ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
коммерциялық емес акционерлік қоғамы
IT-инжиниринг кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 ж.

ДИПЛОМДЫҚ ЖОБА

Тақырыбы: Конвективті нейро желілер негізінде бейнелердегі тұтыну тауарларын анықтау және тану бағдарламасын құру

Мамандығы: 5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету»

Орындаған: Қамен Б.Қ. Тобы: ВТк-15-1
Ғылыми жетекші: аға оқытушы Толыбаев Ш.Д.

Кеңесшілер:
Экономикалық бөлім: э.ғ.к., профессор _____ Ж.Г. Аренбаева
«10» 05 2019 ж.

Өміртіршілік қауіпсіздігі: т.ғ.д., аға оқытушы _____ Ш.Ш. Бекбасаров
«08» 05 2019 ж.

Есептеу техникасын қолдану: аға оқытушы _____ Ж.С. Айткулов
«08» 05 2019 ж.

Норма бақылаушы: аға оқытушы _____ К. Мукапил
«15» 05 2019 ж.

Сын-пікір беруші: т.ғ.д., профессор _____ Б.С. Ахметов
«15» 05 2019 ж.

Алматы 2019

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
коммерциялық емес акционерлік қоғамы

Басқару жүйелері және ақпараттық технологиялар институты

IT-инжиниринг кафедрасы

Мамандығы 5В070400 – «Есептеу техникасы және
бағдарламалық қамтамасыз ету»

Дипломдық жобаны орындауға берілген
ТАПСЫРМА

Білім алушы Қамен Бегзат Қуанышұлы

Жобаның тақырыбы: Конвективті нейро желілер негізінде бейнелердегі тұтыну тауарларын анықтау және тану бағдарламасын құру

2018 жылғы «26» қазан № 124 университет бұйрығымен бекітілген.

Аяқталған жобаны тапсыру мерзімі: «24» мамыр 2019 ж.

Дипломдық жобаның бастапқы мәліметтері (зерттеу (жоба) нәтижелерінің талап етілген параметрлері мен объектінің бастапқы мәліметтері): Ұсынылып отырған дипломдық жобада конвективті нейрондық желілер негізінде тұтыну тауарларын анықтау және тану бағардарламасын құру мақсаты қойылған. Жобаны орындау барысында Mobilenet нейронды желі ядросы және TensorFlow кітапханасы қолданылды.

Дипломдық жобада қарастырылған мәселелер тізімі немесе дипломдық жобаның қысқаша мазмұны:

- талдау бөлімі;
- жобалау бөлімі;
- жүзеге асыру және тестілеу бөлімі;
- өміртіршілік қауіпсіздігі;
- экономикалық бөлім;
- А қосымшасы. Техникалық тапсырма;
- Ә қосымшасы. Программа листингі;
- Б қосымшасы. Ендіру актісі.

Графикалық материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс): 8 кесте, 28 сурет ұсынылған.

Ұсынылатын негізгі әдебиеттер:

1. Хайкин Саймон. Нейронные сети: полный курс, 2-е издание. — Москва: Вильямс, 2006. — 1104 с.
2. Джонс, М. Т. Программирование искусственного интеллекта в приложениях / М. Т. Джонс — 2-е изд. — М. : ДМК-Пресс, 2011 .— 313 с
3. Каллан Р. Основные концепции нейронных сетей = The Essence of Neural Networks First Edition. — 1-е. — «Вильямс», 2001. — С. 288.
4. R Hahnloser, R. Sarpeshkar, M A Mahowald, R. J. Douglas, H.S.Seung (2000). Digital selection and analogue amplification coexist in a cortex-inspired

Дипломдық жобаның бөлімдеріне қатысты белгіленген кеңес берушілер

Бөлімдер	Кеңесшілер	Мерзімі	Қолы
Экономикалық бөлім	Аренбаева Ж.Г.	04.03.2019 – 10.05.2019	
Өміртіршілік қауіпсіздігі	Бекбасаров Ш.Ш.	28.02 -27.03.2019	
Программалық қамтама	Айткулов Ж.С.	04.04.2019 – 10.05.2019	
Норма бақылау	Мукапил К.	04.04.2019 – 10.05.2019	

Дипломдық жобаны дайындау
КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
Талдау бөлімі	14.01.2019	Орындалды
Жобалау бөлімі	20.02.2019	Орындалды
Жүзеге асыру және тестілеу бөлімі	15.03.2019	Орындалды

Тапсырманың берілген күні «25» 10 2018 ж.

Кафедра меңгерушісі _____ Т.С. Картбаев

Жобаның ғылыми жетекшісі _____ Ш.Д Толыбаев

Тапсырманы орындауға алған
білім алушы _____ Б.Қ. Қамен

Аңдатпа

Мобильдік құрылғылардан алынған суреттерді өңдеу, жіктеу және тану қазіргі кезде өзекті мәселе болып табылады, оның көптеген аспектілерінің әлі күнге дейінгі тиімді шешімдері жоқ.

Бұл саладағы қолданбалы зерттеулерді дамыту негізінен, белгілі математикалық модельдерді және суреттерді өңдеу мен айырып тану әдістерін қолданудың жаңа жағдайларына бейімдеу жолымен жүруде. Алайда, бейнелерді тіркеудің функционалдық мобильді құрылғыларының ерекшеліктері, сондай-ақ оларға тән техникалық шектеулер жаңа математикалық модельдер мен әдістерді, есептеу алгоритмдерін және статикалық бейнелерді, сондай-ақ динамикалық бейне ағындарын өңдеудің, талдаудың және айырудың ақпараттық технологияларын әзірлеуді талап етеді.

Бұл дипломдық жобада нақты уақыттағы бейнелердегі тұтыну тауарларын тану үшін бағдарламалық өнімді әзірлеу қарастырылады.

Аннотация

Обработка, классификация и распознавание изображений, получаемых мобильными устройствами их регистрации является актуальной научной проблемой, многие аспекты которой до сих пор не имеют эффективных решений.

Развитие прикладных исследований в этой области идет, в основном, по пути адаптации известных математических моделей и методов обработки и распознавания изображений к новым условиям применения. Однако особенности функциональных мобильных устройств регистрации изображений, а также присущие им технические ограничения требуют разработки новых математических моделей и методов, вычислительных алгоритмов и информационных технологий обработки, анализа и распознавания как статичных изображений, так и динамических видеопотоков.

Целью дипломного проекта является разработка программного продукта для распознавания потребительских товаров на изображениях в реальном времени.

Annotation

Processing, classification and recognition of images received by mobile devices of their registration is an actual scientific problem, many aspects of which still do not have effective solutions.

The development of applied research in this area is mainly on the way of adaptation of known mathematical models and methods of image processing and recognition to new conditions of application. However, the features of functional mobile image recording devices, as well as their inherent technical limitations, require the development of new mathematical models and methods, computational algorithms and information technologies for processing, analysis and recognition of both static images and dynamic video streams.

In this project discusses the development of a software product for the recognition of consumer products in real-time images.

Мазмұны

Кіріспе

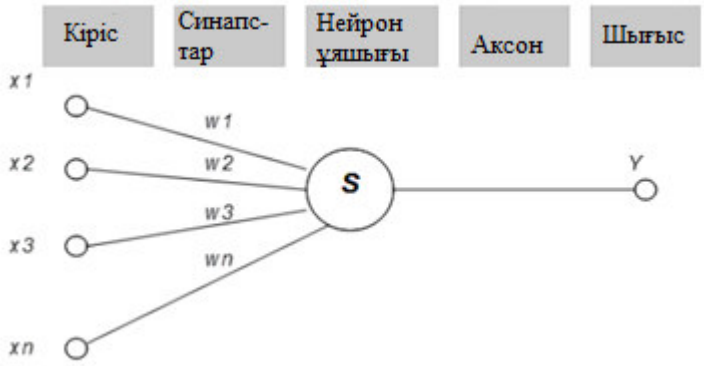
Қазіргі уақытта бейнедегі объектілерді тану және талдау ақпараттық технологияның өзекті міндеті болып табылады. Бұл міндет болашақта да өзінің өзектілігін жоғалтпайды, себебі тез қарқынмен дамып жатыр. Бүгінгі таңда жасалған қосымшалар болақашақта құрылатын жаңа бағдарламалардың негізі болады. Бүгінде компьютерлік көру саласында жасанды нейрондық желіні кеңінен пайдаланады. Нейрондық желілердің артықшылықтарының бірі - бұл барлық элементтер параллель жұмыс істей алады, сол арқылы есептерді шешу тиімділігін, әсіресе бейнелерді өңдеуде айтарлықтай жоғарылатады. Нейрондық желілер басқа статистикалық әдістерге қарағанда, бейнені тану кезінде тұрақты. Қазіргі уақытта түрлі қолданбалы есептер үшін суреттерді автоматты тану жүйелерінің бірнеше түрлері бар. Бірақ нақты уақытта суреттерді тану қосымшалары қарастырылмаған. Нейрондық желілер мұндай жүйелерді әзірлеу үшін теориялық және практикалық негіз бола алады. Осылайша, нейрондық желінің көмегімен бейнедегі объектілерді тану және талдау үшін бағдарламалық қамтамасыз етуді әзірлеу және зерттеу өзекті міндет болып табылады.

Дипломдық жоба барысында үйрітпелі нейрондық желілерді пайдалынып тұтынушы тауарларын тану бағдарламасын әзірлеу әдістері қарасатырылады.

1 Нейрондық желілер түсінігі

1.1 Нейрондық желілер туралы жалпы ақпарат

Нейрон - ақпаратты электр және химиялық сигналдардың көмегімен өңдейтін, сақтайтын жүйке жүйесінің электрлік қоздырғыш жасушасы.



1.1-сурет – Жасанды нейрон сұлбасы

Жасанды нейрон биологиялық сипатқа ұқсас құрылымға сәйкес салынған. Бұл торап бірнеше кіріс және бірнеше шығудан тұрады. Кіріс үшін бірнеше дәлелдер талап етіледі, дәлелдерді толтырады, олардың әрқайсысынан кейбір сызықтық комбинациясы үшін бірыңғай дәлелді көбейтеді немесе шығарады. Шығу-бұл жалпы дәлелдің сызықты емес функциясының мәні. Бұл функция триггер функциясы деп аталады. Нәтиже шығысқа немесе басқа да нейрондарға бірнеше шығу үшін жіберіледі. Бір-бірімен байланысқан нейрон-нейрондық желілер.

Жасанды нейрондық желілердің (ЖНС) пайда болуы ең алдымен мидың жұмысын зерттеумен байланысты. Зерттеумен байланысты алғашқы жұмыстар арасында нейрондық желілер У. Маккалок пен У. Питтстің идеяларын логикалық есептеу және жүйке белсенділігі туралы мақала болды. Ми өте күрделі, сызықты емес, параллель компьютер (ақпаратты өңдеу жүйесі). Ол нейрондар деп аталатын өзінің құрылымдық компоненттерін, олар нақты міндеттерді (бейнелерді тану, сезім органдарының сигналдарын өңдеу, моторлық функциялар сияқты) орындауға қабілетті, ең заманауи компьютерлерге мүмкіндік бере алатындай етіп ұйымдастыруға қабілетті. Дәлірек, ми дәйекті түрде бірнеше тану тапсырмаларын орындайды (мысалы, таныс адамды бейтаныс ортада тану). Оған шамамен 100-200 миллисекунд кетеді, ал ұқсас тапсырмаларды орындау тіпті аз күрделі компьютерде бірнеше күн алуы мүмкін.

Нейрондардың даму түсінігі мидың икемділігі ұғымына байланысты. Адам миындағы ақпаратты өңдеу бірлігі ретінде нейрондардың жұмысында ең маңызды рөл атқарады. Осыған ұқсас, жасанды нейрондық желілерде компьютерлік моделдеу кезінде нейрондық желі жұмысының негізгі принциптері қолданылады:

– білім нейрондық желіге қоршаған ортадан түседі және оқыту процесінде қолданылады.

– білімді жинақтау үшін синаптикалық таразылар деп аталатын нейрондар арасындағы байланыс қолданылады.

Нейрондық желілердің ерекше қабілеті-жалпылау жасау.

Қорытынды деп оқу барысында кездеспеген деректер негізінде нәтижені алу қабілеті түсініледі. Бұл қасиеттер нейрондық желілерге күрделі және ауқымды есептерді шешуге мүмкіндік береді. Бірақ іс жүзінде автономды жұмыс кезінде нейрондық желілер дайын шешімдерді қамтамасыз ете алмайды. Оларды күрделі жүйелерге біріктіру қажет. Атап айтқанда, кешенді тапсырманы бір бөлігі нейрондық желілермен шешілуі мүмкін қарапайым жүйелілікке бөлуге болады.

Нейрондық желілерді пайдалану жүйенің пайдалы қасиеттерін қамтамасыз етеді [1, 33-37]:

Бейімділігі. Нейрондық желілер өзінің синаптикалық салмағын қоршаған ортаның өзгеруіне бейімдеуге қабілетті. Атап айтқанда, белгілі бір жерде әрекет етуге үйретілген нейрондық желілер жұмыс істеу үшін оңай қайта оқытылуы мүмкін. Сонымен қатар, статистика уақыт өте келе өзгеретін стационарлық емес ортада жұмыс істеу үшін нақты уақыттағы синаптикалық салмақты өзгертетін нейрондық желілер құрылуы мүмкін. Жүйенің бейімдік қабілеті жоғары болған сайын, оның тұрақты емес ортадағы жұмысы неғұрлым тұрақты болады, алайда уақыт тез өзгеретін параметрлермен бейімдік жүйе, сондай-ақ өнімділіктің жоғалуына әкеп соқтыратын бөгде қозуларға де әрекет ете алады. Бейімделудің барлық артықшылықтарын пайдалану үшін жүйенің негізгі параметрлері сыртқы кедергілерді ескермеу үшін жеткілікті тұрақты және ортаның Елеулі өзгерістеріне реакцияны қамтамасыз ету үшін жеткілікті икемді болуы тиіс. Бұл міндет әдетте пластикалық тұрақтылық дилеммасы деп аталады.

Жауаптың анық болуы. Бейнелерді жіктеу міндеті контексінде ақпаратты жинайтын нейрондық желіні жасауға болады. Бұл шешім тек нақты классты анықтау үшін және ақпаратты ұлғайту үшін қолданылады. Кейіннен бұл ақпарат күмәнді шешімдерді болдырмау үшін пайдаланылуы мүмкін.

Мәтінмәндік (контекстная) ақпарат. Білім нейрондық желінің ең құрылымында оның іске қосу күйі арқылы ұсынылады. Желінің әрбір нейроны оның барлық басқа нейрондарының әсер етуі мүмкін. Нәтижесінде, нейрондық желінің болуы контекстік ақпаратпен тікелей байланысты.

Ақауларға төзімділік. Нейрондық желідегі ақпаратты сақтаудың бөлінген сипатын ескере отырып, нейрондық желі құрылымына елеулі зиян келтіретіні оның жұмысына айтарлықтай әсер ететінін дәлелдеуі мүмкін. Кішкене зақымдармен нейрондық желінің сапасының төмендеуі баяу.

Масштабталу. Нейрондық желілердің параллель құрылымы кейбір міндеттердің шешімін тездетеді және VLSI (verylarge-scale-inte-grated) технологиясы шеңберінде нейрондық желілердің ауқымдылығын қамтамасыз етеді.

Талдау және жобалау біркелкілігі. Нейрондық желілер болып табылады ақпаратты өңдеудің әмбебап механизмі. Бұл дегеніміз, нейрондық желінің бір жобалық шешімі көптеген пәндік салаларда қолданылуы мүмкін.Бұл қасиет бірнеше жолмен көрінеді.

Басқару жүйелеріндегі нейрожелілік желілер 2 тәсілмен қолданылуы мүмкін.

On-line-Нейросеть режимі білім алады және бір уақытта басқару жүйесінің атқарушы құрылғысының кірісіндегі басқару әсері. Желіні оқыту мақсаты және Объектіні басқару мақсаты жүйенің бірыңғай мақсатты функциясының тапсырмасында көрініс табады. Желі нақты уақытта, жүйедегі процестердің өту қарқынында оқытылады.

Off-line режимі-желі жұмысы екі кезеңнен тұрады: 1) берілген оңтайлы басқару функциясының желісін оқытудың алдын ала кезеңі және 2) сол жағдайларда немесе оларға жақын болған жағдайда объектіні басқару режимінде Осы функцияның аппроксимациясын ойнату кезеңі. Желіні оқытудың және объектіні басқарудың мақсатты функционалдары бір-бірінен өзгеше болуы мүмкін. Басқару үшін нейрондық желіні қолданудың мұндай нұсқасы-супервизорлық басқару деп аталатын-нейрожелілік контроллерді синтездеу процесі және оның параметрлерін баптау бұл жағдайда нақты уақыт өтпесе де, қазіргі уақытқа дейін басым таралуын тапты.

Нейрондық желілерді қасиеттері бойынша жіктеуге болады:

– кіріс ақпаратының түріне;
– оқыту сипаты бойынша;
– сигнал беру сипаты бойынша;
– құрылымы бойынша.

Бірінші ЖНС модельдері 1940-шы жылдардың басында зерттеушілер В. Макколлх пен В. Питтс жасаған [2].

Олар әзірлеген нейрондық желінің компьютерлік моделі математикалық алгоритмдер мен ми белсенділігінің теориясы негізінде құрылды.Макклулх пен Питс электронды нейрондық желілердің кез-келген математикалық және логикалық операцияларды орындауға қабілетті, сондай-ақ интеллекттің кейбір көріністеріне ие: үйрену, тануды үйрену, қорытындылау.

Жасанды нейрон барлық кіретін сигналдардың аккумуляторы болып табылатын жасанды нейрондық желі құрылымдық бірлігі. Нейрон тұтас доменде үздіксіз болып табылатын нәтижелі салмақты сомаға сызықты емес функцияны қолданады. Нәтиже нейрондық желіге жіберіледі. Нейронның ағымдағы жағдайы келесі формула бойынша сипатталуы мүмкін:

$$S = \sum_{i=1}^{n} x_i * w_i + w_0 \quad (1.1)$$

мұндағы, $w_0 -$ нейронның ығысу коэффициенті; $x_i - i -$ ші сигнал.

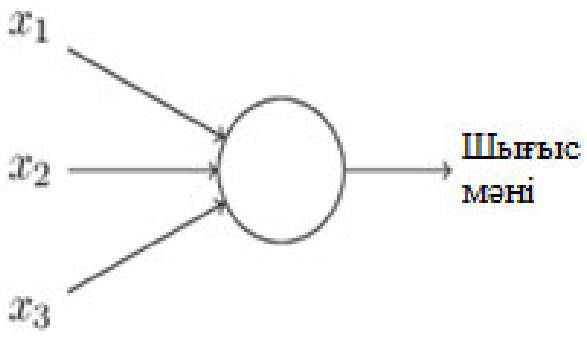
Нейронның шығуы-оның күй функциясы, оның формуласы төменде көрсетілген:

$$y = f(s), \quad (1.2)$$

мұндағы, f() – қосылу функциясы.

Белсендіру функциясы. Нейрондық желілерде активтендіру функциялары күрделі сызықты емес тәуелділіктерді өңдеу үшін қолданылады. Нейрон түріне байланысты белсендіру түрлі функциялары қолданылады. Олардың кейбірін қарастырайық[3].

1.2-суретте бірнеше екілік кіріс бар, x1, x2, шектік активациясы бар нейрон схемасы көрсетілген.



1.2-сурет – Персептрон сұлбасы

Осы түрдегі жасанды нейронда нәтижені есептеу үшін келесі ереже қолданылады: кірулерге келетін сигналдар көбіне салмақтар, w1, w2, ..., олар шығысқа шығудың маңыздылығын білдіретін нақты сандар. Нейрондық шығу мүмкін 0 немесе 1 мәндерін қабылдайды және өлшенген сома Σj $wj * xj$ белгілі

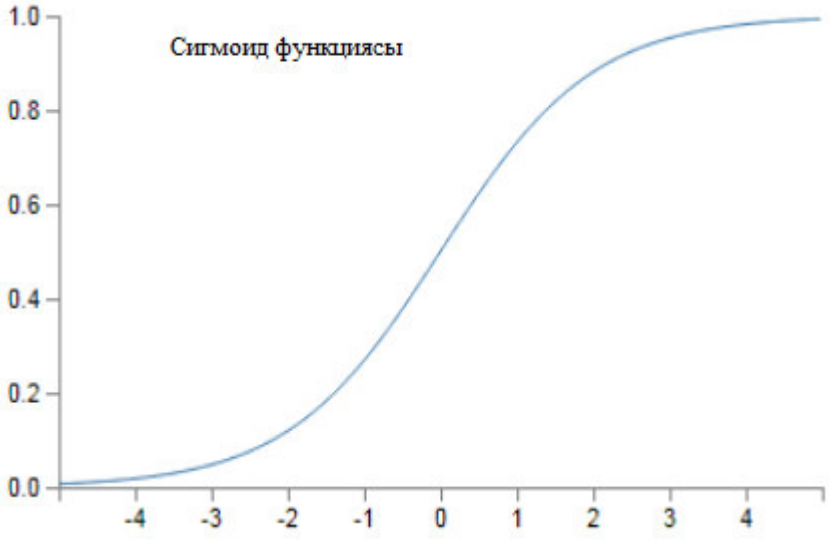
бір шекті мәннен асып кетуіне байланысты анықталады. Салмақ сияқты, шек - нақты сан, бұл параметр нейрон.

$$\text{кіріс мән} = \begin{cases} 0, & \text{егер } \sum_j w_j * x_j \leq \theta \\ 1, & \text{егер } \sum_j w_j * x_j > \theta \end{cases} \quad (1.3)$$

Сигмоидальді нейрон. Логистикалық функция. Бұл модельдің ерекшелігі - салмақ пен өрістердегі шамалы өзгерістер нейронның шығуындағы шамалы өзгерістерді тудырады. Нейтронның шегіне жеткенде, сигналды нейронды нақты сандар болып табылатын x1, x2, ... кірістері бар. Шығу 0-ден 1-ге дейінгі диапазондағы мәндер болуы мүмкін, сызбасы сызықты логистикалық активтендіру функциясы арқылы жүзеге асырылады, графикасы 1.3-суретте көрсетілген. Функцияның келесі формасы бар:

$$f(x) = \frac{1}{1 + e^{-\beta z}} \quad (1.4)$$

мұндағы, $z = \sum x[i] * w[i]$–кіріс мәндерінің өлшенген қосындысы– сигмоидальды белсендіру функциясының еңіс параметрі. Кесте неғұрлым көп болса,соғұрлым күшті. Sigmoid құралы Heviside функциясына ұмтылады.



