б. байланысты статистиканы есептейді.

Аналогты "same service " функциясы тек соңғы екі секундтағы ағымдағы қосылыстағымен бірдей қызметтерді ғана қарастырады.

"same host" және "same service" функциялары байланыс жазбаларында уақытқа негізделген трафик функциялары.

Кейбір зондтаушы шабуылдар, хостарды(немесе порттарды) екі секундтан астам уақыт аралығында сканерлейді, мысалы, бір минутына бір рет. Сондықтан қосылыстар туралы жазбалар тағайындау хосты бойынша сұрыпталды, ал функциялар уақыт терезесінің орнына бір хостпен жүзеге асырылатын 100 қосылымнан тұратын терезені пайдалана отырып жасалды. Бұл хост негізіндегі трафик деп аталатын функциялар жиынтығын береді.

DOS және зондтау шабуылдарының көпшілігіне қарағанда, R2L және U2R шабуылдарының жазбаларында бірізді қалыптың болмауы. DOS пен зондтау шабуылдары кейбір хосттармен өте қысқа уақыт ішінде көптеген қосылыстарды жүргізеді, бірақ R2L және U2R шабуылдары осы пакеттердің бөлігінде орнатылған және әдетте бір ғана қосылысты қамтиды.

Пакеттердегі құрылымсыз деректер порцияларын автоматты түрде алатын пайдалы алгоритмдері зерттеудің ашық сұрағы болып қала береді. Бұл деректер жиынтығында пән саласының білімі, деректер бөліктеріндегі күдікті мінез-құлықты іздейтін функцияларды қосу үшін пайдаланды, мысалы, жүйеге кірудің сәтсіз әрекеттерінің саны. Бұл функциялар "content" деп аталады.

Қосылым туралы жазбалар үшін анықталған функциялардың толық тізімі төмендегі үш кестеде келтірілген [19].

2.2 кесте – Жеке TCP қосылыстарының негізгі ерекшеліктері

функция атауы	сипаттамасы	түрі
duration	қосылым ұзындығы (секундтар саны)	үздіксіз
protocol_type	хаттаманың түрі, мысалы tcp, udp және т.б.	дискретті
service	тағайындалған пункттегі желілік қызмет, мысалы, http, telnet және т.б.	дискретті
src_bytes	көзден тағайындалған пунктке дейінгі деректер байттарының саны	үздіксіз
dst_bytes	тағайындалғаннан пункттен көзге дейінгі деректер байттарының саны	үздіксіз
flag	қосылымның қалыпты немесе қате күйі	дискретті
land	1 егер байланыс бірдей хосттан / порттан әлде хостқа / портқа болса; әйтпесе 0	дискретті
wrong_fragment	«қате» фрагменттер саны	үздіксіз
urgent	шұғыл пакеттер саны	үздіксіз

2.3 кесте – Пән саласымен келтірілген қосылыс шегіндегі мазмұн функциялары

функция атауы	сипаттамасы	түрі
hot	«қызу» индикаторлар саны	үздіксіз
num_failed_logins	сәтсіз кіру әрекеттері саны	үздіксіз

logged_in	1 сәтті кірген болса; әйтпесе 0	дискретті
num_compromised	«бұзылған» жағдайлардың саны	үздіксіз
root_shell	1, егер түбір қабықшасы алынған болса; 0 басқаша	дискретті
su_attempted	1 егер «su root» командасы әрекет етсе; әйтпесе 0	дискретті
num_root	«түбір» қатынау саны	үздіксіз
num_file_creations	файл құру операцияларының саны	үздіксіз
num_shells	қабықша кеңестерінің саны	үздіксіз
num_access_files	қол жеткізуді басқару файлдарына жүргізетін операциялардың саны	үздіксіз
num_outbound_cmds	ftp сеансында шығатын командалардың саны	үздіксіз
is_hot_login	1, егер логин «қызу» тізімге кірсе; әйтпесе 0	дискретті
is_guest_login	1, егер логин «қонақ» болса; әйтпесе 0	дискретті

2.4 кесте – Екі секундтық терезенің көмегімен есептелген трафик функциялары

функция атауы	сипаттамасы	түрі
count	соңғы екі секундтағы ағымдағы хостпен бірдей хостқа қосылу саны	үздіксіз
Ескерту: Келесі функциялар осы хост қосылымдарына қатысты.		
serror_rate	'SYN' 'қателері бар қосылымдардың %	үздіксіз
rerror_rate	«REJ» қатесі бар қосылыстардың %	үздіксіз
same_srv_rate	сол бір қызметке қосылудың %	үздіксіз
diff_srv_rate	әр түрлі қызметтерге қосылу %	үздіксіз
srv_count	соңғы екі секундтағы ағымдағы қосылыммен бірдей қызметке қосылу саны	үздіксіз
Ескерту: Төмендегі функциялар осы қызмет қосылымдарына жатады.		
srv_serror_rate	«SYN» қателері бар қосылымдардың %	үздіксіз
srv_rerror_rate	«REJ» қатесі бар қосылыстардың %	үздіксіз
srv_diff_host_rate	әртүрлі хосттарға қосылым %	үздіксіз

3 Нейрондық желіні құру және оқыту

Дипломдық жобаның бұл бөлімінде, Google Colaboratory сервисі көмегімен, желілік трафикті классификациялайтын жасанды нейрондық

желіні, Python бағдармалау тілінде құрастыру және оны оқыту жүзеге асырылады. Нейрожеліні оқытуда “KDD Cup 1999 Data Set” [20] деректер базасының 10 пайыздық нұсқасы қолданылады.

Бірінші кезекте мәліметтер базасын және оның функцияларының аттарын Google Colab сервисіне, “Linux”-тың “!wget” командасы арқылы жүктеуміз кажет (3.1 сурет).

```
!wget http://kdd.ics.uci.edu/databases/kddcup99/kddcup.names
!wget http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz

--2020-04-08 10:08:13--  http://kdd.ics.uci.edu/databases/kddcup99/kddcup.names
Resolving kdd.ics.uci.edu (kdd.ics.uci.edu)... 128.195.1.86
Connecting to kdd.ics.uci.edu (kdd.ics.uci.edu)|128.195.1.86|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1307 (1.3K)
Saving to: ‘kddcup.names’

kddcup.names        100%[===================>]   1.28K  --.-KB/s    in 0s

2020-04-08 10:08:13 (221 MB/s) - ‘kddcup.names’ saved [1307/1307]

--2020-04-08 10:08:18--  http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz
Resolving kdd.ics.uci.edu (kdd.ics.uci.edu)... 128.195.1.86
Connecting to kdd.ics.uci.edu (kdd.ics.uci.edu)|128.195.1.86|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2144903 (2.0M) [application/x-gzip]
Saving to: ‘kddcup.data_10_percent.gz’

kddcup.data_10_perc 100%[===================>]   2.04M  1.85MB/s    in 1.1s

2020-04-08 10:08:19 (1.85 MB/s) - ‘kddcup.data_10_percent.gz’ saved [2144903/2144903]
```

3.1 сурет — Дерекқор мен функциялар атауларының файлын жүктеу

PEP 8 стандартына сәйкес, барлық керекті кітапханалар, бағдарламалаудың басында импортталады. Ол “import” командасы арқылы жүзеге асырылады (3.2 сурет).

```
[ ] %tensorflow_version 2.x
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense
    from tensorflow.keras.callbacks import ModelCheckpoint
    from sklearn.preprocessing import StandardScaler
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    %matplotlib inline
```

3.2 сурет — Керекті кітапханаларды импорттау

Функциялардың атаулары, алдын ала сервиске жүктеген “kddcup.names” файлында сақталған, оны оқу үшін “read_csv” командасы шақырылады. Алайда бұл файлдың ішінде бізге қажет емес тақырыпат бар және оны елемеу үшін “header” параметріне “0” деген мән беруіміз қажет. Сонымен бірге, файлда функциялар атауларымен қатар олардың типі көрсетілген және олар бір-бірінен «:» символымен ажыратылады. Келешекте бізге “label” функциясы

нейрондық желіні оқытуда жауап ретінде қажет болады, ол трафиктің түрін көрсетеді, оны қосу үшін “append” командасы қолданылады. “features” айнымалысына функциялардың атаулары сақталады (3.3 сурет).

```
[ ] names = pd.read_csv('kddcup.names',
                        header=0,
                        names=['features', 'type'],
                        delimiter=':')
    names = names.append({'features': 'label'},
                         ignore_index=True)
    features = names['features']
```

3.3 сурет — Мәліметтер жиынының функциялар атауларын оқу

Ендігі кезекте “мәліметтер жиынына” да осындай ұқсас әрекеттер жасап, оған “train_data” айнымалысын береміз. Бұл жағдайда, файлда тақырыпат жоқ болғандықтан “header” параметріне “None” деген мән береміз. Функциялардың атауы ретінде, алдын ала дайындаған “features” айнымалысын көрсетеміз (3.4 сурет).

```
[ ] train_data = pd.read_csv('kddcup.data_10_percent.gz',
                             header=None,
                             names=features)
```

3.4 сурет — Оқытуға арналған мәліметтер жиынын оқу

Енді “train_data” айнымалысын шақырайық. Бізге, алдынғы терезеге құрылымдалған мәліметтер мен олардың атаулары яғни функциялардың атаулары(қызылмен көрсетілген) шығуы қажет. Мәліметтер жиыны 494021 қатардан және 42 бағаннан тұрады (3.5 сурет).

```
train_data
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_faile
0	0	tcp	http	SF	181	5450	0	0	0	0	
1	0	tcp	http	SF	239	486	0	0	0	0	
2	0	tcp	http	SF	235	1337	0	0	0	0	
3	0	tcp	http	SF	219	1337	0	0	0	0	
4	0	tcp	http	SF	217	2032	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
494016	0	tcp	http	SF	310	1881	0	0	0	0	
494017	0	tcp	http	SF	282	2286	0	0	0	0	
494018	0	tcp	http	SF	203	1200	0	0	0	0	
494019	0	tcp	http	SF	291	1200	0	0	0	0	
494020	0	tcp	http	SF	219	1234	0	0	0	0	

494021 rows × 42 columns

3.5 сурет — Оқытуға арналған мәліметтер жиынын құрылымы

Функциялар атаулары қызылмен белгіленген. Соңғы, 42-ші бағанға келетін болсақ, жоғарыда “append” функциясы арқылы қосқан “label” бағанын байқаймыз. Ол баған трафиктің түрін көрсетеді (3.6 сурет).

t_host_srv_rerror_rate	label
0.0	normal.
0.0	normal.
0.0	normal.
0.0	normal.
0.0	normal.
...	...
0.0	normal.
0.0	normal.
0.0	normal.
0.0	normal.
0.0	normal.

3.6 сурет — Трафик типінің функциясы

Нейрондық желіні оқыту үшін, мәліметтер жиынында тек қана сандық ақпарат болғаны дұрыс. Біздің жағдайда, кейбір бағандар, яғни функцияларымыз, деректердің жолдық типін қабылдап отыр. Оларды сандық түрге келтіру үшін алдымен жолды типті бағандардың уникалдық тізімін сәйкес айнымалыларға тіркеп құруымыз қажет (3.7 сурет).

```
[ ] list_of_protocol_types = sorted(list(set(train_data.iloc[:].values[:, 1])))
    list_of_services = sorted(list(set(train_data.iloc[:].values[:, 2])))
    list_of_flags = sorted(list(set(train_data.iloc[:].values[:, 3])))
    list_of_labels = sorted(list(set(train_data.iloc[:].values[:, 41])))
```

3.7 сурет — Жолды функциялар мәндерінің уникалды тізімін құру

Тізімнің біреуін ашып көрейік (3.8 сурет).

