AZƏRBAYCAN MİLLİ ELMLƏR AKADEMİYASI

İdarəetmə Sistemləri İnstitutu

Əlyazması hüququnda

Əzimov Rüstəm Bakir oğlu

"Kompüter tanıma sistemində istifadəsi üçün təsvirlərdəki obyektlərin əlamətlərinin hesablama alqoritmlərinin və proqram təminatının işlənilməsi" mövzusunda

MAGİSTR DİSSERTASİYASI

İxtisasın şifri: **060509 "Kompüter elmləri"**

İxtisaslaşmanın adı: **"İntellektual sistemlər"**

Elmi rəhbər: AMEA-nın müxbir üzvü, f.-r.e.d., prof. Ayda-zadə Kamil Rəcəb oğlu
(elmi dərəcəsi, elmi adı, S.A.A.)

Təhsil şöbəsinin müdiri: r.ü.f.d. Quliyeva Sevinc Yunis qızı
(elmi dərəcəsi, elmi adı, S.A.A.)

Bakı – 2022

Mündəricat

GİRİŞ

Mövzunun aktuallığı. Son yarıməsrdə kompüter sistemlərinin intellektinin artımına tələb artmışdır. Maşın intellektinin artımına istiqamətlənən bu tələb məsələlərin həllində insan iştirakının azalması, kompüterlərin insan vəzifələrindən bəzilərini yerinə yetirməsi, habelə insanın hesablama imkanlarını aşan məsələlərin həllərinin reallaşması kimi üstünlüklər gətirmiş, maşın öyrənməsinin fərqli növlərindən istifadə edən surətlərin tanınması, data analitikası və s. konsepsiyalar geniş tətbiq, tədqiq, təhlil olunmuşdur.

Surətlərin tanınması, maşın öyrənməsi kimi konsepsiyalar qısa zamanda uzun inkişaf yolu keçmişdir. Hələ 1943, 1947-ci illərdə Makallok və Pits sadə beyin hüceyrəsinin davranışını tədqiq etmişdilər. 1957-cı ildə Rozenblat neyronun "perseptron" adlandırdığı hesablama modelinin öyrədilməsi üçün alqoritm təklif etmişdir. 1960-cı illərin əvvəllərində Uidrou və Haf ADALİN (ing. ADALIN, ADAptive LINear Elements) adlandırdıqları perseptronvari sistemlər üzərində bəzi məsələlərin həllini nümayiş etdirdilər. Həmin müəlliflər "ən kiçik orta kvadratlar" adlandırdıqları perseptronvari aparatların öyrədilmə alqoritmini də təklif etmişdirlər. 1969-cu ildə Minski və Peypert perseptronları ciddi şəkildə təhlil etmiş və belə qənaətə gəlmişdilər ki, birlaylı süni neyron şəbəkələrin aproksimasiya imkanlarında ciddi məhdudiyyətlər var. O zamanlar hələ çoxlaylı süni neyron şəbəkənin məlum öyrədilmə alqoritmi mövcud deyildi. Bununla yanaşı Hornik və həmmüəlliflərin 1989,1990-cı illərdəki işlərində də istifadə olunan süni neyron şəbəkənin aproksimator kimi istifadə imkanlarını araşdırmışdır. Nəhayət 1974-ci ildə Uerbos, 1985-ci ildə Parker, 1986-cı ildə Rumerlhart və həmmüəllifləri bir-birilərindən xəbərsiz "bekpropaqeyşın" və ya xətaların geriyə ötürülməsi adlanan çoxlaylı süni neyron şəbəkələri öyrətmə alqoritmini təklif etdilər.[17]

Surətlərin tanınması məsələlər sinfinin bir alt sinfi kimi təsvirlərin tanınması üçün də modellər işlənilmiş, yeni model növləri təklif olunmuşdur.

1962-ci ildə neyrofizioloqlar Hyubel və Uiselin tədqiqatları mühüm dönüş nöqtələrindən biridir. Müəlliflər pişiklərin vizual idrakını tədqiq edərək belə nəticəyə gəlmişdilər ki, görmənin ilk mərhələsində görüntüdən, məs., vertikal və horizontal xəttlər kimi sadə əlamətlər çıxarılır, sonra hər mərhələdə əvvəlki mərhələdə alınan nəticədən əlamət çıxarılır. Yəni bioloji görmədə ənənəvi tanıma məsələlərindəki kimi əlamət çıxarılışı sadəcə başlanğıcda yox, prosesin digər mərhələlərində də davam edir. Bu yeni biliklərə əsaslanaraq Fukuşima 1980-ci ildə “Neokoqnitron” (ing. Neocognitron) adlanan şəbəkəni təklif etdi. Sonradan Hyubel və Uiselin ideyaları dərin öyrənmənin ərsəyə gəlməsinə təkan verdi. Nəticədə LiKun, Krizevski və Hinton dərin öyrənmə üsulları və dərin öyrənən şəbəkələrin işlənilməsində iştirak etdilər, öz tövhələrini verdilər. Beləcə bükülmə neyron şəbəkə istifadə olunmağa başlandı, yeni arxitekturalar təklif olundu və olunmağa davam edir. [19]

Azərbaycanda da surətlərin tanınması məsələləri və həll üsulları tədqiq olunmuşdur. G.Abdullayevanın rəhbərliyi ilə əski əlifbada mətnlərin tanıma və identifikasiya sistemi yaradılıb, Azərbaycan əl xalılarının tanınma və identifikasiya sistemi hazırlanmışdır. K.Ayda-zadənin rəhbərliyi ilə E.Mustafayev və C.Həsənov Azərbaycan dilində latın qrafikası ilə yazılmış əlyazmaları tanıma üçün kompüter sistemi işləmişdir. Yenə K.Ayda-zadənin rəhbərliyi ilə S.Rüstəmov Azərbaycan nitqini tanıma sistemi üçün alqoritm və proqram təminatını işləmişdir.

Dünyada informasiyanın artan sürətlə artımı, qlobal miqyasda iqtisadi böyümə və çoxsaylı sosial, təbii problemlərin həllinə olan ehtiyac surətlərin tanınması məsələlərinin aktuallığını qoruyub saxlamasına səbəb olur, habelə tanımada dəqiqliyin və etibarlılığının artması, emal vaxtının azaldılmasına olan tələbi artırır.

Dissertasiya işində ənənəvi tanıma modelləri ilə yanaşı tanıma üçün fərqli yanaşmalar təklif olunmuş, alqoritm və proqram təminatı işlənilmiş fərqli sinif əlamətlər bu yanaşmaların hər birində istifadə olunmuşdur.

İşin əsas məqsədi və qarşıya qoyulan vəzifələr. Dissertasiya işi şəkillərdə təsvir olunan obyektlərin müxtəlif növ əlamətlərinin hazırlanması və təsvirlərdəki obyektlərin tanıması məsələlərinin həllinə fərqli yanaşmalarda istifadəsinin effektivliyinin tədqiqinə həsr olunmuşdur. İşdə qarşıya qoyulan əsas vəzifələr təsvirlərdəki obyektlərin məlum əlamətlərinin araşdırılması, əlamətlərin çıxarılışı və yanaşmaların realizasiyası üçün alqoritm və proqram təminatının hazırlanması, yanaşmaların və əlamətlərin effektivliklərinin qiymətləndirilməsi üçün ədədi eksperimentlərin aparılması, tanımaya fərqli yanaşmaların və təsvirlərdəki obyektlərin fərqli əlamətləri kombinasiyalarının müqayisəli təhlilidir.

Tədqiqatın elmi yeniliyi. Surətlərin tanınması məsələlərinin həllinə fərqli yanaşmalar təklif olunmuş, təsvirlərdəki obyektlərin fərqli əlamətləri bu üsulların hər birində sınanmış, ədədi eksperimentlərin nəticələri əsasında yanaşmaların və əlamətlərin müqayisəli təhlili aparılmışdır.

İşin praktiki əhəmiyyəti. Baxılan işdə təsvirlərdəki obyektlərin əlamətləri çıxarılır, tanıma məsələlərinin həllinə dörd fərqli yanaşmada qiymətləndirilir. Tanıma məsələlərinə həll kimi istifadə oluna biləcək həmin yanaşmaların effektivlikləri Azərbaycan çap əlyazma hərflərinin tanınmasında tədqiq olunur. Təklif olunan əlamətlər və tanıma yollarından biri və ya bir neçəsi çap əlyazma hərflərin tanınması məsələlərinin həllində istifadə oluna bilər.

Tədqiqat işinin metodu və nəzəri əsasları. Dissertasiya işində ilkin olaraq təsvirlərin müxtəlif növ əlamətlərinin istifadəsi tədqiq olunmuşdur, Süni Neyron Şəbəkə, Bükülmə Neyron Şəbəkə kimi təsvirdəki obyektləri tanıma üsulları və K-ortalar klasterləşdirmə üsulu istifadə edilmişdir.

Tədqiqat işinin aprobasiyası və əməli reallaşdırılması. Dissertasiyanın əsas nəticələri “Tətbiqi riyaziyyat və fundamental informatika” mövzusunda IX beynəlxalq elmi-praktiki konfransında, 2021 (Omsk, Rusiya), “Riyaziyyatın tətbiqi məsələləri və yeni informasiya texnologiyaları” mövzusunda onlayn

keçirilən IV Respublika elmi konfransında, 2021 (Sumqayıt), “8-ci Dünya soft kompütinq” beynəlxalq elmi konfransında, 2022 (Bakı), Ümumi lider Heydər Əliyevin anadan olmasının 99-cu il dönümünə həsr olunmuş “Proseslərin avtomatlaşdırılması və informasiya təhlükəsizliyi” mövzusunda onlayn keçirilən Tələbə və Gənc Tədqiqatçıların III Beynəlxalq Elmi Konfranslarında, 2022 (Bakı) məruzə olunmuşdur.

Nəşrlər. Dissertasiya işinin nəticələri 2-si məqalə, 2-si konfrans materialı və 2-si tezis olmaqla 6 elmi əsərdə dərc olunmuşdur. Bunlardan 3-ü həmmüəllifsizdir. Məqalələrin hər ikisi xaricdə dərc olunmuşdur. Bundan əlavə dissertasiya işində alınan nəticələr beynəlxalq səviyyəli 2 və respublika səviyyəli 2 elmi konfransda tezis və konfrans materialları şəklində öz əksini tapmışdır. Onlardan biri xaricdə dərc olunmuşdur.[2,3,12,13,22,46]

Dissertasiyanın həcmi və quruluşu. İş giriş, üç fəsil, nəticə, xülasə, əlavə və ədəbiyyat siyahısından ibarətdir.

Dissertasiya işinin birinci fəsli üç yarımfəsildən ibarət olub, maşın öyrənməsi və onun növləri, həmçinin də maşın öyrənməsinin fərqli növlərində istifadə olunan üsulların tədqiqinə həsr olunmuşdur.

Dissertasiya işinin ikinci fəsli üç yarımfəsildən ibarət olub, optik simvol tanıma məsələlərinin təhlili və məsələnin qoyuluşu, surətlərin tanınması üçün təklif olunan yanaşmalar, həmçinin də şəkildə təsvir olunmuş obyektin əlamətləri və onların çıxarılışının təsvirinə həsr olunmuşdur.