1.3-сурет – Сигмоид функциясы

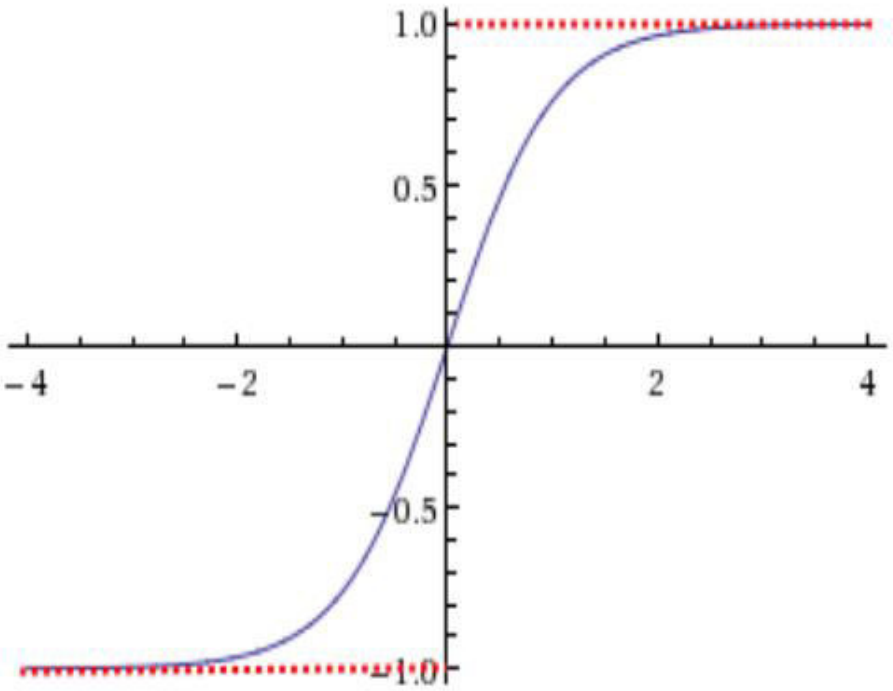
Сигмоидаль функциясы-бұл оның барлық аймағында дифференцияланған оны градиенттік әдістермен оқыту кезінде пайдалануға мүмкіндік беретін анықтамалар.

Гиперболалық тангенс. Гиперболалық тангенс-сигмоидальдық функциядан айырмашылығы бар, мәндер аймағы (-1;1) интервалында жатыр.

Осы белсендіру функциясы төменде берілген:

$$f(z) = \tan h\left(\frac{e^{\beta z} - e^{-\beta z}}{e^{\beta z} + e^{-\beta z}}\right) \tag{1.5}$$

мұндағы, β - функция графикасының тік дәрежесін анықтайды;
z-нейронның кіріс мәндерінің өлшенген сомасы.



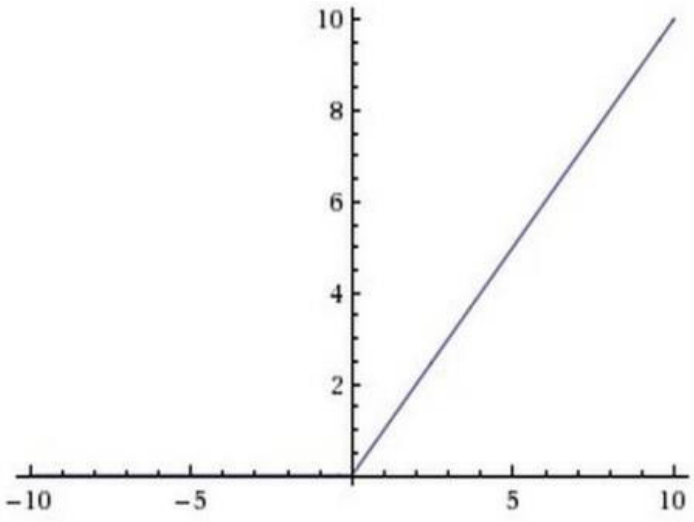
1.4-сурет – Гиперболалық тангенс

Гиперболалық тангенс функциясы сигмоидальды сияқты, анықтаудың барлық аймағында дифференцияланады. Оның ерекшелігі оң және теріс мәндерді модельдеуге мүмкіндік береді.

Іске қосудың түзетілген сызықты функциясы (rectified linear unit, ReLU). Гиперболалық тангенстің сигмоидальді функциясы мен функциясы сызықты емес және оларды анықтаудың бүкіл облысы бойынша сарапануы мүмкін, бірақ оларды қолдану жаттығулар мен ресурстық қарқынды

есептеулердің ұзақ уақытқа жинақталуы арқылы қиындайды. Мұндай проблемаларды болдырмау үшін, ректификацияланған желілік активтендіру функциясын (РТЛ сызықтық блокты - ReLU) [4] пайдалану ұсынылады, бұл оқу үдерісін едәуір жылдамдатуға мүмкіндік береді және сонымен қатар операция арқылы есептеулерді жеңілдетедіскалярдың теріс бөлігін қию. Қосылу функциясы төмендегідей:

$$f(z) = \max(0, z), \tag{1.6}$$



1.5-сурет – ReLu

ReLu артықшылығы градиенттердің оңайлатылған есептеуі болып табылады: осы функцияның туындысы z мәніндегі z мәнінің мәніне тең, егер бұл мән нөлден үлкен болса және нөлге тең болмаса. Осылайша, градиент жарылысының әсері және градиенттердің азаюы азаяды. Бұл функция белсендіру матрицасының қарапайым шекті өзгеруін қолдану арқылы жүзеге асырылуы мүмкін, осылайша, есептеулерде биіктігі талап етілетін,сигмалдықтар мен гиперболалық тангенттерден айырмашылығы, ресурстық белсенді операциялардан айырылады. Алайда, ReLu кемшіліктері төмендегідей: үлкен градиент мәні нейрон ешқашан қайтадан белсендірілмейтін таразылардың осындай жаңартылуына әкелуі мүмкін, яғни ол өшіріледі. Мұндай жағдайда нейрон арқылы өтетін градиент әрдайым нөлге тең болады. Осы типтегі қателіктердің әсерін азайту үшін оқу жиілігін дұрыс таңдау керек.

1.2 Нейронды желі архитектурасы мен түрлері

Жасанды нейрондық желілер биологиялық,жүйке жасушаларының желісін құрайды. Нейрондық желілер бір-бірімен өзара әрекеттесетін нейрондардан тұрады.

Конвективті нейрондық желілер - тербейнелерді тиімді өңдеу үшін қолданылатын тікелей тарату желілердің класы. Конвективті нейрондық желілерде деректерді өңдеу ерекшеліктері төмендегідей:

– жергілікті қабылдау – тұтастай бір емес нейронды енгізу үшін беріледі. Бұл тәсіл суреттің топологиясын қабаттан қабатқа сақтауға мүмкіндік береді;

– ортақ салмақ - үлкен мөлшердегі байланыстар үшін аз салмақ жиынтығы;

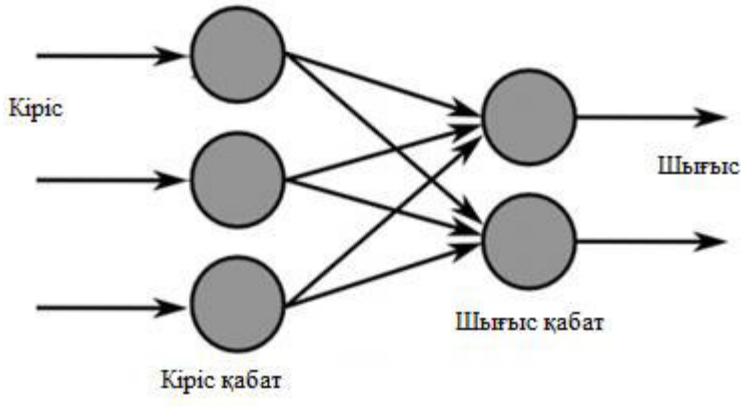
– өлшемді азайту - кеңістік өлшем түсірілетін қабаттарға байланысты кескін бірте-бірте азаяды.

Объектіні тану масштабқа қарамастан орындалады. Бұл жағдайда белгілердің болу фактісі оның кескіндегі нақты орналасқан жерін білуден маңызды.

Конвективті нейрондық торап конволюцияның және субгумуляцияның дәйекті қабаттарынан тұрады. Бұл алдыңғы қабаттың ерекшеліктер карталарын өңдеу негізінде ағымдағы қабаттағы ерекшеліктер карталарын жасауға мүмкіндік береді. Осылайша, түрлі ерекшеліктердің күрделі иерархиясын терең талдау жүргізіледі, жергілікті ерекшеліктерді таңдау және тану қарастырылған.

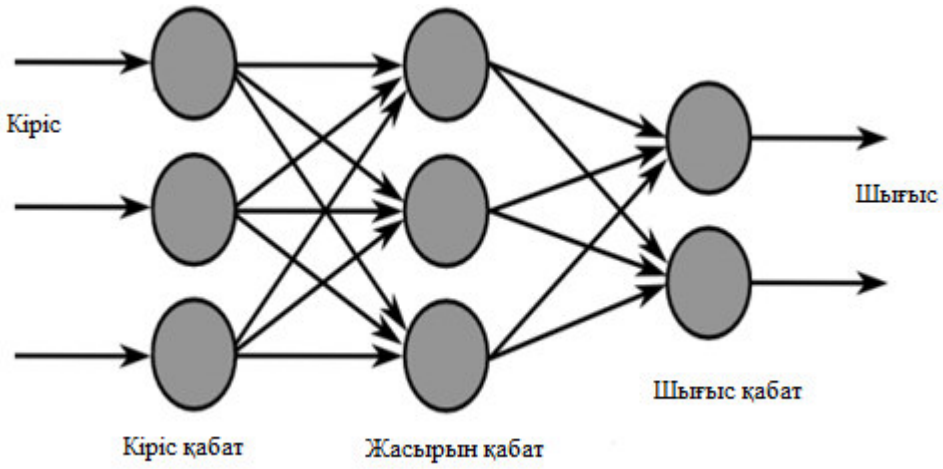
Жасанды нейрондық желілер бір қабатты және көп қабатты желілерге бөлінеді.

1.6-суретте бір қабатты нейрондық желі (Single-layer neural network) көрсетілген. Бұл желінің ерекшелігі кіріс қабатынан сигналдар олар өзгеретін Шығыс қабатына бірден беріледі,соңғы нәтижені қалыптастыра отырып. [5]



1.6-сурет – Бір қабатты нейронды желі

Кіріс және шығыс қабаттарынан басқа, 1.7-суретте келтірілген көп қабатты нейрондық желіде жасырын қабаттар бар,олардың арасында орналасқан.



1.7-сурет – Көп қабатты жасанды нейронды желі

Міндеттерді шешуге байланысты нейрондық желіде бір немесе бірнеше жасырын қабаттар болуы мүмкін. Әр қабаттағы қабаттар саны мен элементтердің саны функцияның күрделілігін анықтайды, сондықтан нейрондық желіні жобалау кезінде осы параметрлерді дұрыс анықтау маңызды. Осылайша, көп қабатты нейрондық желі күрделіліктің кез келген дәрежесіндегі функцияны модельдеуі мүмкін.

Бүгінгі күні әртүрлі құрылымдармен нейрондық желілерді көптеген іске асыру бар:

– Розенблатт персептронны;
– радиалды-базистік функциялар;
– өткізгіштер картасы;
– Хакимовтың сплайн моделі;
– Джордан желісі;
– нейрондық газ;
– хаотикалық нейрон желісі;
– осцилляторлық нейрон желісі;
– қарсы тарату желісі;
– Смирнов желісі;
– Решетовтың ықтимал желісі;
– бейімді резонанс желілері;
– үйіртпелі нейронды желілер;
– импульсті;
– терең сенім желісі (deep belief network) немесе Байесов желісі;
– генетикалық алгоритм.

Тікелей тарату желілері. Тікелей тарату желілері әдетте көп қабатты перцептрон болып табылады. Бұл желі, әдетте, кіріс көздерден, бастапқы түйіндерден, әйтпесе жасанды нейро желі сыртқы ортасын зерттейтін сенсорлық элементтерден тұрады. Бұл кіріс жиыны кіріс қабатын құрайды. Содан кейін олар есептеу операцияларын орындайтын жасырын қабаттарға сигналдарды жібереді.

Содан кейін сигнал шығыс қабатына жібереді, бұл өз кезегінде шығу нейрондарына ақпарат береді.

Тікелей тарату желілерінде көрініп тұрғандай, сигнал үнемі бір бағытта қозғалады: кіріс кірісінен барлық қабаттарды бір уақытта өтіп кетеді. Бұл көптеген оқыту алгоритмдерін пайдалану үшін ыңғайлы.

Үйітрпелі нейрондық желілер. Бұл нейрондық желі архитектурасын француз ғалымы Ян Лекун ұсынды. Ең алдымен, конструкциялық жасанды нейронды желі бейнені тануға бағытталған. Бұл Google компаниясы ұсынған «тереңдетіп оқыту» технологиясының бір бөлігі.

Желі құрылымы: көп қабатты, бір бағытты. Оқыту әдетте стандартты әдістер негізінде қолданылады, мысалы, қатені кері тарату әдісі. Белсендіру функциясы-кез келген. Үйрітпелі нейрондық желілердің негізгі ерекшелігі, неге ол үйіртпе деп аталады, оның үйіртпе операциясы бар. Үйіртпе операциясы-бұл бейненің әрбір фрагменті орама матрицасына көбейтіледі, нәтижесі жинақталады және жаңа бүктелген бейнедегі тиісті позицияға жазылады. Әр жолы, үйірті операциясы арқылы өтіп, сурет мәселе үшін маңызды бөлшектерді сақтай отырып, шуды бөліп, өлшемде азаяды.

Кәдімгі перцепроннан айырмашылығы, жасанды нейрондық желінің әрбір нейроны бұрынғы барлық байланысты және олардың әрқайсысы жеке үйіртпелі нейронды желі коэффициенті пайдаланылған кішкентай матрица қолданылады, ол бүкіл қабаттың үстінен «жылжытады». Бұл матрица келесі қабаттың тиісті нейроны үшін белсендіру сигналын құрайды. Яғни әрбір қабат қабаттағы барлық нейрондар үшін бірдей салмақ матрицаларын пайдаланады. Оның қолданылуы компьютерде іске қосылған кездегі ANN өлшемін азайтады және барлық нейрондардың бірдей өңдеуін қамтамасыз етеді.Осы матрица-ақ конволюция ядросы деп аталады. Бұл белгішені графикалық интерпретациялау құралы ретінде түсіндіріледі. Мысалы, кіріс кескінін талдайтын бірінші қабат кескіннің белгілі бір бөлігіндегі белгілі бір бұрышпен көлбеу сызықты бөліп алады. Сонымен қатар, конвективті нейрондық желіде салмақтың жиынтығы біреу емес, ал конволюцияның нәтижелері әр түрлі белгілер болуы мүмкін. Олар қабаттан қабатқа дейін әр түрлі болуы мүмкін. Осылайша, кейінгі қабаттағы революциядан кейінгі сурет қабілетін белгілердің картасы ретінде алынады. Келесі немесе сол матрицаны талдаудан кейін қарапайым фигуралар: шеңбер, шаршы және т.б. сияқты жаңа белгілерді көреді. Соңында, соңғы қабат оның белгілерін талдай отырып, үйдің немесе біреудің бетінің қай жерде

орналасқанын анықтай алады. Енгізілген матрицаның матрицалары алдын-ала белгіленбегенін, бірақ білім алудың нәтижесі екенін ескеріңіз. Бұл перцепроналық синапстар жиынтығы ретінде қабылдануы мүмкін: барлығы. Дегенмен, мұнда матрица, дәлірек айтқанда, нейрондық тиісті топтар үшін, конволитациялық кескін блоктары үшін бірдей салмақты сақтайды. Әрбір жаттығу жиыны өз ерекшеліктерінің жиынтығын жасайды, яғни бір қабаттағы жасанды нейро желі көп арналы - көптеген тәуелсіз ерекшеліктер карталарын жасайды. Сурет блоктары (жоғарыда аталған) 5 * 5-тен 100 * 100-ге дейін түрлі өлшемдерде болуы мүмкін, бұл блокта матрицаның өлшемі орнатылады. Бір матрица блоктарда «жүрмейді», белгілерге «қадам» бере алады, бірақ әрқайсысы 1-3 пиксельден аз қадамдар алады.

Субдискретизация - алынған сипаттамалар картасының өлшемін азайтуды жүзеге асыратын операция. Әдетте, конветтуттық жасанды желілерді енгізгенде, осы мүмкіндіктің болуы туралы ақпарат осы мүмкіндіктердің бейнесінде координаттарды білуден әлдеқайда маңызды. Демек, сіз суб-импорттау әрекетін пайдалана аласыз. Онда бірнеше көршілес нейрондардан

Традиционалдық карта ең маңызды немесе анық көрінеді және бір нейрон ретінде қабылданады. Сығылған атрибут картасы алынады, бұл келесі есептерді әрі қарай жылдамдатады, сондай-ақ жасанды нейро желі бастапқы енгізу векторының өлшеміне инвариантты суреттер.

Үйіртпелі жасанды желі жұмысын толығырақ қарастырайық. Желі бірненше қабаттардан тұрады. Бастапқы қабатқа, кіріске, сурет түседі. Бұл сурет үйіртпе және субдискретизация кезектесетін үйіртпе қабаттарының сериясынан өтеді. Үйрітпе картасын жасайды. Одан әрі ең маңыздысы бөлінеді. Қабаттан қабатқа дейін белгілер картасы мөлшері азаяды, бірақ оның арналарының саны артады. Бұл жасанды нейронды желі белгілердің күрделі иерархиясы бар неғұрлым күрделі белгілерді тану қабілетін қамтамасыз етеді. Бірнеше қабаттан өткеннен кейін белгілер картасы векторға немесе тіпті скалярға шығарылады. Мұндай белгілер картасы өте көп. Үйіртпе нейрондық желінің шығуында перцептрон мысалы, толық байланыс нейрондық желінің бірнеше қабаты қосымша пайдаланылуы мүмкін. Оған кіруге белгілердің соңғы карталары беріледі, перцептрон пайдаланушыға түсінікті түрде аяқталған нәтижені шығарады.

Рекуррентік желілер. Рекуррентті нейрондық желілердің ерекше ерекшелігі (РНЖ) кем дегенде бір кері байланыстың болуы болып табылады. Бұл нейрондар байланысты бір бағытта емес, сондай-ақ алыс (кіруден) және аз қашықтағы нейрондармен. Кері байланыстарды қолдану арқасында НС кіріс әсерлерінің реттілігін есте сақтау және ойнату қабілетті. Ал рекуррентті Нейронды желі есте сақтау негізінде жіктеуді шеше алады. Кері байланыс, бағдарламалау тұрғысынан-бұл циклдік орындаудың аналогы.

1-ші РНЖ Хопфилд желісі болды. Ол нейрондар осы жадтың ұяшықтары болып табылатын ассоциативті жады идеясын жасанды нейрондық желіде іске асырады. Бұл РНЖ бір, алдын ала берілген, өлшемінің тізбектерімен жұмыс істейді. Хопфилд желісі жеңілдетілген жағдайда бір Нейрон қабаты бар, оның нейрондары өзара байланысты. Байланыс осы нейронның маңыздылығын анықтайтын салмақпен бекітіледі. Бұл физикалық мағынада "энергия" баламасымен байланыстыруға болады. Бұл энергия жүйедегі барлық таразыға байланысты. Осылайша, "энергия" өзін жұмсауға және барынша азайтуға ұмтылған кезде желіні градиентті түсіру әдісімен оқытуға болады. Бұл минимум 1011101 сияқты белгілі бір үлгіні "есте сақтаған" кезде желі жағдайына әкелуі мүмкін. Және оған майланған немесе бұрмаланған сигнал бергенде, мысалы 1010100, НЖ "еске түсіреді" және осы сигнал жататын шаблонды қалпына келтіре алады. Мұны адамның ассоциативті жадымен салыстыруға болады. Дегенмен, оны тікелей қабылдау керек емес,, біз жалпы қағидаттар туралы айтамыз, және ми жұмыстарының барлық аспектілері Хопфилд желісінде іске асырылған жоқ. Хопфилд желілері өз міндеттерін жақсы орындайды және құру кезінде перцептрондарды мүмкіндіктері бойынша оңай айналып өтті.

Уақыт өте келе РНЖ басқалармен өзгертіліп, дамыды. Хопфилд торларынан кейін Джефф Элманның «қарапайым қайталанатын желісі» пайда болды [6]. Осы желіні құра отырып, автор уақыттық жүйені тану үшін нейрондық желіні оқыту мәселелері мен проблемаларына тоқталды. Нейрондық желі тіпті дәйектілікті уақытында тани алады ма? Егер солай болса, оны қалай жасауға болады? Мысалы, егер сіз 1011101 және 01011101 сигналдарын жіберсеңіз, онда NA осы сигналдардың ұқсастығын анықтайды, олардың біреуі уақытты ауыстырады? Әрине, сіз мұны қалай істей аласыз? Егер біз Хопфилд желілерінің жұмысына қарасақ, онда ол ұсынылған мысалдарды «есте» қоямыз. Хопфиллдің сызығы оның эволюциясы емес, сигналдарды салыстыру мәселесін шешеді және бір сигналды ауыстыруды анықтай алмайды. Элманның шешімі қарапайым қайталанатын желіге тағы бір «контекстік» қабат қосылғанына негізделген. Ішкі қабаттың күйі бұл қабатқа, әрбір оқу циклі мен РНЖ жұмысына көшірілді. Контекстік қабаттағы бұл байланыстар да оқытылуы мүмкін. Осындай қасиеттерге байланысты мұндай РНЖ уақытша серияларды оңай анықтайды, ерікті ұзындығын реттейді. Бұл қасиеттер Элманның «қарапайым РНЖ» басқа тәсілдер мен желілік модельдерден сапалы ажыратады. Сонымен қатар, Элманның желісі тек қана сөз тәртібіне негізделген етістіктерді, сын есімдер мен сөздерді жіктеуге және ажыратуға қабілетті болды. Сол кезде бұл нақты серпіліс болды.