```
list_of_labels

['back.',
 'buffer_overflow.',
 'ftp_write.',
 'guess_passwd.',
 'imap.',
 'ipsweep.',
 'land.',
 'loadmodule.',
 'multihop.',
 'neptune.',
 'nmap.',
 'normal.',
 'perl.',
 'phf.',
 'pod.',
 'portsweep.',
 'rootkit.',
 'satan.',
 'smurf.',
 'spy.',
 'teardrop.',
 'warezclient.',
 'warezmaster.']
```

3.8 сурет — Трафик типтерінің функциясында кездесетін мәндер тізімі

Байқап отырғанымыздай, “list_of_labels” тізімі, мәліметтер жиынымыздың “label” қатарында кездесетін мәндерін қамтиды.

Енді біздің тізімдеріміз дайын. Бұл тізімдерде, сәйкес бағандарда кездесетін уникалды мәндер ғана жазылған, яғни бір рет жазылған мән қайталанбайды. Сонымен қатар тізімдер Python-ның “sorted” ішкі функциясы арқылы сұрыпталған.

Тізімді құрастырғаннан кейін одан “сөздік” жасаймыз. Сөздік дегеніміз, екі мәннен тұратын(яғни кілт(key) және мән(value)) құрылымдық дерек.

Сөздігіміз кілт ретінде тізімнің мәндерін қабылдайды ал мәні ретінде тізімдегі мәндердің позициялық номерін қабылдайды. Оны іске асыру үшін жаңа функция құрастырамыз (3.9 сурет).

```
[ ] def get_dictionary_from_list(n):
        unique_dict = {}
        for index, value in enumerate(n):
            unique_dict[value] = index + 1
        return unique_dict
```

3.9 сурет — Тізімдерден сөздік құратын функция

Желіні оқыту кезінде, мәліметтер жиынының “label” бағанының мәндері, жауаптар ретінде қолданылады, алайда біздің нейрондық желі, трафикті “жақсы” және “жаман” деп классификациялайтын болады, яғни бинарлы классификатор болып табылады. Сондықтан жауаптарымыз тек қана 0(“жақсы”) және 1(“жаман”) деген мәндерді қабылдау қажет. Барлығын ескере отырып, жауаптардың сөздігін құрастыру үшін басқа функция жасаймыз (3.10 сурет).

```
def get_classified_dict(classification_list, replaceable_word):
    dict_of_something = {}
    for item in classification_list:
        if item == replaceable_word:
            dict_of_something[item] = 0
        else:
            dict_of_something[item] = 1
    return dict_of_something
```

3.10 сурет — Трафик типтерің тізімінен сөздік құратын функция

Құрған функцияларымызды енді тізімдерге қолданайық және оларды сәйкес айнымалыларға сақтайық. Жауаптар сөздігін құрған кезде, функцияның екінші аргументі ретінде “normal.” жолын енгіземіз, бұл жағдайда “normal.” кілтінің мәні 0-ге тең болады қалған кілттердің мәні 1-ге тең болады (3.11 сурет).

```
dict_of_labels = get_classified_dict(list_of_labels, 'normal.')
dict_of_protocol_types = get_dictionary_from_list(list_of_protocol_types)
dict_of_services = get_dictionary_from_list(list_of_services)
dict_of_flags = get_dictionary_from_list(list_of_flags)
```

3.11 сурет — Тізімдерден сөздік құру

Біз жолды типін қабылдайтын бағандардың уникалды сөздіктерін құрдық. Ол сөздіктердің кілттері ретінде ауыстырылатын жол типті мәліметтер, ал сөздіктердің мәндері ретінде оларға сәйкес сандар бекітілген. Нейрондық желілер тек қана сандармен жұмыс істейтінін біле отырып, мәліметтер жиынындағы барлық жолды типін қабылдайтын деректерді сөздіктегі сәйкес сандарға ауыстырайық (3.12 сурет).

```
[ ] train_data.label = [dict_of_labels[item] for item in train_data.label]
    train_data.protocol_type = [dict_of_protocol_types[item] for item in train_data.protocol_type]
    train_data.service = [dict_of_services[item] for item in train_data.service]
    train_data.flag = [dict_of_flags[item] for item in train_data.flag]
    train_data = train_data.sample(frac=1).reset_index(drop=True)
    train_data
```

3.12 сурет — Жолды мәндерді сөздіктегі сәйкес мәндерге алмастыру

Бір ескеретін жайт, 3.12 сурет суретке назар аударсақ, жол типін қабылдайтын мәліметтерді сандық типіне ауыстырғаннан кейін біз мәліметтер жиынын араластырып жібердік. Бұны нейрондық желіні оқыту кезінде, валидациялық жиынның мәндері әртүрлі ақпарат қамту үшін жасаймыз.

Мәліметтер жиынының терезесін шақыратын болсақ барлық жол типті деректер сандық типке ауысқанын байқаймыз (3.13 сурет).

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	nu
0	0	1	15	10	1032	0	0	0	0	0	
1	0	1	15	10	1032	0	0	0	0	0	
2	0	2	46	6	0	0	0	0	0	0	
3	0	3	46	10	105	0	0	0	0	0	
4	0	2	46	6	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
494016	0	1	15	10	520	0	0	0	0	0	
494017	0	2	46	6	0	0	0	0	0	0	
494018	0	2	46	6	0	0	0	0	0	0	
494019	0	3	41	10	516	4	0	0	0	0	
494020	0	2	46	2	0	0	0	0	0	0	

494021 rows × 42 columns

3.13 сурет — Дерекқордың жаңа құрылымы

Осы жерге дейін біз мәліметтер жиынын керекті форматқа келтірдік. Ендігі кезекте біз нейрондық желіні оқытуға арналған жиынын және олардың жауаптарының жиынын бөліп “x_train” және “y_train” сәйкес айнымалыларға сақтамыз (3.14 сурет).

```
[ ] x_train = train_data.iloc[:, 0:41].values
    y_train = train_data.iloc[:, 41].values
```

3.14 сурет — Дерекқорды оқыту және жауаптар мәліметтер жиынына бөлу

Бақылап отырғанымыздай “x_train”, яғни оқытуға арналған жиыны 0-ден 41-ші бағанға(Python тілінің ерекшелігі, санақ жүйесін 0-ден бастайды және санап шығу кезінде соңғы элемент есепке алынбайды) дейін мәндерді қабылдайды, “y_train” жауаптар жиыны 41-ші баған мәндерін қабылдайды.

“x_train” айнымалысын шақырып көрейік (3.15 сурет).

```
x_train

array([[ 0.,  1., 15., ...,  0.,  0.,  0.],
       [ 0.,  1., 15., ...,  0.,  0.,  0.],
       [ 0.,  2., 46., ...,  1.,  0.,  0.],
       ...,
       [ 0.,  2., 46., ...,  1.,  0.,  0.],
       [ 0.,  3., 41., ...,  0.,  0.,  0.],
       [ 0.,  2., 46., ...,  0.,  1.,  1.]])
```

3.15 сурет — Оқытуға арналған мәліметтер жиынының форматы

3.15 сурет суретке қарасақ, оқытуға арналған жиынымыз “Numpy” массивы ретінде сақталған. Барлық пакеттердің параметрлері тік жақшамен ажыратылған, өз кезекте әрбір пакеттің әр параметрлері үтірмен бөлінген.

Енді біздің оқытуға арналған мәліметтер жиынымыз нейрондық желіні үйрету үшін дайын болғанымен, параметрлердің мәндері әртүрлі масштабта болып келеді, яғни кейбір параметрлер 0-ден 1-ге дейін ғана ауысса, басқа параметрлер үлкен аралықтағы сандарды қамтиды. Әртүрлі масштабтағы параметрлермен, нейрожелі тиімсіз оқытылады. Сол себептен, оқытуға арналған мәліметтер жиынымызды “scikit-learn” кітапханасының “StandardScaler” модулі арқылы стандарттайық (3.16 сурет).

```
[ ] scaler = StandardScaler()
    x_train = scaler.fit_transform(x_train)
```

3.16 сурет — Оқытуға арналған мәліметтер жиынын стандарттау

Оқытуға арналған жиынымызды қайта тексеріп көрейік (3.17 сурет).

```
x_train
```

```
array([[-0.06779172, -0.81154961, -0.69498244, ..., -0.46320239,
        -0.25203952, -0.249464  ],
       [-0.06779172, -0.81154961, -0.69498244, ..., -0.46320239,
        -0.25203952, -0.249464  ],
       [-0.06779172,  0.92575306,  1.5948145 , ...,  2.16202721,
        -0.25203952, -0.249464  ],
       ...,
       [-0.06779172,  0.92575306,  1.5948145 , ...,  2.16202721,
        -0.25203952, -0.249464  ],
       [-0.06779172,  2.66305573,  1.22549241, ..., -0.46320239,
        -0.25203952, -0.249464  ],
       [-0.06779172,  0.92575306,  1.5948145 , ..., -0.46320239,
         4.08467564,  4.09571547]])
```

3.17 сурет — Оқытуға арналған мәліметтер жиынының жаңа форматы

Нейрондық желіні оқыту үшін барлық деректеріміз дайын. Ендігі кезекте нейрондық желіні құруымыз қажет. Нейрондық желіміздің архитектурасы ретінде толықбайланысты перцептронды таңдаймыз. “Keras” кітапханасында мұндай қабаттар “Dense” деп аталады. Желіміз, жалпы алғанда, бес қабаттан тұрады, бірінші қабат кіріс қабаты, келесі үш қабат жасырын қабаты және соңғы қабат шығыс қабаты. Бірінші қабат, оқытуға арналған мәліметтер жиынына сәйкес 41 кірістен тұрады. Жасырын үш қабаты 45, 29 және 17 нейрондардан тұрады, белсендіру функциясы ретінде “ReLU” қолданылады. Нейронды желіміз бинарлы классификатор болғандықтан шығыс қабатымыз бір ғана нейроннан тұрады, белсендіру функциясы ретінде “sigmoid” қолданылады (3.18 сурет).

```
[ ] model = Sequential()
    model.add(Dense(45, activation = 'relu', input_dim = 41))
    model.add(Dense(29, activation = 'relu'))
    model.add(Dense(17, activation = 'relu'))
    model.add(Dense(1, activation = 'sigmoid'))
```

3.18 сурет — Нейрондық желіні құру

Нейрондық желіні құрғаннан соң оны оның компиляциясын іске асырайық. Оңтайлағыш ретінде “adam”, шығын функциясы ретінде “binary_crossentropy”, метрика ретінде “accurasy” таңдаймыз (3.19 сурет).

```
[ ] model.compile(optimizer = 'adam',
                  loss = 'binary_crossentropy',
                  metrics = ['accuracy'])
```

3.19 сурет — Құрылған нейрондық желінің компиляциясы

Моделімізге “summary” функциясын шақыратын болсақ, құрған нейрондық желі туралы ақпарат аламыз (3.20 сурет).

```
model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 45)                1890
_________________________________________________________________
dense_5 (Dense)              (None, 29)                1334
_________________________________________________________________
dense_6 (Dense)              (None, 17)                510
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 18
=================================================================
Total params: 3,752
Trainable params: 3,752
Non-trainable params: 0
_________________________________________________________________
```

3.20 сурет — Құрылған нейрондық желі жайлы ақпарат

Желіні үйрету кезінде, оның ең жақсы нәтижені сақтау үшін “.h5” форматындағы файлды құрастырайық және бұл файлды “callback” параметріне тіркейік. Ең жақсы нәтиже валидациялық дәлдік негізінде сақталады, бұны іске асыру үшін “monitor” параметріне “val_accurasy” деген мәнді береміз (3.21 сурет).

```
[ ] model_save_path = 'best_model.h5'
    checkpoint_callback = ModelCheckpoint(model_save_path,
                                          monitor='val_accuracy',
                                          save_best_only=True,
                                          verbose=1)
```

3.21 сурет — Колбэк құру

Бұл кезеңде нейрондық желіну оқыту үшін барлық іс-әрекеттер орындалды. Ендігі кезекте желіні оқытудың өзі ғана қалды. Бұны іске асыру үшін “model.fit” методын шақырайық және оның параметрлері ретінде “x_train” оқытуға арналған жиынын, “y_train” жауаптар жиынын көрсетейік, дәуірлер санын 100-ге, топтама көлемін 256-ға, валидация мөлшерін 20%-ға теңестірейік сонымен қатар алдыңғы кезеңде құрастырған колбэкты көрсетейік (3.22 сурет).