Dissertasiya işinin üçüncü fəsli iki yarımfəsildən ibarət olub, ədədi eksperimentlərin təsviri, onun nəticələri, həmçinin də nəticələrin təhlilinə həsr olunmuşdur. İkinci fəsildə təsvir olunan əlamətlərin həmin fəsildə təklif olunan yanaşmalarda istifadəsinin effektivliyi qiymətləndirilmiş, təklif olunan yanaşmaların müqayisəli təhlili aparılmışdır.

I FƏSİL. KOMPÜTER TANIMA SİSTEMLƏRİNDƏ İSTİFADƏ OLUNAN MAŞIN ÖYRƏNMƏSİ ÜSULLARININ TƏHLİLİ

1.1. Müəllimlə öyrənmə üsulları

1.1.1. Maşın öyrənməsi üsulları və onların növləri.

A.Jeron 2019-cu ildə ikinci dəfə nəşr olunan “Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow” adlı monoqrafiyasında maşın öyrənməsi terminini “öyrənmə” sözünü diqqətə almaqla şərh etməyə çalışıb. Artur Samuelin maşın öyrənməsini “kompüterlərə birbaşa proqramlaşdırılmadan öyrənmə qabiliyyəti verən tədqiqat sahəsi” adlandırmasının həmin nəşrdə iqtibas gətirilməsi də təsadüfi deyil. Tom Mitchell bu tərifi bir az da genişləndirərək maşın öyrənməsini müəyyən T tapşırığını həll etmək üçün E təcrübəsi (müşahidələri) ilə öyrənən və müəyyən P ilə performansı yoxlanılan kompüter programı adlandırır.

Maşın öyrənməsi öyrətmədə insan nəzarəti səviyyəsinə görə aşağıdakı növlərə bölünür:

- Müəllimlə öyrənmə. Müəllimlə öyrənmə surətlərin tanınması məsələsinin həlli olacaq modelin qurulması üçün istifadə olunur. Surətlərin tanınması isə öz növbəsində obyektin müəyyən məlum siniflərdən hansına daxil olduğunun avtomatik tapılması üsullarını araşdırır. Bu cür yanaşma ədəbiyyatlarda sinifləşdirmə (ing. classification) adı ilə gedir.
- Müəllimsiz öyrənmə. Müəllimsiz öyrənmə etiketlənməmiş verilənlərin qruplaşdırılması, dəstələşdirilməsi məsələlərində istifadə olunur. Bu halda müəllimlə öyrənmədəkindən fərqli olaraq dəstələrin nələr olduqları aprior olaraq müəyyən deyil. Dəstələrin sayının əvvəlcədən bəlli olub-olmamasına görə isə dəstələşdirmə üsulları iki yerə bölünür.
- Yarımmüəllimlə öyrənmə. Yarı-müəllimlə öyrənmədə (ing. “semi-supervised learning”) müəllimlə öyrənmə ilə müəllimsiz öyrənmə birlikdə istifadə olunmaqla həllin effektivliyin artımı hədəflənir.

- Səhvlərdən öyrənmə. Bu yanaşma digərlərindən kifayət qədər fərqlənir. Beynəlxalq ədəbiyyatda “reinforcement learning” adlı ilə məlum olan bu yanaşmada mühitdən məlumat alan agent fəalliyətinin cavabında mükafat və cəza almaqla ən yaxşı hərəkət strategiyası seçməyə çalışır.

Maşın öyrənməsinin yuxarıdakı növlərindən üçü dissertasiyanın digər fəsillərində geniş şərh olunmuş, məlum üsulları verilmişdir.

1.1.2. Surətlərin tanınması məsələlərinin qoyuluşlarının ümumi təsviri. Öyrətmə bazasının hazırlanması mərhələsində nümunə şəkillər yığılır, sonra hər şəklin hansı mövcud sinfə aid olduğu ekspert və ya ekspertlər tərəfindən müəyyənləşdirilir. Hər şəkildən əlamət çıxarılmaqla əlamət vektoru hazırlanır, yaranmış əlamət vektoru və çıxış qiyməti cütlərindən ibadət öyrətmə bazası hazırlanır. Yəni öyrətmə bazası $G = \{g^h\}$, $h = \overline{1,H}$, $g^h = (E^h, Y^h)$ cütlərindən yaradılır, burada H bazadakı müşahidələrin sayı, $E^h = \left(e_1^h, \dots, e_n^h\right)$ və $Y^h = \left(y_1^h, \dots, y_m^h\right)$, uyğun olaraq, h-ci şəklin əlamət və çıxış vektorları, m siniflərin sayıdır. Hər çıxış vektoru şəklin siniflərə mənsubiyyətini bildirən ədədlərdən ibarətdir. Deməli, h-ci şəklin aid olduğu sinfin nömrəsi j^*: $y_{j^*}^h = \max\limits_{j=\overline{1,m},} y_j^h$ kimi tapıla bilir.

Çıxış vektorlarının $Y^h = \left(y_1^h, \dots, y_m^h\right)$, $h = \overline{1,H}$ kimi formalaşdırılması həm modelin hazırlanmasında asanlıqlar yaradır, həm də bu cür yanaşma kifayət qədər interpretasiya qabiliyyətlidir. Bu cür təsvir strategiyası “1-in ecazkar təsviri” (ing. “One-Hot Encoding”) adlanır və realizasiyada təklif etdiyi sadəlikdən əlavə fərqli siniflərin bir-biri ilə süni asılılığını da aradan qaldırmaqda effektiv sayılır.[30]

Hərflərin tanınması məsələsi G-nin köməyi ilə parametrik identifikasiya olunan modelin qurulması ilə həll olunur. Verilmiş $\tilde{\mathrm{X}}$ şəklinin $\tilde{\mathrm{E}}$ əlaməti çıxarıldıqdan sonra $\tilde{\mathrm{E}}$ hazır modelə verildikdə, modelin çıxışında $\Upsilon =$

$(\tilde{y}_1, \dots, \tilde{y}_m)$ vektoru alınır. Beləliklə, şəklin sinfi $j^*: \tilde{y}_{j^*} = \max\limits_{j=\overline{1,m,}} \tilde{y}_j$ şəklində tapılır.

Növbəti bənddə müəllimlə öyrənmə ilə öyrədilən modellərin növləri təsvir olunmuşdur.

1.1.3. Surətlərin tanınması üsullarının təhlili.

Aşağıda surətlərin tanınması üçün istifadə olunan bəzi sinifləşdirmə üsullarının təsviri verilmişdir.

Sadəlövh Bayes üsulu. Sadəlövh Bayes üsulu spam filterləmədə geniş istifadə olunur.[1,38] Üsul şərti ehtimallara əsaslanır:

$$P(\tilde{\text{E}}/\upsilon_j) = \frac{P(\tilde{\text{E}} \text{ və } \upsilon_j)}{P(\upsilon_j)}, \qquad P(\upsilon_j/\tilde{\text{E}}) = \frac{P(\tilde{\text{E}} \text{ və } \upsilon_j)}{P(\tilde{\text{E}})}, \qquad (1.1.1)$$

burada $P(\tilde{\text{E}}/\upsilon_j)$ və $P(\upsilon_j/\tilde{\text{E}})$ uyğun olaraq υ_j çıxışının varlığında $\tilde{\text{E}}$ əlamətinin olması ehtimalı və $\tilde{\text{E}}$ əlamətinin varlığında υ_j çıxışının olması ehtimalı; $P(\upsilon_j)$, $P(\text{E})$, $P(\tilde{\text{E}} \text{ və } \upsilon_j)$ uyğun olaraq υ_j çıxışının ehtimalı, E əlamətinin ehtimalı, υ_j çıxışının və E əlamətinin birgə ehtimalı; m – siniflərin sayıdır və növbəti ifadə doğrudur: $j = \overline{1,m}$. (1)-dəki düsturlardan aşağıdakı ifadə alınır:

$$P(\tilde{\text{E}}/\upsilon_j)P(\upsilon_j) = P(\upsilon_j/\tilde{\text{E}})P(\tilde{\text{E}}). \qquad (1.1.2)$$

(1.1.2) bərabərsizliyindən isə aşağıdakı ifadə alınır:

$$P(\upsilon_j/\tilde{\text{E}}) = \frac{P(\tilde{\text{E}}/\upsilon_j)P(\upsilon_j)}{P(\tilde{\text{E}})} \qquad (1.1.3)$$

Əlamət $\tilde{\text{E}}$ vektoru $(\tilde{\epsilon}_1, \dots, \tilde{\epsilon}_n)$ şəklindədir. Bu vektorun elementlərinin varlığı ehtimallarının və bu elementlərin υ_j-nin olması şərtindəki ehtimallarının bir-birindən asılı olmadığı var saysaq, (1.1.3) bərabərsizliyini aşağıdakı kimi yazmaq olar:

$$P(\upsilon_j/\tilde{\mathrm{E}}) = \frac{\prod_{i=1}^{n} P(\tilde{\epsilon}_i/\upsilon_j)\, P(\upsilon_j)}{\prod_{i=1}^{n} P(\tilde{\epsilon}_i)}. \qquad (1.1.4)$$

(1.1.4) bərabərliyinə Bayes düsturu deyilir. Əlaməti $\tilde{\mathrm{E}}$ olan obyektin sinfi aşağıdakı kimi tapıla bilər [28,32]:

$$j^*: P(\upsilon_{j^*}/\tilde{\mathrm{E}}) = \max_{j=\overline{1,m,}} P(\upsilon_j/\tilde{\mathrm{E}}),$$

burada j^* – əlaməti $\tilde{\mathrm{E}}$ olan surətin sinfinin nömrəsini göstərir.

Bayes üsulunun surətlərin tanınmasına tətbiqində çətinliklərdən biri əlamət vektorlarının komponentlərinin qiymətlərinin tezliklərini hesablamaqdır. Əlamətlərin qiymətləri və ya sinif qiymətləri kəsilməz olduqda Bayes sinifləşdirməsinin Qaus Bayes sinifləşdirilməsi adlı modifikasiyasından istifadə oluna bilir.[36] Bu modifikasiya istifadə olunduqda əgər ehtimalların normal paylandığı hesab olunursa, hər hadisənin ehtimalı aşağıdakı ifadə ilə tapılır:

$$P(\tilde{\epsilon}_i/\upsilon_j) = \frac{1}{\sqrt{2\pi}\sigma_{\upsilon_j}} e^{-\frac{\left(\tilde{\epsilon}_i - \mu_{\upsilon_j}\right)^2}{2\sigma_{\upsilon_j}^2}},$$

burada μ_{υ_j}, σ_{υ_j} uyğun olaraq orta qiymət və dispersiyanın kökdən çıxarılmış qiymətidir (ing. standart deviation).[45]

Bayes üsulunun üstün cəhətləri aşağıdakılardır [38]:

- Realizasiyası sadədir, əlaməti verilmiş surətin siniflərin hər birinə aidiyyəti ehtimalı tapıla bildiyindən kifayət qədər interpretasiya qabiliyyətlidir: verilmiş surətin siniflərə aid olma şansları müqayisə asanlıqla oluna bilər.
- Digər bir çox sinifləşdirmə alqoritmlərinə nəzərən daha sürətli proqnozlaşdırma aparır.
- Kiçik öyrətmə bazalarıyla işlədilə bilir.