Элманның қарапайым РН желілері барлық оқиға басамасы болып, осдын кейін 1997 жылы Хохрайтер және Шмидхубер [6] қазіргі заманғы РНЖ үшін өз мақалаларын жариялады. Өз жұмысында авторлар қарапайым РНЖ-нің ұзақ мерзімді есте сақтау проблемасын шешетін модификацияны сипаттады: олардың

нейрондары жақында қабылданған ақпаратты жақсы «есте ұстайды», бірақ олардың бұл ақпарат қаншалықты маңызды болғанына қарамастан, көптеген циклдар өңделген нәрсені үнемі есте сақтау мүмкіндігі жоқ. LSTM желілерінде ішкі нейрон күрделі деп аталатын күрделі жүйемен жабдықталған, сондай-ақ Ұзақ мерзімді жады тәрізді жасушалық мемлекеттің тұжырымдамасы. Қақпа ақпараттар ұялы күйге түсетінін анықтайды, ол одан тазартылады және бұл қадам осы РНЖ-ге беретін нәтижеге әсер етеді. Біз LSTM-ті егжей-тегжейлі талдай алмаймыз, алайда біз қазір RNS-нің бұл нұсқалары, мысалы, Google-дың машина аудармалары үшін кеңінен пайдаланылатындығын ескереміз.

1.3 Нейрондық желілерді оқыту

Нейрондық желілердің маңызды ерекшелігі - олардың экологиялық деректерден үйрену қабілеті және оқыту нәтижесі ретінде олардың жұмысын жақсарту. Белгілі бір ережелер бойынша уақыт өте келе өнімділік артады. Нейрондық желіні оқыту синаптикалық таразыларды түзетудің интерактивті процесі арқылы жүргізіледі. Ең дұрысы, нейрондық желі әрбір итерацияда қоршаған орта туралы білім алады.

Кіріс сигналын қолданғанда дұрыс жұмыс істейтін жасанды нейрондық желі тиісті шығу сигналына түрлендіреді. Нейрондық желіні үйрену процесі - осындай салмақтарды таңдау, онда желіден өткен кіріс сигналы қажетті нәтижеге айналдырылады. Тиімді оқытуға арналған NA үшін оқу және сынақ үлгісі болуы керек.

Оқу жиынтығы - желілік тренинг өткізілетін кіріс кескіндерінің жиынтығы. Оқытушы оқытумен бірге, кіріс сигналдарына қосымша, шығыс жиынтығы жасалады. Оқу кезеңі жүйенің жұмыс істеу критерийінің қолайлы бағасына жеткенде аяқталған деп есептеледі.

Нейрондық желілердің сапасын бағалау үшін олар тестіленеді. Бұл үшін сынақ жиынтығы - кіріс кескіндерінің жиынтығы (дұрыс шығыс сигналдарымен мұғаліммен оқыту жағдайында), ол бойынша желінің жұмыс сапасы сынақ үлгісіндегі дұрыс танылған бейнелердің пайыздық көрсеткіші ретінде бағаланады.

Нейрондық желілік жүйе арқылы шешілетін міндеттерге байланысты оқытудың тиісті түрі екі ең кең тараған әдістерден таңдалады: мұғалімсіз немесе мұғалімсіз оқыту. Мұғаліммен жаттығу кезінде бұрын «мұғалім» белгілеген деректер немесе оқу үлгілері өңделеді. Мұғалімдерді оқыту жүйесі осындай деректерді талдайды және белгілі деректердің даналары туралы жаңа, таңбаланбаған деректерді көрсету үшін пайдаланылуы мүмкін ішкі көрсетілім функциясын қалыптастырады.Мұғалімсіз оқу барысында олар бастапқыда өңделмейді.белгіленген деректер. Мұғалімсіз оқытудың мақсаты деректердегі

жасырын құрылымдарды іздеу, олардың ішкі көрінісін анықтау, деректерді кластерлеу және оларды таңбалауды жеңілдету.

Мұғаліммен оқыту. Егер нейрондық желі алдын ала белгілі дұрыс жауаптарды пайдалана отырып оқытылса, онда мұндай оқыту алгоритмі деп аталады - мұғаліммен оқыту.Мұғаліммен оқыту кезінде жұмыс және икемді нейрондық желіні жеткілікті мөлшерде қалыптастыру үшін үлкен таңдау талап етілетінін атап өту қажет.

Мұғалімсіз оқыту. Мұғалімсіз оқыту алгоритмі тек кіріс деректері белгілі болған кезде қолданылады. Олардың негізінде кіріс және шығыс деректерді беру нақты болып есептелінеді. Олар кластеризациялауға және обьектілердің өзінен бейнелерді табуға қабылдануы мүмкін.

Нығайтумен оқыту. Нейрондық желіні оқытудың тағы бір түрі, ортамен өзара әрекеттеседі. Орта пікірлері мұғаліммен оқыту кезіндегі қателіктер емес, бекіту сигналдары болып табылады.

Қателерді түзету негізінде оқыту. Фрэнк Розенблатт ұсынған перцепронды оқыту әдісі. Алгоритм - бұл перцепронның ағымдағы реакциясы дұрыс болғанша, байланыс салмағы өзгермейтін оқыту әдісі. Егер шығыс ақпарат қанағаттандырмаса, желі параметрлері түзетіледі. Түзету мәнін есептеу үшін желінің нақты және қалаған шығу арасындағы айырмашылық қолданылады.

Көп қабатты нейрондық желіні оқыту әдістерінің бірі қателерді артқа қарай таратудың алгоритмі болып табылады. Бұл итеративті градиент алгоритмі, оның басты идеясы желідегі шығу сигналдарынан оның кірістеріне, керісінше қалыпты жұмыс кезінде тікелей сигналды таратуды таратуға бағытталған. Қатені бағалау үшін әдетте жауаптардың орташа қателігі ретінде анықталған сапалы функционалдылықты енгізу.

Нейрондық желіні артқы тарату әдісінің әдісін қолданумен оқыту процесін егжей-тегжейлі қарастырайық: шығу қателігінің таңдалған функциясына негізделген E (y_i, $y`_i$), мұндағы у - мақсатты шығыс мәні, $y`$ - ағымдағы шығу

Нейрондық желінің әрбір қабатының салмағы төмендегідей түзетіледі:

$$w_i^{(l)} = w_i^{(l)} + \eta \Delta w_i^{(l)} \tag{1.7}$$

мұндағы, l - қабаттың нөмірі;

i - ағымдағы қабаттағы салмақ нөмірі;

η - оқу жылдамдығының параметрі.

Салмақты түзету жүргізілетін шама келесідей анықталады:

$$w_i^{(l)} = -\delta_i^{(l)} \frac{\partial a(z_i)}{\partial z_i} y_i^{(l-1)} \tag{1.8}$$

мұндағы, $a(z_i) -$ қазіргі қабаттың қосу функциясы;

$z_i = \sum_{j=1}^{N} w_j^{(l)} y_j^{(l-1)}$ - алдыңғы қабаттың sығыс мәндерінің өлшенген сомасы;

N-қабаттағы нейрондар саны.

$\delta_i^{(l)}$ - келесідей анықталатын қате параметрінің мәні:

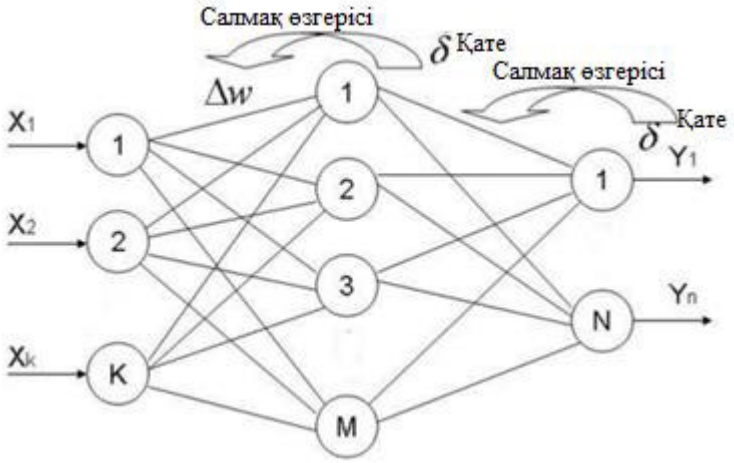
шығу қабаты үшін

$$\delta_i^{(l)} = \frac{\partial E(y_i, y`_i)}{\partial y`_i} \tag{1.9}$$

жабық қабат үшін

$$\delta_i^{(l)} = \sum_{j=1}^{N} \delta_i^{(l+1)} w_i^l \tag{1.10}$$

Егер салмақты реттегеннен кейін, қате белгіленген шекті мәннен аз болса, жаттығу тоқтайды, әйтпесе процесс сапа өлшемі орындалмайынша итеративно қайталанады.



1.8-сурет – Қателерді кері тарату алгоритмі

Нейрондық желіні үйрену процесі оның ішкі параметрлерін нақты тапсырманы оңтайлы орындау үшін конфигурациялау болып табылады. Нейрондық желіні оқыту алгоритмі итеративті болып табылады, оның қадамдары дәуір деп аталады. Epoch - жаттығу процесінде бір итерация, оның ішінде жаттығу жиынынан барлық мысалдарды өңдеу, сонымен қатар желінің салмағын түзету. Оқу кезеңінде дәуірлердің санын таңдаудан басқа, жүйені

өңдеуге арналған нейрондық желіге жеткізілетін сериялардың (партиялардың) мөлшері де түзетіледі - алға және кері өтудің бір иерархасына арналған оқу үлгілерінің саны. бір дәуір.

Хебб оқытуы. Хебб оқытуы физиологиялық және психологиялық зерттеулерге негізделген Оқыту алгоритмін 2 бөлімнен ереже түрінде ұсынуға болады:

– егер екі нейрон синапстың екі жағында бір уақытта белсендірілсе, онда бұл қосылыстың беріктігі артады;

– егер екі Нейрон синапстың екі жағында асинхронды белсендірілсе, онда мұндай синапс әлсірейді немесе толықтай жойылады.

Бәсекелестік оқыту. Көптеген нейрондар бір мезгілде қозғала алатын Хебб оқытудың айырмашылығы, бәсекелі оқыту кезінде демалыс нейрондары бір-бірімен белсендіруге жарасады. Яғни барлығы көптеген шығу нейрондары тек бір ғана пайдаланылады – ең үлкен шығумен. Оқыту кезінде жеңген нейронның салмағы ғана өзгереді.Және ең жақсы қолайлы нейрон мысалға жақын болады.

Бәсекелестік оқыту классификация есептерінде қолданылатын статистикалық қасиеттерді зерттеу үшін өте ыңғайлыкіріс бейнелері. Бұл жағдайда әрбір нейрон шығу қабаты бір сурет үшін жауап береді.

Хопфильд оқытуы. Бұл оқыту моделі рекуррентті толық байланыс нейрондық желілер үшін қолданылады. Жұмыс барысында мұндай желілердің динамикасы тепе-теңдік ережелерінің біріне ұқсас болады. Бұл жүйе әрқашан тыныштыққа ұмтылатын термодинамикамен салыстыруға болады. Бұл тепе-теңдік ережелері оқу процесінде алдын ала анықталады, олар желі энергиясы деп аталатын функционалдың жергілікті минимумы болып табылады.

Алгоритм. Желіні оқыту - бұл векторлар-эталондық бейнелерді есте сақтау үшін өзара әрекеттесу матрицасының салмағы құрайтын "жады". Коэффициенттерді есептеу келесі ережеге негізделген: барлық есте қалған бейнелер үшін байланыс X_i матрицасы теңдеуді қанағаттандыруы тиіс.

$$X_i = WX_i \tag{1.11}$$

Есте сақталатын векторлар екілік түрі болуы керек. Салмақ коэффициенттерін есептеу мынадай формула бойынша жүргізіледі:

$$W_{ij} = \frac{1}{N} \sum_{d-1\ldots m} X_{id} X_{jd} \tag{1.12}$$

мұндағы, N - векторлардың өлшемі, m - есте сақталатын Шығыс векторларының саны, d - есте сақталатын Шығыс векторының нөмірі, есте

сақталатын j - ші вектордың і компоненті. Бұл өрнек анық болуы мүмкін, егер өлшеу матрицасы W әр есте сақталатын вектордың сыртқы туындысын өздерімен есептеп және осылайша алынған матрицаларды қосумен табылуы мүмкін.Бұл былай жазылуы мүмкін:

$$W = \frac{1}{N}\sum_{i} X_i X_i^T \tag{1.13}$$

мұндағы, Xi - і-ші естелік баған векторы. Осы салмақтарды есептеу тек бір дәуірге ғана арналған желілік оқыту деп аталады.

Больцман оқытуы. Бұл оқыту моделі Людвиг Больцманның және оның жұмысының құрметіне өз атауын алды. Стохастикалық оқыту алгоритмі болып табылады, стохастикалық механика идеяларына негізделген. Оқыту үшін рекуррентті нейрондық желілер қолданылады.

Алгоритм. Хопфилд желісіндегі сияқты, Больцман машинасында "энергия" ұғымы бар.Осы энергияны есептеу мына тәсілмен жүргізіледі:

$$E = \sum_{i<j} w_{ij} s_i s_j - \sum_{i} \theta_i s_i \tag{1.14}$$

мұндағы, w_{ij} - j және i нейрондары арасындағы байланыс күші,

S_i – күй;

i –нейронының - $s_i \in \{0,1\}$ тиісті аралығы;

$\theta_i - i$ нейронының шегі.

Хопфилд машинасында сияқты Больцман машинасын оқытудың міндеті желі жағдайын тұрақтандыру болып табылады. Алайда (Хопфилд желісіне қарағанда) Больцман машинасы жергілікті минимумдарды емес (және оларда "ату" емес) іздейді,жаһандық минимумдарға кіруге тырысады. Жергілікті минимумдардан шығу және терең минимумға түсу ықтималдығын арттыру үшін "жылу шуын" пайдалану идеясы С. Кирпатрика тиесілі. Осы идеяның негізінде күйдірудің имитация алгоритмі әзірленді.

$$P_k = \frac{1}{1 + e^{-E_k/t}} \tag{1.15}$$

мұндағы, t – жылу шуының аналогы;

E_k – байланыс таразысының соммасы.

Больцман машинасының оқу мүмкіндіктері шектеулі болғанымен, бұл проблемаларды Boltzmann машинасының архитектурасын қолдану арқылы шешуге болады. Бұл архитектурада қосылыстар жасырын және көрінетін нейрондар арасында ғана кездеседі, бірақ сол класс нейрондары арасында болмайды. Бұл архитектураны бастапқыда Пол Смоленский 1986 жылы Harmonium атымен қолданған, бірақ Хинтон 2000-жылдардың ортасында жылдам оқыту алгоритмдерін ойлап тапқаннан кейін ғана танымал болды.

Генетикалық алгоритм. Генетикалық алгоритм - бұл шешімдердің ең жақсы және ең нашар вариациялары таңдалатын таңдау арқылы ең жақсы комбинацияны іздеу әдісі. Бұл процесті табиғаттағы табиғи іріктеумен салыстыруға болады. Іріктеу жүргізіледі, онда талаптарды қанағаттандыратын адамдар ғана қалады. Жақсы нәтижелер айқындалады-шешімдердің ұсынымы жасалады.

1.4 Нейрондық желілердің қолданылуы

Нейрондық желілер адамның миына ұқсас аналитикалық есептеулерді қажет ететін күрделі мәселелерді шешу үшін қолданылады. Нейрондық желілердің ең көп қолданылатын түрлері:

Жіктелу – деректерді параметрлер бойынша бөлу. Мысалға, адамдар тобын кіре берсе, олардың қайсысы несие беріп жатқанын шешуге тура келеді. Бұл жұмыс нейрондық желі арқылы жасалуы мүмкін: жас, төлем қабілеттілігі, несие тарихы және т.б.

Болжау – келесі қадамды болжау мүмкіндігі. Мысалы, қор нарығындағы жағдайға негізделген қорлардың өсуі немесе құлдырауы.

Тану – қазіргі уақытта нейрондық желілердің ең кең қолданылуы болып табылады. Фотосуретті немесе камералық телефондарды іздеген кезде, Google сіздің бетіңіздің орнын анықтаған және оны көрсететін кезде және т.б.

Бұл тарауда нейрондық желілермен байланысты жалпы түсініктер ұсынылды. Биологиялық нейрондық желілер және олардың жұмыс істеу ерекшеліктері қарастырылады. Жасанды нейрон модельдері, жасанды нейрондық желілер ұсынылған. Сондай-ақ, негізгі нейрондық желілік архитектуралар берілген. Сонымен қатар, ЖНС оқытудың негізгі қағидалары көрсетіледі, қателерді артқа тарату әдісі оқытушымен ЖНС оқытудың негізгі әдісі ретінде ұсынылады.

Нейрондық желілер шешетін мәселелер. Қазіргі уақытта нейрондық желілер адам миы қызметінің қалыптасқан моделі болып табылады, кең ауқымды мәселелерді шешуге қолданылады. Төменде [3] жұмыста келтірілген сұрыптамаға сәйкес жасанды нейрондық желілер (ЖНЖ) көмегінде шешілетін негізгі проблемалық мәселелер анықталған [17]:

– классификация/бейнелерді ганымдау. Тапсырма белгілер векторымен ұсынылған бір немесе бірнеше анықталған кластардан гұратын кіріс образьшьщ кімге тиісті екендігін корсету (мысалы дыбыстық белгі немесе қолтаңба белгісі арқылы).Атақты қосымшаларға эріптерді тану, дыбыстық тану, электрокардиограммалар сигналының классификациясы, қан жасушаларының классификациялары, бармақ іздерін тану, бет пішінін тану т.б.

– кластеризация/категоризация. Тапсырмаларды шешуде кластеризация «ұстассыз» образдар классификациясы деген атпенде танымал. Мұнда кластар белгісі арқылы үйретуші таңдау жоқ. Кластеризация алгоритмі образдар тэрізді негізделген. Олар ұқсас образдарды бір кластерге жинақтайды. Кластеризация әдісі мэліметтер құрамын анықтағанда, мәліметтерді сыққанда, қайта қалпына келтіруде қолданылады.

– функция аппроксимациясы. шуылмен бұрамаланган анықталмаған (х) функциясы арқылы генерленген үйретуші таңдау бар делік. Аппроксимация тапсырмасы анықталмаған (х) функциясыныц бағасын табу. Функция аппроксимациясы көпмәнді инженерлік жэне ғылыми тапсырма моедльдерін анықтау үшін қажет.

– болжамдау/болжау. t уақытында сәттері тізбектелген п дискретті санағы берілген делік. Мақсат t >п+1 уақыттың келесі сэті тізбектегі у(() мэндерін болжамдаудан тұрады. Болжамдау/болжау бизнесте, ғылымда жэне техникада шешімдер қабылдауда колемді эсерге ие. Қор биржасындагы баға болжамы жэне ауа-райы, бейтаныс қоршаган ортаға бейімделу жағдайында өзі оқушы мобильдік дербес жүйелерді шешетін негізгі мэселе болып табылады.

– оптимизация. Математикадағы, статистикадағы, техникадағы,ғылымдағ ымедицинадағы жэне экономикадағы көптеген мэселелер тиімдестіру мәселелері ретінде қарастырылуы мүмкін. Тиімдестіру ізбе -ізділіктің міндеті - мақсатты тиістілікті азайту немесе көбейту жолы мен шектеу жүйесін қанағаттандырудың шешімін табу. NP тольтқ класында енуші коммивояжер мысалы тиімдестіру мэселесніц классикалық үлгісі болып табылады.

– ассоциативті жады. Мазмұнына қарай бағытталған ассоциативті жады немеес жады құрамы толық емес кіріс немесе бұрмаланған мазмұннан пайда болуы мүмкін. Ассоциативті жады мультимедиалық ақпараттар қорын қуруда пайдаланылады. Сонымен қатар ол мобильдік роботтардың басқару жүйесінің негізі болып табылады.

– басқару. у(t) уақытымен берілген динамикалық жүйесін қараймыз мұнда: u(t)-кірісті басқарушы эсер, у(t)-t уақыттық сәттегі жүйелік шығыс. Эталондық модельді басқару жүйелерінде жүйе эталондық модельмен берілген бағытта қозғалатын кірістік эсерді u(t) есептеп шығу болып табылады. Қозғалтқышты басқару траекториясы мысал бола алады [17-19].

Зерттеу барысында Нейрондық желілердің негізгі модельдері қарастырылып, нейрондық желілердің модельдеріне салыстырмалы мінездемесі

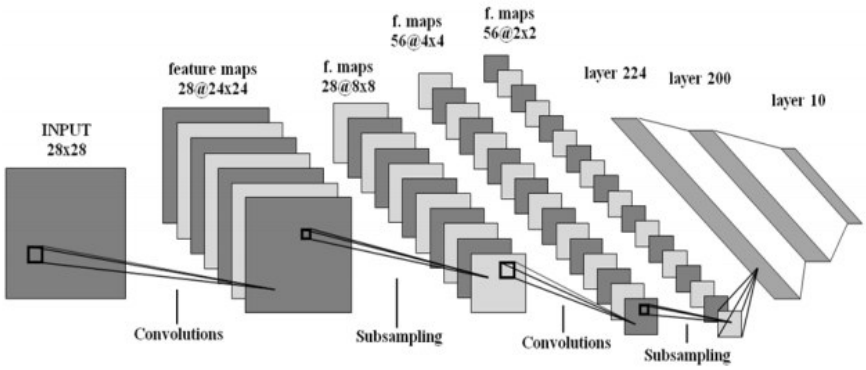
жасалды. Көпқабатты нейронның ақпаратгық жүйе моделінің жеке қасиеттері мен Хопфильд моделінің мүмкіндіктері ұсынылды. Нейрондық желілердің қолданылу аясымен тиімділігі анықталды.

1.5 Үйіртпелі нейрондық желілер

Үйіртпелі нейрондық желі (үйіртпелі нейрондық желі - CNN) - кескінді талдау үшін кеңінен қолданылатын терең, тікелей таратылатын жасанды нейрондық желілер класы.

Кескінді талдаудың басқа әдістерімен салыстырғанда, конвективті НЖ алдын ала өңдеусіз суреттерді тиімді талдау мүмкіндігін береді. Желілік әдеттегі алгоритмдерде қолмен орнатылған сүзгі коэффициенттерін автоматты түрде таңдайды. Бұл желі мүмкіндіктері оның басты артықшылығы болып табылады. Қазіргі кезде конвектуралық нейрондық желілер бейнелердегі күрделі визуалды объектілердің көп санын сыныптау үшін дәлдік және жылдамдық әдістерінде үздік болып табылады [8].

2.1-суретте ұсынылған конветтутенттік НЖ құрылымы әртүрлі қабаттардан тұрады: үйіртпелі (convolutional) қабаттар, субдискретизация(subsampling) қабаттар және жіктеу үшін қолданылатын толығымен қосылған қабаттар.