```
[ ] history = model.fit(x_train,
                        y_train,
                        epochs=100,
                        batch_size=256,
                        validation_split=0.2,
                        callbacks=[checkpoint_callback])
```

3.22 сурет — Оқыту параметрлерін беру және оқыту

Google Colab сервисінде жоғарыдағы кодты орындаған соң нақты уақыт тәртіптемесінде нейрондық желіні оқыту жөніндегі есептеме пайда болады (3.23 сурет).

```
Epoch 1/100
1541/1544 [==============================>.] - ETA: 0s - loss: 0.0202 - accuracy: 0.9957
Epoch 00001: val_accuracy improved from -inf to 0.99893, saving model to best_model.h5
1544/1544 [==============================] - 5s 3ms/step - loss: 0.0202 - accuracy: 0.9957 - val_loss: 0.0040 - val_accuracy: 0.9989
Epoch 2/100
1535/1544 [==============================>.] - ETA: 0s - loss: 0.0046 - accuracy: 0.9989
Epoch 00002: val_accuracy improved from 0.99893 to 0.99916, saving model to best_model.h5
1544/1544 [==============================] - 4s 3ms/step - loss: 0.0046 - accuracy: 0.9989 - val_loss: 0.0039 - val_accuracy: 0.9992
Epoch 3/100
1542/1544 [==============================>.] - ETA: 0s - loss: 0.0042 - accuracy: 0.9990
Epoch 00003: val_accuracy did not improve from 0.99916
1544/1544 [==============================] - 5s 3ms/step - loss: 0.0042 - accuracy: 0.9990 - val_loss: 0.0042 - val_accuracy: 0.9990
Epoch 4/100
1538/1544 [==============================>.] - ETA: 0s - loss: 0.0033 - accuracy: 0.9992
Epoch 00004: val_accuracy did not improve from 0.99916
1544/1544 [==============================] - 5s 3ms/step - loss: 0.0033 - accuracy: 0.9992 - val_loss: 0.0038 - val_accuracy: 0.9991
Epoch 5/100
1531/1544 [==============================>.] - ETA: 0s - loss: 0.0030 - accuracy: 0.9992
```

3.23 сурет — Нейрондық желіні оқыту есептемесі

Бұл есептеменің көкпен белгіленген аймақта дәуірлердің кезеңдері және сонымен қатар орындалған топтамалар саны көрсетіледі. Қызылмен белгіленген аймақта оқытудың бір дәуірге кеткен уақыты , бір топтамамен оқыту уақыты, жоғалту мөлшері, оқытуға арналған жиынындағы дұрыс жауаптардың үлесі, тексеруге арналған жиынындағы жоғалтулар, тексеруге арналған жиынындағы дұрыс жауаптардың үлесі. Жасылмен белгіленген аймақта val_accurasy(тексеруге арналған жиынындағы дұрыс жауаптардың үлесі) мәні жақсарған соң, алдын ала дайындаған “best_model.h5” файлына нейрондық желіні сақтағанын көруге болады.

```
Epoch 98/100
1532/1544 [==============================>.] - ETA: 0s - loss: 6.6651e-04 - accuracy: 0.9998
Epoch 00098: val_accuracy did not improve from 0.99960
1544/1544 [==============================] - 5s 3ms/step - loss: 6.6529e-04 - accuracy: 0.9998 - val_loss: 0.0029 - val_accuracy: 0.9995
Epoch 99/100
1532/1544 [==============================>.] - ETA: 0s - loss: 7.1177e-04 - accuracy: 0.9998
Epoch 00099: val_accuracy did not improve from 0.99960
1544/1544 [==============================] - 5s 3ms/step - loss: 7.1329e-04 - accuracy: 0.9998 - val_loss: 0.0025 - val_accuracy: 0.9995
Epoch 100/100
1528/1544 [==============================>.] - ETA: 0s - loss: 6.9701e-04 - accuracy: 0.9998
Epoch 00100: val_accuracy did not improve from 0.99960
1544/1544 [==============================] - 5s 3ms/step - loss: 6.9075e-04 - accuracy: 0.9998 - val_loss: 0.0036 - val_accuracy: 0.9995
```

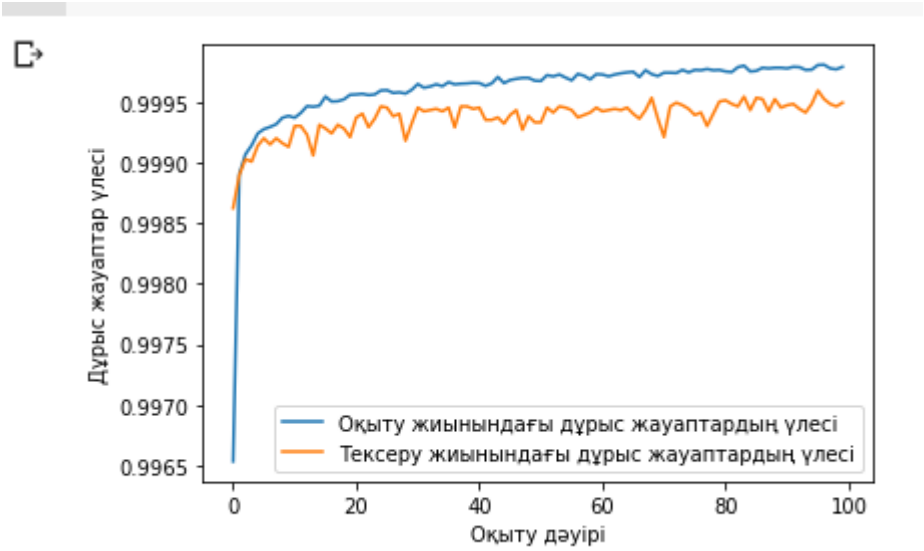
3.24 сурет — Оқытудың аяқталуы

Жүзінші дәуір аяқталғаннан соң нейрондық желіні оқыту тоқтатылады (3.24 сурет). Желіні оқыту жайлы графигін алу үшін “Matplotlib” кітапханасын қолданайық (3.24 сурет).

```
[28] plt.plot(history.history['accuracy'],
             label='Оқыту жиынындағы дұрыс жауаптардың үлесі')
     plt.plot(history.history['val_accuracy'],
             label='Тексеру жиынындағы дұрыс жауаптардың үлесі')
     plt.xlabel('Оқыту дәуірі')
     plt.ylabel('Дұрыс жауаптар үлесі')
     plt.legend()
     plt.show()
```

3.24 сурет — Оқыту жайлы график тұрғызу

Осы кодты орындаған соң, консольде нейрондық желіні оқыту туралы график шығуы қажет (3.25 сурет).



3.25 сурет — Нейрожелінің оқытылуының графигі

Жоғарыдағы графикке қарасақ, нейрондық желіні оқыту кезінде, тексеруге арналған жиынындағы дурыс жауаптар үлесі әр дәуір сайын өсіп жатпайды, тіпті кейбір кезеңдерде, алдынғы дәуірмен салыстырғанда дұрыс жауаптардың үлесі әлдеқайда төмен шаманы қабылдайды. Сол себепті, тым көп дәуірлер саны нейрондық желінің “шамадан тыс оқытылу”(overfitting) процессіне әкеліп соғады.

Біздің желіміз 100 дәуір бойы оқытылды және ең жоғарғы дұрыс жауаптардың үлесі 96-шы дәуірде 99.96%-ды құрады. Бұл нейрондық желінің моделі “best_model.h5” файлында сақталды және оның салмақтарын желімізге қолдану үшін “load_weights” функциясын шақырамыз (3.26 сурет).

```
[ ] model.load_weights(model_save_path)
```

3.26 сурет — Колбэкте сақталған желінің салмақтарын жүктеу

Осы кезеңде біздің жасанды нейрондық желіміз оқытылған және қолдануға толығымен дайын. Алайда желіміздің жұмыс істеуінің дұрыстылығын тексеруіміз қажет. Желіні тексеру үшін, оны оқытуға қолданылған мәліметтерден тыс деректерді қолдануымыз қажет. “KDD Cup 1999 Data Set” дерекқорында арнайы тестілеуге арналған мәліметтер жиыны бар. Бұл мәліметтер жиыны, оқытуға арналған мәліметтер жиыны қарағанда өзгешелеу болып келеді. Оның ішінде оқытуға қатыспаған желілік шабуылдардың қосымша 14 типі бар. Бұл тапсырманы неғұрлым шынайылыққа жақындатады. Кейбір желілік шабуылдар сарапшылары жаңа шабуылдардың көпшілігі белгілі шабуылдардың нұсқалары деп санайды және белгілі шабуылдардың «нышандары» жаңа нұсқаларды анықтауға жеткілікті деп есептейді.

Оқытылған желімізді тестілеу жиынында тексеру үшін алдымен оны оқытуға арналған жиынымызды сияқты керекті форматқа келтіру керек. Бұның іске асырудың алгоритмі алдында жасағанымыздай. Келесі кезекте тестілеу жиынын “x_test” және “y_test” мәліметтер жиыны және олардың жауаптары ретінде бөлуіміз қажет.

Тестілеу жиынын керекті форматқа келтіргеннен соң нейрондық желімізді тексеріп көрейік. Ол үшін моделімізге “evaluate” функциясын шақырамыз (3.27 сурет).

```
[ ] model.evaluate(x_test, y_test, verbose=1)

9720/9720 [==============================] - 17s 2ms/step - loss: 1.7513 - accuracy: 0.9267
[1.7513374090194702, 0.9266852736473083]
```

3.27 сурет — Оқытылған нейрондық желіні тестілеу жиынында тексеру

Біздің нейрондық желіміз, тестілеуге арналған жиынында, жоғарыдағы суретте көріп отырғанымыздай, 92.67% жағдайда трафикті дұрыс талдай алды. Бұл жиында оқытуда қатыспаған жаңа типті желілік шабуылдар қатысқанын ескере отырсақ, нейрондық желіміз өте жоғары нәтиже көрсетті деуге болады.

Нейрожелілік жүйелердің басқа жүйелермен салыстырғандағы артықшылықтардың бірі, кез келген жаңа типтегі деректерге бейімделінуі. Осы артықшылықты қолдану, желілік трафикті талдаудың тиімділігін бірнеше есе арттырады.

Бұл дипломдық жұмыста нейрондық желіміз “KDD Cup 1999 Data Set” дерекқорының 10% нұсқасында оқытылды. Толық нұсқасы қосылулардың 5 млн жазбасын қамтиды. Оқытылған нейрондық желімізді дерекқордың толық нұсқасында тексеріп көрейік (3.28 сурет).