Bayes üsulunun zəif cəhətləri aşağıdakılardır [38]:

- Elə $\exists i \in \overline{1,n}$ və ya $\exists j \in \overline{1,m}$ varsa ki, $P(\tilde{\epsilon}_i/\upsilon_j) = 0$ olsun, $P(\upsilon_j/\tilde{\text{E}}) = 0$ olacaq. Yəni hansısa sinfin varlığı şərtində əlamət vektorunun hansısa elementinin varlığının ehtimalı sıfırdırsa, ümumilikdə həmin əlamət vektorunun həm sinfə məxsusluğu ehtimalı 0 olur. Bu reallığa adekvat deyil.
- Üsulun əlamətlərin siniflərlə əlaqəsinin asılı olmamasına söykənməsi bəzi çətinliklər yarada bilir.

K ən yaxın qonşu. Üsulun istifadəsi zamanı verilmiş surətin sinfi bu surətə öyrətmə bazasındakı yaxın müşahidələrin sinifləri əsasında təyin olunur. Bunun üçün verilmiş surətə bazadakı ən yaxın K sayda surətin müvafiq çıxışları arasında əksəriyyət təşkil edən çıxış verilmiş surətin çıxışı hesab olunur.

Verilmiş $\tilde{\text{E}}$ əlamətli obyektin K ən yaxın qonşu üsulunun köməyi ilə sinfinin təyin edilməsi aşağıdakı kimi aparılır. [49]

1. Obyektin surəti ilə öyrətmə bazasındakı surətlərin məsafələri hesablanır:

$$d_h = \mu(\tilde{\text{E}},\ \ E^h),\ \ h = \overline{1,H},$$

burada E^h – bazadakı h-ci surətin əlaməti, H – bazadakı nümunələrin sayı, d_h isə sinfi tapılacaq obyektin əlaməti $\tilde{\text{E}}$ ilə h-ci surətin əlaməti arasındakı məsafədir.

2. $\tilde{\text{E}}$ əlamətli obyektin surətinə bazadakı ən yaxın K sayda surətlərin indeksləri təyin olunur:

$$i_1^*: \mu(\tilde{\text{E}},\ \ E^{i_1^*}) = \min_{h\,\in\overline{1,H}} \mu(\tilde{E},\ E^h),$$

$$i_{j+1}^*: d_{i_{j+1}^*} = \min_{h\,\in\overline{1,H},\ \ i_h^* \neq i_l^*,\ \ l=\overline{1,j}} \{d_{i_h^*}\},\ \ \ j = 1,2,\ldots,K-1,$$

burada $i_1^*,\ i_2^*,\ \ldots,\ i_K^*$ uyğun olaraq 1-ci, 2-ci, ..., K-ci ən yaxın surətlərin bazadakı indeksləridir.

3. Addım 2-də tapılmış indekslərdəki surətlərin siniflərə aidiyyatı tezliyi hesablanır:

$$q_{i_j^*} = \left|\left\{y_l:\ y_l = y_{i_j^*},\ \ l = i_1^*,\ i_2^*,\ \ldots,\ i_K^*\right\}\right|,\ \ j = 1,\ 2,\ \ \ldots,\ \ K,$$

burada $q_{i_j^*} - i_j^*$-ci yaxın müşahidənin sinifinin K müşahidələrdəki ümumi sayı, |·| - çoxluğun elementlərinin sayı əməliyyatı, y_l, $y_{i_j^*}$ sinfi bildirən qiymətlərdir.

4. Seçilmiş K müşahidə arasında müşahidə sayına görə əksəriyyət təşkil edən sinif verilmiş $\tilde{E}$ əlamətli obyektin sinfi qəbul olunur:

$$\tilde{y} = y_{i_j^*},\quad harada\quad i_j^*: q_{i_j^*} = \max_{1 \le l \le K} q_{i_l^*}.$$

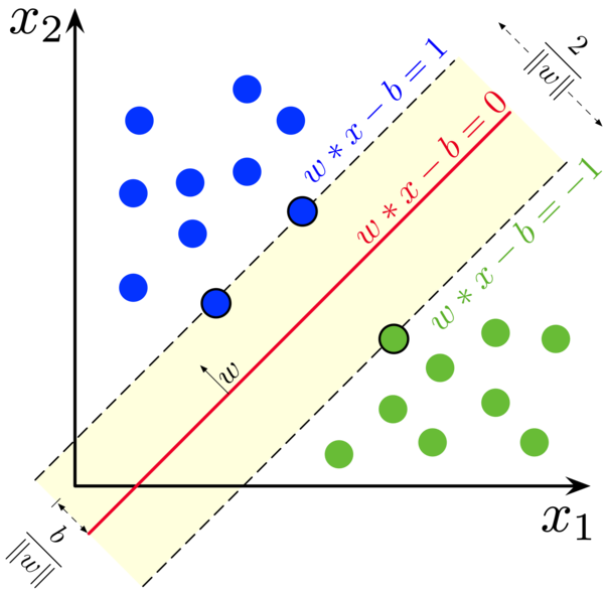
K ən yaxın qonşu üsulunun pis cəhətləri aşağıdakılardır [37]:

- Verilmiş surətin sinfi təyin edilməli olduqda verilmiş surətlə öyrətmə bazasındakı bütün surətlər arasındakı məsafələr tapılmalıdır, bu da verilmiş surətin emal müddətini artırır.
- Verilmiş surətin emalı zamanı bütün öyrətmə bazası istifadə olunur. Məsələn, digər bəzi üsullarda öyrətmə bazası modelə öyrədildikdən sonra modelin sadəcə strukturu və parametrlərinin qiymətləri yadda saxlanılmaqla verilmiş surətin sinfinin təyini zamanı öyrətmə bazasından istifadəyə ehtiyac duyulmur. Bu da, xüsusən, bazadakı öyrətmə cütlərinin sayı çox olduqda çətinliklər yaradır.
- Əlamət fəzasının ölçüsü artırdıqca verilmiş surətlə bazadakı surətlərin arasındakı məsafənin adekvat tapılması çətinləşir. Bəzən bir əlamətdəki fərq digər bir çox əlamətdəki fərqləri üstələyə bilir.

- Az sayda kənaraçıxma belə üsulun effektivliyini azalda bilir.

Dayaq Vektor Üsulu. Üsul öyrətmə bazasındakı müşahidələrin siniflərinin bir-birindən asılı olmayan və ayrıla bilən çoxluqlar olmasına söykənir. Üsul realizasiyada qabarıq proqramlaşdırma məsələsinə gətirilə bildiyindən dəqiq həll tapmaq mümkün ola bilir. Əlamət vektorunun elementlərinin sayı öyrətmə bazasındakı verilənlərin sayını aşdıqda digər bəzi modellərdən fərqli olaraq dayaq vektorlar üsulu istifadə oluna bilər və ya yaxşı nəticələr də alınır.

Bir çox icmalda üsulun izahı ikiölçülü əlamətlər fəzasında təyin olunan müşahidələrin ikisinifli - iki sinifdən birinə (binar) sinifləşdirilməsi nümunəsi üzərində aparılır (şəkil 1.1.1). [4,5,48,21,31]



Şəkil 1.1.1. Dayaq vektor üsulunun sinifləşdirmə məqsədi ilə istifadəsi nümunəsi.

Öyrətmə bazasındakı fərqli sinifli surətlər hipermüstəvilərlə ayrılır və üsul istifadəsi zamanı elə hipermüstəvi tapmaq lazımdır ki, onun hər iki tərəfində biri olmaqla ona dayaq hipermüstəvilər arasında məsafə maksimum olsun, eyni zamanda həm öyrətmə bazasında bir sinifdəki surətlər bu dayaq hipermüstəvilərinin bir, digəri o biri dayaq hipermüstəvinin o biri tərəfində

qalsın. Deməli, öyrətmə bazasındakı surətlərin çıxış qiymətləri $Y^h \in \{-1,1\}$, $h = \overline{1,H}$ olduqda elə bir $g(W,b,\mathrm{E})$ funksiyası tapılmalıdır ki, əlaməti E olan surətin sinfini $f(\mathrm{E}) = sign\big(g(W,b,\mathrm{E})\big)$ göstərsin.[29] Burada $g(\mathrm{E})$ – diskriminant və ya qərar funksiyası, $f(\mathrm{E})$ – indikator funksiya, W,b qərar funksiyasının parametrləridir [39] və aydındır ki, çıxış qiyməti -1 olan surətlər bir sinfin, 1 olan surətlər isə digər sinfin müşahidələridir.

Deməli, diskriminantın parametrlərinin qiymətləri elə təyin olunmalıdır ki, aşağıdakı şərtlər ödənsin:

$$< W, E^h > + b > 1, \text{ əgər } Y^h = 1, \tag{1.1.5}$$

$$< W, E^h > + b > -1, \text{ əgər } Y^h = -1, \tag{1.1.6}$$

burada $h = \overline{1,H}$ doğrudur və $<\cdot,\cdot>$ - skalyar hasil əməliyyatıdır.

(1.1.5), (1.1.6) şərtlərini vahid şərtdə yazmaq olar:

$$Y^h(< W, E^h > + b) \geq 1, \quad h = \overline{1,H}. \tag{1.1.7}$$

Nöqtədən müstəviyə qədər məsafə aşağıdakı kimi ifadə olunur:

$$d(W,b,\mathrm{E}) = \frac{Y^h(< W, E^h > + b)}{\|W\|},$$

burada $\|\cdot\|$ – norma əməliyyatıdır. (1.1.7)-dən məlum olur ki, aşağıdakı bərabərlik doğrudur:

$$\frac{Y^h(< W, E^h > + b)}{\|W\|} \geq \frac{1}{\|W\|}.$$

Bir sinifdəki – hipermüstəvinin bir tərəfindəki ən yaxın müşahidənin üzərində olduğu paralel hipermüstəvi ilə digər sinifdə ilk rast gəlinən müstəvi arasındakı məsafə aşağıdakı kimi tapılır, bu məsafənin minimumunu tapmaq arzuolunandır ki, minimum olan halın maksimumunu tapaq:

$$\rho(W,b) = \min_{E^h : Y^h = 1} \frac{\left| < W, E^h > + b \right|}{\|W\|} + \min_{E^h : Y^h = -1} \frac{\left| < W, E^h > + b \right|}{\|W\|} = \frac{2}{\|W\|}.$$