1.9-сурет – CNN құрылымы

Үйіртпе қабаттар мен субдискретизация бір-бірімен кезектесіп, көп қабатты толық байланысты классификатор үшін белгілердің шығу картасын қалыптастырады.

– бұл қабаттардың саны желінің топологиясына байланысты болады, ол келесі бағыттарда анықталады:

– шешілетін есеп (классификация, болжау, генерация және т. б.);

– шешілетін есептегі шектеулер (жылдамдық, жауаптың дәлдігі және т.б.);

– кіріс деректер (типі, өлшемі, форматы);

– шығыс деректер;

– желі гиперпараметрлерінің құрылымын оңтайландыру және таңдау әдістері.

Үйіртпелі желі арқылы объектілерді тануда нейрондық желі топологиясын таңдау маңызды орын алады.

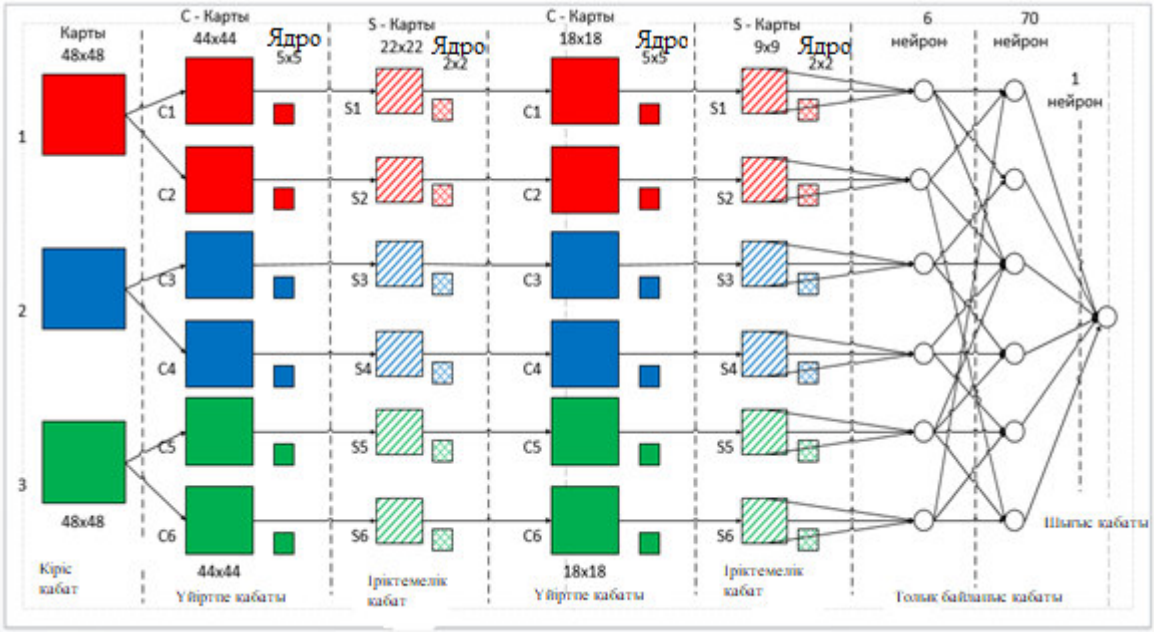
Топологияны таңдауға келесі қадамдар әсер етуі мүмкін:

– нейрондық желілермен шешілетін міндеттерді анықтау (жіктеу, болжамдау, модификациялау);

– шешілетін проблемадағы шектеулерді анықтау (жылдамдық, жауаптың дәлдігі);

– кірісті (тип: сурет, дыбыс, өлшем: 100x100, 30x30, формат: RGB, сұр реңкте) және шығу деректерін (сыныптар саны) анықтаңыз.

Менің құрған нейронды желі тапсырмасы ол бейнелердің классификациясын анықтау болып табылады. Нейронды желі шектеулігі бұл оның жауапты жылдам беруінде(1 секундтан аз) және тану дәлдігі(70% дан кем емес). Нейронды желінің жалпы топологиясы төмендегі суретте көрсетілген (1.10-сурет).

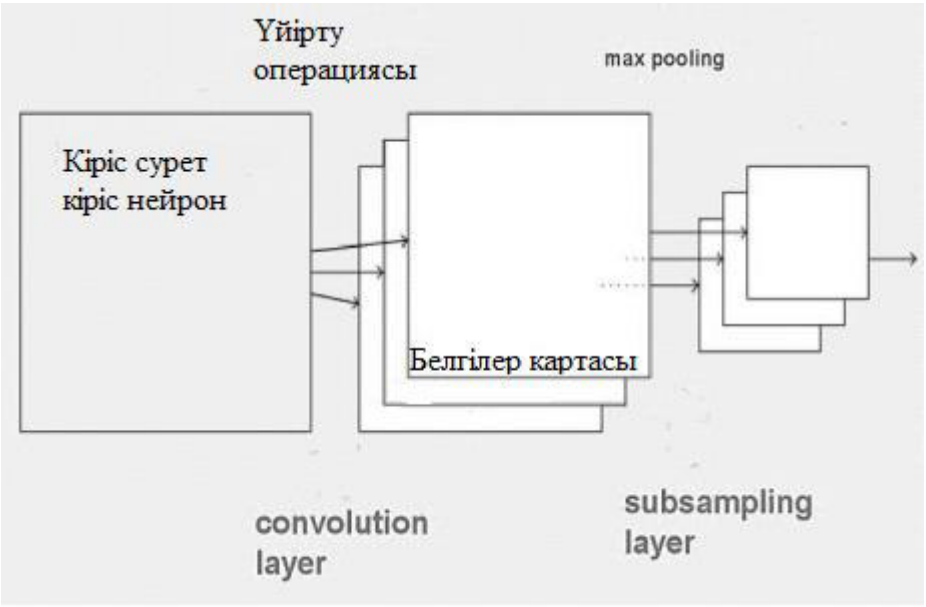


1.10-сурет – Үйіртпелі нейрожелінің топологиясы

1.6 Үйіртпелі нейрондық желілердің архитектурасы

Жоғарыда айтылғандай, конвективті НЖ-нің типтік бөлігі біріктіруден, біріктіруден және субдескритизациядан тұрады.

Үйіртпелі қабат - кірісте қабылданған функция картасын өңдейтін әрбір арна немесе үйкеліс ядролары үшін сүзгілер жинағын қамтиды. Дәнекерлеу ядросының салмағы оқу үдерісінде белгіленеді. Субдискретизация (pooling, subsampling) операциясы жасалатын ерекшеліктер картасының өлшемін азайтады.



1.11-сурет – Үйіртпелі нейрожелінің типтік блогы

Үйіртпе қабаттар мен субдискретизация бір-бірімен кезектесіп, көп қабатты толық байланысты классификатор үшін белгілердің шығу картасын қалыптастырады.

Бұл қабаттардың саны желінің топологиясына байланысты болады, ол келесі бағыттарда анықталады:

– шешілетін есеп (классификация, болжау, генерация және т. б.);
– шешілетін есептегі шектеулер (жылдамдық, жауаптың дәлдігі және т.б.);
– кіріс деректер (типі, өлшемі, форматы);
– шығыс деректер;
– желі гиперпараметрлерінің құрылымын оңтайландыру және таңдау әдістері.[9]

Үйірту операциясы. Үйірту ядросы салмақтың жиынтығы. Бұл операция нәтижесі белгілер картасы деп аталатын суретті енгізуге болады. Үйірту ядросына байланыты белгілер картасы әр түрлі болады. Кіріс сипаттамаларын барынша толық таңдау үшін бірнеше конвукцияның ядролары пайдаланылады, сондықтан конвективтік қабатта бірнеше ерекшеліктік карталар алынады.

Ядро - бұл сүзгі немесе терезе, ол алдыңғы картаға жылжытады және объектілердің белгілі бір атрибуттарын табады.

Мысалы, желі бірнеше тұлғаны үйренсе, онда біреуі оқу процесінде көз, ауыз, қас немесе мұрынды ең үлкен сигнал шығаруы мүмкін, ал басқа ядро басқа белгілерді анықтай алады. Негізгі өлшемі конвективті қабаттағы карталардың мөлшері тіпті тең етіп таңдалады, бұл қабаттың өлшемі азаятындай ақпаратты жоғалтпауға мүмкіндік береді.

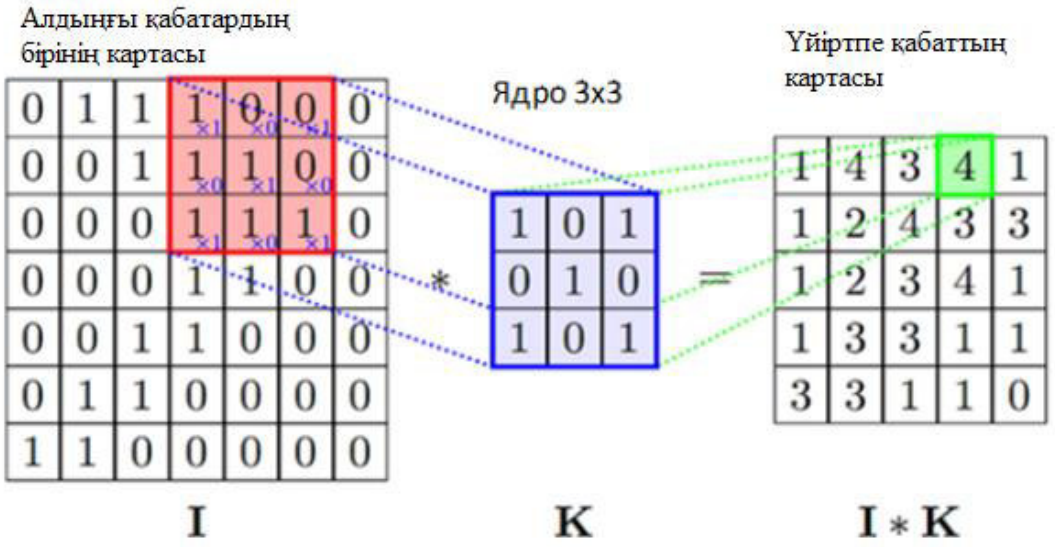
Үйрітпелі нейрондық желілердің басты ерекшелігі өңделген бейнелерді талдауға арналған конвульция операциясын пайдалану болып табылады. Конвульсиялық операция келесідей орындалады: ядро конверсия ядро коэффициенттерінің мәндеріне қатысты осы аймақтың элементтерінің салмақты сомасын өңдейді:

$$(f * g)[m,n] = \sum_{k.l} f[m-k, n-l] * g[k,l], \tag{3.1}$$

мұндағы, f – бейненің бастапқы матрицасы;

g – үйірпе ядросы.

1.12-суретте үйіртпе операциясын жұмысы көрсетілген.



1.12-сурет – Үйірту операциясы

Математикалық, ұю қабатындағы операцияны келесідей көрсетуге болады:

$$x^l = f(x^{l-1} * k^l + b^l), \tag{1.16}$$

мұндағы, $x^l - l$ қабатының шығысы;

f() – іске қосу функциясы;

b^l – l қабатының жылжу коэффициенті;

* - k ядросының х үйіртпе операциясың кірісі.

Бастапқы матрицалардың мөлшері азаяды:

$$x_j^l = f\left(\sum_i x_j^{l-1} * k_j^l + b_j^l\right), \tag{1.17}$$

мұндағы, $x_j^l - \ j$ белгілер картасы(l қабатының шығысы);

f() –іске қосу функциясы;

b^l – l қабатының j картасына қатысты жылжыту коэффициент

$k_j^l - l$ қабатсына қатысты j картасының үйртпе ядросы;

* - k ядросының х үйіртпе операциясың кірісі.

Әрбір ұю қабаты көптеген теңшелетін синаптикалық ядролардан тұрады. Мұндай көрініс нейрондық желіде бөлінетін таразылар жүйесін іске асыруға мүмкіндік береді. Толық байланыс НС қарағанда, ұю желісінде жалпы салмақ байланыстар санын қысқартуға және іске асыруға мүмкіндік береді

Суреттің барлық аймағында ұқсас белгілерді бөлу мүмкіндігі. Ұю қабаты жұмыс нәтижесінде белгілер карталарының жиынтығын қалыптастырады (feature maps). Ұю қабатымен қалыптасатын барлық белгілер карталарының мынадай формула бойынша есептелетін бірдей мөлшері болады:

$$(w, h) = (mW - kW + 1, mH - kH + 1), \tag{1.18}$$

мұндағы, (w,h) – есептелетің картаның өлшемі;

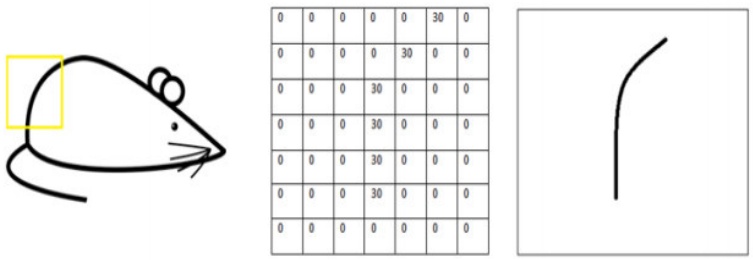
mW – алдыңғы картаның ені;

mH – алдыңғы картаның биіктігі;

kW – ядроның ені;

kH – ядроның биіктігі.

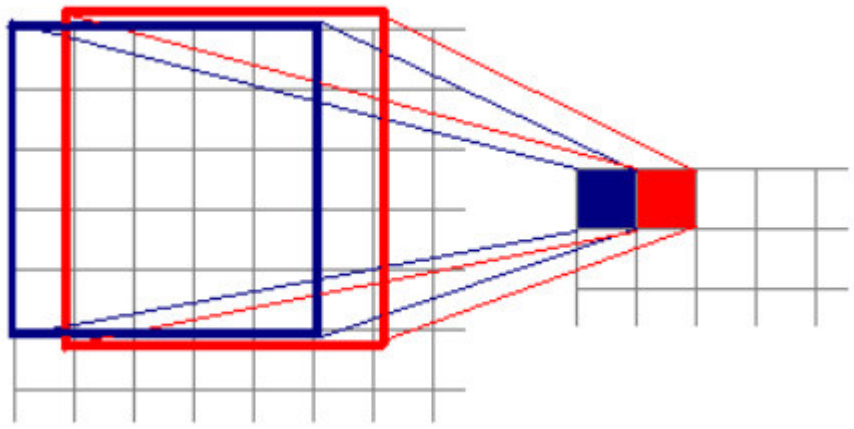
Ядро-бұл сүзгі, ол алдыңғы картаның барлық аймағында объектілердің белгілерін анықтайды.



1.13-сурет – Белгіні бөлуге үйретілген ядроның үлгісі

Оқыту процесінде әрбір ядро белгілі бір белгілерді бөледі,мысалы, бет, көз және т. б. ядроның өлшемі бөлінетін белгілердің мөлшеріне сәйкес таңдалады. Әдетте оны 3х3-тен 7х7-ге дейін береді.

Субдискретизация қабаты. Осы кезеңде карталардың өлшемін азайту міндеті орындалады жоғары қабатта алынған белгілер. Субдискретизация кезеңінде белгілерді іріктеу орындалады. Бұл келесідей қол жеткізіледі: орама қабатынан алынған белгілердің өңделетін картасында неғұрлым маңызды белгілер таңдалады. Бұл операция пулинг (pooling) деп аталады. Ең жиі пулингті жүзеге асыру үшін өңделетін картада максималды мәнді есептейтін функция қолданылады.



1.14-сурет – Субдискретизация операциясы

Бұл қабатты төмендегі формула арқылы сипаттауға болады:

$$x^l = f(a^l * subsample(x^{l-1}) + b^l)\ , \qquad (1.19)$$

мұндағы, x^l – l қабатының шығуы;

f() – активаиция функциясы;

$a^l, b^l - 1$ қабатының жылжыту коэффициенті;

subsample() – таңдау операциясың жұмыс істеу функциясы.

Қорытындылай келе конвективті нейронды желі объектінің масштабының өзгерісіне, оның орын ауыстыруына, бұрылуына, оның әртүрлі жақтан көрінуіне басқа да ақауларға төзімді болып келеді.

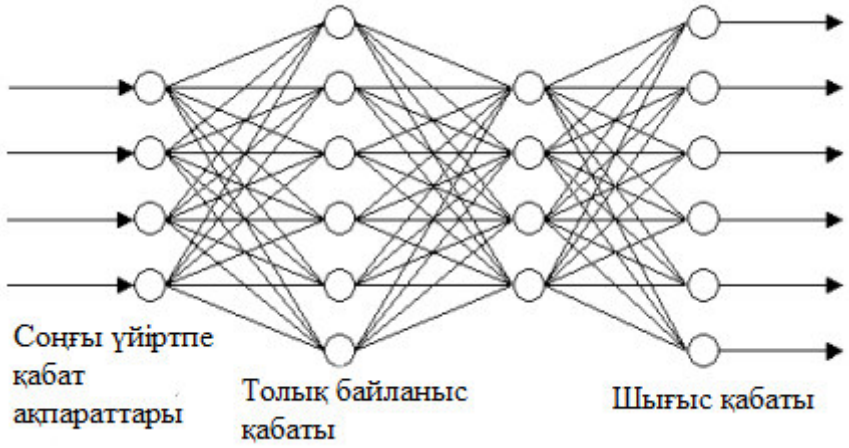
Субпозиция қабатының тиімділігі:

– есептеу жыладғының артуы(кем дегенде 2-есе), яғни бұл карталар санынның азайып отыруына тікелей байланысты;

– керек емес бөлшектерді сүзгіден өткізу;

– жоғарғы деңгейдегі қасеттерді іздеу(келесі қабат үшін).

Толық қосылған және шығыс қабаты. Толық қосылған қабат жіктеу міндетін орындайды. Бұл қабат күрделі сызықты емес функцияны модельдейді, оның параметрлерін тану сапасын жақсартады. Жасырылған қабаттың нейрондарының саны субсамляторлық қабаттың карталарының санына тең, себебі алдыңғы субполяциялық қабаттың әр картасының нейрондары жасырын қабаттың бір нейронымен байланысты.



1.15-сурет – Толық байланыс қабаты

Шығарушы қабаты алдыңғы қабаттың барлық нейрондарына қосылған. Нейрон саны танылатын сыныптар санына сәйкес келеді.

Осы тарауда конволюциялық нейрондық желілердің сәулеттік ерекшеліктері, оларды пайдалану қағидалары және қолдану салалары қарастырылды. Терең конвективті нейрондық желілерді пайдалану көрсетілген алдын-ала өңдеусіз сандық суреттерді тиімді талдау мүмкіндігін береді.

1.7 Бағдарламаны әзірлеу технологияларын сипаттау

Соңғы жылдары машиналық оқыту және жасанды интеллект жүйелері гиометриялық прогрессиямен артуда. Жасанды интеллект жүйелерінің қарқынды дамуы, оның қолдану аясының артуы программистер қатарынан қызғушылық артуына байланысты көптеген жаңа технологиялар мен фреймворктердің пайда болуына негіз болды.

1.7.1 Tensor Flow

Google әзірлеген Tensor Flow - терең оқуды қолдайтын сенімді ашық бастапқы платформа, оған тіпті смартфоннан да қол жеткізуге болады.

Tensor Flow - бұл статистикалық бағдарламаларды құру мен дамытудың тамаша құралы. Негіздеме таратылған оқытуды ұсынғандықтан, барлық AI үлгілері пайдаланушы таңдаған абстракцияның кез-келген деңгейінде әлдеқайда тиімді түрде оқытылатын болады.[10]

Артықшылықтары:

– Tensor Flow python тілінде құрылған.

– бұл фреймворк өте ауқымды есепту күшіне ие. Кез келген GPU қолдануға болады.

– жүйе жасанды интелект модельдерін құру үшін графикалық абстрациялық есептеулерді қолданады.

Кемшіліктері:

– шешім қабылдау мен жауап алу үшін бұл фреймворк бірнеше сатылы кіру ақпараттарын қолданады, бұл біраз уақытты талап етеді.

– жүйеде жасанды интелек желілеріндегі бар модельдер әлі қарастылырмаған.

1.7.2 Microsoft CNTK

Microsoft CNTK мәтінді, хабарларды және дауысты қайта құруды қолдайтын нейрондық желілерге негізделген, жылдам және жан-жақты ашық көзі болып табылады. Платформа бағалаудың сапасына сәйкес компьютердің жедел жалпы бағалауы есебінен тиімді масштабты орта болып табылады.[10]

Microsoft CNTK көптеген деректер массивтерімен біріктірілген, бұл осы негізді өздерінің ең жақсы түрлерін, соның ішінде Skype, Cortana және т.б. сияқты жобалар үшін біріктіреді. Бұдан басқа, бұл құрылым қарапайым және түсінікті құрал болып табылады. онымен тиімді жұмыс істеуге көмектеседі.

Артықшылықтары:

– платформа Python және C++ тілдерін қолданатындықтан, фреймворк бірнеше қызметпен бір уақытта жұмыс істеуге мүмкіндік береді, бұл өз кезегінде оқу үрдісін айтарлықтай жылдамадаты.

– негіздеме жасанды интеллект әлеміндегі соңғы оқиғаларды ескере отырып жасалған. Microsoft CNTK архитектурасы GAN, RNN және CNN-ді қолдайды.

– мұның бәрі бөлінген оқыту арқылы ЖИ үлгілері тиімді оқытуға мүмкіндік береді.

Кемшіліктері:

– фреймворкте визуализация панелі мен мобильді микропроцессорлық архитектураға қолдау жоқ.

1.7.3 Caffe

Caffe - бұл оқыту жүйесіндегі алдын-ала орнатылған жиынтықтарды қамтитын платформа. Бұл жүйе кескінді өңдеу мүмкіндіктері үшін белгілі, ал платформа MATLAB қолданбалы бағдарламалық жасақтамасының пакетін қолдайды.[10]

Артықшылықтары:

– C,C++ және Python тілдерін ұштастыру және қолдау, CNN қолдау (нейрондық желілік технологиялар);

– фреймворк сондай-ақ әртүрлі есеп айырысу мәселелерін шешуге маманданған.

Кемшіліктері:

– Caffe күрделі деректер жиынтығын өңдей алмайды, бірақ визуалды кескін өңдеумен салыстырмалы түрде жылдам.