```
[ ] model.evaluate(x_test, y_test, verbose=1)

153076/153076 [==============================] - 344s 2ms/step - loss: 0.0049 - accuracy: 0.9995
[0.0049059116281569, 0.999539852142334]
```

3.28 сурет — Оқытылған нейрондық желіні мәліметтер жиынының толық нұсқасында тестілеу

Толық нұсқалы мәліметтер жиынында нейрондық желіміз трафиктің 99.95%-ын дұрыс талдай алды.

4 Өмір тіршілігінің қауіпсіздігі

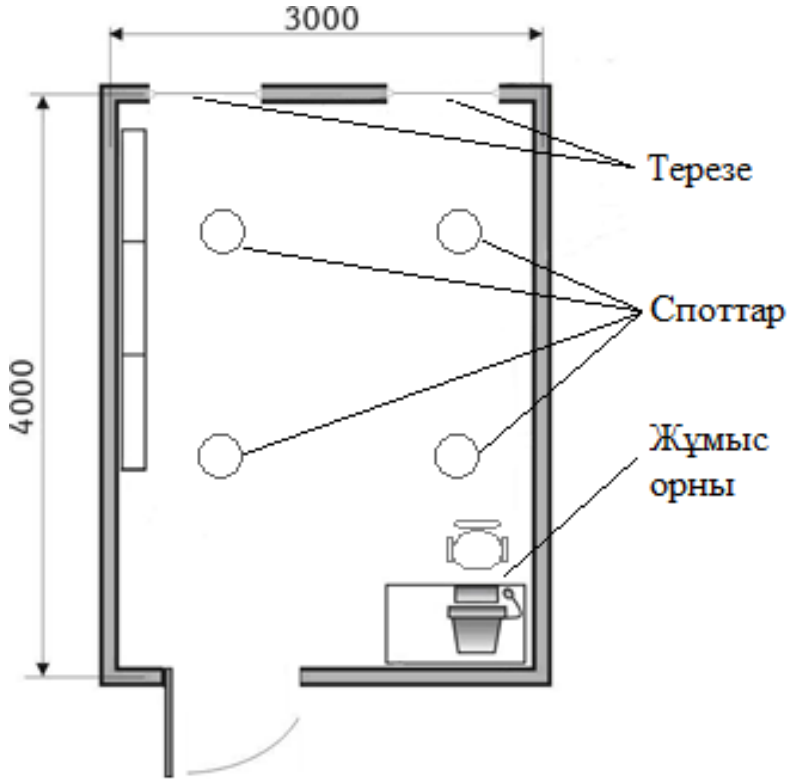
4.1 Еңбек шарттарын талдау

Бұл дипломдық жұмыстың мақсаты Python бағдарламалау тілі арқылы нейрондық желіні құру болып табылады. Жоба стандартты перифериясы бар дербес компьютерімен жабдықталған жұмыс орнында жасалды. Жабдықтардың толық техникалық жарамдылығы, оның электр, өрт қауіпсіздігі, жұмысқа ыңғайлы болу үшін оңтайлы микроклимат және эргономикалық жұмыс орны талап етіледі. Жұмыс кезінде инженерге әсер ететін факторлар: компьютер жұмысының шуы, монитордың жыпылықтауы, ұзақ уақыт отыру және адамның көру жүйесіне шамадан тыс көп күш түсіру, сондай-ақ жерге тұйықтау жүйесінің немесе қауіпсіздік техникасының бұзылуы, экранның немесе бөлменің жеткіліксіз жарықтандыруы кезінде жұмыс істеу.

4.1.1 Жұмыс орнына сипаттама беру. Бөлмені талдау:

- жұмыс бөлмесі бірінші қабатта орналасқан;
- бөлме түрі: үй бөлмесі;
- бөлменің өлшемі 4x3x2.6 (сәйкесінше ұзындығы, ені, биіктігі);
- жасанды жарықтандыру: споттар - 4 дана;
- бөлмеде екі терезе бар, терезенің өлшемдері 1,6x1,3 (терезенің шамадан тыс жарықтығынан қорғау үшін перделер қолданылады).

Бөлменің жоспары 4.1 суретте көрсетілген.



4.1 сурет – Жұмыс бөлмесінің жоспары

Жұмыс орнында компьютермен жұмыс істеуге арналған былғары компьютерлік кресло бар. Креслоның арқасы белді ұстап тұратын иілген пішінді. Арқасының биіктігі-750 мм, ені-520 мм, тереңдігі-520 мм.

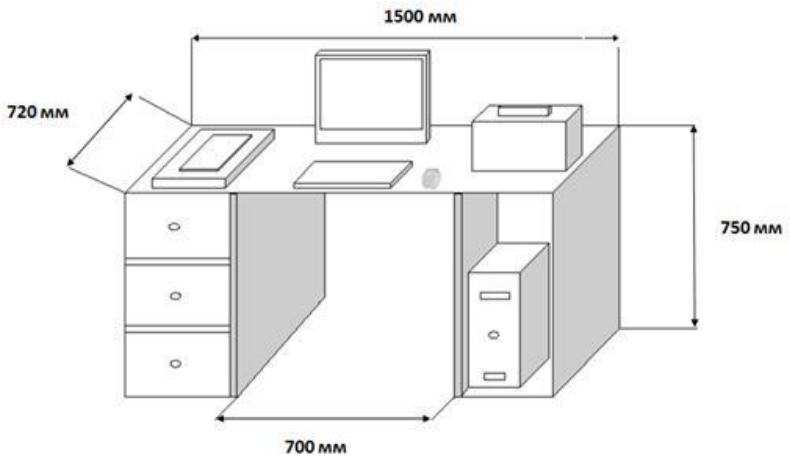
Оңтайлы жұмыс орны.

Ақпаратты дисплейге шығару құрылғыларымен жабдықталған жұмыс орнын әзірлеу Ақпараттық жүйелер саласындағы эргономикалық жұмыс орнын жобалаудың маңызды міндеті болып табылады.

Жұмыс орнының барлық құрамдас бөліктерінің бір-біріне қатысты орналасуы психологиялық, әлеуметтік, физиологиялық және т.б. сияқты әр түрлі талаптарға сәйкестендіріледі.

Мамандық сипаты айтарлықтай әсер етеді. Мысалы: бағдарламашы-әзірлеушінің жұмыс орнын ұйымдастыру үшін жабдықты оңтайлы орналастыру, еркін қозғалатын орын және т.б. сияқты жағдайлар ескеріледі.

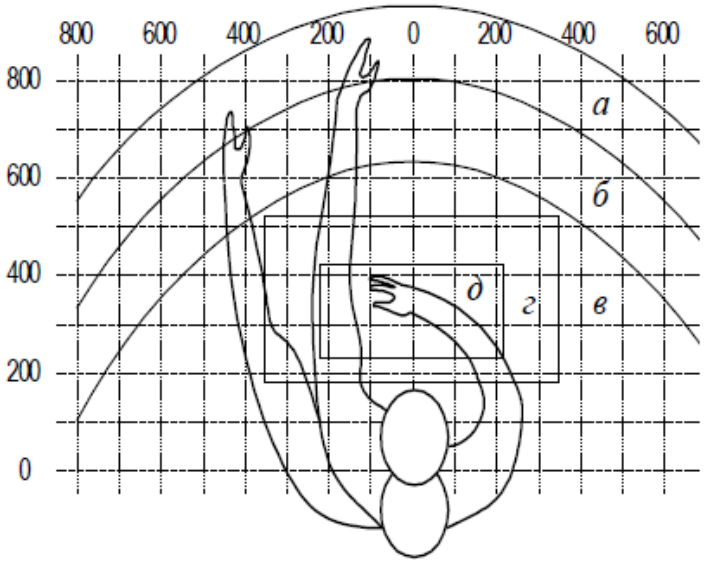
Бағдарламашыға ыңғайлы жұмыс орнын жобалаудың негізгі аспектілері келесідей сипаттамалар орын алады: жұмыс орнының элементтерінің реттелуі, жұмыс беті, жұмыс бетінің биіктігі, аяқ кеңістігі, жұмыс бойынша құжаттардың орналасуына қойылатын талаптардың болуы(құжат стендінің болуы, оның өлшемдері, құжаттарды әртүрлі орналастыру мүмкіндігі, пайдаланушының көзінен экранға, құжатқа, пернетақтаға дейінгі қашықтық және т.б.), орындықтың сипаттамалары, жұмыс үстелінің бетіне қойылатын талаптар, жұмыс орнының реттелетін элементтері (4.2 сурет).



4.2 сурет – Жұмыс кеңістігін жоспарлау

Бағдарламашы өз жұмысын отырған күйде орындайды және оның жұмыс кеңістігінің негізгі құрамдас бөліктері - үстел мен орындық.

Бағдарламалаушының жұмыс істейтін кейіпі барынша ыңғайлы, шаршауды тудырмауы тиіс. Жұмыс істеуде жиі қолданылатын нәрсе жеңіл қол жеткізу аймағында орналасады. Яғни, жұмыс орнын жоспарлау еңбек заттарының, құжаттамалардың ұтымды орналасуын көздеуі тиіс (4.3 сурет).

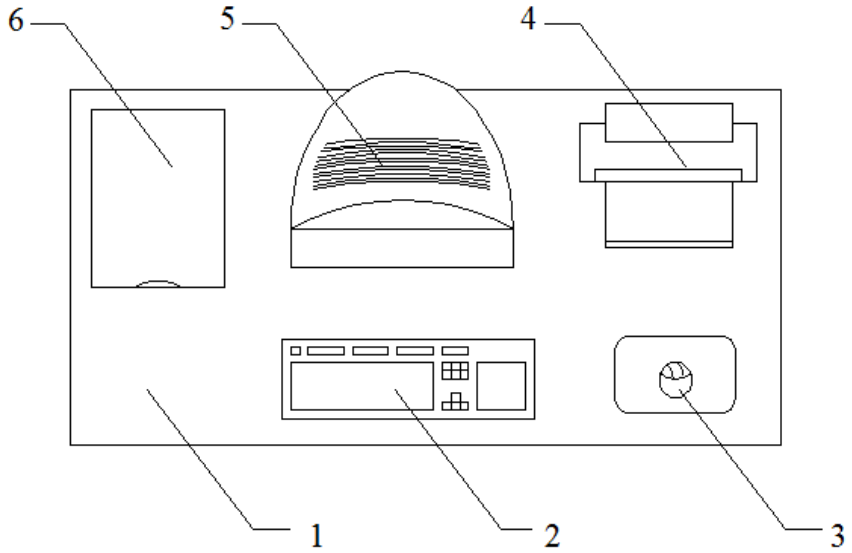


4.3 сурет – Программистің жұмыс кеңістігіндегі қол жетімділік аймағы. а - барынша қол жеткізу аймағы; б - қол созылған кезде саусақтардың қол жеткізу аймағы; в - алақанның жеңіл қол жеткізу аймағы; г-күш қажет ететін жұмыс істеуге арналған оңтайлы кеңістік; д - ұсақ жұмыс істеуге арналған оңтайлы кеңістік.

Бағдарламалаушының жұмыс орнындағы құрылғылардың оңтайлы орналсуы:

- дисплей а аймағында (орталықта) орналасады;

- жүйелік блок үстелдің арнайы қуысына орналастырылады;
- пернетақта-г/д аймағында;
- "тышқан" - оң жақта;
- а/б аймағындағы сканер (сол жақта);
принтер а аймағында (оң жақта).



4.4 сурет – Негізгі периферияльық құрылғыларды орналастыру

4.4-сурет дербес компьютердің негізгі және қосымша құрамдастарының әзірлеушінің жұмыс орнында болжамды орналасуын қамтиды:
- 1 – жұмыс үстелінің беті;
- 2 – пернетақта;
- 3 – компьютерлік «тышқан»;
- 4 – принтер;
- 5 – монитор;
- 6 – сканер.

Программист-әзірлеушінің жұмыс орындығы көтеру-бұрылыс механизмімен жабдықталады. Орындықтың биіктігін реттеу 400-500 мм шегінде жүргізіледі. Жұмыс орындығының тереңдігі 380 мм – ден астам, ал ені 400 мм – ден астам, арқаның тірек бетінің биіктігі 300 мм – ден кем емес, ені 380 мм-ден кем емес болу талап. Орындықтың арқасының көлбеу бұрышы 90 -110° шегінде өзгертіледі.

4.1.2 Жұмыс орнындағы жарықтандыру. Бөлмелер мен жиһаздардың бояуы; көзге жағымды болу және жақсы көңіл-күйді қамтамасыз ету үшін белгілі бір талаптарға сәйкестендіріледі.

Жұмыс станциялары орналасқан үй-жайларда осындай шағылысу коэффициенттері:
- төбе үшін - 60-70%;
- қабырғалар үшін - 40-50%;

- еден үшін - ~ 30%;

- жұмыс жиһазы және басқа беттер үшін - 30-40%.

Өндірістік жарықтандыруды дұрыс жобалау қажеттілігі, жеткіліксіз жарықтандыруда көру жүйесіне күш түсіруі, зейіннің әлсіреуі, уақытынан бұрын шаршағандықтан пайда болады, ал шамадан тыс жарықтандыру – көздің соқырлығын, ауырсынуын тудырады.

Өндірістік жарықтандыруды дұрыс жобалау және орнату арқасында жұмыс жағдайлары жақсарады, шаршағыштық төмендейді, жұмыскерге жағымды психологиялық әсер етеді, жұмыс орнында жарақат алу мүмкіндігін азайтуға көмектеседі, сонымен қатар еңбек қауіпсіздігін арттырады.

Жарықтың дұрыс орналастырылмауы өткір көлеңкелер тудырады, жұмысшының бағытсыздануына әкеледі. Мұндай жағдайлар апатқа немесе кәсіби ауруға алып келеді, бұл жарықтандыруды дұрыс есептеуді одан да маңызды етеді.

Жарықтандырудың 3 түрі бар: жасанды, табиғи және біріктірілген (табиғи жасанды бірге).

Жасанды жарықтандыру жұмыста түнде және күндіз табиғи жарықтың нормаланған мәндерін қамтамасыз ету мүмкін болмаған кезде қолданылады (бұлтты ауа-райында немесе күндізгі қысқа уақыт болған кезде).

Табиғи жарықтандыру нормалары жеткіліксіз болған кезде жасанды жарықтандыруды толықтыратын жарықтандыру біріктірілген жарықтандыру деп аталады. Ол авариялық, күзеттік, жұмысқа арналған және эвакуациялық болып бөлінеді. Сонымен қатар, жұмысқа арналған жарықтандыру жалпы немесе біріктірілген болып бөлінеді.

Біріктірілген жарықтандыруды пайдаланған кезде жергілікті жарықтандыру біртіндеп жалпы жарықтандыруға қосылады.

ҚР ЕЖ 2.04-104-2012 [21] сәйкес есептеу орталықтарының үй-жайларында аралас жарықтандыру жүйесін қолданылады.

Жоғары дәлдіктегі жұмыстарды орындау үшін (0,3 және 0,5 мм арасындағы объектінің ажыратуға арналған ең аз мөлшері) – ТЖК шамасы (табиғи жарықтандыру коэффициенті) кемінде 1,5% құрауы тиіс. Дәлдігі орташа көру жұмыстары үшін (объектінің көлемі 0,5-тен 1,0 мм-ге дейін) ТЖК 1% - дан кем болмайды. Жасанды жарықтандыруды жасау үшін ДРЛ немесе ЛБ типті люминесцентті шамдар қолданылады, олар жұп бойынша жұмыс үстелдерінің үстінде біркелкі орналасқан жарық көздеріне біріктіріледі.

Жарықтандырудың негізгі гигиеналық талабы - барлық көрінетін аймақты жеткілікті түрде біркелкі жарықтандыруды қамтамасыз ету, атап айтқанда бөлменің жарықтандыру дәрежесі мен компьютер дисплейінің жарықтығы мүмкіндігінше біркелкі болуын қамтамасыз ету. өйткені периферияльық көру аймағында жарқырау көздің талуын едәуір арттырады және тез шаршауға әкеледі.

Компьютерлер орнатылған жерлерде жарықтандыру талаптары төменде көрсетілген: жоғары дәлдікті қажет ететін көрнекі жұмыстарды орындау үшін

жалпы жарықтандыру 300 люкс, ал аралас жарық үшін 750 люкс, орташа дәлдікті қажет ететін көрнекі жұмыстарға ұқсас талаптар - 200 люкс және 300 люкс

4.1.3 Жұмыс орнындағы микроклимат параметрлері. Есептеу техникасы бөлмедегі температура көрсеткіштерінің өсуіне және салыстырмалы ылғалдылықтың көрсеткішінің төмендеуіне әкеліп соғатын жылудың айтарлықтай бөлінуінің көзі болып табылады. Сәйкесінше, компьютерлермен жабдықталған үй-жайларда микроклиматтың қалыпты көрсеткіштерін сақталынады.

Өндірістік үй-жайлардағы микроклимат организмнің температурасына, ауа жылдамдығының, ылғалдылықтың және қыздырылған беттердің жылу сәулеленуінің бір мезгілдегі әсерінен ықпал етеді. Сондықтан әр түрлі үй-жайлардың микроклиматы сыртқы метеожағдайлардың өзгеруіне, тәулік, жыл уақытына, жылыту, желдету жүйелерінің ерекшеліктеріне, сондай-ақ өндірістік процестің ерекшеліктеріне байланысты. СанЕмН 2.2.4.548-96 [22] санитарлық нормалары, микроклиматтың қажетті шамалар өлшемдерінің шарттарын ұсынады. Бұл нормалар жыл мезгіліне, еңбек процесінің сипатына және өндірістік үй-жайдың сипатына байланысты өзгереді, 4.1-кестеде көрсетілген.