Bu məsafənin minimumu belə maksimum qiymət almalı olduğundan iki sinfi ən yaxşı ayıracaq hipermüstəvinin parametrləri aşağıdakı (1.1.8), (1.1.9) məsələsinin həllidir:

$$\min_{W,b} \frac{1}{2}\|W\|, \tag{1.1.8}$$

$$Y^h(< W, E^h > +b) \geq 1, \;\; h = \overline{1,H}, \tag{1.1.9}$$

(1.1.8), (1.1.9) məsələsinin həlli üçün Laqranj funksiyası qurulur:

$$L(W,b,\mu) = \frac{1}{2}\|W\| \; - \sum_{h=1}^{H} \mu_h (Y^h(< W, E^h > +b) \; - \; 1),$$

burada $\mu_h, h = \overline{1,H}$, Laqranj vuruqlarıdır. Zəruri şərtlərdən aşağıdakı bərabərliklər alınır:

$$\nabla_W L(W,b,\mu_1,\dots,\mu_H) = W \; - \sum_{h=1}^{H} \mu_h Y^h E^h = 0 \Rightarrow W = \sum_{h=1}^{H} \mu_h Y^h E^h, \tag{1.1.10}$$

$$\frac{\partial L(W,b,\mu_1,\dots,\mu_H)}{\partial b} = - \sum_{h=1}^{H} \mu_h Y^h = 0 \Rightarrow \sum_{h=1}^{H} \mu_h Y^h = 0. \tag{1.1.11}$$

(1.1.8) və (1.1.9)-dakı W vektorunu (1.1.10), (1.1.11)-dəki ifadəsi ilə əvəz etdikdə (1.1.8), (1.1.9) məsələsi aşağıdakı (1.1.12), (1.1.13), (1.1.14) məsələsinə çevirilir:

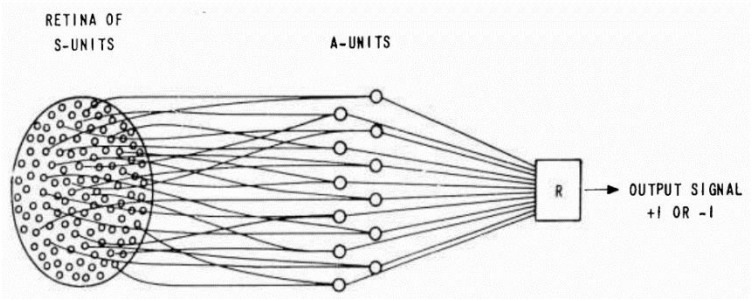
$$min\, J(\mu_1,\dots,\mu_H) = \frac{1}{2} \sum_{h_1=1}^{H} \sum_{h_2=1}^{H} \mu_{h_1}\mu_{h_2} Y^{h_1} Y^{h_2} < E^{h_1}, E^{h_2} > \; - \sum_{h=1}^{H} \mu_h, \tag{1.1.12}$$

$$\sum_{h=1}^{H} \mu_h Y^h = 0, \tag{1.1.13}$$

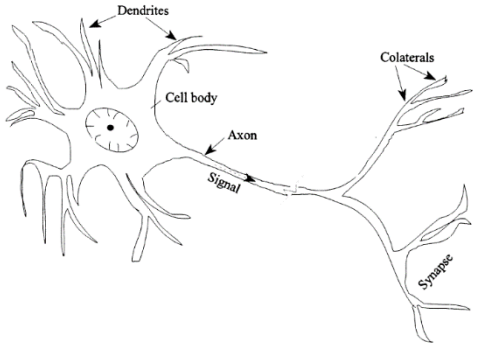
$$0 \leq \mu_h \leq C, \qquad h = \overline{1,H}, \tag{1.1.14}$$

burada $C > 0$ cərimə parametridir, dayaq vektorları arasına düşməsinə icazə verən istisnaların miqdarını bildirir.

Süni Neyron Şəbəkə. Perseptron. Süni Neyron Şəbəkə (SNŞ) mürəkkəb real məsələlərin həlli üçün istifadə olunan elə bir modeldir ki, küyə dözümlü, ümumiləşdirmə qabiliyyəti yüksək hesablama modelidir. SNŞ-lərin fəaliyyəti insan beyninin bioloji quruluşuna əsaslanır. Orta insan beynində 10^{11}-ə yaxın sinir hüceyrəsi olur. Hər belə hüceyrə 10000-ə qədər digər hüceyrə ilə əlaqələr qura bilir və beləliklə hüceyrələr arası əlaqələrin sayı 10^{15}-ə çatır. Bu əlaqələr vasitəsi ilə bir neyrondan digərinə elektrik və ya kimyəvi siqnal ötürülür. Neyronun membranındakı gərginlik müəyyən bir dəyəri aşdıqda digər neyronlara siqnal ötürür.[50] Sinir sisteminin işi bu neyronların əlaqələnməsinə əsaslanır. Reseptor sinirlərdən alınan siqnal şəbəkəyə ötürülür, siqnal effektor sinirlərə daşınır (şəkil 1.1.2). [23, səh.35]



Şəkil 1.1.2. Süni neyronun modeli.

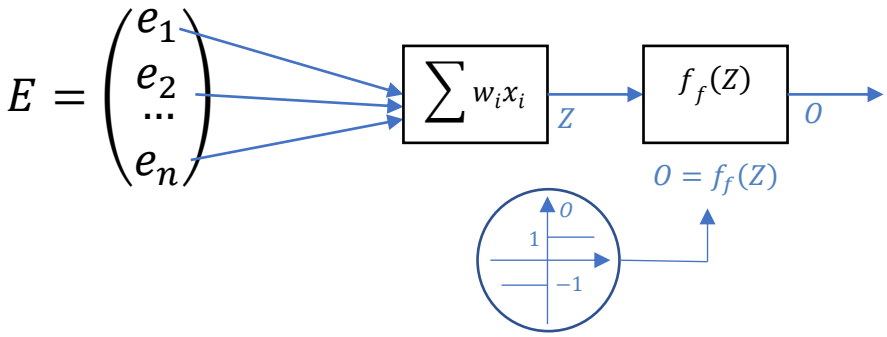


Şəkil 1.1.3. Bioloji neyronun modeli.

20-ci əsrin 50-ci illərində Makkallok və Pits insanın sinir hüceyrələrinin fəaliyyətinin hesablama modelini tədqiq etməklə, bioloji neyronun imitasiyası olan süni neyronun hesablama modeli kimi istifadəsinə yol açdılar. Bioloji və

neyronun işi şəkil 1.1.3-də təsvir olunmuşdur [16,52**Error! Reference s ource not found.**]:

Bioloji neyronda dendritlər vasitəsi ilə digər neyronlardan alınan siqnal hüceyrənin nüvəsinə keçir. Neyronda aktivlik olduqda siqnal digər neyronlara aksonlar vasitəsi ilə ötürülür. Bioloji neyronda süni neyronun dendritlərinin analoqu kimi girişlər, nüvədə baş verənlərin imitasiyası kimi girişlərin çəkilər nəzərə alınaraq cəmlənməsi, bioloji aktivləşmənin baş verib-verməməsi kimi aktivasiya funksiyası dayanır.[47] Şəkil 1.1.4-də $f_f(Z)$ aktivasiya



Şəkil 1.1.4. Süni neyronun riyazi modeli.

funksiyasıdır, xəttiliyi aradan qaldırmaq üçün istifadə olunur.

Aktivasiya funksiyaları. Bioloji neyron ona daxil olan siqnallar müəyyən bir həyəcanlandırma əmsalını aşdıqda həyəcanlanır, elektrik buraxırsa, süni neyronda da həyəcanlandırma – aktivasiya funksiyası neyronun çıxış qiymətinə təsir edir. Aktivasiya funksiyasına nümunə kimi aşağıdakı qeyri-xətti funksiyaları istifadə etmək olar [43]:

1. Vahid sıçrayış funksiyası [47]:

$$f_f(Z) = \begin{cases} 1, & Z \geq 0, \\ 0, & əks\ halda. \end{cases}$$

2. Siqmoid funksiyası:

$$f_f(Z) = \frac{1}{1 + e^{-Z}},$$

harada ki, $0 \leq f_f(Z) \leq 1$.

3. Hiperbolik tangens funksiyası:

$$f_f(Z) = \frac{e^Z - e^{-Z}}{e^Z + e^{-Z}},$$

harada ki, $-1 \leq f_f(Z) \leq 1$.

4. Softmaks funksiyası. Adətən, şəbəkənin çıxışında vektor alındıqda, başqa sözlə, şəbəkənin son layında birdən çox neyron olduqda, xüsusən, sinifləşdirmə məsələlərində geniş istifadə olunur. Son laydakı hər neyronun çıxış qiyməti aşağıdakı kimi tapılır:

$$f_f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_j}}, \quad i = \overline{1,m},$$

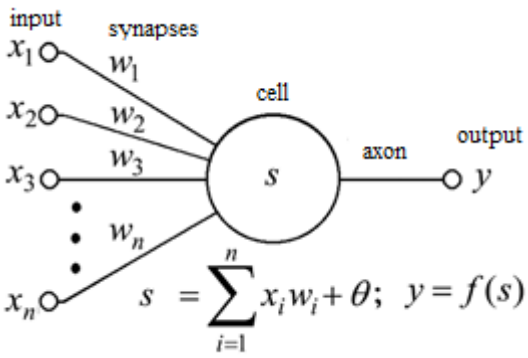
burada, $z_1, \ldots, z_m$ son laydakı neyronların uyğun giriş siqnallarının uyğun çəkilərlə skalyar hasilinin həyəcanlandırma əmsalı ilə cəmləridir və növbəti bərabərlik doğrudur: $\sum_{i=1}^{m} f_f(z_i) = 1$.

5. Rectified Linear Unit (ReLU) funksiyası. İlk dəfə 2010-cı ildə təklif olunmuşdur. 2018-ci ildə archive.org saytında dərc olunan məqalədə ReLU aktivasiya funksiyasının dərin öyrənmənin tətbiq olunduğu işlərdə “susmaya görə” mahiyyətini daşıyır.[43] Çox sadə olmasına baxmayaraq xəttiliyi aradan qaldırır və yaxşı nəticə verir[20]. Analitik ifadəsi aşağıdakı kimidir:

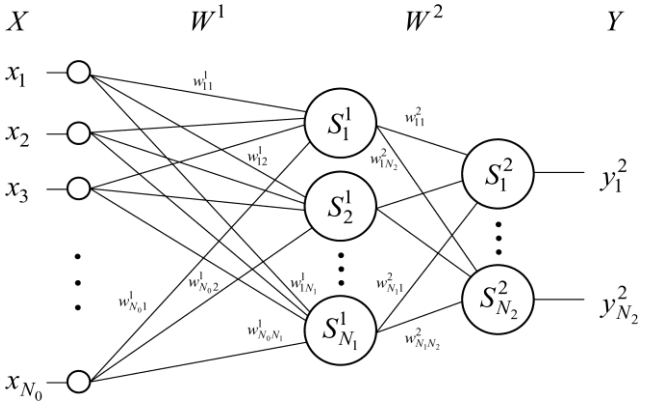
$$f_f(Z) = max\{0, Z\} = \begin{cases} Z, & Z > 0, \\ 0, & əks\ halda. \end{cases}$$

Çoxlaylı Tam Əlaqəli Süni Neyron Şəbəkə. Birdən çox səviyyədə çoxsaylı neyronun əlaqəli işi çoxsəviyyəli SNŞ adı altında ifadə olunur və approksimasiya imkanının genişliyi səviyyələrin, səviyyələrdəki neyron sayları və aktivasiya funksiyalarının seçimi ilə təyin olunur (şəkil 1.1.5, şəkil 1.1.6). Çoxsəviyyəli SNŞ-nin tam əlaqəli növü surətlərin tanınması məsələsində xüsusi yer tutur. Belə tam əlaqəli SNŞ-lərin girişinə surətin əlaməti verildikdə əlamətin əvvəlcədən mahiyyəti bilinən siniflərdən hansına uyğun gəldiyi

tapılır. Lakin belə bir şəbəkəni işlətməzdən əvvəl xətanın geriyə yayılması (ing. error backpropagation) adlanan yanaşmanın köməyi ilə öyrətmə bazasını modelə öyrətmək lazım gəlir.