1.7.4.Theano

Графикалық процессорларды (GPU) пайдалану арқылы, орталық процессорлар (CPU) орнына, Теанодағы жасанды интеллект модельдері үлкен есептік қуатты талап ететін есептеу операциялары кезінде жоғары дәлдікті қамтамасыз етеді. Көп өлшемді деректер массивтерін есептеу үшін сізге жоғары өнімділік қажет, ал Theano оны қамтамасыз ете алады! Theano жылдам пысықтауды және жауапты талап ететін тапсырмаларда ұзақ уақыт бойы жұмыс істейтін Python бағдарламалау тіліне негізделген.[10]

Артықшылықтары:

– Theano барлық деректерді қажет ететін қосымшаларды тиімді қолдауды қамтамасыз етеді, бірақ басқа кітапханалармен біріктіруді талап етеді;

– платформа CPU және GPU-мен жұмыс істеу үшін оңтайландырылған.

Кемшіліктері:

– Theano нұсқасы жаңартуларды шығарып, функционалдықты қосуды жоспарламайды.

1.8 Бағдарламаға арналған технологияны сипаттау

Жоғарыда көрсетілген технологияларды салыстыра отырып, дипломдық жұмысты орындау үшін мен TensorFlow технологиясын таңдадым.

Tensorflow - бұл Google-дің ашық дерек көзімен шығаратын компьютерлік оқыту және таратылатын есептеу техникасы. Нейрондық желілердегі негізгі қосымшаны тапты, бірақ деректерді өндіру және күрделі математикалық есептеулер сияқты басқа да көптеген тапсырмаларға сай келеді.

TensorFlow - машина жасау және Google-ден нейрондық желілерді терең зерттеуге арналған кітапхана, «Machine Intelligence» зерттеу ұйымы коммерциялық емес ұйым болып табылады, оның басты мақсаты қауіпсіз жасанды интеллект жасау, сондай-ақ әлеуетті зерттеу жасанды интеллект жасау кезінде туындауы мүмкін қауіптер мен мүмкіндіктер.

Бұл икемді және ауқымды жүйені смартфондар мен қуатты компьютерлерде пайдалануға болады. Мысалы, бұл кітапхана көптеген қолданбаларда қолданылады: Google Photos, Google Translate, Inbox және басқа бағдарламалар.

1.16-сурет – TensorFlow технологиясын қолдануға болатын бағдарлмалау тілдері

TensorFlow технологиясы ашық түрдегі кітапхана болғандықтан ол өте үлкен ауқымды қолданысқа ие жаңа технология болып табылады. Компьтерлік көру және машиналық оқыту жүйесі бойынша бұл технологияға ұсқсас OpenCv кітапханасы бар. Бірақ менің таңдауым TensorFlow технологиясына түсті. Себебі,

TensorFlow технологиясы жаңа әрі кросплатформалы технологияя болып табылады. Тіпті бұл технологияны қарапайым браузерлерде оқып, жүзеге асыра алады. Қазіргі кездегі ең танмал бағдарламалық тіл ретінде қалыптасқан JavaScrip тілінде де кеңінен қолданылып, браузер арқылы виртуалды кеңісікте біраз жоблар мен бағдарламалар жасалып браузерде қолданылуда.

Мен өзімнің дипломдық жобамда TensorFlow технологиясын Java бағдарламалық тілі негізінде қолданып, android платформасына қолданбалы бағдарлама жазуда пайдаланып, тексерістер жүргіздім.

1.8.1 MobileNet нейрожелі моделі

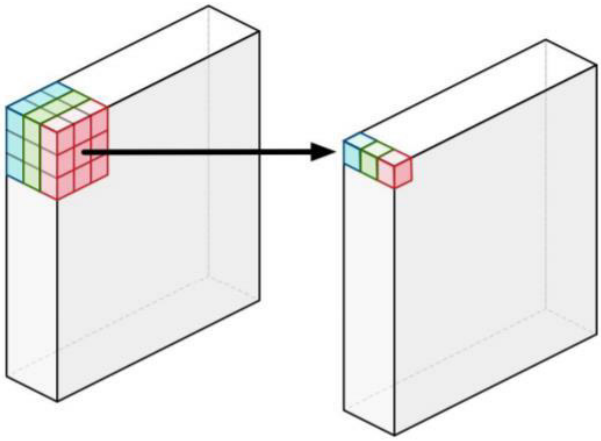
MobileNet (MobileNetwork-дан қысқартылған, мобильді желі) – бұл мобильді және кіріктірілген компьютерлік көру бағдарламаларында қолдануға арналған тиімді нейрондық желілік моделдердің класы.

Бұл архитектура ортақ тереңдіктік конволюцияны (тереңдікті бөлектелген конволюция) пайдалануға негізделген. Көптеген басқа модельдерден айырмашылығы, нейрондық желілердің көлемін азайтудың басты мақсаты - құрылғылардың бір сыныптарына бағытталған, MobileNet-тің дамуының басты мақсаты желінің жұмыс істеуі кезінде туындайтын уақытты кідіртуді азайту және өлшемді азайтудың бүйірлік мақсаты ғана болды. тереңдігі алғашқы бірнеше қабаттарда орындалған есептеулердің көлемін азайтуға мүмкіндік береді, сондықтан осы модельдің басты мақсаты шешіледі.

MobileNet бірінші қабаты кіріс суретінің барлық арналарына стандартты орауды іске асырады. Ұю ядросы әрбір қадамда ядро қамтитын кіріс пиксельдерінің өлшенген сомасын барлық кіріс арналары бойынша есептеп, сурет бойынша жылжиды. Әсіресе маңызды бұл операция барлық кіріс арналарын біріктіреді. Егер сурет 3 кіріс арнасы болса, онда осы сурет бойынша бір ядроны іске қосу 1 арна пиксельге шығу суретіне әкеледі.

Сондықтан әрбір кіріс пикселі үшін, оның қанша арналарына қарамастан, орама тек бір арнамен ғана жаңа Шығыс пикселін жазады (іс жүзінде біз кіріс суретінде орама ядроларының жиынтығын іске қосамыз және олардың әрқайсысы шығуда өз арнасын алады).

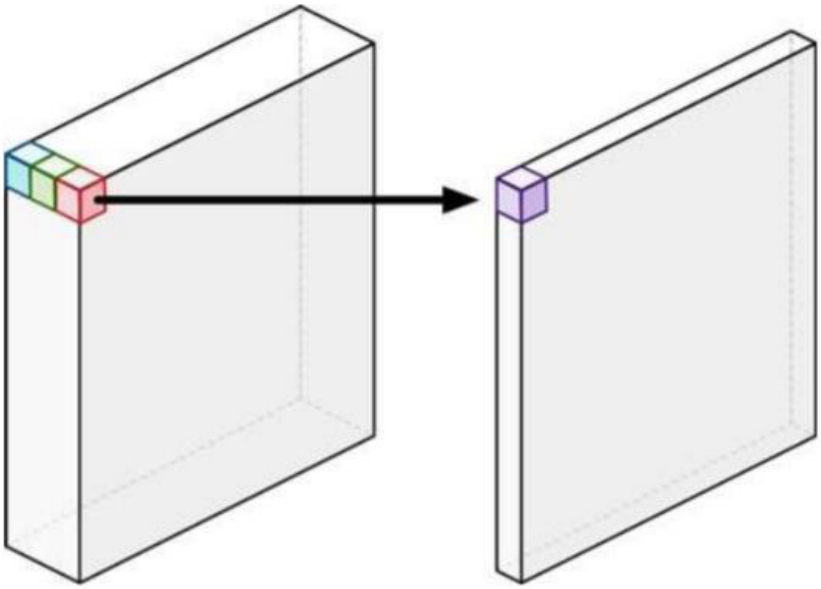
Mobile Net моделі стандартты орауды тек бірінші деңгейде қолданады. Қалған қабаттарда тереңдікке бөлетіг орама қабаттарын қолданады. Ол екі комбинациясын білдіреді: тереңдікке орауыштар және нүктелі орауыштар.
Тереңдікке үйірту операциясының жұмыс 1.17-суретте көрсетілген.



1.17-сурет – Тереңдікке үйірту

Қарапайым үйірту операциясына қарағанда, ол кіріс арналарын біріктірмейді, әр арна бойынша бөлек орауды орындайды. 3 арнасы бар сурет үшін, тереңдікке орау 3 арнасы бар Шығыс суретін жасайды. Әрбір арна алады өз жиынтығы таразы. Тереңдікке ораудың мақсаты кіру арналарын сүзу болып табылады. Бұл шектерді анықтау, түстік сүзу және т.б. есептерде қолданылады.

Егер ол 2-ге тең болса, онда әрбір кіріс арнасы үшін түйреуіш 2 Шығыс арнасын жасайды (және 2 түрлі таразылар жиынтығын қалыптастырады). Бірақ MobileNet бұл параметр емес пайдаланылады. Тереңдікке орауды орындағаннан кейін 1=1 өлшемдік ядросы бар орама операциясын білдіретін нүктелік дәнекерлеу операциясы жүргізіледі, бұл 1.18-суретті көрсетеді.

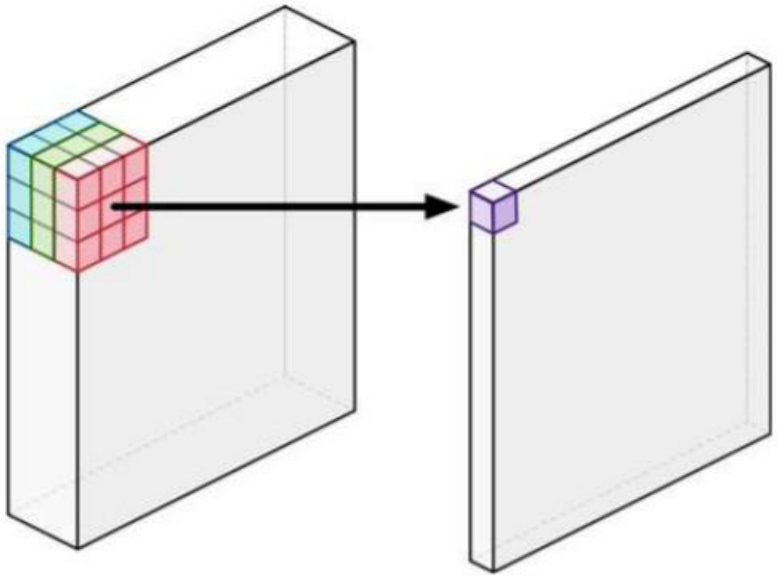


1.18-сурет – Нүктелік үйірту

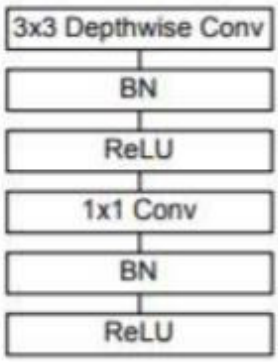
Басқаша айтқанда, мұндай операция барлық арналарды жойып, олардың өлшенген сомасын есептеп шығарады. Стандартты орау жағдайында сияқты, әдетте мұндай ағынды ядролардың жиынтығы шығыс каналдары арқылы бейнені алуға пайдаланылады. Нүктелік үйірті операциясының мақсаты жаңа белгілер картасына қол жеткізу болып табылады

Бұл екі тәсіл біріктірілген кезде-тереңдікке орау және жүйелі ағынды орауышқа бөлінеді. Стандартты ораудан айырмашылығы, ол орындалады сүзу, сондай-ақ бір өтуге біріктіру, бұл екі операция тереңдігі жеке қадам ретінде орындалады. Тереңдікке бөлу операциясы 1.19-суретте көрсетілген.

Екі көзқарастың түпкі нәтижесі өте ұқсас - олар деректерді сүзеді және жаңа белгілерді қалыптастырады, бірақ әдеттегі конволюция дәл осындай нәтижелерге жету үшін есептеудің әлдеқайда көп мөлшерін қажет етеді және ол көп салмақты қажет етеді. Нәтижесінде олар (шамамен) бірдей нәрсені жасаса да, бөлінген терең конволюция жылдамырақ өтеді [11]. 1.20-сурет бөлінген икемділік блогын тереңдікте көрсетеді.



1.19-сурет – Тереңдікті бөліп үйірту операциясы



1.20-сурет – MobileNet архитектурасы

MobileNet бұл ортақ түйісулердің қатарынан 13 қатарға дейін қатарын пайдаланады. MobileNet толық желісі 30 қабатқа ие. Желілік топологиясы өте қарапайым:

– 2 қадаммен конволюция қабаты;
– тереңдіктің қабаты;
– арналар сомасын екі есе көбейтетін нүктелі конвективті қабат;
– 2 қадаммен тереңдіктің қабаты;
– арналар сомасын екі есе көбейтетін нүктелі конвективті қабат;
– қабатты тереңдікте икемдеу;
– айналымдық қабаттағы қабат;
– 2-ші қадаммен қабаттың тереңдігі;
– арналар сомасын екі есе көбейтетін нүктелі конвективті қабат.

Бірінші қабаттан кейін (стандартты үйкеліс) терең және нүктелі қабаттар топтары ауыстырылады. Уақыт бойынша, тереңдік қабаты желінің көмегімен өтетін деректер өлшемін азайту үшін 2 қадам өлшемін алады. Кейде нүктелі қабат арналардың санын екі есе көбейтеді. Барлық конвективті қабаттарды ReLU белсендіру функциясы бақылайды.

Бұл бастапқы өлшемдік кескінге дейін жалғасады 224 × 224 7 × 7 пиксель өлшеміне дейін азайтылмайды, бірақ 1024 арнамен. Осыдан кейін ол барлық суретте жұмыс істейтін орташаның негізіндегі агрегациясы бар қабатқа түседі, оның шығуында 1 × 1 × 1024 өлшемдік суретті аламыз, ол шын мәнінде 1024 элементтен тұратын вектор болып табылады. Егер біз MobileNet-ті классификатор үшін негіз ретінде пайдалансақ, мысалы, 1000 мүмкін кластары бар ImageNet-те, онда соңғы қабат softmaxактивациясы бар және 1000 шығу жолы бар толық байланыс қабаты болып табылады. Егер MobileNet басқа деректер жиынтығында немесе классификатор емес, белгілерді алатын желі ретінде пайдаланылса, онда оның орнына басқа қабатты түпкі ретінде пайдалануға болады.

MobileNet өзі модель болса да,жылдам өңдеуді жүзеге асыру, сонымен қатар, осы архитектураның қиылған нұсқасын пайдалануға мүмкіндік береді. Желінің өлшемін анықтауға арналған үш өлшемді параметрлері бар:

– ен (α) мультипликаторы: арналардың санын реттейді. Егер ен мультипликаторы 1 болса, онда желі 32 арналардан басталып, 1024 аяқталады;

– рұқсат мультипликаторы (ρ): кіріс имиджінің өлшемдеріне жауап береді (кіріс тензорларының кеңістіктік өлшемдері). Әдепкі енгізу өлшемі - 224 × 224 пиксел;

– тереңдігі төмендеуі: толық желіде бес қабаттан тұратын топ бар орташа дәлдігі айтарлықтай жоғалтпай пайдалануға болмайды.

Бұл параметрлер жылдамырақ өңдеуді жүзеге асыратын кішірек желіні жасау үшін пайдаланылуы мүмкін. Алайда мұндай үзілістер желінің дәлдігіне әсер етеді [12, 13]

1.8.2 TensorFlow

TensorFlow - Google әзірлеген ашық бастапқы кітапхана, ол машинада оқыту тапсырмаларын шешу үшін қолданылады.

Кітапхананың мүмкіндіктері әртүрлі архитектуралармен ANN-ті құру және оқытуға мүмкіндік береді, проблемалардың әртүрлі кластарын шешеді (үлгілерді анықтау және тану, қатынастарды іздеу және т.б.) [16].

Өнімділікті арттыру үшін TensorFlow процессор мен графикалық процессор арасындағы міндеттерді бөледі. Жұмыстың жылдамдығын арттыру үшін көптеген операциялар C ++ бағдарламасында Egin және cuDNN кітапханаларын қолдану арқылы жүзеге асырылады.

TensorFlow-тегі әрбір операция есептеулердің қалай орындалатынын сипаттайтын есептік графикті (деректер ағынын кестесі) анықтайды. График қиғаштардан (толтырғыштардан), айнымалылардан (айнымалылардан) және операциялардан (операциядан) тұрады. Ол тензорлар бойынша есептеулерді жүзеге асырады - 0-ден n-ге дейінгі өлшемдердің массивтері.

Графикте әрбір шыңы операцияны жүзеге асыру болып табылады, графиктің шеттері тензорлар болып табылады (кесте құру барысында көрсетілетін ерікті өлшемдердің массивтері). Тәуелділікті бақылайтын арнайы шыңдар болуы мүмкін. Олар алдыңғы түйіннің алдында міндетті екенін көрсетеді келесі біреуді бастамас бұрын есептеуді аяқтау керек.

Әрбір операция - дерексіз сипаттамасы тиісті негізгі операциялары бар есептеулер. Операцияның негізі - нақты құрылғымен, орталық процессормен немесе графикалық процессормен жүзеге асырылуы мүмкін нақты іске асыру.

Айнымалы ток трансформаторы арнайы тензор болып табылады, ол TensorFlow сеансының әр түрлі іске қосылған кезде қолданылуы мүмкін, сондай-ақ сеанстар арасындағы аралық күй. Сонымен қатар айнымалылармен жұмыс

істеу үшін тиісті тензордың мазмұнын өзгерте алатын арнайы операциялар айқындалады.

TensorFlow кітапханасының мүмкіндіктері:

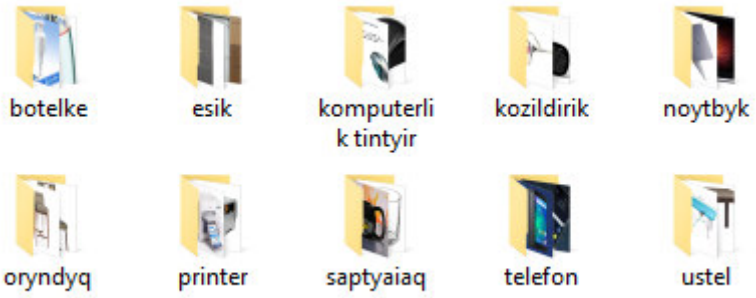
– кітапхана терең әдіс ретінде қолдануға жарамды;
– оқыту және машина жасаудың өзге де әдістері;
– ыңғайлы тіркеу жүйелерін, визуализацияларды қамтиды;
– оқу үдерісі және есептік графиктер;
– төмендегілер үшін кросс-платформа жүзеге асырылған (Linux, MacOS, Windows, Android, iOS);
– бірнеше тілде бағдарламалау интерфейсі бар (Python, C ++, Java, Go, Haskell).

2 Бағдарламаны жобалау

2.1 Әзірленіп жатқан бағдарламаның сипаттамасы

Дипломдық бағдарламаны жобалау екі кезеңнен тұрады. Бірінші кезең бағдарламаға қажетті деректер қорын қалыптастыру. Екінші кезең осы деректер қорын бағдарламада қолдану. Деректер қоры ретінде бағдарламаға объектілер жайлы ақпарат қажет. Дипломдық жұмыс тапсырмасы бойынша үйіртпелі нейрондық желі негізінде объектілерні тану және анықтау болатын. Ол үшін біз ең алдымен қарапайым нейрондық желі ядросына алып, мен бұл жайғдайда mobilenet_0.50 ядросын алдым. Осы қарапайым нейронды желі ядросы негізінде оған үйіртпе қабаттар қосу арқылы қарайпайым желіге жаңа объектілерді тануды оқыттым ол процесс төменде көрсетілген.

Ең алдымен біз керекті объектілердің бейнелерін жинақтаймыз.



2.1-сурет – Объектілер жайлы ақпарат

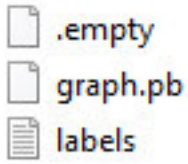
Жоғарыдағы суретте мен жаңадан оқытылатын объектілер бейнелерін жинақтап алдым. Бұл біздің қарапайым нейрондық желі үстіне оқытылатын жаңай үйіртпе қабаттар арқылы алынатын жаңа объектілер. Бұл ақпараттар арқылы біз жаңа класстар аламыз.

Үйіртпелі желі оқыту төмендегі скрипт арқылы жүзеге асырылады. Бұл кезең біз қажетті ақпараттарды жинақтаған соң жүзеге асырылады.

```
    //retrain
    python -m scripts.retrain --bottleneck_dir=tf_files/bottlenecks --
how_many_training_steps=6000                                        --
model_dir=tf_files/models/"mobilenet_0.50_224"                      --
summaries_dir=tf_files/training_summaries/"mobilenet_0.50_224"      --
learning_rate=0.00001    --output_graph=tf_files/retrained_graph.pb --
output_labels=tf_files/retrained_labels.txt                         --
architecture="mobilenet_0.50_224" --image_dir=tf_files/newdataset
    //check retrained on image
```

```
python -m tensorflow.python.tools.optimize_for_inference
  --input=tf_files/graph.pb
  --output=tf_files/optimized_graph.pb
  --input_names="input"
  --output_names="final_result"
```

Жоғарыдағы скрипт жүзеге асып болған соң біз төмендегі файлдарды аламыз.(2.2-сурет). Келесі кезеңде бағдарламамызға осы алынған файлдарды қолданып жаңа классификатор құраймыз.



2.2-сурет – Нейронды желі негізінде оқытылған
жаңа үйіртпелі нейронды желі файлдары

2.2 UML диаграммалмары

2.2.1 UML тілі

UML - бірыңғай модельдендіру тілі. Оны құруға индустрияның қандай да бір дәрежесінде болмасын барлық салалары қатысты болғандықтан, "UML -бұл бағдарламалық жүйелердің артефактілерін көрсету, спецификациялау, конструкциялау және құжаттандыру, сондай-ақ бизнес-процесстердің және бағдарламалық емес жүйелердің тілі. UML «міндеттерінің» тізбесіне басты назарды аудара кетейік. Спецификацциялау, көрсету, конструкциялау және құжаттандыру - бұлардың барлығы да жоғары деңгейлі жобалауға тікелей қатысты болып отыр[20]. Ал жоғары деңгейлі «құрал-саймандар» қолданудың бір-бірегей аспектісі болуы мүмкін емес, сондықтан, UML анықтамасын келесідегі көпаспектілі интерпретациямен толықтыруға болады:

– UML әдіс есебінде жүйелердің қозғалысын тану үшін пайдаланылады;
– UML тіл есебінде пәндік аймақ туралы білімді «есептен шығару» үшін падаланылады;
– UML модельдендіру тілі ретінде жүйелердің байланысу заңдылықтарын түсіну үшін (және мүмкін формальдауда) пайдаланылады;
– UML бірыңғайландыру түрі ретінде құрастырушылардың қызметін үйлестіру үшін пайдаланылады.