Жұмысшылар тұратын кеңістіктегі таза ауа нормасы 4.2-кестеде келтірілген. Орталық қызметкерлері орналасатын үй-жайлардың көлемі, бір ауысымда жұмыс істейтін қызметкерлердің ең көп санын ескере отырып, бір адамға кемінде 19,5 м3 құрайды.

4.1 кесте – Компьютерлер орнатылған бөлмелер үшін микроклимат параметрлері

Жыл мезгілі	Микроклимат параметрі	Мәні
Суық	Бөлме температурасы	22…24°С
	Салыстырмалы ылғалдылық	40…60%
	Ауа жылдамдығы	0,1м/с дейін
Жылы	Бөлме температурасы	23…25°С
	Салыстырмалы ылғалдылық	40…60%
	Ауа жылдамдығы	0,1…0,2м/с

4.2 кесте – Компьютерлер орналасқан үй-жайларды таза ауамен қамтамасыз ету нормалары

Бөлменің сипаттамасы	Бөлмеге келетін таза ауаның көлемді шығыны, бір адамға/м3/сағ
Көлемі бір адамға 20 м3 дейін	30-дан кем емес
Бір адамға 20 ... 40 м3	20-дан кем емес
Бір адамға 40 м3 астам	Табиғи желдету

Қолайлы жағдай жасау үшін техникалық әдістер(желдету, ауаны баптау, жылыту жүйесі), сондай-ақ ұйымдастыру құралдары(Еңбек және демалыс кезектесуі, жыл және тәулік уақытына байланысты жұмыстарды жүргізуді ұтымды ұйымдастыру) қолданылады.

4.1.4 Діріл және шу. Шудың әсерінен концентрация, зейін азаяды, физиологиялық функциялар бұзылады, жанасқыштық нашарлайды, шаршау жинақталады. Мұның бәрі адамның жұмыс жағдайына, оның өнімділігіне, қауіпсіздігі мен өндіріс сапасына теріс әсер етеді. Ұзақ шу әсерімен жұмыс істейтін адамдар тітіркене бастайды, бас айналуы басталады, есте сақтау қабілеті төмендейді және шаршағыштық жоғарылайды.

Әзірлеуші-бағдарламашылардың жұмыс орнындағы шу деңгейі 50 дБ және ақпаратты өңдеу залдарындағы 65 дБ аспауы тиіс. Шу деңгейін төмендету үшін бөлменің қабырғалары мен төбесін қаптауға арналған дыбыс жұтатын материалдар қолданылады. Жұмыс орнындағы діріл деңгейін арнайы дірілді оқшаулағышқа қондырғыны орнату арқылы төмендетеді. 4.3-кесте қауіпсіздік пен жұмыс қабілеттілікті қамтамасыз ету үшін қажетті еңбек санатына байланысты дыбыстың шекті деңгейлерін қамтиды.

4.3 кесте – Жұмыс орнындағы дыбыстың шекті деңгейі, дБ.

Еңбек қауырттылығы	Еңбек ауырлығының категориясы			
	I. Жеңіл	II. Орташа	III. Ауыр	IV. Өте ауыр
I. Аз	80	80	75	75
II. Орташа	70	70	65	65
III. Қауыртты	60	60	-	-
IV. Өте қауыртты	50	50	-	-

4.1.5 Иондаушы және электромагниттік сәулелену. Инженер-программистің жұмыс орнында инфрақызыл және ультракүлгін сәулеленудің қарқындылығы 10 және 100 мВт / м2 аралығында жатыр, ал рентген сәулесінің ең жоғары деңгейі кейде ғана 10 мкбэр / сағ-тан аса құрайды.

4.4-кестеде компьютер дисплейінен иондамайтын электромагниттік сәулеленудің рұқсат етілген мәндері көрсетілген.

4.4 кесте – Иондамайтын электромагниттік сәулелену параметрлерінің рұқсат етілген мәндері [23]

Параметр атауы	Рұқсат етілген мәндер
Видеомонитор бетінен 50см қашықтықта электромагниттік өрістің электр құраушысының кернеулігі	10В/м
Видеомонитордан 50см қашықтықта электромагниттік өрістің магниттік құраушысының кернеулігі	0,3А/м
Электростатикалық өрістің қарқындылығы аспауы керек: ересек пайдаланушылар үшін, мектепке дейінгі	20кВ/м 15кВ/м

мекемелердің балалары және орта арнаулы және жоғары оқу орындарының студенттері үшін	

4.2 Есептік бөлім

4.2.1 Қорғаныштық жерге тұйықтауды есептеу. Қорғаныштық жерге тұйықтау есептеу жұмысы МЕМСТ 12.1.030-81 [24] бойынша жүргізілді. Бұл стандарт жиілігі 400 Гц дейінгі тұрақты және ауыспалы токтың электр қондырғыларын қорғаныштық жерге тұйықтау және нөлдеу үшін бағытталған және электр қауіпсіздігін қамтамасыз етуге қойылатын талаптарды белгілейді Қорғаныштық жерге тұйықтау, адамды оқшауланыу бұзылған электрқұрылғылардың беткі қабатымен жанасқан кезде электр тогының соғуынан сақтайды. Жерге тұйықтау электр қондырғыларының металл бөлшектерін жерге (жерге тұйықтау құрылғысына) қосу арқылы жүзеге асырылады. Егер адам қуатталған, бірақ жерге тұйықталған өткізбейтін бөлікке кездейсоқ тисе, зақымдану мүмкіндігі қауіпсіз мәнге дейін азаяды.

4.5-кестеде стандартты электрлік сипаттамалар көрсетілген.

4.5 кесте – Электрлік сипаттамалар

Айнымалы ток көзі			Тұрақты ток көзі	
Кіріс ток күші, А	Кіріс кернеуі, V	Жиілік Гц	Кіріс ток күші, А	Кіріс кернеуі, V
2.5	220	50	3	36 - 72

Ғимараттың сыртқы жағынан тік электродтарды контуры бойынша орналастыра отырып, жерге тұйықтағыштың құрылысы болжанады. Вертикальды жерге тұйықтағыш ретінде диаметрі 15 мм және ұзындығы 3 м болат өзекшелерді қарастырамыз; электродтардың жоғарғы ұштары 0,7 м тереңдікте, оларға сол болаттан жасалған өзекшелі типті көлденең электродтар дәнекерленеді (4.5 сурет). Шешім келесідей:

- жерге тұйықтау құрылғысын орнату ережелеріне сәйкес тұйықтау құрылғысының кедергісі R3=4 Ом, есептік R3 = 3 Ом-қабылдаймыз;
- тік электродтар арасындағы қашықтық-3 м;
- табиғи RH болмаған кезде жасанды жерге тұйықтау кедергісі R3 = 3 Ом;
- көлденең және тік электродтар үшін пневматикалық коэффициенттерді есепке ала отырып, топырақтың есептік үлестік кедергісі тиісінше: $p_{рг}$ = 273 Ом·м; $p_{рв}$ =105 Ом·м.

Өзек түріндегі бір тік электродтың ағу кедергісі:

$$R_{ОВЭ} = \frac{\rho_{РВ}}{2 \cdot \pi \cdot l_В} \cdot \left(\ln\frac{2 \cdot l_В}{d} + \frac{1}{2} \cdot \ln\frac{4 \cdot t_В}{4 \cdot t_В}\frac{+l_В}{-l_В}\right) \quad (4.1)$$

мұндағы $l_В$ - бір электродтың ұзындығы (тік). $l_В$ = 3 м;

$t_В$ - бұл электродты толтыру тереңдігі. $t_В$ = 0,7 + 1,5 = 2,2 м;

d - өзектің диаметрі

$$R_{ОВЭ} = \frac{105}{2 \cdot \pi \cdot 3} \cdot \left(\ln\frac{2 \cdot 3}{15 \cdot 10^{-3}} + \frac{1}{2} \cdot \ln\frac{4 \cdot 2{,}2 + 3}{4 \cdot 2, -3}\right) = 35{,}35 \text{ Ом}$$

Тік жерге тұйықтау өткізгіштерінің саны:

$$N = \frac{R_{ОВЭ}}{K_{ИВ} \cdot R_И} \quad (4.2)$$

мұндағы $K_{ИВ}$ $l_В$ мен жерлендіргіштердің арасындағы қатынас кезіндегі жерлендіргіштің қолдану коэффициенті

$$N = \frac{35{,}35}{0{,}73 \cdot 3} \approx 16{,}14 \approx 16$$

Көлденең электродтардың ағуының есептік кедергі:

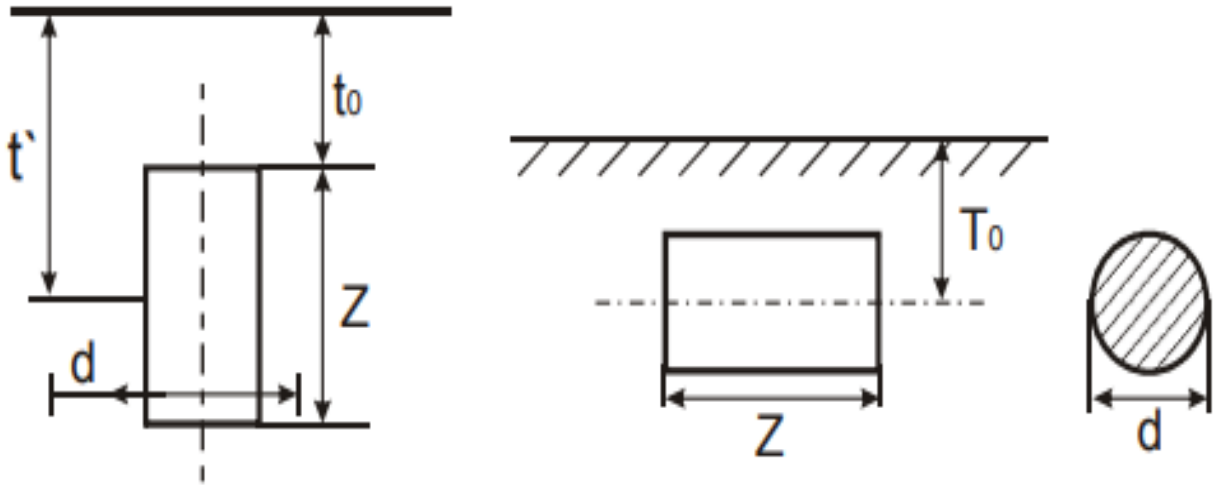
$$R_{РГЭ} = \frac{\rho_{РГ}}{K_{ИГ} \cdot 2 \cdot \pi \cdot l_Г} \cdot \ln\frac{2 \cdot {l_Г}^2}{d \cdot t_Г} \quad (4.3)$$

мұндағы $K_{ИГ}$-пайдалану коэффициенті. $K_{ИГ}$ =0,77;

$l_Г$-көлденең электродтардың ұзындығы. $l_Г$ = 4х3 = 12 м;

$t_Г$-толтыру тереңдігі. $t_Г$ = 0,7 м;

d-электродтардың диаметрі. d = 15 мм.



4.5 сурет – Тік (сол) және көлденең (оң) жерлендіргіштердің орналасуы

$$R_{РГЭ} = \frac{273}{0{,}77 \cdot 2 \cdot \pi \cdot 12} \cdot \ln \frac{2 \cdot 12^2}{15 \cdot 10^{-3} \cdot 0{,}7} = 44{,}8 \text{ Ом}$$

Тік электродтардың соңғы саны:

$$N = \frac{R_{ОВЭ} \cdot (R_{РГЭ} - R_И)}{К_{ИВ} \cdot R_{РГЭ} \cdot R_И} = \frac{35{,}35 \cdot (44{,}8 - 3)}{0{,}73 \cdot 44{,}8 \cdot 3} = 15{,}1 \approx 16 \qquad (4.4)$$

Жерлендіргіштердің жалпы кедергісі мына формула бойынша есептеледі:

$$R_{общ} = \frac{R_{ОВЭ} \cdot R_{РГЭ}}{R_{ОВЭ} + К_{ИГ} + R_{РГЭ} + К_{ИВ} \cdot N} \qquad (4.5)$$

Жалпы кедергіні есептейік:

$$R_{общ} = \frac{35{,}35 \cdot 44{,}8}{35{,}35 + 0{,}77 + 44{,}8 + 0{,}73 \cdot 16} = 2{,}8 \text{ Ом}$$

4.2.2 Су және көбік өрт сөндіру қондырғыларын есептеу. Су және көбік өрт сөндіру қондырғыларын есептеу жұмысы [25] әдістемелік нұсқау бойынша жүргізілді және барлық мәндер сол нұсқаулықтың кестелерінен алынды.

Суландырғыш(генератор) арқылы судың, көбік түзгіш ерітіндісінің Q, л/с есептік шығынын мына формула бойынша анықталады

$$Q = k * \sqrt{H} \qquad (4.6)$$

мұндағы k – 2.2-кесте [25] бойынша қабылданатын суландырғыштың (генератордың) өнімділік коэффициенті 0,2-ге тең;

H – 2.2-кесте бойынша қабылданатын суландырғыш(генератор) еркін су күші 5 м тең.

$$Q = 0{,}2 * \sqrt{5} = 0{,}45 \text{ л/с}$$

Үй-жайдағы спринклерлік қондырғы үшін судың, көбік түзгіш ерітіндісінің Q, л/с шығыны мынадай формула бойынша анықталады

$$Q = a * qn \tag{4.7}$$

мұндағы a — стеллаждың бір мезгілде суландырылатын бөлігінің есептік ұзындығы 15 м тең деп қабылданады [25];

qn — суландыру қарқындылығы 2.1-кесте [25] бойынша қабылданады, қатты жанатын материалдар үшін 0,45 л/см^2 тең.

$$Q = 15 * 0{,}45 = 6{,}75 \text{ л/с}$$

Құбырлардың есептеу учаскесіндегі су күшінің шығыны H, м, мынадай формула бойынша анықталады:

$$H = \frac{Q^2}{B} \tag{4.8}$$

мұндағы Q-құбырдың есептік учаскесіндегі судың, көбік түзуші ерітіндісінің шығыны, л / с;

B-құбырдың сипаттамасы, мынадай формула бойынша анықталады:

$$B = \frac{k}{l} \tag{4.9}$$

мұндағы k - коэффициент, 2.3-кесте [25] бойынша қабылданады, 0,18 тең;

l - құбырдың есептік учаскесінің ұзындығы, 4 м тең.

$$B = \frac{0{,}18}{4} = 0{,}045 \text{ м};$$

$$H = \frac{6{,}75^2}{0{,}045} = 1012{,}5 \text{ м}.$$

Н қондырғыларын басқару тораптарындағы су күшінің ысырабы мынадай формула бойынша анықталады:

$$H = \varepsilon_{ц} * Q^2 \quad (4.10)$$

мұндағы $\varepsilon_{ц}$ — басқару торабында қысым шығындарының коэффициенті 2.4-кесте [25] бойынша қабылданады, спринклерлік су толтырғыш қондырғы үшін 3,02* 10^3 тең;

Q — басқару торабы арқылы судың, көбік түзуші ерітіндінің есептік шығыны, л / с.