Şəkil 1.1.5. Perseptron.



Şəkil 1.1.6. Çoxlaylı tam əlaqəli süni neyron şəbəkə.

Şəbəkənin hazırlanması iki mərhələdə aparılır:

1. Struktur seçimi. Bu mərhələdə səviyyələrin və səviyyələrdəki neyronların sayı, fərqli səviyyələrdəki neyronların necə əlaqələndiriləcəyi - aktivasiya funksiyaları seçilir.
2. Parametrlərin seçimi. Şəbəkədə neyronlar arasındakı sinaps qiymətləri və neyronların həyəcanlandırma əmsallarının qiymətləri təyin olunur.

2-ci mərhələ xətanın geriyə yayılması vasitəsi ilə aparılır. Bunun üçün şəbəkədə son səviyyədən ilk səviyyəyə doğru getməklə sinaps və həyəcanlandırma əmsallarına görə müəyyən məqsəd funksiyasının xüsusi törəmələri rekursiv şəkildə tapılır.

Səviyyələrin sayını L, səviyyələrdəki neyron saylarını N_l, $l = \overline{1,L}$, l-ci səviyyədəki j-cu neyronun $(l-1)$-ci səviyyədə əlaqədə olduğu neyronların indeksləri çoxluğunu I_{lj}^-, l-ci səviyyədəki j-cu neyronun $(l-1)$-ci səviyyədəki i-ci neyronla arasındakı sinapsın (əlaqənin) çəkisini w_{ij}^l, l-ci səviyyədəki j-cu neyronun həyəcanlandırma əmsalını θ_j^l ilə işarə edək.

Nümunələrdən h nörməli olanı şəbəkədən tam əlaqəli keçirildikdə l-ci səviyyənin j-cu neyronunun çıxışını $\bar{y}_{j,h}^{l},\ h=\overline{1,H},\ l=\overline{1,L},\ j=\overline{1,N_l}$, ilə işarə edək, harada ki, şəbəkənin $\bar{y}_{j,h}^{0},\ h=\overline{1,H},\ j=\overline{1,N_0}$, giriş dəyərləridir.

Şəbəkəyə giriş kimi bazadakı surtələrin əlamətləri $E^h=\left(e_1^h,\dots,e_n^h\right),\ h=\overline{1,H}$, çıxış vektorları $Y^h=\left(y_1^h,\dots,y_m^h\right),\ h=\overline{1,H}$, olduqda aşağıdakılar doğrudur:

$$\bar{y}_{j,h}^{0}=e_j^h,\qquad h=\overline{1,H},\qquad j=\overline{1,N_0}, \tag{1.1.15}$$

$$\bar{y}_{j,h}^{l}=f_f\left(\bar{z}_{j,h}^{l}\right),\qquad h=\overline{1,H},\qquad l=\overline{1,L},\qquad j=\overline{1,N_l}, \tag{1.1.16}$$

$$\bar{z}_{j,h}^{l}=\sum_{i\in I_{lj}} w_{ij}^{l}\,\bar{y}_{i,h}^{l-1}+\theta_j^{l},\qquad h=\overline{1,H},\qquad l=\overline{1,L},\qquad j=\overline{1,N_l}. \tag{1.1.17}$$

Öyrətmə prosesinin aparılması üçün qurulan məqsəd funksiyası beynəlxalq sferada “xəta funksiyası” (ing. cost function) adlandırılır. Öyrətmə cütlərinin giriş qiyməti şəbəkəyə verildikdə şəbəkənin çıxışında alınanlarla həmin giriş qiymətinə uyğun real çıxış qiyməti arasındakı fərq isə “itki funksiyası” (ing. loss function) ilə ifadə olunur. Demək baza üçün xəta funksiyasının qiyməti hər nümunə üçün çıxarılan itki funksiyalarının qiymətləri cəmidir. Aşağıda xəta funksiyası kimi ən kiçik kvadratlar üsulunda istifadə olunan orta kvadrat xəta funksiyası verilmişdir:

$$\mathcal{J}(\mathbb{W},\mathbb{O},\overline{\mathbb{Z}},\overline{\mathbb{Y}},\mathbb{E},\mathbb{Y})=\frac{1}{H}\sum_{h=1}^{H}\eta_h J_h(\mathbb{W},\mathbb{O},\bar{Z}_h,\bar{Y}_h,E^h,Y^h), \tag{1.1.18}$$

$$J_h(\mathbb{W},\mathbb{O},\bar{Z}_h,\bar{Y}_h,E^h,Y^h)=\frac{1}{2}\sum_{j=1}^{N_L}\left(\bar{y}_{j,h}^{L}-y_j^h\right)^2, \tag{1.1.19}$$

burada $\mathcal{J}(\mathbb{W},\mathbb{O},\overline{\mathbb{Z}},\overline{\mathbb{Y}},\mathbb{E},\mathbb{Y})$ – xəta funksiyası, $J_h(\mathbb{W},\mathbb{O},\bar{Z}_h,\bar{Y}_h,E^h,Y^h)$ – itki funksiyası, $\mathbb{W}$ və $\mathbb{O}$ uyğun olaraq çəki və həyəcanlandırma əmsalları çoxluğu,

$\overline{\mathbb{Z}}$ və $\overline{\mathbb{Y}}$ uyğun olaraq bütün surətlər şəbəkədən keçirildikdə neyronların çıxışaqədərki cəm və çıxış qiymətləri, $\bar{Z}_h$ və $\bar{Y}_h$ uyğun olaraq h-cı surət şəbəkədən keçirildikdə neyronların çıxışaqədərki cəm və çıxış qiyməti, $\mathbb{E} = \{E^h\}_{h=1}^{H}$ – öyrətmə bazasındakı surətlərdən ibarət çoxluq, $\mathbb{Y} = \{Y^h\}_{h=1}^{H}$ – öyrətmə bazasındakı surətlərin çıxışlarından ibarət çoxluq, η_h – h-cı nümunənin öyrətmədə əhəmiyyətini – çəkisini bildirən qiymətdir.

SNŞ-nin asılı olduğu parametrlərin – sinaps və həyəcanlandırma əmsalı qiymətlərin tapılması üçün birinci tərtib optimallaşdırma üsulları istifadə olunur. Birinci tərtib optimallaşdırma üsullarına qradient, qoşma qradient və s. daxildir. Bu üsulların köməyi ilə optimal $\mathbb{W}, \mathbb{O}$ seçilməsi üçün $\frac{\partial \mathcal{J}}{\partial w_{ij}^l}, \frac{\partial \mathcal{J}}{\partial \theta_j^l}$, $h = \overline{1,H}$, $l = \overline{1,L}$, $j = \overline{1,N_l}$, tapılmalıdır. Bu parametrlərin tapılması üçün aşağıdakılar daxil olunur:

$$q_{j,h}^l = \frac{\partial J_h}{\partial z_{j,h}^l}, \qquad r_{j,h}^l = \frac{\partial J_h}{\partial \bar{y}_{j,h}^l},$$

$$Q_j^l = \frac{\partial \mathcal{J}}{\partial z_j^l} = \sum_{h=1}^{H} q_{j,h}^l, \qquad R_j^l = \frac{\partial \mathcal{J}}{\partial \mathcal{y}_j^l} = \sum_{h=1}^{H} r_{j,h}^l, \tag{1.1.20}$$

$$l = \overline{1,L}, \qquad j = \overline{1,N_l},$$

burada $q_{j,h}^l$ və Q_j^l uyğun olaraq şəbəkədən bir müşahidə və bütün müşahidələr keçirilərkən l-ci səviyyənin j-cu neyronunun çıxışa qədərki cəm qiymətlərinə görə, $r_{j,h}^l$ və R_j^l uyğun olaraq şəbəkədən bir müşahidə və bütün müşahidələr keçirilərkən l-ci səviyyənin j-cu neyronunun çıxışına görə xüsusi törəmələri, $z_j^l \in \overline{\mathbb{Z}}$ və $\mathcal{y}_j^l \in \overline{\mathbb{Y}}$ isə bütün müşahidələr şəbəkədən keçirildikdə uyğun olaraq l-ci səviyyənin j-cu neyronunun çıxışa qədərki cəm və çıxış qiymətlərindən ibarət çoxluqlardır.

Sinaps və həyəcanlandırma əmsalı qiymətlərini aşağıdakı kimi ifadə edə bilərik:

$$\frac{\partial J_h}{\partial w_{ij}^l} = \frac{\partial J_h}{\partial z_{j,h}^l}\frac{\partial z_{j,h}^l}{\partial w_{ij}^l} = q_{j,h}^l \bar{y}_{i,h}^{l-1}, \tag{1.1.21}$$

$$\frac{\partial J_h}{\partial \theta_j^l} = \frac{\partial J_h}{\partial z_{j,h}^l}\frac{\partial z_{j,h}^l}{\partial \theta_j^l} = q_{j,h}^l. \tag{1.1.22}$$

(1.1.18), (1.1.19) və (1.1.21), (1.1.22)-dən aşağıdakı ifadələr alınır:

$$\frac{\partial \mathcal{J}}{\partial w_{ij}^l} = \frac{1}{H}\sum_{h=1}^{H}\eta_h \frac{\partial J_h}{\partial w_{ij}^l} = \frac{1}{H}\sum_{h=1}^{H}\eta_h q_{j,h}^l \bar{y}_{i,h}^{l-1}, \tag{1.1.23}$$

$$\frac{\partial \mathcal{J}}{\partial \theta_j^l} = \frac{1}{H}\sum_{h=1}^{H}\eta_h \frac{\partial J_h}{\partial \theta_j^l} = \frac{1}{H}\sum_{h=1}^{H}\eta_h q_{j,h}^l. \tag{1.1.24}$$

(1.1.23), (1.1.24)-də istifadə istifadə olunacaq $q_{j,h}^l,\ h=\overline{1,H},\ l=\overline{1,L},$ $j=\overline{1,N_l}$, üçün reskursiv düsturların alınması axırıncı laydan başlayır:

$$q_{j,h}^L = \frac{\partial \bar{y}_{j,h}^L}{\partial z_{j,h}^L} r_{j,h}^L = \frac{\partial f_f\left(z_{j,h}^L\right)}{\partial z_{j,h}^L} r_{j,h}^L, \tag{1.1.25}$$

burada $r_{j,h}^L = \bar{y}_{j,h}^L - y_j^h$ doğrudur harada ki, $l=\overline{1,L},\ j=\overline{1,N_l}$.