Көптеген программисттердің көзқарасы көрсетіп отырғандай, жобаны іс ке асыру мәселесі бойынша ойтолғаулар шамаменоған код жазу үшін балама болып

отыр. Кейбір заттар бағдарламаландырудың қандай да бір болмасын кодында тікелей алғанда өте жақсы мәнерленеді, себебі программаның мәтіні - бұл алгоритмдерді және өрнектерді жазу үшін өте қысқа және қарапайым жол[20].

Бірақ мұндай жағдайлардың өзінде программист бейресми болса да модельдендірумен айналысады. Ол өз ойының жобасын тақтада немесе майлықта жазды делік. Бірақ мұндай жақындасу жағымсыз лаң туғызуы мүмкін. Біріншіден, концепциялық модельге қатысты сылтаулар бойынша ой алмасуларға қатысатын пікірталастың қатысушылары бір тілде сөйлесетін кезде ғана мүмкін болады. Жобаларда жасау кезінде компаниялар сөздерінің тілін жаңалық есебінде ашуы тиіс болатындығы ереже сияқты болып кеткен.

Бағдарламалық құралдарды шығаратын компания, орындалатын кодпен қатар басқа да артефактілерді шығарады, соның ішінде келесідегілерді:

- жүйеге қойылатын талаптар;
- архитектура;
- жоба;
- бастапқы код;
- жобалау жоспарлары;
- тесттер;
- прототип;
- болжам нұсқамалары, және т.б.

Өңдеу жөнінде қабылданған әдістемеге байланыссыз бір жұмысты орындау басқаларға қарағанда ресмиірек жүргізіледі.Айтылған артефактілер -бұл жобаның құрамалы бөлігінің жеткізушілері ғана емес, олар басқару үшін, нәтижені бағалау үшін, сондай-ақ жүйені жасау және оған өрбіту жасап болған соңғы уақытта ұжым мүшелері арасындағы қарым-қатынас құралы есебінде қажет. UML жүйелік архитектураны және барлық оның бөлшектерін құжаттандыру жөніндегі проблеманы шешуге мүмкіндік береді, Жүйеге қойылатын талаптарды тұжырымдау және тесттерді анықтау үшін тілді ұсынады, және соңында жобаны жоспарлау және болжамдарды басқару кезеңінде жұмыстарды модельдендіру үшін құралдарды ұсынады. UML тілі біріншіден бағдарламалық жүйелерді жасауға арналған[21].

UML қолдану аясы бағдарламалық қамтамасыз етуді модельдендірумен шектелмейді. Оның мәнерлілігі заң жүйесіндегі құжаттандыру айналымын, ауруханалардағы емделушілерге қызмет көрсету жүйесінің құрылымын және қызмет көрсетуін, аппарат құралдарына жобалау жасауды модельдендіруге мүмкіндік береді. UML жасауда басты болып келесідегі мақсаттар танылған:

– пайдаланушыларға мағыналы модельдерді жасауға және олармен алмасуға мүмкіндік беретін, модельдендіруді көрсетушіліктің мәнерлі тілін қолдануға дайын етіп ұсыну;

– базалық тұжырымдаманы кеңейту үшін кеңейту және мамандандыру механизмдерін қарастыру;

– бағдарламаландырудың нақты тілдерінен және жасау процестерінен тәуелсіз болуын қамтамасыз ету;

– модельдендірудің осы тілін түсіну үшін формальды негізін қамтамасыз ету.

UML патенттелген құрал болып табылмайды және барлығы үшін ашық. Ол өзі негізделіп жасалған әдістерді пайдаланудың тәжірибе жүзінде расталған ғылыми қоғамдастықтары мен тұтынушыларының қажеттілігін қанағаттандыру үшін тағайындалған. Методология бойынша көптеген мамандар, ұйымдар және инструменталды құрал-жабдықтарды тасымалдаушылар бұл тілді пайдаланамыз деп міндеттеме алған. UML Бучтың, OMT, OOSE әдістемелерінде және басқа да озық әдістерінде пайдаланылуы сияқты, сондай-ақ UML және кең қауымдастық серіктестерінің ұсыныстарын енгізетіндіктен, семантиканың және нотацияның негізінде құрылғандықтан бұл тілді мойындап, қадірлеу кең, жаратынды түрде болуы тиіс.

UML атауындағы «бірыңғайлау» эпитетінің екі аспектісі бар. Біріншіден, ол модельдендірудің ертеректегі әдістерінің тілдері арасындағы маңызды емес көптеген ерекшеліктерді іс жүзінде жоққа шығарады. Екіншіден, мүмкін аса ерекше шығар, ол жүйелердің көптеген түрлі түрлерінің болашағын, жасау фазаларын (талаптарды талдау, жобалау және іске асыру) және ішкі тұжырымдарды бірыңғайластырады (олармен байланысты бағдарламалық қамтамасыз етуді емес, бизнесті).

Дегенмен, UML нақты тілді анықтаса да, бұл модельдендірудің тұжырымдарын болашақта жетілдіру үшін тосқауыл бола алмайды. UML жасау кезінде көптеген озық әдістер назарға алынған болатын, бірақ UML келешектегі болжамына өзге де әдістер өз ықпалын тигізетін болады.

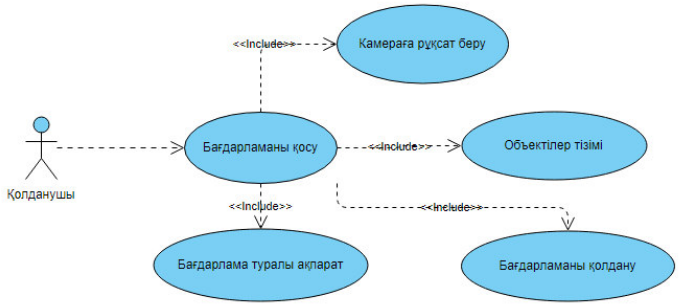
Сонымен қатар, UML негізінде жаңа перспективалық (болашақ) әдістер анықталуы мүмкін. UML оның ұйытқысын қайта анықтамастан ақ кеңейтілуі мүмкін. Объектілік технологияларды дамытуда 1989 жылы құрылған Object Management Group (OMG) консорциумының құрылуы басты негіз болып отыр, оның мақсаты - интероперальды біртекті емес бөлінген объектілік орталарды құру үшін индустриалды стандарттарды жасау болып табылады.

Осы мақсатта OMG қабылдаған көрсеткіштердің нәтижесінде 1997 жылдың қыркүйек айында қабылданған модельдендірудің бірыңғайландырылған тілі (UML) деп аталған тіл стандартының қабылдануы болды[21].

UML стандартының негізін объектілік талдау және жобалау жөніндегі аты әйгілі технологияларының үш ойларымен жинақталған Г.Буч, OOSE И.Якобсона және OMT Д.Рэмбо технологияларының Rational Software компанияларының ұсыныстарына бірқатар ірі корпорациялардың қолдау көрсеткендігі табылды.

2.2.1 Прецеденттер диаграммасы

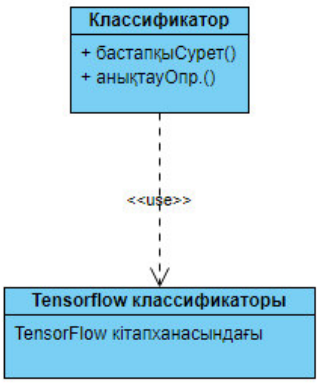
Ең алдымен прецеденттер диаграмасынан бастайық. Бұл диаграммада қолданушының бағдарламамен жұмыс істеу процессі көрсетілген(2.3-сурет).



2.3-сурет – Прецеденттер диаграммасы

Диграммада қолданушының бағдарламаны қолдану мүмкіндіктері көрсетілген. Ең алдымен қолданушы бағдарламаны іске қосу барысында бағдарлама қолданушыдан бағдарлама қолданылып отырған девайстың камерасына рұқсат сұрайды. Егер қолданушы оған рұқсат бермеген жағдайда бағдарламаны толық қолдануға мүмкіндік болмайды. Қолдану камера рұқсат бергеннен соң бағдарламаның барлық функциясына рұсат беріледі. Бағдарлам үйірпетлі нейрондық желі негізінде объектілерді тануға мүмкіндік береді. Диаграммадан көріп тұрғанымыздай бағдарламада объектілер тізімін көруге мүмкіндік бар. Бұл объектілер тізімі алдын ала үйіртпелі нейрондық желі негізінде оқытылып дайындалып алынған объектілер. Және де қолданушы бағдарламы туралы ақпаратты да көре алады.

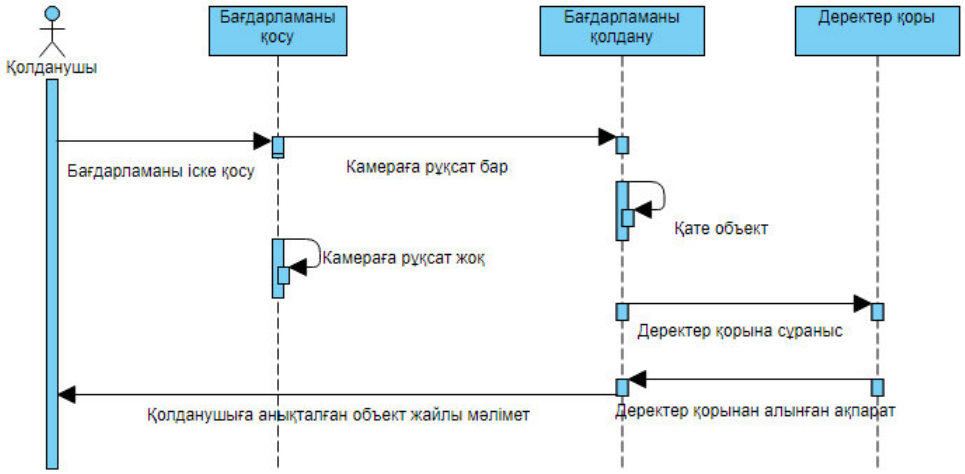
2.2.2 Класстар диаграммасы



2.4-сурет – Класстар диграммасы

2.4-суретте бағдарламада қолданылатын класстар диаграммасы көрсетілген. Бағдарламаны жобалау барысында біз дайын кітапханалардан көптеген дайын класстарды қолданамыз. Бірақ ол класстардың жұмыс істеу объектілері мен функциялары белгілі болғандықтан бұл жобалау кезеңінде оларға сипаттама бермеуді шештім. Бұл кезеңде тек жаңадан қолданылатын классқа сипаттама берілген. Диграммадан көріп тұрғанымыздай біз классификатор классын құрамыз. Бұл класс бізге үйрітпелі нейронды желі негізінде оқытылған объектер классымен жұмыс істейді. Осы құрылған класс біздің бағдарламамыздың негізі ядросы болып табылады. Бағдарламаның ары қарайғы жұмыс істеу процессінің барлығы осы классқа байланысты. Біздің бағдарламамызда. TensorFlow технологиясы да қолданылғандықтан біздің классификатор классымен тығыз байланысты.

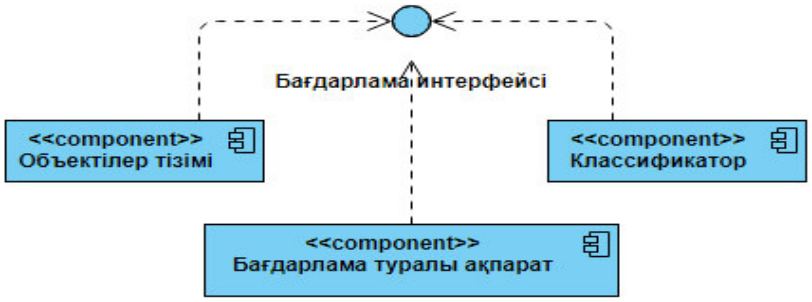
2.2.3 Реттілік диаграммасы



2.5 - сурет – Реттілік диаграммасы

Реттілік диаграммасында қолданушының бағдарламамен жұмыс істеу әркеттерінің кезегі көрсетілген. Диаграммаға қарасақ ең алдымен қолданушы бағдарламаны іске қосады. Бағдарлама қолданушыдан девайс камерасына рұқсат сұрайды. Рұқсат берілген жағдайда бағдарлама ары қарай өз жұмысын жасайды. Егер рұқсат берілмеген жағдайда бағдарлама толық функциналды болып жұмыс жасамайды. Камера рұқсат берілген соң, қолданушы бағдараманы қолдануына мүмкіндік алады. Бағдарламаны қолдану барысында бағдарлама қолданушы көрсеткен объектіні суретке түсіріп оны деректер қорындағы объектілермен салысытырып қолданушыға ол объект деректер қорында болса ол туралы толық ақпаратты шығарып береді. Егер болмаған жағдайда қате не болмаса ол объект туралы жалған ақпарат шығуы мүмкін.

2.2.4 Компонеттер диаграммасы



2.6 - сурет – Компонеттер диаграммасы

Компоненттер диаграммасында біздің бағдарламамызда бар компоненттер көрсетілген. Бағдарлама интерфейсі 3 компоненттен тұрады. 1-ші ол объектілер тізімі, яғни біздің бағдарламамыздың деректер қорындағы объектілер тізімі. 2-ші бағдарлама туралы ақпарат. 3-ші бағдарлама ядросы, яғни классификатор объектіні анықтау терезесі.

3 Бағдарламаны іске асыру

3.1 Бағдарлама интерфейсі

Бағдармламаны смартфоннан іске қосқаннан кейін бағдарлма интерфейсі төмендегідей болады(3.1-сурет).

3.1 - сурет – Бағдарлама интерфейсі

Суретте көріп тұрғанымыздай бағдарлама 3 компоненттен тұрады:
– OBEKTTER TIZIMI;
– ANYQTAP TANY;
– BAGDARLAMA TYRALY.
Енді осы компоненттерге жеке-жеке тоқталып өтейік.

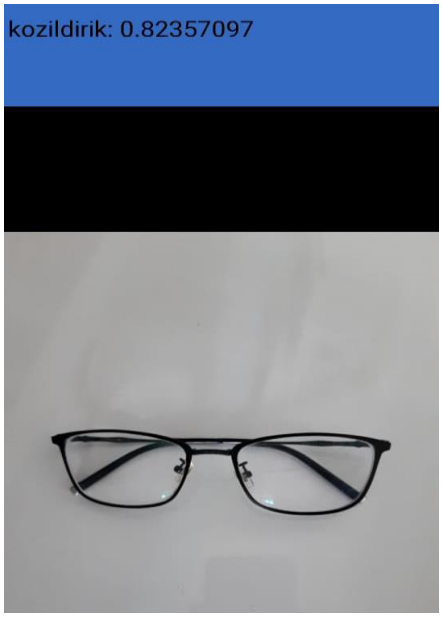
3.1.1. OBEKTTER TIZIMI

3.2 - сурет – OBEKTTER TIZIMI

3.2-суретте бағдаралма ішінде бар объекттер тізімі көрсетілген. Яғни, қазіргі кездегі бағдарлама анықтай алатын объектілер тізімі. Бұл тізім әзірше 10 объектіден тұрады. Оны сұранысқа байланысты арттырып отыруға болады

3.1.2. ANYQTAP TANY

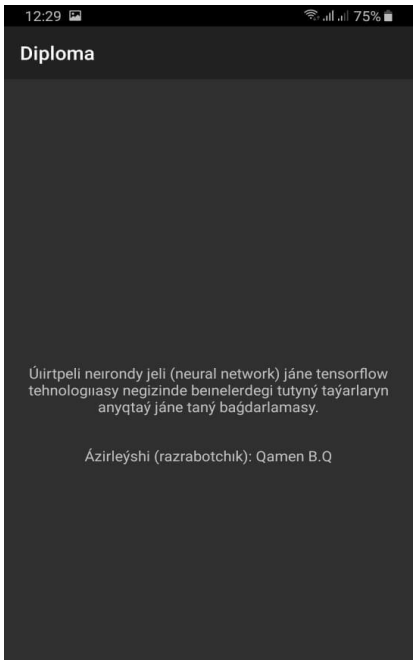
3.3-суретте бағдарлама жұмыс істеу барысы көрсетілген. Суретте бағдарлама көзілдірікті тауып оны анықтағана бейнеленген.



3.3-сурет – ANYQTAP TANY

3.1.3. BAGDARLAMA TYRALY

Бұл терезе арқылы қолдану бағдарлама туралы толық ақпаратты ала алады(3.4-сурет).



3.4-сурет – BAGDARLAMA TYRALY

3.2 Бағдарлама арқылы жүргізілген зерттеулер.

Бағдарламның объектілерді тану және анықтау дәлдігіен тексеру барысында бірнеше объектілермен зерттеулер жүргізілді. Оның нәтижесі төмендегі кестеде көрсетілген(3.1-кесте).

3.1-кесте – Тәжірбие нәтижелері

№	Объект	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Көзілдірік	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
2	Ноутбук	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	0	1
3	Бөтелке	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Кестедегі алынған ақпараттар бойынша бағдарламаның әрбір объект үшін дәлідігін тексеріп көрейік:

Көзілдірік

$$X = \frac{100 * 19}{20} = 95\%$$

Ноутбук

$$X = \frac{100 * 18}{20} = 90\%$$

Бөтелке

$$X = \frac{100 * 19}{20} = 95\%$$

Орташа дәлдік:

$$X = \frac{95 + 90 + 95}{3} = 93{,}33\%$$

Бағдарламаның объектілерді тану орташа дәлідігі 93,33% ды құрады.

4 Экономикалық бөлім

4.1 Бағдарламалық қамтамасыз етуді әзірлеудің еңбек сыйымдылығын есептеу

4.1-кесте - Жұмыстарды кезеңдер мен түрлер бойынша бөлу және олардың еңбек сыйымдылығын бағалау

БӨ әзірлеу кезеңдері	Жұмыс түрі	Әзірлеудің еңбек сыйымдылығы, адам× сағ.
1 кезең	Техникалық тапсырманы құрастыру	10
2 кезең	Орындау үшін қажетті деректерді жинау	15
3 кезең	Қосымшаны жобалау	20
4 кезең	Қосымша үшін базаны әзірлеу	50
5 кезең	Қосымшаның құрылымын әзірлеу	40
6 кезең	Бағдарламаны жазу	100
7 кезең	Сынақ жүргізу	70
8 кезең	Есеп және талдау	15
Бағдарламалық өнімді орындаудың еңбек сыйымдылығы		320

Бағдарламалық қамтамасыз етуді әзірлеуге арналған шығындарды есептеу. Бағдарламалық қамтамасыз етуді әзірлеуге арналған шығындарды анықтау тиісті сметаны жасау жолымен жүргізіледі, ол мынадай баптарды қамтиды:

- материалдық шығындар;
- еңбекке ақы төлеу шығындары;
- әлеуметтік салық;
- негізгі қорлардың амортизациясы;
- өзге де шығындар.

4.2-кесте - Материалдық шығындар

Материалдық ресурс атауы	Өлшем бірлігі	Саны	Бір данаға шаққандағы құн, тг	Сомасы, тг
Ноутбук Asus ROG J771	Дана	1	300000	300000
Смартфон Samsung A8+	Дана	1	150000	150000
Материалдық шығын қорытындысы				450000

Алматы қаласы бойынша 2019 жылы заңды тұлғалар үшін Тариф ҚҚС есебімен 1 кВт/с үшін 18,32 теңгені құрайды ("АлматыЭнергоСбыт"ЖШС ресми сайтында ұсынылған деректерге сәйкес). Шығындар 3-кестеде көрсетілген.

Электр энергиясына жұмсалатын шығындардың жалпы сомасы (ЗЭ) мынадай формула бойынша есептеледі:

$$З_э = \sum_{i=1}^{n} M_i \cdot K_i \cdot T_i \cdot Ц \qquad (4.1)$$

мұндағы, Mi – I электр жабдығының паспорттық қуаты, кВт;

Кі – электр жабдығының қуатын пайдалану коэффициенті (Кі=0.70.9 қабылданады);

Ті – І-ші жабдықтың барлық әзірлеу кезеңіндегі жұмыс уақыты;

Ц – электр энергиясының бағасы, тг / кВт×сағ;

і – электр жабдығының түрі;

n – электр жабдықтарының саны.

4.3-кесте – Электроэнергия шығындары

Құрал-жабдық атауы	Паспорттық қуат, кВт	Қолдану қуатының коэфициенті	Әзірлеуге жұмсалған құрал-жабдық уақыты, сағ.	Электро энергия бағасы, тг/кВт ч	Сомасы, тг
Ноутбук Asus ROG J771	0.8	0.9	250	18,32	4580
Смартфон Samsung A8+	0.6	0.7	70	18,32	1282,4
Электроэнергияға жұмсалған шығын қорытындысы					5862,4

4.2 Бағдарламалық қамтамасыз етуді әзірлеуге арналған шығындарды есептеу

Android-әзірлеушінің орташа жалақысы 2019 жылы 200000 теңгені құрайды (Алматы қаласы үшін).

Қызметкердің бір айдағы жұмыс сағаттары мынадай формула бойынша анықталады:

$$Чм=Nм \cdot Чрд ,$$

мұндағы, Чм – бір айдағы қызметкердің жұмыс сағаты;
Nм – бір айдағы жұмыс күндерінің саны(23);
Чр – күндегі жұмыс сағаттарының саны(8).

$$Чм=23 \cdot 8=184$$

Қызметкердің сағаттық ставкасы мынадай формула бойынша есептеледі:

$$Ч_М = N_М \cdot Ч_{рд} \quad (4.2)$$

Android-әзірлеуші:

$$ЧС_i = \frac{200000}{184} = 1086{,}95 \text{ тг}$$

мұндағы, $ЗП_i$ – і қызметкердің айлық жалақысы, тгтг;
$ФРВ_i$ – і қызметкердің жұмыс уақытының айлық қоры, сағ.

БӨ әзірлеудің еңбек сыйымдылығын анықтау үшін 1-кестедегі деректер пайдаланылады.

Android әзірлеушінің БӨ әзірлеуінің еңбек сыйымдылығы 320 адам×сағ. техникалық тапсырманы құрастыру, орындау үшін қажетті деректерді жинау, қосымшаларды жобалау, қосымшалар үшін деректер базасын әзірлеу, қосымшалардың струтурасын әзірлеу, қосымшаларды жазу, талдау және есеп беру).

$$T_1 = 10 + 15 + 20 + 50 + 40 + 100 + 70 + 15 = 320$$

Еңбекақы төлеуге жұмсалатын шығындардың жалпы сомасы ($З_{ТР}$) мынадай формула бойынша анықталады:

$$З_{тр} = \sum_{i=1}^{n} ЧС_i \cdot T_i, \quad (4.3)$$

мұндағы, ЧСі – і қызметкердің сағаттық ставкасы, тг;
Ті – ПП әзірлеудің еңбек сыйымдылығы, адам×сағ;
І – қызметкердің санаты;

n – ПҚ әзірлеумен айналысатын қызметкерлердің саны.