$$H = 3{,}02 * 103 * 0.452 = 611{,}55 \text{ м}$$

Көлемдік өрт сөндіру кезінде V көбік түзгіш ерітіндісінің көлемі мына формула бойынша анықталады:

$$V = k1 * \frac{V}{k2} \quad (4.11)$$

мұндағы k1 - қатты жанғыш материалдар үшін 2.5 кестесі [25] бойынша қабылданатын көбік бүліну коэффициенті 3-ке тең;

V-қорғалатын үй-жайдың көлемі, м3;

k2-көбік еселігі, химиялық көбік үшін 5-ке тең.

$$V = 3 * \frac{31{,}2}{5} = 18{,}72 \text{ м}^3$$

Бір мезгілде жұмыс істейтін n көбік генераторларының саны мынадай формула бойынша анықталады

$$n = \frac{V}{Q * t} \quad (4.11)$$

мұндағы t - Орташа еселі көбікпен қондырғының жұмыс істеу ұзақтығы, мин, 2.5 кесте [25] бойынша қабылданады, 25 тең.

$$n = \frac{18.72}{0.45 * 25} = 1{,}64 \approx 2$$

Осылайша, жалпы жұмыс көлемі V = 31,2 м³ болатын жұмыс бөлмесін өртке қарсы қамтамасыз ету үшін екі көбік генераторы (спринклерлер) жеткілікті.

5 Техникалық-экономикалық бөлім

5.1 Жұмыстың мақсаты

Жобаның басты мақсаты Python бағдарламалау тілінде желілік трафикті классификациялайтын нейрондық желіні құру болып табылады. Бұл нейрондық желі, желілік трафикті тез және тиімді талдауына, сонымен қатар жаңа желілік шабуылдар типіне жылдам бейімделінуіне негізделген. Қазіргі таңда желіге қосылған технологиялардың саны жылдан жылға өсуде және пайдаланушылардың қауіпсіздігін қамтамасыз ету үшін нейрондық желілерді қолдану таптырмайтын амал болып табылады.

5.2 Қаржылық жоспары

Қаржы жоспары жалпы күрделі шығындар, кірістер, операциялық шығындарды, пайданы және өзін-өзі ақтау кезеңінің есептеуін қарастырады.

Жобаны іске асыру барысында қолданылған құралдардың аты, бағасы, саны және техникалық сипаттамалары 5.1-кестеде көрсетілген.

Ал 5.2-кестеде жобаға тартылған қызметкерлер туралы мәліметтер және олардың жалақысы туралы тізімі берілген.

5.1 кесте – Техникалық құрылғыларға кететін шығындар

Атауы	Сипаттамасы	Саны	Бір данасы үшін бағасы, теңге	Жалпы бағасы, теңге
Жүйелік блок	Intel® Core™ i7-3770 CPU @ 3.4GHz	1	150000	150000
Монитор	BenQ FP93G X	1	25000	25000
Тышқан	A4Tech Bloody V8	1	8000	8000
Пернетақта	USB Genius SlimStar 320	1	4500	4500
Бағдарламалық қамтамасыз ету	MS Windows 10 Professional	1	60000	60000
Жалпы бағасы				247500

Жабдықтарға баға ҚҚС-сыз есептемей келтірілген.

5.2 кесте – Жобаға тартылған қызметкерлер туралы мәліметтер және олардың жалақысы

Атауы	Саны	Жалақысы, теңге
Инженер - бағдарламалаушы	1	285000
Барлығы	1	285000

5.2.1 Жобаны іске асыру мерзімі. Бағдарламалық жасақтама өнімін жобалау мен әзірлеу кезеңдерден тұрады және мынадай жұмыс түрлерін қамтиды:

- 1-кезең – тапсырманы қою, қажетті ақпаратты жинау, дипломдық жобаның құрылымын жасау;
- 2-кезең – бағдарламалауға қажетті материалдарды жинау;
- 3-кезең – бағдарламалау құралдарын таңдау;
- 4-кезең – бағдарламалау;
- 5-кезең – презентацияны әзірлеу;
- 6-кезең – тексеру және есеп беру.

Жоба әзірлеу кестесі 5.3-кестеде келтірілген.

5.3 кесте – Жобаны іске асыру кезеңдері мен мерзімі

Кезең атауы		Жұмыстың жасалауы апталарға бөлінген									
		1	2	3	4	5	6	7	8	9	10
1-кезең	Тапсырма қойлуы										
	Әдебиеттерді іріктеу және талдау										
	Жобаның құрылымын әзірлеу										
2-кезең	Материалдарды жинау										
3-кезең	Бағдарламалық құралдарды таңдау										
4-кезең	Бағдарламалау										
5-кезең	Презентацияны әзірлеу										
6-кезең	Тексеру және есеп беру										

5.2.2 Жалақы қорын есептеу. Жұмысшы еңбегінен бөлек, техникалық құрылғылар және программалық қамтамасыздандырулар бастапқы қаржылай салымды қажет етеді. Осы қаржылай салымның негізінде жобаның соңғы бағасы есептеледі. Жобаны жүзеге асыру барысындағы жұмсалған қаржы келесі формуламен анықталады:

$$С = ЕАҚ + Сн + А + Э + ЖШ \quad (5)$$

мұндағы ЕАҚ – еңбек ақының қоры;
Сн – әлеуметтік салық;
А – амортизациялық аударым;
Э – электр энергиясына жұмсалған қаражат;
ЖШ – жалпы шығын.

Жалақы қоры – барлық қызметкерлер үшін негізгі және қосымша жалақыдан қалыптасқан және белгілі бір мерзімге еңбек ақысын төлеуге арналған жалпы шығындар болып табылады және 5.1 формуласымен анықталады:

$$ЕАҚ = ЖК_{н} + ЖК_{кос} \quad (5.1)$$

мұндағы $ЖК_{н}$– негізгі жалақы;
$ЖК_{кос}$ – қосымша жалақы.

БӨ-ді әзірлеу кезеңінде жұмысшылар біркелкі емес түрде жұмыс істейді, бұл үшін орташа күндік жалақыны есептеу керек, содан кейін жалақының жалпы мөлшерін.

Әр қызметкердің орташа күндік жалақысын формула бойынша есептеледі:

$$D = \frac{ЖК_{а}}{К_{ж}} \quad (5.2)$$

мұндағы $ЖК_{а}$ – айлық жалақы, теңге;
$К_{ж}$ – бір айдағы жұмыс күндерінің саны (22 күн – бес күндік жұмыс аптасы).

$$D = \frac{285000}{22} = 12955(\text{тенге/күн})$$

Бір сағатқа еңбекақысы формула бойынша есептеледі: күнге

$$Н = \frac{ЖК_{а}}{(К_{ж} \cdot С_{ж})} \quad (5.3)$$

мұндағы $ЖК_а$ – айлық жалақы, теңге;

$К_ж$ – бір айдағы жұмыс күндерінің саны (22 күн – бес күндік жұмыс аптасы);

$С_ж$ – жұмыс күнінің ұзақтығы, сағат, $C_ж$= 8 сағат.

$$D = \frac{285000}{22 \cdot 8} = 1619 \left(\frac{\text{теңге}}{\text{сағ}}\right)$$

Жұмыстың әрбір түріне арналған циклдің ұзақтығы формуламен анықталады:

$$t_n = \frac{T}{q_n \cdot z \cdot K} \qquad (5.4)$$

мұндағы T – кезеңін еңбек қарқындылығы, норма-сағат;

q_n – орындаушылар саны;

z – жұмыс күнінің ұзақтығы, сағат z =8;

K – уақыт нормаларына сәйкестік коэффициенті, K =1,1.

Алынған t_n мән үлкен бүтін бөлікке қарай ықшамдаймыз.

Тапсырма қою:

$$t_1 = \frac{5}{1 \cdot 8 \cdot 1.1} \approx 1(\text{күн})$$

Әдебиеттерді іріктеу және зерттеу:

$$t_2 = \frac{25}{1 \cdot 8 \cdot 1.1} \approx 3(\text{күн})$$

Жоба құрылымын әзірлеу:

$$t_3 = \frac{8}{1 \cdot 8 \cdot 1.1} \approx 1(\text{күн})$$

Материалдарды жинау:

$$t_4 = \frac{7}{1 \cdot 8 \cdot 1.1} \approx 1(\text{күн})$$

Бағдарламалық құралдарды таңдау:

$$t_5 = \frac{15}{1 \cdot 8 \cdot 1.1} \approx 2(\text{күн})$$

Бағдарламалау:

$$t_5 = \frac{110}{1 \cdot 8 \cdot 1.1} \approx 13(\text{күн})$$

Әзірлеуші, жетекші, жоба презентациясын дайындау:

$$t_7 = \frac{8}{1 \cdot 8 \cdot 1.1} \approx 1(\text{күн})$$

Әзірлеуші, жетекші, тексеру және есеп беру:

$$t_8 = \frac{10}{1 \cdot 8 \cdot 1.1} \approx 2(\text{күн})$$

Негізгі жалақы бойынша шығындарды есептеудің жиынтық нәтижелері 5.4-кестеде көрсетілген.

5.4 кесте – Негізгі жалақы бойынша шығындарды есептеудің жиынтық нәтижелері

Жұмыс атауы	Орындаушы	Еңбек қарқындылығы			Еңбек жалақысы, сағатына теңге	Жалақы мөлшері,теңге
		Норма-сағат	Жалпы еңбек сыйымдылығының% -ы	Цикл уақыты,күн		
Тапсырма қою	Инженер - бағдарламалаушы	5	2,70	1	1619	8095
Әдебиеттерді іріктеу және зерттеу	Инженер - бағдарламалаушы	25	13,51	3	1619	40475
Жобаның құрылымын әзірлеу	Инженер - бағдарламалаушы	8	2,70	1	1619	12952
Материалдарды жинау	Инженер - бағдарламалаушы	7	3,78	1	1619	11333
Бағдарламалық құралдарды таңдау	Инженер - бағдарламалаушы	15	8,11	2	1619	24285

Бағдарламалау	Инженер - бағдарламалаушы	110	59,46	13	1619	178090
Жоба презентациясы н дайындау	Инженер - бағдарламалаушы	8	4,32	1	1619	12952
Тексеру және есеп беру	Инженер - бағдарламалаушы	10	5,41	2	1619	16190
Барлығы		188	100	24		304372

Қосымша жалақы негізгі жалақының 10%-ын құрайды және 5.5 формула бойынша есептеледі:

$$ЖК_{кос} = ЖК_{нег} \cdot 0,1 \quad (5.5)$$

$$ЖК_{кос} = 304372 \cdot 0,1 = 30437,2(\text{тенге})$$

Осылайша, жалақының жалпы қоры болады:

$$ЕАҚ = 304372 + 30437,2 = 334809,2(\text{тенге})$$

5.2.3 Әлеуметтік салық бойынша шығындарды есептеу. Әлеуметтік салық қызметкердің табысынан 9,5% (ҚР Салық кодексінің 358-бабы 1-тармағы) болып табылады және 5.6 формуласына сәйкес есептеледі:

$$С_{н} = (ЕАҚ - ЗЖ) \cdot 0,095 \quad (5.6)$$

Онда ЗЖ – зейнетақы жарналары, 10% құрайды және әлеуметтік салық төленуге жатпайды.

$$ЗЖ = ЕАҚ \cdot 0,1 \quad (5.7)$$

$$ЗЖ = 334809,2 \cdot 0,1 = 33480,92(\text{тенге})$$

$$Сн = (334809,2 - 33480,92) \cdot 0,095 = 28626,2(\text{тенге})$$

Әлеуметтік аударымдар ЕАҚ-ның 3,5% құрайды:

$$Ш_{АА} = (ЕАК - ЗК) \cdot 3,5\% \quad (5.9)$$

$$Ш_{АА} = (334809,2 - 33480,92) \cdot 0,035 = 10546,5(\text{тенге})$$

Міндетті әлеуметтік сақтандыруға арналған аударымдар (ЕАҚ-ның 1,5%):

$$Ш_{МАСАА} = ЕАК \cdot 1{,}5\% \quad (5.10)$$

$Ш_{МАСАА} = 334809{,}2 \cdot 0{,}015 = 5022{,}138$ теңге

5.2.4 Амортизациялық аударымдарды есептеу. Амортизациялық аударымдар 5.11 формуласы бойынша есептеледі:

$$A_i = \frac{H_A \cdot C_{тар} \cdot N}{100 \cdot 12 \cdot n} \quad (5.11)$$

мұндағы H_A – амортизация нормасы;

$C_{тар}$ – жабдықтың бастапқы құны;

N – жұмыс орындауына керек күндері саны;

n – жұмыс айындағы күндер саны.

Компьютерлік техника мен бағдарламалық қамсыздандыруға арналған H_A-ның құнсыздануының жалпы құнын 40% құрайды.

Жабдықтар мен пайдаланылған бағдарламалық жасақтаманың амортизациялық шығыны болады:

$$A_1 = (25 \cdot 150000 \cdot 24)/(100 \cdot 12 \cdot 22) = 30409(тг);$$

$$A_2 = (25 \cdot 25000 \cdot 24)/(100 \cdot 12 \cdot 22) = 568(тг);$$