Sondan əvvələ getməklə $I_{lj}^+ - l$-ci səviyyənin $(l+1)$-ci səviyyədəki neyronların indeksləri çoxluğu olduqda $r_{j,h}^l,\ h=\overline{1,H},\ l=\overline{1,L-1},\ j=\overline{1,N_l}$, üçün aşağıdakı rekursiv düstur alınır:

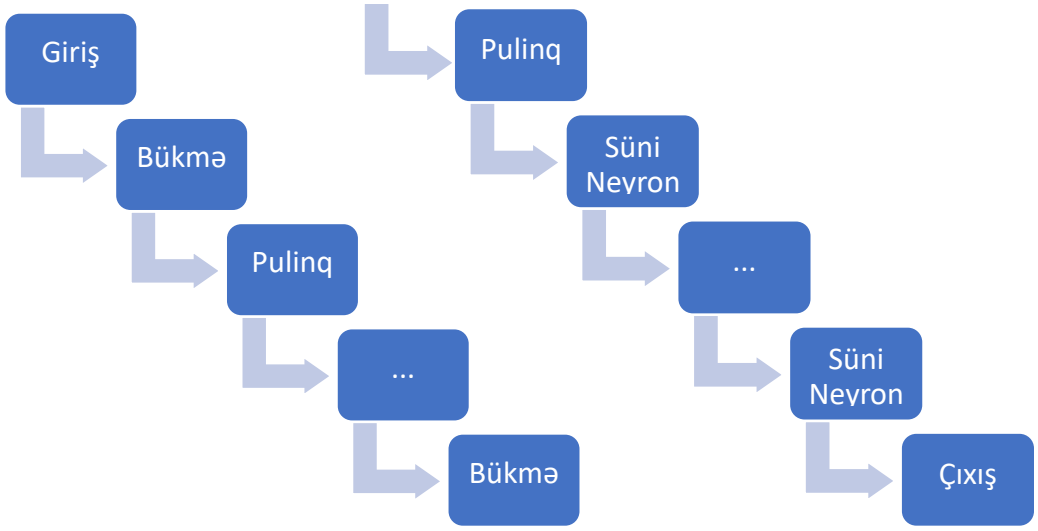
$$r_{j,h}^l = \sum_{k\in I_{lj}^+}\frac{\partial J_h}{\partial z_{k,h}^{l+1}}\frac{\partial z_{k,h}^{l+1}}{\partial \bar{y}_{j,h}^l} = \sum_{k\in I_{lj}^+}\frac{\partial z_{k,h}^{l+1}}{\partial \bar{y}_{j,h}^l} q_{k,h}^{l+1} = \sum_{k\in I_{lj}^+} w_{jk}^{l+1} q_{k,h}^{l+1}. \tag{1.1.26}$$

(1.1.25) və (1.1.26)-dan $q_{j,h}^l,\ h=\overline{1,H},\ l=\overline{1,L-1},\ j=\overline{1,N_l}$, üçün aşağıdakı rekursiv düstur alınır:

$$q_{j,h}^{l} = \frac{\partial f_f(z_{j,h}^{l})}{\partial z_{j,h}^{l}} \sum_{k \in I_{ij}^{+}} w_{jk}^{l+1} q_{k,h}^{l+1}. \quad (1.1.27)$$

(1.1.21), (1.1.22), (1.1.27) düsturları birlikdə istifadə edilərək $\frac{\partial \mathcal{J}}{\partial w_{ij}^{l}}, \frac{\partial \mathcal{J}}{\partial \theta_{j}^{l}}$, $h = \overline{1,H},\ l = \overline{1,L},\ j = \overline{1,N_l}$, qiymətlərini hesablamaq mümkündür.

Bükülmə Neyron Şəbəkə. Ənənəvi Bükülmə Neyron Şəbəkə (BNŞ) bir və ya birdən çox bükmə-pulinq səviyyələri bloku və tam əlaqəli səviyyə adlanan SNŞ-dəki perseptronlardan ibarət laylar blokundan ibarətdir (şəkil 1.1.7). [25]



Şəkil 1.1.7. Bükülmə Neyron Şəbəkənin quruluşu.

Bükülmə və pulinq əməliyyatları aşağıdakı kimi aparılır:

- Bükülmə əməliyyatı. Şəkil $X = (x_{ij})_{i,j=1}^{m}$, bükülmənin nüvəsi $C = (c_{ij})_{i,j=1}^{k}$ olduqda bükmə əməliyyatının nəticəsi olacaq şəkli $\dot{X} = (\dot{x}_{ij})_{i,j=1}^{m-k+1}$ deyə işarə etsək, aşağıdakı doğrudur:

$$\dot{x}_{ij} = \sum_{o=i}^{i+k} \sum_{p=j}^{j+k} x_{op} c_{o-i+1,p-j+1}, \quad i,j = \overline{1, m-k+1}. \quad (1.1.28)$$

Bükülmə əməliyyatı zamanı təsvirdən regionların necə çıxarılacağını bildirən iki xarakteristika var:

1. Addım genişliyi. Bu xarakteristika inglisdilli ədəbiyyatda “stride” adlanır, regionu bildirən çərçivənin sütunlar və sətirlər üzrə neçə xana – piksel hərəkət etməsini bildirir. Susmaya görə 1 götürülür.
2. Əhatəliliyin miqdarı. Bu xarakteristika isə inglisdilli ədəbiyyatda “padding” adlanır, regionlar çıxarılarkən təsvirin sərhədlərindəki xanalara nə qədər əhəmiyyət verildiyini göstərir. Susmaya görə 0 götürülür, əhatəliliyin miqdarından asılı olaraq şəklin sərhədinə boş – ağ və ya qara piksellər əlavə olunur.

(1.1.28)-də təsvir olunmuş əməliyyatda addım genişliyi və əhatəliliyin miqdarı susmaya görə olduğu kimidir, nəticədə $(m-k+1)\times(m-k+1)$ ölçülü çevrilmiş təsvir alınır. Addım genişliyi və əhatəliliyin miqdarını s və p götürsək, alınmış yeni təsvirin ölçüsü $\left(\frac{m-k+p}{s}+1\right)\times\left(\frac{m-k+p}{s}+1\right)$ olacaq.[25]

- Pulinq əməliyyatı. BNŞ-lərdə bükülmə laylarından əlavə pulinq laylarından da istifadə olunur. Bu lay ümumiləşdirməyə/icmala kömək etməklə SNŞ-lərin mühüm komponentini təşkil edir. Ədədlərdən və ya vektorlardan ibarət matris formasında verilmiş şəkildən düzbucaqlı şəklində regionlar çıxarılır, həmin region matrisi üzərində əməliyyat aparılır. Çıxarılan regionların ölçüsü eynidir, əvvəlcədən verilir. [24,51,41]

  Şəkil $X=\left(x_{ij}\right)_{i,j=1}^{m}$ və şəkildən çıxarılan region $k\times k$ ölçülü olarsa, pulinq əməliyyatının nəticəsində alınacaq şəkil $\dot{X}=\left(\dot{x}_{ij}\right)_{i,j=1}^{m-k+1}$ olacaq.

Ümumilikdə bükülmə neyron şəbəkələrdə pulinq əməliyyatının iki növündən geniş istifadə olunur.

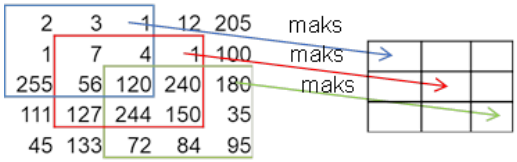
1. Orta qiymət. Bu yanaşma ilə seçilmiş regiondakı piksellərin ədədi ortası tapılır (şəkil 1.1.8). Pulinq əməliyyatı ilə çevrilmiş şəklin pikselləri aşağıdakı kimi müəyyən olunur:

$$\dot{x}_{ij} = \frac{1}{k^2}\sum_{o=i}^{i+k}\sum_{p=j}^{j+k} x_{op}\,, \quad i,j = \overline{1, m-k+1}.$$

2. Maksimum. Bu yanaşma ilə seçilmiş regiondakı piksellərin maksimumu tapılır (şəkil 1.1.9). Həmin pulinq əməliyyatı ilə çevrilmiş şəklin pikselləri aşağıdakı kimi müəyyən olunur:

$$\dot{x}_{ij} = max\{x_{op}\}_{o=i,\ p=j}^{i+k,\ j+k}, \quad i,j = \overline{1, m-k+1}.$$

Şəkil 1.1.8. Maks pulinq əməliyyatı.



Şəkil 1.1.9. Orta pulinq əməliyyatı.

Məlum BNŞ Arxitekturaları. BNŞ-nin fərqli arxitekturaları mövcuddur. Bu arxitekturalar aşağıdakı xarakteristikalara görə bir-birindən fərqlənir:

- giriş şəklinin ölçüsü,
- bükmə və pulinq səviyyələrinin sayı,
- fərqli bükmə səviyyələrindəki bükmə əməliyyatlarının sayı,
- bükmə əməliyyatının nüvəsinin ölçüsü,
- pulinq əməliyyatı zamanı təsvirdən çıxarılan regionların ölçüsü,
- tam əlaqəli süni neyronlar səviyyələrinin sayı və həmin səviyyələrdəki neyronların sayı,
- məsələnin qoyuluşu.

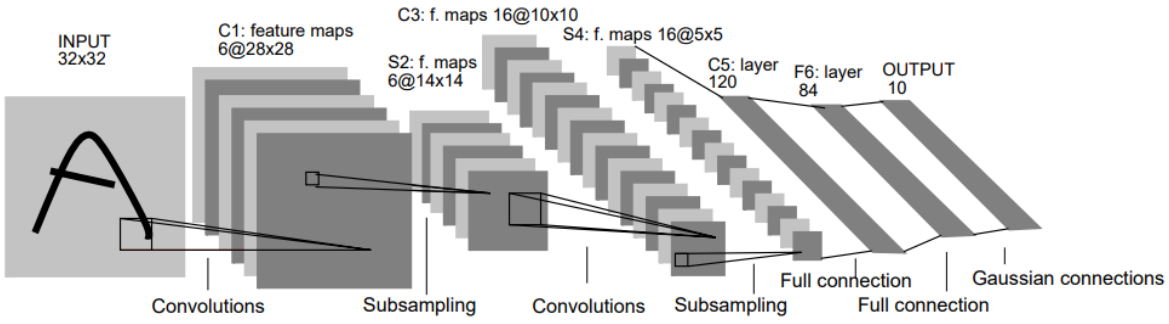
Yuxarıdakılardan sonuncusuna görə bükülmə neyron şəbəkə arxitekturaları üç cür olur:

Birinci növ BNŞ arxitekturaları obyektlərin təyin olunması (ing. object detection) məsələsində istifadə olunur. Obyektlərin tanınması məsələsi təsvirdəki obyektlərin vizual surətlərinin yeri, ölçüsü və sinfinin tapılmasını həyata keçirəcək modelin hazırlanması ilə həll olunur. Region-based CNN, SPP-Net, Fast R-CNN, Faster R-CNN, Mask R-CNN, YOLO kimi obyektlərin təyin olunması şəbəkələri mövcuddur.