Android-әзірлеуші:

$$З_{тр} = 1086{,}95 \cdot 320 = 347824 \text{ тг}$$

4.4-кесте – Еңбек ақы шығыны

Біліктілігі	БӨ әзірлеудің еңбек сыйымдылығы, адам×сағ	Сағаттық ставка, тг/сағ	Сомасы, тг
Android-әзірлеуші	320	1086,95	347824
Еңбек ақы шыныдарының қорытындысы			347824

Қосымша еңбекақы:

$$З_{доп} = З_{тр} \cdot 10\%$$
$$З_{доп} = 347824 \cdot 0{,}1 = 34782{,}4 \text{ тг}$$

Еңбекақы қоры:

$$Ф_{зп} = З_{тр.о} + З_{доп}$$
$$Ф_{зп} = 3478264 + 34782{,}4 \ = 382606{,}4 \text{ тг}$$

Әлеуметтік салықты есептеу:

$$Н_{с} = (Ф_{зп} - ОПВ) \cdot 11\%$$

мұндағы, ОПЗ - міндетті зейнетақы жарналары - жалақы жобасының 10%.

$$Н_{с} = \left(382606{,}4 \ - (382606{,}4 \ \cdot 0.1)\right) \cdot 0.11 = 37878{,}03 \text{ тг}$$

4.2.1 Негізгі қорлардың амортизациясын есептеу

Амортизациялық аударымдардың жалпы сомасы мынадай формула бойынша анықталады:

$$З_{АМ} \ = \sum_{i=1}^{n} \frac{Ф_{i} \cdot Н_{Ai} \cdot Т_{НИРi}}{100 \cdot Т_{Эфi}}, \qquad (4.4)$$

мұндағы, $Ф_i$ – ОФ құны, тг;

$Н_{Ai}$ – ОФ амортизациясының жылдық нормасы, %;

$Т_{НИРi}$ – БӨ әзірлеудің барлық кезеңіндегі і-ОФ жұмыс уақыты, сағ.;

$Т_{Эфi}$ – і-ОФ бір жылдағы жұмыс уақытының тиімді қоры, сағ / жыл;

і – ОФ түрі;

n – ОФ саны.

4.2.2 ОФ амортизациясының жылдық нормасын есептеу

Құрал-жабдықтар:

$$Н_{Ai} = \frac{100}{Т_{Ni}} \quad (4.5)$$

$$Н_{Ai} = \frac{100}{4} = 25$$

мұндағы, $Т_{Ni}$ - ОФ пайдаланудың ықтимал мерзімі, жыл;

БӨ әзірлеу үшін жұмыс уақытын анықтау үшін 5 кестедегі деректер пайдаланылады.

БӨ әзірлеу үшін Android Studio бойынша жұмыс уақыты 100 сағатты құрайды (қосымшаларды жазу).

Құрал-жабдықтар:

$$З_{АМН} = \frac{300000 \cdot 25 \cdot 250}{100 \cdot 1920} = 9765{,}62 \text{ тг}$$

$$З_{АМН} = \frac{150000 \cdot 25 \cdot 100}{100 \cdot 1920} = 1953{,}25 \text{ тг}$$

4.5-кесте – Негізгі қорлардың амортизациясы (ОФ)

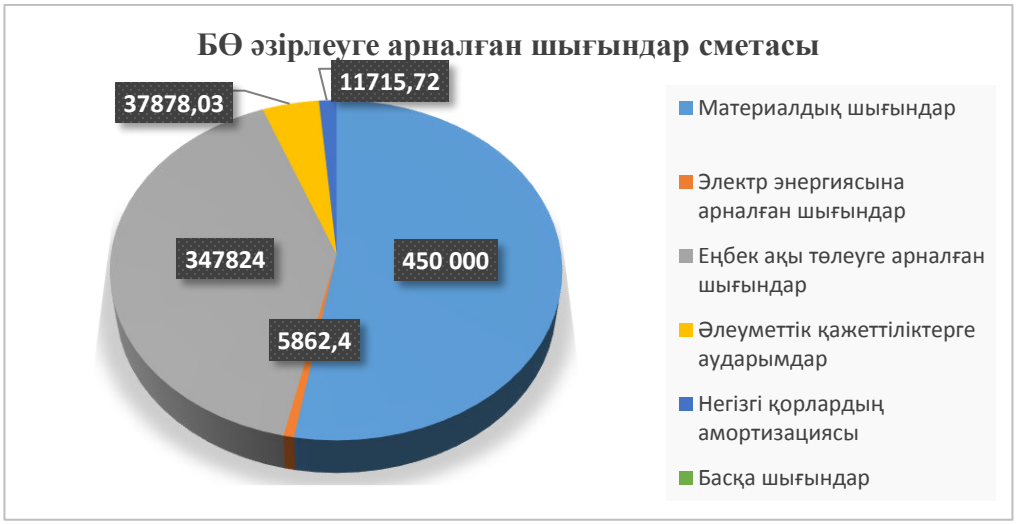
БҚ мен құрал-жабдық атауы	БҚ мен құрал-жабдық құны, тг	Жылдық амортизация нормасы, %	Жабдықтың және БҚ жұмыс уақытының тиімді қоры, сағ / жыл	БӨ әзірлеуге арналған жабдық және БҚ жұмыс уақыты, адам	Сомасы, тг
Ноутбук Asus	300000	25	1920	250	9765,52

4.5-кестенің жалғасы

БҚ мен құрал-жабдық атауы	БҚ мен құрал-жабдық құны, тг	Жылдық амортизация нормасы, %	Жабдықтың және БҚ жұмыс уақытының тиімді қоры, сағ / жыл	БӨ әзірлеуге арналған жабдық және БҚ жұмыс уақыты, адам	Сомасы, тг
Смартфон Samsung A8+	150000	25	1920	70	1953,25
Қорытынды					11715,72

4.6-кесте – БӨ әзірлеуге арналған шығындар сметасы

Шығын атаулары	Сомасы, тг
1. Материалдық шығындар	450 000
2. Электр энергиясына арналған шығындар	5862,4
3. Еңбек ақы төлеуге арналған шығындар	347824
4. Әлеуметтік қажеттіліктерге аударымдар	37878,03
5. Негізгі қорлардың амортизациясы	11715,72
6. Басқа шығындар	0
Қорытынды	853279,46



5.1-сурет – БӨ әзірлеуге арналған шығындар сметасы

4.2.3 БӨ ықтимал (шарттық) бағасын анықтау

Шарттық бағаны есептеу (ЦД):

$$Ц_д = З_{НИР} \cdot (1 + \frac{Р}{100}), \quad (4.6)$$

мұндағы, $З_{НИР}$ - әзірлеуге арналған шығындар БӨ (6-кесте), тг;
Р - ПП табыстылығының орташа деңгейі - 20% .

$$Ц_д = 853279,46 + 170655,892 = 1023935,352тг$$

2019 жылға ҚҚС ставкасы 12% мөлшерінде белгіленген.
ҚҚС есебімен өткізу бағасы мынадай формула бойынша есептеледі:

$$Цр=Цд + Цд * НДС$$

$$Ц_р = 1023935,352 + 1023935,352 \cdot 0.12 = 1146807,94245 тг$$

4.3 Экономикалық бөлім бойынша қорытынды

Әзірленген қосымша тұтыну тауарларын тану процесін автоматтандыруға мүмкіндік береді. Бұл бөлімде экономикалық тиімділік тұрғысынан әзірленген бағдарламалық өнімге талдау жасалды. Бағдарламалық өнімді жасау бойынша жұмысты орындау мерзімі мен еңбек сыйымдылығы анықталды. Өндірілген есептеулер мен көрсеткіштер бұл өнімді тұтыну тауарларын тануды автоматтандыру үшін сатудың көптеген салаларында пайдалануға болады деп айтуға мүмкіндік береді.

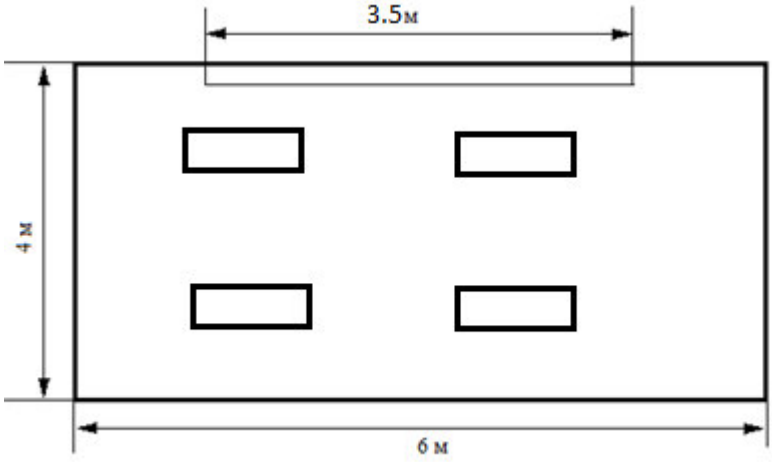
Осылайша, өткізу бағасы ҚҚС есебімен 1146807,94245 тг тең, өзіндік құн 853279,46 тг тең, рентабельділік(пайда) 170655,892 тг тең.

5 Өміртіршілік қауіпсіздігі

Дипломдық жобаның тақырыбы: «Үйіртпелі нейрондық желілер негізінде бейнелердегі тұтыну тауарларын анықтау және тану бағдарламасын әзірлеу». Жобаны әзірлеу кезінде программист жұмыс кабинетінде болады. Бұл кабинетте жұмыс орнының желдету, шу және эргономикасы талаптарға сәйкес келеді. Ал жарықтандыру жеткіліксіз болды. Сондықтан дипломдық жобаның осы бөлімінде бөлмені жарықтандыруды есептеу және программист үшін қолайлы жағдай жасау туралы шешім қабылданды.

5.1 Жарықты есептеу

Әзірлеуші орналасқан жұмыс кабинеті ұзындығы 6 м, ені 4 м және биіктігі 3 м., биіктігі 1,4 м, терезенің басталу биіктігі 0,6 м (5.1-сурет).



5.1-сурет – Әзірлеуші орналасқан жұмыс кабинетінің сұлбасы

Бұл бөлмеде бір терезе бар: ені 2,5 м және биіктігі 1,4 м.Жарық ойықтарының ауданы S=3,5 м2 бөлмені қалыпты жарықтандырылуы үшін осы терезе жеткілікті ма екенін тексереміз. Есептеу мынадай формула бойынша бүйірлік жарықтандыру кезінде жарық ойықтарының ауданын алдын ала анықтаудан тұрады.

$$100\frac{S_0}{S_n} = \frac{\mathrm{e}_N K_з \eta_0}{\tau_0 r_1} K_{зд} \qquad 5.1)$$

мұндағы, S_0 – бүйірлік жарықтандыруы бар жарық саңылауларының алаңы, м^2;

S_n – бөлме ауданы, м^2;

$e_н$ – КЕО-ның нормаланған мәні;

$K_з = 1{,}2$ – қабылданған жеңіл таратушы материалдың тік орналасуымен қауіпсіздік коэффициенті [14].

τ_0 – жалпы өткізгіштігі;

r_1 – бөлменің бетінен және ғимаратқа іргелес жатқан қабатынан көрінетін жарықтың КЭО-нің өсуін ескере отырып алынатын коэффициент;

$K_{зд} = 1$ – ғимараттарға қарсыласу арқылы терезелердің қараңғы болуын ескере отырып коэффициент [14].

Еден аумағы формула бойынша анықталады:

$$S_n = L * B \tag{5.2}$$

Осылайша, (4.2) формула бойынша еден ауданы тең

$$L = 6 \text{ м},$$
$$B = 4 \text{ м},$$
$$S_n = 6 * 4 = 24 \text{ м}^2$$

Табиғи жарық коэффициентінің нормаланған мәні формула бойынша есептеледі

$$e_N = e_н * m_N \tag{5.3}$$

мұндағы, N - табиғи жарықтың болуына қарай әкімшілік-аумақтық аймақтың топтарының саны;

бөлме және табиғи жарықтандыру жүйесінде көрнекі жұмыстарды ерекшеліктеріне қарай СНиП 23-05-95 арқылы таңдалатын коэффициент табиғи жарық, яғни $e_н = 1{,}5$. [14];

$m_N = 0{,}65$ – оңтүстігінде сыртқы қабырғасында жарық саңылау түріне, көкжиек жағынан олардың бағдар және жеңіл климаттың коэффициентін әкімшілік ауданы тобы санына байланысты коэфициенті;

(4.3) формулаға сәйкес, әртүрлі аумақтарда орналасқан ғимараттар үшін КЕО e_N нормаланған мәндері тең:

$$e_N = e_н * m_N = 1.5 * 0.65 = 0.975\ \%$$

Бір жақты жарықтандырылған бөлменің тереңдігі

$$l = B - 1 = 4 - 1 = 3 \text{ м}$$

η_0 жарық сипаттамасының мәнін алу үшін келесі қатынастар есептелді

$$\frac{L}{l} = \frac{6}{3} = 2$$

$$\frac{l}{h_{\text{расч}}} = \frac{l}{(h_{\text{ок}} + h_{\text{нок}}) - h_{\text{рп}}} = \frac{3}{(0.7 + 0.8) - 1} = 6$$

Жарық сипаттамасының мәні кестеден алынып тасталады. Алынған мәндер кестедегі мәндерге сәйкес келмегендіктен, интерполяция жасаймыз

$$x = 1.3 + \frac{(15 - 13) * (6 - 5)}{7{,}5 - 5} = 13.8$$

$$\eta_0 = 13.8$$

Жалпы жарық өткізгіштік τ_0 формула бойынша анықталады

$$\tau_0 = \tau_1 * \tau_2 * \tau_3 * \tau_4 \quad (5.4)$$

мұндағы τ_1– кестеге сай анықталған материалды жарық беру коэффициенті. Терезе парақтары үшін: бір, τ_1= 0.9;

τ_2 – терезенің байланыстарында жарықтың жоғалуын ескере отырып коэффициент. Ағаш бір терезе рамалары үшін $\tau_2 = 0.75$.

τ_3 – тірек конструкцияларындағы жарық жоғалуын есепке ала отырып, кестеге сәйкес бүйірлік жарықтандыру 0,8-ке тең болған коэффициент;

τ_4 = 1 – кестеге сай [14] кестеге сәйкес күн сәулесінен қорғайтын құрылғылардағы жарықтың жоғалуын ескере отырып коэффициент.

(5.4) формулада белгілі мәндермен алмастырамыз, τ_0 табамыз:

$$\tau_0 = \tau_1 * \tau_2 * \tau_3 * \tau_4 = 0.9 * 0.75 * 0.8 * 1 = 0.54$$

кестесіне сәйкес жанама жарықтандыру үшін r_1 коэффициентін анықтаңыз. Бұл үшін біз:

бөлменің тереңдігін әдеттегі жұмыс бетінің деңгейінен терезенің жоғарғы жағына дейін қатынасы:

$$\frac{l}{h_{\text{расч}}} = \frac{3}{1} = 6$$

есептік нүкте мен сыртқы қабырға арасындағы бөлменің тереңдігіне ара қатынасы:

$$\frac{l}{B} = \frac{3}{4} = 0.75$$

бөлменің ұзындығына тереңдікке қатынасы:

$$\frac{L}{l} = \frac{6}{3} = 2$$

Төбенің, қабырғалардың және едендің орташа өлшенген коэффициенті формула бойынша анықталады

$$\rho = \frac{\rho_{\text{пот}} + \rho_{\text{ст}} + \rho_{\text{пол}}}{3} \qquad (5.5)$$

Осылайша, формула бойынша (5.5)

$$\rho = \frac{\rho_{\text{пот}} + \rho_{\text{ст}} + \rho_{\text{пол}}}{3} = \frac{50 + 30 + 10}{3} = 30\% = 0.3$$

Алынған мәндер кестедегі мәндерге сәйкес келмейтіндіктен r_1, біз интерполяция жасаймыз

0.8 – 0.6	0.75 – 0.6
1.9 – 1.5	x – 1.5

$$x = 1.5 + \frac{(0.75 - 0.6) * (1.9 - 1.5)}{0.8 - 0.6} = 1.8$$

мұндағы, $r_1 = 1.8$.

Формула бойынша біз S_0 жарықтандырумен жарық саңылауларының аумағын табамыз

$$S_0 = \frac{S_n e_N K_з \eta_0 K_{зд}}{100 \tau_0 r_1} = \frac{24 * 0.975 * 1.2 * 13.8 * 1}{100 * 0.54 * 1.8} = 3.987 \text{ м}^2 \approx 4 \text{ м}^2$$

Бұл бөлмеде жарық ашу аумағы тексерілді, үйде және бұл жеткіліксіз. Осы деректерден қосымша жарық көздері қажет деп қорытынды жасауға болады, яғни. жасанды жарықтандыруды есептеу қажет.

5.2 Жасанды жарықтандыруды есептеу

Бөлмеде жасанды жарықтандыру жалпы жарықтандыру аспаптарында 3050 лм жарық ағыны бар 4 дана 65 Вт люминесцентті лампалармен жүзеге асырылады. 300 л-нен жарықтандыру үшін қажетті кәдеге жарату коэффициентін пайдаланып шамдардың санын есептеп шығарамыз, бұл III-ші санаттағы визуалды жұмыстарды қамтамасыз ету үшін жеткілікті.

Жақыртандыру көздерін анықтау үшін келесі формуланы пайдаланамыз:

$$N_Л = \frac{S_{помещ} K_з Z E_Н}{\Phi_Л n \eta} \quad (5.6)$$

мұндағы, ең төменгі жарықтандыру коэффициенті (орташа және минималды жарықтылық арақатынасы). Есептеулерде z коэффициенті 1.1 ÷ 1.2 ауқымында қабылданады;

n = 2 - шамдағы шамдар саны;

η - көрініс коэффициенттеріне және бөлме индексіне байланысты пайдалану коэффициенті.

$K_з$ – қауіпсіздік факторы. Шаң бөлетін құрылғылар жоқ бөлме үшін $K_з$= 1,5.

Жарықтандыру қондырғысын пайдалану кезінде жұмыс беттеріндегі жарықтандыру жарық көздерінің жарық сәулесін, лампалар мен жарықтандыру құрылғыларының ластануын, сондай-ақ жарықтандырылған бөлменің қабырғаларын және төбесін ластау арқылы азайтылады. Сондықтан, жарықтандыру қондырғысының қуатын анықтау кезінде қауіпсіздік коэффициенті енгізіледі. Қауіпсіздік коэффициенті ауаның ластану деңгейіне байланысты шаңнан, түтіннен, түктің және т.б.

Біркелкі жарықтылықты сипаттайтын Z факторы көптеген айнымалылардың функциясы болып табылады және шамдар арасынан ілгіштің (L / h) есептік биіктікке дейінгі арақатынасына тәуелді болады, бұл ретте ұсынылған Z мәндері күрт артады. Λ ұсынылған мәннен асып кетпегенде, сіз:

Z = 1,15 - қыздыру және DRL үшін;

Z = 1.1 - флуоресцентті лампалар үшін [15].

Жарықтандыру қондырғыларын пайдалану коэффициенті - сәуле түсетін ағынның жұмыс бетіне көзден шыққан жарық ағынына қатынасы.

Пайдалану жылдамдығын анықтау үшін, бөлме индексін анықтау керек:

$$i = \frac{S_{\text{п}}}{(L+B)h_{\text{расч}}} = \frac{A*B}{(L+B)h_{\text{расч}}} \quad (5.7)$$

мұндағы, S, L, B тиісінше ауданы, ұзындығы және ені.

Барлық мәндерді қойып келесіні нәтижені аламыз

$$i = \frac{A*B}{(L+B)h_{\text{расч}}} = \frac{6*4}{(6+4)*2{,}2} = 1.1$$

$$\eta = 0.42$$

Есептеулерді жүргізу барысында жарықтың ағын көрсеткіші төмендегі формула бойынша есептелінеді:

$$\text{Фл} = \frac{\text{E}_{\text{min}} \cdot \text{S} \cdot \text{Z} \cdot \text{K}_{\text{з}}}{\text{N} \cdot \eta \cdot n} \quad (5.8)$$

мұндағы, E_{min} - ең төменгі қалыпты жарықтандыру болып табылады, лк;

k - қауіпсіздік факторы (люменесцентті лампалар үшін, k = 1.3);

S – жарықтандырылған алаң, м^2;

Z-ең аз жарықтандыру коэффициенті (біркелкі жарықтылық коэффициенті);

N - шамдар саны;

n - шамдағы шамдар саны;

h – жарықтандыру ағымының бірлік өлшеу коэфициенті.

Жасанды жарықтың жеткілікті болуын тексеру үшін сіз Emin-ді есептеуіңіз керек, ол 300-ден астам болуы керек.

$$\text{E}_{\text{min}} = \frac{\Phi_{\text{л}} \cdot \text{N} \cdot \text{n} \cdot \eta_{\text{з}}}{\text{S} \cdot \text{Z} \cdot K} = \frac{3050 \cdot 4 \cdot 2 \cdot 0.42}{24 \cdot 1.1 \cdot 1.5} = 260$$

Есептеулер бойынша E_{min} бізге керек шамадан аз. Сондықтан біз бұл мәселені шешіудің жолданыр қарастырамыз. Мен бұл мәселені шешу үшін бұрын жұмыс кабинетінде орналасқан жарық ағыны 3050лм құрайтын 65Вт люменсцентік шамдарды жаңа жарық ағыны 3560лм 80Вт люменсценті

шамдармен алмастырдым. Жаңа шамдарды ауыстырғаннан кейін қайта проверил E_{min} тексердім:

$$E_{min} = \frac{\Phi_л \cdot N \cdot n \cdot \eta_з}{S \cdot Z \cdot K} = \frac{3560 \cdot 4 \cdot 2 \cdot 0.42}{24 \cdot 1.1 \cdot 1.5} = 302$$

Нәтижесінде біз жайлы бөлмені жарықтандыруды қамтамасыз ету үшін жеткілікті болды, шамдарды күшейту керек, яғни, қуаты 65 ватт артық шамды таңдаңыз. Осылайша жалпы жарықтандыру үшін 3560 лм жарық ағыны бар 80 Вт люминесцентті шамдар таңдалды.

Формулаға сәйкес біз жарық шамдарын анықтау үшін жарық шамдарын анықтау үшін 300 лкс шамасында жарық шамдарын анықтауымыз керек, 3560 лм

$$N = \frac{S_{помещ} K_з Z E_Н}{\Phi_Л n \eta} = \frac{24 * 1{,}5 * 1{,}1 * 300}{3560 * 2 * 0{,}42} = 3{,}973 \approx 4$$

Формула бойынша бізге 4 шам қажет.