$$A_3 = (25 \cdot 8000 \cdot 24)/(100 \cdot 12 \cdot 22) = 182(тг);$$

$$A_4 = (25 \cdot 4500 \cdot 24)/(100 \cdot 12 \cdot 22) = 102(тг);$$

$$A_5 = (25 \cdot 60000 \cdot 24)/(100 \cdot 12 \cdot 22) = 1364(тг);$$

$$A = 30409 + 568 + 182 + 102 + 1364 = 5625(тг).$$

5.2.5 Электр энергиясының шығындарын есептеу. Электр жабдықтарын өндіру кезінде электр энергиясының құнын есептеу қажет. Өндірістік қажеттіліктерге арналған электр энергиясының құны жабдықтар мен қосымша қажеттілік үшін электр энергиясының құнын қамтиды.

$$Э = З_{эл.эн.кур} + З_{кос.каж} \quad (5.12)$$

мұндағы $З_{эл.эн.кур}$ – жабдықтардың электр тұтыну құны;

$З_{кос.каж}$ – қосымша қажеттілікке арналған энергия шығыны.

Жабдықтардың электр энергиясын тұтыну 5.14 формуласы бойынша есептеледі:

$$З_{эл.эн.кур} = W \cdot T \cdot S \cdot К_{кол} \quad (5.13)$$

мұндағы W – энергияны тұтыну, Вт;

T – жабдықтарды пайдалану сағаттарының саны;

S – киловатт-сағат электр энергиясының құны, (1кВтсағ=24 теңге);

$К_{кол}$ – қолдану коэффиценті, ($К_{кол}$=0,9).

W = 90Вт = 0.09кВт (компьютердің қуаты);

T = 24·5=120 сағат;

S = 24 теңге.

Негізгі жабдықтарға арналған электр қуатының құны:

$$З_{эл.энер} = 0.09 \cdot 120 \cdot 24 \cdot 0.9 = 233{,}3(теңге)$$

Қосымша қажеттілікке арналған шығындар жабдықтың өзіндік құнын 5% мөлшерінде жиынтық индикатор бойынша қабылданады:

$$З_{кос.каж} = 0{,}05 \cdot З_{эл.эн.кур} \quad (5.14)$$

$$З_{кос.каж} = 0.05 \cdot 233{,}3 = 11{,}7(тг)$$

Электр энергиясының жалпы құны:

$$Э = 233{,}3 + 11{,}7 = 245(тг)$$

5.2.6 Үстеме шығындар. Жалпы шығындар келесі формула бойынша есептеледі:

$$ЖШ = (ЕАҚ + Сн + А + Э) \cdot 0{,}5 \quad (5.15)$$

5.15 формуласына сәйкес үстеме шығыстар:

$$ЖШ = (334809{,}2 + 28626{,}2 + 5625 + 245) \cdot 0.5 = 184652{,}7(тг)$$

Осылайша, 5 формуласына сәйкес, бағдарламалық жасақтама өнімін жасаудың жалпы құны:

$$С = 334809{,}2 + 28626{,}2 + 5625 + 245 + 184652{,}7 = 553958{,}1(тг)$$

Жүйенің имитациялық моделін құру құны 5.5-кестеде көрсетілген.

5.5 кесте – Жүйенің имитациялық моделін құру құны

Шығын атауы	Құны, теңге	Шығын мөлшері, %
Жалақы қоры	334809,2	60,43944
Әлеуметтік салық	28626,2	5,167575
Амортизациялық аударымдар	5625	1,01542
Электр қуаты шығындары	245	0,044227
Есептік шығындар	184652,7	33,33333
Жалпы құны	553958,1	100

Жүйенің имитациялық моделін құру құнының диаграммасы 5.1-суретте көрсетілген.

5.1 сурет – Жүйенің имитациялық моделін құру құнының құрылымы

5.2.7 Жүзеге асыру құны. Бағдарламалық өнімді жүзеге асыру бағасы оның құндылығы мен пайдасынан тұрады:

$$Жб = С + П \quad (5.16)$$

мұндағы Сб – өнімнің құны;
П – пайда.

Бастапқы бағаны анықтағанда, бағдарламалық өнімді іске асыру үшін кірістілік деңгейін (20%) белгілеу керек:

$$Цп = Сб \times \left(1 + \frac{P}{100}\right) \quad (5.17)$$

мұндағы Р – рентабельділік (20%).

$$Цп = 553958 \cdot (1 + 0.2) = 664749{,}72(тг)$$

Енді іске асыру бағасы қосылған құн салығын (ҚҚС) ескере отырып есептейміз:

$$Жб = Цп + ҚҚС \quad (5.18)$$

мұндағы ҚҚС – қосылған құн салығы.

ҚҚС келесі формула бойынша есептеледі:

$$ҚҚС = Ц_р \cdot 0.12 \quad (5.19)$$

$$ҚҚС = 664749{,}72 \cdot 0.12 = 79769{,}9664(тг)$$

$$Жб = 664749{,}72 + 79769{,}9664 = 744519{,}6864(тг)$$

5.2.8 Экономика бөлімі бойынша қорытынды. Желілік трафикті классификациялайтын нейрондық желіні әзірлеудің өзіндік құны 553958 теңгені құрады. Соның ең үлкен 60,4% бөлігі жалақы қорын 5,1%-ы әлеуметтік салығын, 1%-ы амортизациялық аударымдарын, 0,04%-ы электр қуаты шығындарын, 33,3%-ы жалпы шығындар құрайды. Бұл технологияның жүзеге асыру құны 744519тг тең. Осы таңда, нейрондық желілердің сан алуан жетістіктерін ескере кетсек бұндай шығын қысқа мерзімде ақталады.

Қорытынды

Бұл дипломдық жұмыста желілік трафикті талдайтын нейрондық желі Python бағдарламалау тілінде құрастырылып, “KDD Cup 1999 Data Data Set” дерекқорының 500000 жазбаларын құрайтын 10%-дық нұсқасында оқытылды. Нәтижесінде нейрондық желі оқытуға кірмеген трафик түрлері бар тестілеуге арналған мәліметтер жиынында трафиктің 92,7%-ын дұрыс талдай алды. Дерекқордың 5млн жазбасын құрайтын толық нұсқасында желінің жұмысының дәлдігі 99,95% құрады.

Әзірленген трафикті талдау әдісі Қазақстанда аналогы жоқ, сонымен қатар басқа бағдарламалық жасақтамалардан айырмашылығы, нейрондық желілер негізінде жасалған бағдарламалар жаңа типті деректерге бейімделінуі.

Сондай-ақ, бұл жұмыста еңбекті қорғау бойынша есептер жасалды, онда қорғаныштық жерге тұйықтау және өрт қондырғыларының есебі жасалды.

Жоба үшін техникалық-экономикалық негіздеме орындалды, бұл оның экономикалық тиімді екенін және салыстырмалы түрде қысқа мерзімде өтеуге қабілетті екенін көрсетеді. Бағдарламаның функционалдығын арттыру мүмкіндігі қол жеткізуге болатын экономикалық тиімділікті айтарлықтай арттырады.

Әдебиеттер тізімі

1 Swaroop C. H. A Byte of Python [Электронный ресурс]. — URL: https://python.swaroopch.com/ (дата обращения: 17.02.2020).

2 Нейронные сети [Электронный ресурс]. — URL: https://www.asozykin.ru/courses/nnpython (дата обращения: 17.02.2020).

3 pandas Documentation [Электронный ресурс]. — URL: https://pandas.pydata.org/docs/pandas.pdf (дата обращения: 17.02.2020).

4 Keras Documentation [Электронный ресурс]. — URL: https://keras.io (дата обращения: 01.09.2019).

5 Matplotlib Documentation [Электронный ресурс]. — URL: https://matplotlib.org/Matplotlib.pdf (дата обращения: 17.02.2020).

6 Боршевников А. Е. Сетевые атаки. Виды. Способы борьбы [Текст] // Современные тенденции технических наук: материалы Междунар. науч. конф. (г. Уфа, октябрь 2011 г.). — Уфа: Лето, 2011. — С. 8-13. — URL https://moluch.ru/conf/tech/archive/5/1115/ (дата обращения: 14.03.2020).

7 Удалённые сетевые атаки [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Удалённые_сетевые_атаки (дата обращения: 15.03.2020).

8 Нейронные сети, перцептрон [Электронный ресурс]. – URL: https://neerc.ifmo.ru/wiki/index.php?title=Нейронные_сети,_перцептрон (дата обращения: 16.03.2020).

9 Ахтёров А.В., Кирильченко А.А. Основы теоретической робототехники. Искусственные нейронные сети. [Электронный ресурс]. –

URL: https://keldysh.ru/papers/2008/prep02/prep2008_02.html (дата обращения: 17.03.2020).

10 Искусственный_нейрон [Электронный ресурс]. – URL: https://vlab.wikia.org/ru/wiki/Искусственный_нейрон (дата обращения: 19.03.2020).

11 Практики реализации нейронных сетей [Электронный ресурс]. – URL: https://neerc.ifmo.ru/wiki/index.php?title=Практики_реализации_нейронных_сетей (дата обращения: 20.03.2020).

12 Созыкин А.В. Обзор методов обучения глубоких нейронных сетей // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 3. С. 28–59. DOI: 10.14529/cmse170303.

13 Python [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Python (дата обращения: 21.03.2020).

14 TensorFlow [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/TensorFlow (дата обращения: 22.03.2020).

15 Keras [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://en.wikipedia.org/wiki/Keras (дата обращения: 23.03.2020).

16 NumPy [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/NumPy (дата обращения: 24.03.2020).

17 Pandas [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Pandas (дата обращения: 25.03.2020).

18 Matplotlib [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Matplotlib (дата обращения: 26.03.2020).

19 [Электронный ресурс]. – URL: http://kdd.ics.uci.edu/databases/kddcup99/task.html (дата обращения: 1.04.2020).

20 [Электронный ресурс]. – URL: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (дата обращения: 1.04.2020).

21 ҚР ЕЖ 2.04-104-2012* Табиғи және жасанды жарықтандыру. Қазақстан Республикасы Ұлттық экономика министрлігінің Құрылыс, тұрғын үй-коммуналдық шаруашылық істері және жер ресурстарын басқару комитеті. – Астана, 2018

22 СанПиН 2.2.4.548-96 Гигиенические требования к микроклимату производственных помещений. – Москва: Минздрав России (2001 г.). Дата актуализации: 01.02.2020.

23 СанПиН 2.2.2.542-96 Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы. – Москва: Минздрав России (1996 г.). Дата актуализации: 01.02.2020

24 ГОСТ 12.1.030-81 Система стандартов безопасности труда. Электробезопасность. Защитное заземление, зануление. Министерство

монтажных и специальных строительных работ СССР. Дата актуализации: 01.02.2020

25 Абикенова А.А., Методические указания к выполнению раздела «Пожарная профилактика» - Алматы: АИЭС, 2009.

26 Жандаулетова, Ф. Р. Охрана труда: учебник для вузов / Ф.Р. Жандаулетова, Т.Е. Хакимжанов, Т.С. Санатова; МОН РК, НАО АУЭС. – Алматы: АУЭС, 2019. - 399 с.

27 Базылов Қ.Б., Алибаева С.А., Нурмагамбетова С. С. Бітіруші жұмысының экономикалық бөлімі үшін әдістемелік нұсқаулар. 050719 – Радиотехника, электроника және телекоммуникация мамандығының барлық оқу түрінің студенттеріне арналған. Алматы 2009.

А қосымшасы