İkinci növ bükülmə neyron şəbəkə arxitekturaları təsvirlərin nümunələr əsasında seqmentləşdirilməsi (ing. instance segmentation) üçün istifadə olunur. Təsvirlərin seqmentləşdirilməsi məsələsində hər biri rəng ifadə edən piksellərdən ibarət matrisin hansı xanalarının hansı obyekt olduğu verilmiş təsvirin məqsədə uyğun rənglənməsi ilə aydınlaşdırılır. Fully Convolutional Network (FCN), DeepLab, SegNet və s. arxitekturalar mövcuddur.

Nəhayət üçüncü növ məlum bükülmə neyron şəbəkə arxitekturaları təsvirlərin sinifləşdirilməsi üçün istifadə olunur. Bu halda təsvirdə bir obyekt var və həmin təsvir obyektin surətidir. Belə arxitekturalardan bəziləri aşağıdakılardır [25]:

1. LeNet-5. Bu arxitektura 1998-ci ildə Yan LeCun tərəfindən təklif olunmuş, MNIST adlanan əlyazma rəqəm şəkillərindən ibarət baza ilə sınanmışdır. 60000-ə yaxın öyrədilə bilən parametri var (şəkil 1.1.10). [50]



Şəkil 1.1.10. LeNet-5 arxitekturalı BNŞ-nin sxemi.

2. AlexNet. 2012-cu ildə ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) adlanan və hər il aparılan yarışda təxminən 1000

sinfin müşahidələrindən ibarət haradasa 1.2 milyona yaxın müşahidənin olduğu baza ilə ən aşağı xəta dərəcəli modellər arasında birinci yerə layiq görülmüşdür. Arxitekturanın təqdim olunduğu əsas məqalənin müəlliflərindən biri Cefrey Hintondur. Şəbəkəyə giriş kimi $224 \times 224 \times 3$ ölçülü şəkil götürür, ümumilikdə 60 milyona yaxın parametri var.[15,25]

3. ZFNet. ILSVRC-2013 yarışında AlexNet-i geridə qoymuşdur. AlexNet-dən fərqli olaraq bükmə nüvəsi 11×11 yox, 7×7 ölçülüdür, əhatəliliyin miqdarı isə 4 yox, 2 götürülmüşdür.
4. VGGNet. 2014-cü ildə təqdim olunmuşdur. AlexNet-ə nəzərdə daha dərin şəbəkə hesab olunur, ən məhşur arxitekturalardandır. Bu arxitekturanın sınanmasıyla kiçik ölçülü bükmə nüvələrinin çoxsaylı istifadəsi böyük ölçülü bükmə nüvələrinin effektivliyini əvəz edə bildiyi aydınlaşdırıldı. Daha dərin şəbəkə qurmağa bu imkan verir.
5. GoogLeNet. 8 səviyyəli AlexNet-dən fərqli olaraq 22 səviyyəlidir, amma parametr sayı AlexNet-in parametr sayından 12 dəfə azdır. ILSVRC-2014 yarışının qalibidir.
6. ResNet. Qradient probleminin həlli üçün bir ideya təklif olunur. ILSVRC-2015 yarışının qalibidir.
7. DenseNet. ILSVRC-2016 yarışının qalibidir.

1.2. Müəllimsiz öyrənmə üsulları

1.2.1. Klasterləmə və onun növləri

Keçən yarımfəsildə maşın öyrənməsinin növləri təsvir edildi və maşın öyrənməsinin müəllimlə öyrənmə növünün köməyi ilə parametrik identifikasiyası aparılan və surətlərin tanınması üçün istifadə olunan modellərin bəzi növləri izah olundu. Bu yarımfəsildə müəllimsiz öyrənmənin

geniş tətbiq sahələrindən biri olan klasterləmə üsullarının növləri qeyd olunmuş, bəzi klasterləmə üsulları təsvir edilmişdir.

Müəllimsiz öyrənmə adı altında çox tədqiq olunan istiqamətlərdən biri klasterləmə üsullarıdır (ing. clustering methods). Klasterləmə oxşar nümunələri eyni dəstədə qruplaşdırmaqdır. [27, səh. 238]

Verilənlərin müəyyən sayda klasterlərə ayrılmasına bölüşdürücü klasterləmə (ing. partitional clustering) deyilir. İyerarxik klasterləmədə isə nümunələr klasterləşdirilərək və ya klasterlərə bölünərək klasterlər daha böyük klasterlərə birləşdirilir və ya böyük klasterlər parçlanır. Bu yanaşmalardan birincisi iyearxik klasterləmənin toplanan, digəri parçalanan növünə daxildir.

Bölüşdürücü klasterləmənin məlum üsullarının ən çox istifadə olunanlarından biri orta qiymətlər üsuludur.

Aşağıda orta qiymətlər üsulunun təsviri verilmişdir.

1.2.2. K-ortalar üsulu

Əlamətləri $E^h = \left(e_1^h, \dots, e_n^h\right),\ h = \overline{1,H}$ olan bazadakı nümuələri K sayda klasterə k-ortalar üsulu (ing. k-means method) ilə qruplaşdırmaq lazım gəldikdə, elə $C^k = \left(c_1^k, \dots, c_n^k\right) \in R^n,\ k = \overline{1,K}$ mərkəzləri tapmaq lazım olur ki, müşahidələrin daxil olduğu dəstələriin mərkəzlərinə məsafələri cəmi minimum olsun.[26] Bunun üçün əvvəla hər hansı müşahidənin hər hansı dəstəyə mənsubunu bildirən $s_{hi} \in \{0,1\}$ binar dəyişəni daxil olunur:

$$s_{hi} = \begin{cases} 1, & əgər\ h\ nömrəli\ surət\ i\ nömrəli\ dəstəyə\ daxildirsə, \\ 0, & əks\ halda, \end{cases}$$

burada $h = \overline{1,H}$, $i = \overline{1,K}$ olur və $\sum_{i=1}^{K} s_{hi} = 1$ olmaqla bir nümunənin yalnız bir dəstəyə daxil edilə biləcəyini ifadə edir.

Kvadratlar cəmi (ing. sum of squares) yanaşmasına görə $c_i,\ i = \overline{1,K}$ mərkəzləri aşağıdakı (1.2.1)-(1.2.3) məsələnin həll edilməsi ilə tapılır:

$$\sum_{h=1}^{H}\sum_{k=1}^{K} s_{hi}\left\|E^h - C^k\right\|^2 \rightarrow^{min}_{C^1,\dots,C^K,\ s_{11},\dots,s_{HK}}, \quad (1.2.1)$$

$$s_{hi} \in \{0,1\}, \qquad h = \overline{1,H},\ \ i = \overline{1,K}, \quad (1.2.2)$$

$$\sum_{i=1}^{K} s_{hi} = 1, \qquad h = \overline{1,H}, \quad (1.2.3)$$

harada ki, $\|A - B\| - A = (a_1,\dots,a_N)$ və $B = (a_1,\dots,a_N)$ vektorları arasındakı məsafədir. Aşağıda məlum məsafələrdən bir neçəsi verilmişdir. [44]

1. Minkovski ölçüsü. Tam α ədədi verildikdə yuxarıdakı A və B vektorları arasındakı Minkovski məsafəsi aşağıda verilmişdir:

$$d^{\alpha}(A,B) = \left(\sum_{i=1}^{N}|a_i - b_i|^{\alpha}\right)^{\frac{1}{\alpha}}.$$

2. Manhatton ölçüsü:

$$\|A - B\| = d^2(A,B) = \sum_{i=1}^{N}|a_i - b_i|.$$

3. Evklif ölçüsü:

$$d^1(A,B) = \sqrt{\sum_{i=1}^{N}|a_i - b_i|}.$$

Mərkəzlərə görə zəruri şərt alınır:

$$\sum_{h=1}^{H} s_{hi}\left(c_i^k - e_i^h\right) = 0, \qquad k = \overline{1,K},\ \ i = \overline{1,n}. \quad (1.2.4)$$

(1.2.4) şərtindən mərkəzlərin aşağıdakı ifadəsi alınır.

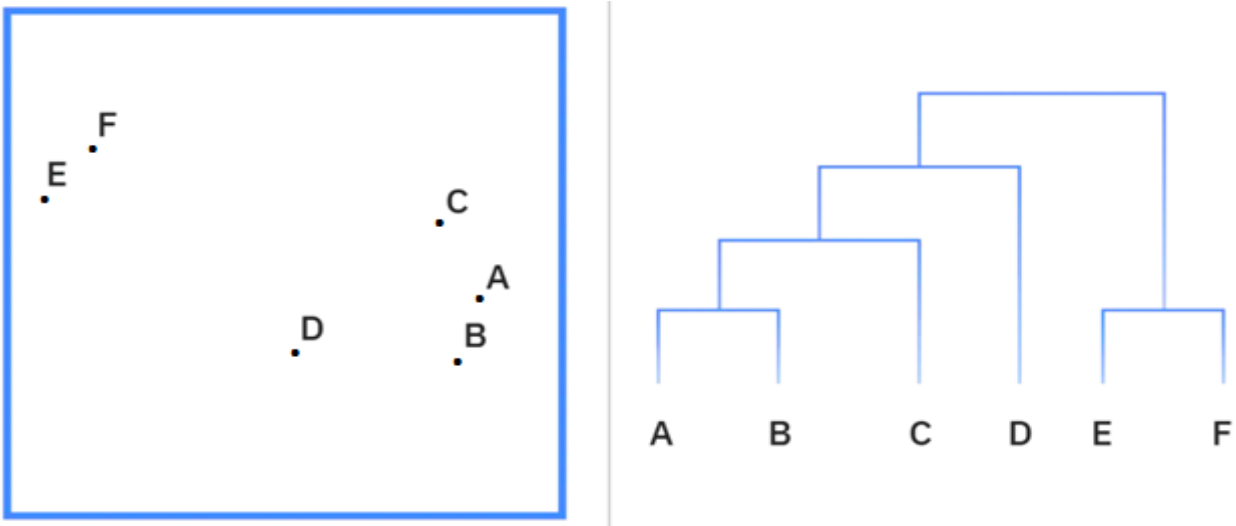
$$c_i^k = \frac{\sum_{h=1}^{H} s_{hi} e_i^h}{\sum_{h=1}^{H} s_{hi}}, \qquad k = \overline{1,K}, \;\; i = \overline{1,n}. \tag{1.2.5}$$

(1.2.5) ifadəsindən aydın olur ki, kvadratlar cəmi ifadəsinin qiymətini minimallaşdıran mərkəzlər həm də onlara daxil olan nümunələrin ağırlıq mərkəzi - orta qiymətidir. Buna əsaslanaraq bu mərkəzlərin optimal qiymətlərini tapmaq üçün aşağıdakı iterativ alqoritm təklif olunur:

1. $C^k, \; k = \overline{1,K}$ mərkəzlərinin ilkin qiyməti ixtiyari şəkildə verilir.
2. $E^h \in Q_{k^*}$: $\left\|E^h - C^{k^*}\right\| = \min\limits_{k=1,\dots,K} \left\|E^h - C^k\right\|$, harada ki, Q_{k^*} k^* sıra nömrəli dəstəyə düşən müşahidələr çoxluğudur.
3. $c_i^k = \frac{1}{|Q_k|} \sum_{\tilde{E}=(\tilde{e}_1,\dots,\tilde{e}_n)\in Q_k} \tilde{e}_i \,, \; k = \overline{1,K}, \;\; i = \overline{1,n}$, harada ki, $|Q_k|$ – Q_k çoxluğunun elementlərinin sayıdır.
4. Əgər $\forall C^k, \; k = \overline{1,K}$ dəyişirsə, addım 2-yə qayıdılır, əks halda iterasiyalar dayandırılır.