Шамдардың орналасу биіктігін анықтау қажет:

$$h_{расч} = H_{пом} - (H_{св} + H_{р.п.}),$$

мұндағы, $H_{св} = 0$ – шамдардың көтерілу биіктігі, м;

$H_{р.п.} = 0{,}8$ – еденнен жұмыс бетінің қашықтығы, м;

$H_{пом} = = 3$ - бөлме биіктігі, м.

Содан кейін аспалы шамдардың биіктігі

$$h_{расч} = H_{пом} - \left(H_{св} + H_{р.п.}\right) = 3 - (0 + 0{,}8) = 2{,}2 \text{ м}$$

Формула бойынша лампалар арасындағы қажетті қашықтықты анықтаймыз:

$$L = \lambda * h$$

мұндағы, L - көршілес шамдар немесе флуоресцентті лампалар арасындағы қашықтық;

h - шамның жұмыс жасайтын жерінен асып кетуінің биіктігі.

Шамдарды люменесцентті шамдармен қолданған кезде

$$\lambda = 1.2 \div 2.4$$

Осылайша, шамдар арасындағы қажетті қашықтық

$$L = \lambda * h = 1{,}5 * 2{,}2 = 3{,}3 \text{ м}$$

Лампалардың арасындағы қашықтық:

$$Lb=\lambda*hp=2{,}4\cdot 0{,}8 = 2 \text{ м}.$$

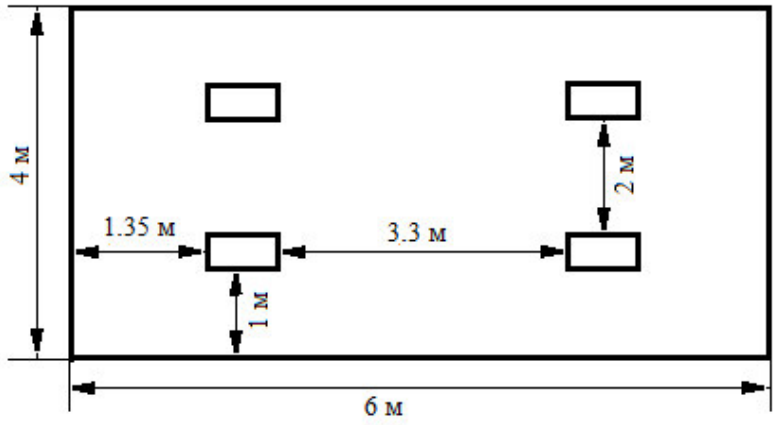
Шам мен қабырға арасындағы қашықтық:

$$la=lb=Lb/3 + [0{,}3;0{,}5] =2{,}5/3+0{,}47 =1{,}3 \text{ м}.$$

Шамдар арасындағы қашықтық:

$$La= Lb - 1=2 - 1=1 \text{ м}.$$

Шамдар мен жарық саңылауларының орналасуы 5.2 суретте көрсетілген:



5.2-сурет – Шамдар мен жарық саңлауларының орналасуы

Жүргізілген есептеулер негізінде жасанды жарықтандыру қайта жаңартылды. Жарық ағыны 3050 лм болатын 65Вт-дан ескі люминесцентті шамдарды 80вт-дан жарық ағыны 3560 лм жаңа люминесцентті шамдарға ауыстыру туралы шешім қабылданды.

Қорытынды

Дипломдық жұмыстық орындау барысында келесі нәтижелерге қол жеткіздік:

– бейнелердегі тұтыну тауарларын тану және анықтау тапсырмасы бойынша мәселені шешу негізінде үйіртпелі желі архитектуралары мен жұмыс істеу процесстері қарастырылып тиімді нейронды желі анықталды;

– Mobilenet нейронды желі ядросы бойынша үйіртпелі нейрондық желі арқылы жаңа объектілер класстары құрастырылды;

– құрастырылыған класстар мәліметтері бойынша бағдарламаның ақпарат қоры жинақталды;

– Android платформасына құрылған класстар бойынша бағдарлама жазылды;

– бағдарламаны әртүрлі объектілерді анықтау барысында зерттеу жұмыстары жүргізіліп, нәтижелер бойынша бағдарлама дәлдігі анықталды;

Әдебиеттер тізімі

1 Хайкин Саймон. Нейронные сети: полный курс, 2-е издание. —Москва: Вильямс, 2006. — 1104 с.

2 Джонс, М. Т. Программирование искусственного интеллекта в приложениях / М. Т. Джонс — 2-е изд. — М. : ДМК-Пресс, 2011 .— 313 с

3 Каллан Р. Основные концепции нейронных сетей = The Essence of Neural Networks First Edition. — 1-е. — «Вильямс», 2001. — С. 288.

4 R Hahnloser, R. Sarpeshkar, M A Mahowald, R. J. Douglas, H.S. Seung (2000). Digital selection and analogue amplification coexist in a cortex-inspired

5 Тарик Рашид. Создаем нейронную сеть. : Пер. с англ. — СПб. : ООО “Альфа-книга”, 2017. — 272 с. : ил. — Парал. тит. англ. - С. 51-61.

6 Elman Jeffrey L. Finding structure in time // Cognitive science. — 1990. — Vol. 14, no. 2. — Pp. 179–211.

7 Hochreiter Sepp, Schmidhuber J¨urgen. Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — Pp. 1735–1780.

8 LeCun Y., Bottou L., Bengio Y., Haffner P.. Gradient-based learning applied to document recognition//Proceedings of the IEEE. – 1998. – Т. 86. – №. 11. – С. 2278-2324

9 [LeCun Y., Bengio Y., Hinton G. Deep learning //nature. – 2015. – Т.521. – №. 7553. – С. 436.]

10 Frameworks in the World of Artificial Intelligence by Abhishek Kothari | may 20, 2018

11 Andrew G. Howard, Menglong Zhu, Bo Chen and others. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision // arXiv. – 2017 – 17 April

12 Matthijs Holleman. Google’s MobileNets on the iPhone // Machine, Think! – 2017

13 Roryorangepants. MobileNet: меньше, быстрее, точнее // Хабр. – 2018

14 Хакимжанов Т.Е. Расчет аспирационных систем. Дипломное проектирование. Для студентов всех форм обучения всех специальностей. – Алматы: АИЭС, 2002. – 30 с.

15 Бекишева А.И. Методические указания к выполнению экономической части дипломной работы для бакалавров специальности 5В0703 Информационне системы – Алматы: АУЭС; 2013. –24 с

16 TensorFlow [Электрондық ресурс]. URL: www.tensorflow.org

17 Ширяев В.И. Финансовые рынки: нейронные сети, хаос и нелинейная динамика. - М.: Либроком, 2009. 230с.

18 Волчихин В. И. Основы обучения искусственных нейронных сетей. Учебное пособие. -Пенза: Изд-во Пенз.гос.ун-та, 2004.-110 с.

19 Р. Рохас. Нейронные сети: систематическое введение / Р. Рохас; Forew. Фельдман, - Берлин [и др.]: Springer, 1996. - стр. 502.

20 Боггс Уэнди, Майкл Боггс «UML и Rational Rose 2002» /Пер. с англ. – М. «Лори», 2004

21 Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. М.: ДМК, 2000

А қосымшасы
(міндетті)

Техникалық тапсырма

А.1 Талаптарды талдау

Бірінші кезекте қосымшаны қанағаттандыруы тиіс талаптар анықталды:

– қолданба ұялы құрылғы камерасынан ағысты(ақпаратты) оқу керек;

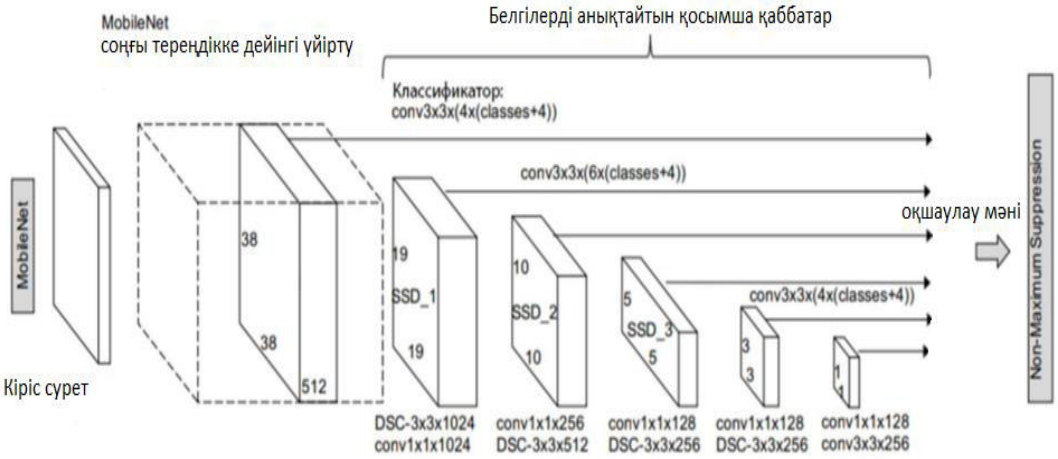
– әр оқылған кадр TesnsorFlow технологиясына түсінікті матрицалы сандарға түрлендіріледі;

– алынған матрицаны нейрожелі кірісіне беру арқылы қолданбада тұтыну тауарларын тану мүмкіндігінің болуы;

– нейрожелі моделінің құрылуы.

А.2 Нейрожелі моделі

Дипломдық жоба барысында А.1-суретте көрсетілген нейрожелі моделі құрылды.



А.1-сурет – Нейрожелі архитектурасы

Бұл модель келесі принцип бойынша жұмыс істейді: сурет, соңғы қабат болып бөлінетін ораудың соңғы қабаты болып табылатын MobileNet біріктірілген моделінің кіруіне түседі. Оның белгілері шығыс карталары бір уақытта детекция және жіктеу блогына, сондай-ақ бөлінетін ораудың келесі қабатына түседі.

Деректер картаның өлшемі 1x1xN құрайтын қабатқа жеткенше процесс қайталанады, мұнда N-белгілер саны. Бұл қабатта бөлінетін орауды жүргізудің қажеті жоқ, сондықтан белгілер карталары тікелей детекция және жіктеу блогына түседі.

А.3 Бағдарламаны әзірлеу

Бағдарламаны әзірлеу үшін TensorFlow кітапханасы және Python тілінде оқытылған MobileNet нейронды желі моделі қолданылды.

Бағдарлама әзірлеу бірінші бізге керекті классификаторды оқытудан негіз алады. Біз ең алдымен Python тілінде жазылған скрипт арылы MobileNet нейро желі модельіне жаңа үйірту қабаттарын қосу арқылы жаңа объектілерді оқытып жаңа классификатор аламыз. Төмендегі скрипт осы процесті жүзеге асырады:

```
//retrain

python -m scripts.retrain --bottleneck_dir=tf_files/bottlenecks --how_many_training_steps=6000 --model_dir=tf_files/models/"mobilenet_0.50_224" --summaries_dir=tf_files/training_summaries/"mobilenet_0.50_224" --learning_rate=0.00001 --output_graph=tf_files/retrained_graph.pb --output_labels=tf_files/retrained_labels.txt --architecture="mobilenet_0.50_224" --image_dir=tf_files/newdataset

//check retrained on image

python -m tensorflow.python.tools.optimize_for_inference
  --input=tf_files/graph.pb
  --output=tf_files/optimized_graph.pb
  --input_names="input"
  --output_names="final_result"
```

Классификатор оқытылып дайын болғаннан кейін біз оны android-studio қосымшасы арқылы android платформасына арналған бағдарлама жасау үшін қоладанмыз.

А.4 Android-studio қосымшасын іске асыру

Камерадан сурет ағыны нейрожелінің кірісіне беріледі. Содан кейін біздің құрылған нейрожелі ядросы арқылы объект үйірту операциясынан өтеді. Толық өңдеуден өткен сурет қолданушыға мобильді телефон экранына сол объект жайлы ақпаратты қайтарады.

Бағдарлама интерфейсі келесідей болады.

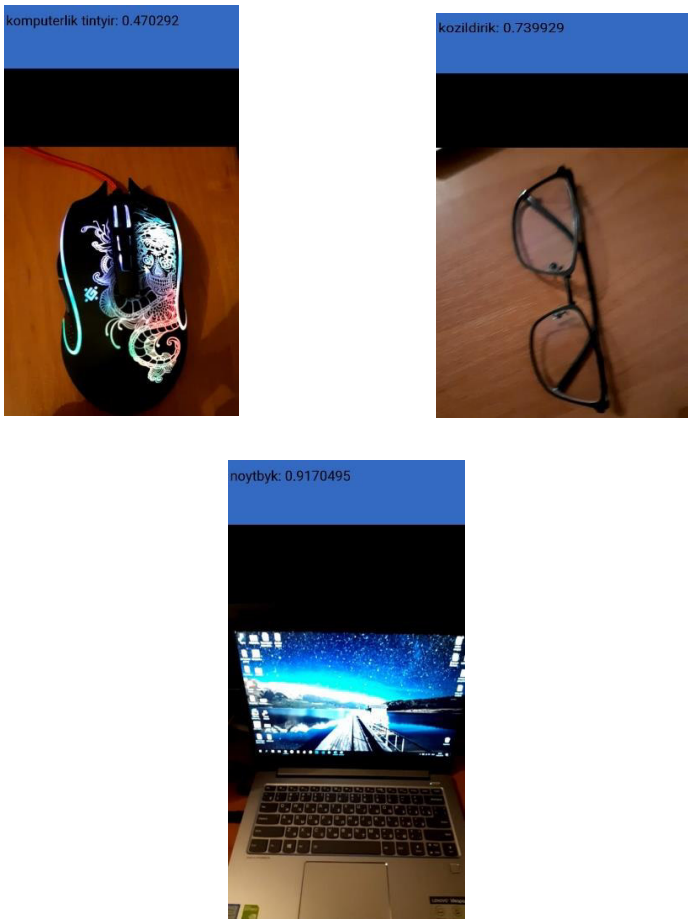
Объектілер тізімі. Бұл компонентте қолданба анықтай алатын барлық объектілер тізімі толықтай көрсетіледі.

Анықтап тану. Бұл мәзір қолданбаны толық жұмыс істеу алаңы. Яғни бағдарламаның жұмыс істеуі.

Бағдарлама туралы. Қолданба жайлы барлық ақпаратты көру мүмкіндігі.

А.5 Қолданбаның жұмыс істеу процесі

Төмендегі суреттерде қолданбаның жұмыс істеу процесі көрсетілген.



А.2-сурет – Қолданбаның жұмыс істеу процессі

Б қосымшасы
(міндетті)

Программа листингі

```
package org.tensorflow.demo;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Button object_list = (Button) findViewById(R.id.object_list);
      Button scan = (Button) findViewById(R.id.scan);
      final Button about_app = (Button) findViewById(R.id.about_app);
      object_list.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            showObjectList();
         }
      });
      scan.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            scan();
         }
      });
      about_app.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            showAppInfo();
         }
      });
   }


   private void showObjectList() {
```

```
        startActivity(new Intent(this,ObjectList.class));
    }
    private void showAppInfo() {
        startActivity(new Intent(this,AboutAppActivity.class));
    }
    private void scan() {
        startActivity(new Intent(this,ClassifierActivity.class));
    }
}
```

```
package org.tensorflow.demo;

import android.graphics.Bitmap;
import android.graphics.RectF;
import java.util.List;

/**
 * Generic interface for interacting with different recognition engines.
 */
public interface Classifier {
  /**
   * An immutable result returned by a Classifier describing what was recognized.
   */
  public class Recognition {
    /**
     * A unique identifier for what has been recognized. Specific to the class, not the instance of
     * the object.
     */
    private final String id;

    /**
     * Display name for the recognition.
     */
    private final String title;
    /**
     * A sortable score for how good the recognition is relative to others. Higher should be better.
     */

    private final Float confidence;
```

```
/** Optional location within the source image for the location of the recognized object. */
private RectF location;

public Recognition(
    final String id, final String title, final Float confidence, final RectF location) {
  this.id = id;
  this.title = title;
  this.confidence = confidence;
  this.location = location;
}

public String getId() {
  return id;
}

public String getTitle() {
  return title;
}

public Float getConfidence() {
  return confidence;
}

public RectF getLocation() {
  return new RectF(location);
}

public void setLocation(RectF location) {
  this.location = location;
}
@Override
public String toString() {
  String resultString = "";
  if (id != null) {
    resultString += "[" + id + "] ";
  }


  if (title != null) {
    resultString += title + " ";
  }
```

```
    if (confidence != null) {
      resultString += String.format("(%.1f%%) ", confidence * 100.0f);
    }

    if (location != null) {
      resultString += location + " ";
    }

    return resultString.trim();
   }
  }

  List<Recognition> recognizeImage(Bitmap bitmap);

  void enableStatLogging(final boolean debug);

  String getStatString();

  void close();
}
```

```
package org.tensorflow.demo;

import android.content.res.AssetManager;
import android.graphics.Bitmap;
import android.os.Trace;
import android.util.Log;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.PriorityQueue;
import java.util.Vector;


/** A classifier specialized to label images using TensorFlow. */
public class TensorFlowImageClassifier implements Classifier {
  private static final String TAG = "TensorFlowImageClassifier";
  // Only return this many results with at least this confidence.
```

```
private static final int MAX_RESULTS = 3;
private static final float THRESHOLD = 0.1f;
// Config values.
private String inputName;
private String outputName;
private int inputSize;
private int imageMean;
private float imageStd;

// Pre-allocated buffers.
private Vector<String> labels = new Vector<String>();
private int[] intValues;
private float[] floatValues;
private float[] outputs;
private String[] outputNames;



private boolean logStats = false;

private TensorFlowInferenceInterface inferenceInterface;

private TensorFlowImageClassifier() {}

/**
 * Initializes a native TensorFlow session for classifying images.
 *
 * @param assetManager The asset manager to be used to load assets.
 * @param modelFilename The filepath of the model GraphDef protocol buffer.
 * @param labelFilename The filepath of label file for classes.
 * @param inputSize The input size. A square image of inputSize x inputSize is assumed.
 * @param imageMean The assumed mean of the image values.
 * @param imageStd The assumed std of the image values.
 * @param inputName The label of the image input node.
 * @param outputName The label of the output node.
 * @throws IOException
 */
public static Classifier create(
    AssetManager assetManager,
    String modelFilename,
    String labelFilename,
    int inputSize,
```

```
  int imageMean,
  float imageStd,
  String inputName,
  String outputName) {
TensorFlowImageClassifier c = new TensorFlowImageClassifier();
c.inputName = inputName;
c.outputName = outputName;

// Read the label names into memory.
// TODO(andrewharp): make this handle non-assets.
String actualFilename = labelFilename.split("file:///android_asset/")[1];
Log.i(TAG, "Reading labels from: " + actualFilename);
BufferedReader br = null;
try {
  br = new BufferedReader(new InputStreamReader(assetManager.open(actualFilename)));
  String line;
  while ((line = br.readLine()) != null) {
    c.labels.add(line);
  }
  br.close();
} catch (IOException e) {
  throw new RuntimeException("Problem reading label file!" , e);
}

c.inferenceInterface = new TensorFlowInferenceInterface(assetManager, modelFilename);

// The shape of the output is [N, NUM_CLASSES], where N is the batch size.
final Operation = c.inferenceInterface.graphOperation(outputName);
final int numClasses = (int) operation.output(0).shape().size(1);
Log.i(TAG, "Read " + c.labels.size() + " labels, output layer size is " + numClasses);

// Ideally, inputSize could have been retrieved from the shape of the input operation.  Alas,
// the placeholder node for input in the graphdef typically used does not specify a shape, so it
// must be passed in as a parameter.

c.inputSize = inputSize;
c.imageMean = imageMean;
c.imageStd = imageStd;

// Pre-allocate buffers.

c.outputNames = new String[] {outputName};
c.intValues = new int[inputSize * inputSize];
```

```
  c.floatValues = new float[inputSize * inputSize * 3];
  c.outputs = new float[numClasses];

  return c;
 }

 @Override
 public List<Recognition> recognizeImage(final Bitmap bitmap) {
  // Log this method so that it can be analyzed with systrace.
  Trace.beginSection("recognizeImage");

  Trace.beginSection("preprocessBitmap");
  // Preprocess the image data from 0-255 int to normalized float based
  // on the provided parameters.
  bitmap.getPixels(intValues, 0, bitmap.getWidth(), 0, 0, bitmap.getWidth(), bitmap.getHeight());
  for (int i = 0; i < intValues.length; ++i) {
   final int val = intValues[i];
   floatValues[i * 3 + 0] = (((val >> 16) & 0xFF) - imageMean) / imageStd;
   floatValues[i * 3 + 1] = (((val >> 8) & 0xFF) - imageMean) / imageStd;
   floatValues[i * 3 + 2] = ((val & 0xFF) - imageMean) / imageStd;
  }
  Trace.endSection();

  // Copy the input data into TensorFlow.
  Trace.beginSection("feed");
  inferenceInterface.feed(inputName, floatValues, 1, inputSize, inputSize, 3);
  Trace.endSection();
  // Run the inference call.
  Trace.beginSection("run");
  inferenceInterface.run(outputNames, logStats);
  Trace.endSection();

  // Copy the output Tensor back into the output array.
  Trace.beginSection("fetch");
  inferenceInterface.fetch(outputName, outputs);
  Trace.endSection();



  // Find the best classifications.
  PriorityQueue<Recognition> pq =
     new PriorityQueue<Recognition>(
         3,
```

```
        new Comparator<Recognition>() {
          @Override
          public int compare(Recognition lhs, Recognition rhs) {
            // Intentionally reversed to put high confidence at the head of the queue.
            return Float.compare(rhs.getConfidence(), lhs.getConfidence());
          }
        });
  for (int i = 0; i < outputs.length; ++i) {
    if (outputs[i] > THRESHOLD) {
      pq.add(
          new Recognition(
              "" + i, labels.size() > i ? labels.get(i) : "unknown", outputs[i], null));
    }
  }
  final ArrayList<Recognition> recognitions = new ArrayList<Recognition>();
  int recognitionsSize = Math.min(pq.size(), MAX_RESULTS);
  for (int i = 0; i < recognitionsSize; ++i) {
    recognitions.add(pq.poll());
  }
  Trace.endSection(); // "recognizeImage"
  return recognitions;
}

@Override
public void enableStatLogging(boolean logStats) {
  this.logStats = logStats;
}
```

В қосымшасы.
(міндетті)

Енгізу актісі