```
!wget http://kdd.ics.uci.edu/databases/kddcup99/kddcup.names
!wget http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz

# %tensorflow_version 2.x
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import ModelCheckpoint
from sklearn.preprocessing import StandardScaler
from google.colab import files
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# %matplotlib inline

names = pd.read_csv('kddcup.names',
                    header=0,
                    names=['features', 'type'],
                    delimiter=':')
names = names.append({'features': 'label'},
                     ignore_index=True)
features = names['features']
```

```
train_data = pd.read_csv('kddcup.data_10_percent.gz',
                         header=None,
                         names=features)
train_data

list_of_protocol_types = sorted(list(set(train_data.iloc[:].values[:, 1])))
list_of_services = sorted(list(set(train_data.iloc[:].values[:, 2])))
list_of_flags = sorted(list(set(train_data.iloc[:].values[:, 3])))
list_of_labels = sorted(list(set(train_data.iloc[:].values[:, 41])))

list_of_labels

def get_dictionary_from_list(n):
    unique_dict = {}
    for index, value in enumerate(n):
        unique_dict[value] = index + 1
    return unique_dict
```

А қосымшасының жалғасы

```
def get_classified_dict(classification_list, replaceable_word):
    dict_of_something = {}
    for item in classification_list:
        if item == replaceable_word:
            dict_of_something[item] = 0
        else:
            dict_of_something[item] = 1
    return dict_of_something

dict_of_labels = get_classified_dict(list_of_labels, 'normal.')
dict_of_protocol_types = get_dictionary_from_list(list_of_protocol_types)
dict_of_services = get_dictionary_from_list(list_of_services)
dict_of_flags = get_dictionary_from_list(list_of_flags)

train_data.label = [dict_of_labels[item] for item in train_data.label]
train_data.protocol_type = [dict_of_protocol_types[item] for item in
train_data.protocol_type]
train_data.service = [dict_of_services[item] for item in train_data.service]
train_data.flag = [dict_of_flags[item] for item in train_data.flag]
train_data = train_data.sample(frac=1).reset_index(drop=True)
train_data
```

```
x_train = train_data.iloc[:, 0:41].values
y_train = train_data.iloc[:, 41].values
x_train

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_train

model = Sequential()
model.add(Dense(45, activation = 'relu', input_dim = 41))
model.add(Dense(29, activation = 'relu'))
model.add(Dense(17, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))

model.compile(optimizer = 'adam',
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])
```

А қосымшасының жалғасы

```
model.summary()

model_save_path = 'best_model.h5'
checkpoint_callback = ModelCheckpoint(model_save_path,
                                      monitor='val_accuracy',
                                      save_best_only=True,
                                      verbose=1)

history = model.fit(x_train,
                    y_train,
                    epochs=100,
                    batch_size=256,
                    validation_split=0.2,
                    callbacks=[checkpoint_callback])

plt.plot(history.history['accuracy'],
         label='Оқыту жиынындағы дұрыс жауаптардың үлесі')
plt.plot(history.history['val_accuracy'],
         label='Тексеру жиынындағы дұрыс жауаптардың үлесі')
plt.xlabel('Оқыту дәуірі')
plt.ylabel('Дұрыс жауаптар үлесі')
plt.legend()
plt.show()
```

```
model.load_weights(model_save_path)

!wget http://kdd.ics.uci.edu/databases/kddcup99/corrected.gz

test_data = pd.read_csv('corrected.gz',
                        header=None,
                        names=features)

test_data

list_of_test_labels = sorted(list(set(test_data.iloc[:].values[:, 41])))
list_of_test_services = sorted(list(set(test_data.iloc[:].values[:, 2])))
list_of_test_services = sorted(list(set(list_of_test_services) -
set(list_of_services)))

dict_of_test_labels = get_classified_dict(list_of_test_labels, 'normal.')
dict_of_test_services = dict_of_services.copy()
```

А қосымшасының жалғасы

```
for item in list_of_test_services:
   dict_of_test_services[item] = len(dict_of_test_services) + 1
dict_of_test_services

test_data.label = [dict_of_test_labels[item] for item in test_data.label]
test_data.protocol_type = [dict_of_protocol_types[item] for item in
test_data.protocol_type]
test_data.service = [dict_of_test_services[item] for item in test_data.service]
test_data.flag = [dict_of_flags[item] for item in test_data.flag]

x_test = test_data.iloc[:, 0:41].values
y_test = test_data.iloc[:, 41].values
x_test

x_test = scaler.fit_transform(x_test)
x_test

model.evaluate(x_test, y_test, verbose=1)
```