1.2.3. İyerarxik klasterləmə üsulları

İyerarxik klasterləmə, adətən, dendoqramlar vasitəsi ilə vizual olaraq göstərilə bilir (şəkil 1.2.1, [33]).



Şəkil 1.2.1. İyerarxik klasterləmənin nəticəsinin dendoqramda nümayişi.

İyerarxik dəstələşdirmə üsullarının alqoritmlərinin yalançı kodu aşağıda verilmişdir:

Sadə iyerarxik toplanan dəstələşdirmə alqoritminin təsviri:

1. Hər nümunəyə bir elementli bir dəstə yaradılır.
2. İki yaxın dəstə bir dəstələrdə birləşdirilir.
3. Bütün nümunələr cəmi bir dəstədə birləşənə kimi addım 2 təkarlanır.

Sadə iyerarxik parçalanan dəstələşdirmə alqoritminin təsviri:

1. Bütün nümunələrdən ibarət bir dəstə yaradılır.
2. Bir dəstə iki bir-birindən uzaq iki dəstəyə bölünür.
3. Hər dəstədə cəmisi bir nümunə qalana kimi addım 2 təkrarlanır.

İyearxik dəstələşdirmə aparmaq üçün dəstələr arasında məsafə anlayışını daxil etmək lazımdır. Q_1, Q_2 iki nümunələr dəstəsi olduqda aşağıdakı dəstələr arası məsafə meyarlarını daxil etmək olar [10]:

1. Minimum yanaşması:
$$D_{min}(Q_1, Q_2) = \min_{A \in Q_1, B \in Q_2} d^{\alpha}(A, B),$$
burada $d^{\alpha}(A, B) - A, B$ əlamətli nümunələr arasındakı Minkovski məsafəsidir, bu məsafə $\alpha = 2$ olduqda Evklid məsafəsi olur.
2. Maksimum yanaşması:
$$D_{max}(Q_1, Q_2) = \max_{A \in Q_1, B \in Q_2} d^{\alpha}(A, B).$$
3. Müşahidələrin cüt-cüt məsafələrinin orta qiymətləri yanaşması:
$$D_{avg}(Q_1, Q_2) = \frac{1}{|Q_1||Q_2|} \sum_{A \in Q_1} \sum_{B \in Q_2} d^{\alpha}(A, B),$$
burada $|Q_i| - Q_i$ çoxluğunun elementlərinin sayıdır, harada ki, $\forall i = 1,2$.
4. Mərkəzlərin məsafəsi yanaşması:
$$D_{mean}(Q_1, Q_2) = d^{\alpha}(C_1, C_2),$$
burada C_1, C_2 uyğun olaraq Q_1, Q_2 dəstələrindəki nümunələrin ağırlıq mərkəzləridir.

1.3. Qarışıq öyrənmə üsulları

Texnoloji inkişaf rəqəmsal verilənlərin artımına səbəb oldu. Lakin bu verilənlərin böyük bir qismi etiketlənməmiş haldadır. Hansı ki, bu etiketlənməmiş verilənlər müəllimlə öyrənmədə birbaşa istifadə oluna bilmir. Verilənlərin yalnız bir qismi etiketlənmiş olduqda həmin etiketlənmiş qisim əsasında etiketlənməmiş verilənlərin etiketlənməsi üçün fərqli yanaşmalar yarımmüəllimlə öyrənmə (ing. semi supervised learning) başlığı altında araşdırılır.

Yarımmüəllimlə öyrənmə üçün istifadə olunan fərqli yanaşmalar var. Onlardan birini öz-özünə öyrənmə, öz-özünə etiketləmə və ya proqnoz əsaslı öyrənmə (ing. self-training, self-labeling, decision-directed learning) adlandırmaq olar. Bu yanaşma yarımmüəllimlə öyrənmədə ilk ağla gələn yanaşmadır, müəllimlə öyrənmədən ardıcıl istifadə etməklə etiketlənməmiş verilənlərin etiketlənməsini özündə ehtiva edir. Bu yanaşmada bir model etiketlənmiş məlumatların kiçik bir hissəsi ilə öyrədilir, sonra bu modelin köməyi ilə - model proqnozu əsasında etiketlənməmiş verilənlər etiketlənir. Sonra artıq istifadə olunmuş etiketlənmiş verilənlərlə yenicə model tərəfindən etiketlənən verilənlər bir bazada birləşdirilərək, yenidən bir model öyrədilir və bu addımlar bütün verilənlər etiketlənənə qədər davam edir. Bu yanaşma etiketlənməmiş verilənlərin sayı etiketlənmiş verilənlərdən çox olduqda istifadə olunur. Etiketlənmiş və etiketlənməmiş verilənlər uyğun olaraq G_+, G_- olduqda öz-özünə öyrənmə yanaşmasının alqoritmi aşağıdakı kimidir:

1. Etiketlənmiş verilənlər G_+-dən müəyyən $g_+ \subset G_+$ seçilir.
2. Bir model g_+ çoxluğundakı ilə öyrədilir.
3. Etiketlənməmiş verilənlər G_--dən müəyyən $g_- \subset G_-$ seçilir.
4. Addım (2)-də öyrədilmiş modelin köməyi ilə g_- çoxluğundakı verilənlərin sinfi proqnozlaşdırılır.
5. Addım (4)-də alınan proqnoz nəticələri və g_- G_+ bazasına əlavə olunur.

6. $g\_$ verilənləri çoxluğu $G\_$ bazasından silinir.
7. $G\_$ boşdursa, alqoritm bitir, əks halda Addım(1)-ə qayıdılır.

Yarımmüəllimlə öyrənmədə istifadə olunan digər bir yanaşma qarşılıqlı öyrənmə (ing. co-training) adlanır. Bu halda dataset şərti olaraq iki hissəyə bölünür. Hər hissə üçün ayrı bir model qurulur. Bu modellərdən biri müəyyən bir ehtimal etibarlılığını (ing. probability confidence) aşdıqda həmin "inamlı" modelin cavabları əsasında digər model təzədən öyrədilir.

Bunlarla yanaşı qraf əsaslı yarımmüəllimlə öyrənmə (ing. graph based Semi Supervised Learning), Qaus qarışdırma modelləri (ing. Gaussian Mixture Models), klasterlə və etiketlə (ing. cluster and label), yarımmüəllimlə öyrənən dayaq vektor üsulu (S3VM, ing. Semi Supervised Support Vector Machine), generativ üsullar (ing. generative methods) yanaşmaları mövcuddur. Qraf əsaslı müəllimlə öyrənmə böyük və real datasetlər üçün asanlıqla istifadə oluna bilir, məlumatların düyün nöqtələrində, əlaqələrin tillərdə saxlanılmasına əsaslanır. Qaus qarışdırma modellərində yarımmüəllimlə öyrənmənin köməyi ilə klasterləmənin keyfiyyətini artırmaq, klasterləmə vaxtının azaldılmasına nail olmaq mümkün ola bilir. Klasterlə və etiketlə yanaşmasında öyrətmə üçün fərqli bazaların istifadəsində müəllimlə öyrənmənin nəticəsinin ciddi şəkildə dəyişməsinin aradan götürülməsi hədəflənir. Bazadakı müşahidələr sinifləşdirilməsi sadə olacaq şəkildə siniflərə ayrılır. S3VM-də bölüşdürmə aparan qərar funksiyası haqqında hipotez etiketlənməmiş məlumatlardan asılı olur. [18,40]

II FƏSİL. TƏSVİRLƏRDƏKİ OBYEKTLƏRİN TANINMASINA BƏZİ YANAŞMALARIN TƏTBİQİ

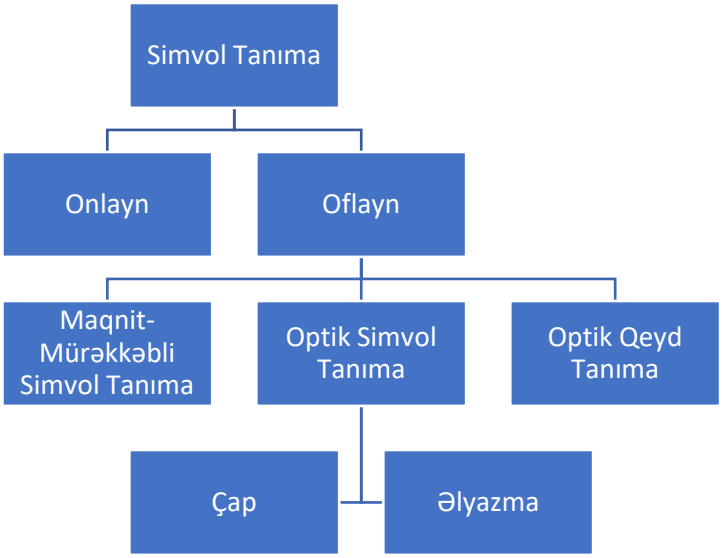
2.1. Təsvirlərdəki obyektlərin tanınması məsələsinin qoyuluşu.

2.1.1. Optik simvol tanıma məsələlərinin təsviri.

Optik simvol tanıma (OCR, ing. optical character recognition) məsələləri elektron sənəd dövriyyəsi sistemlərinin ayrılmaz hissəsidir, avtomobil nömrələrinin avtomatik tanınması, görmə məhdudiyyətli insanların yazılı məlumatlarla işləmə imkanının yaradılması və s. məqsədlərlə geniş istifadə olunur. Bununla yanaşı latın qrafikalı mətnlərin, o cümlədən yapon, çin, koreya, hind, ərəb və s. dillər kimi latın qrafikalı olmayan əlifbalarla yazılmış mətnlərin elektron hesablama cihazlarına daxil edilməsinin sadələşdirilməsi də OCR məsələlərinin mühüm tətbiq sahələrindəndir. Ona görə də, Azərbaycan əlifbası ilə yazılmış mətnlərin şəkillərinin emalının köməyilə həmin mətnlərin elektronlaşdırılması üçün yanaşmaların, üsul, alqoritm və proqram təminatının işlənilməsi əhəmiyyət kəsb edir.

Fərqli əlifbalar üçün işlənilən OCR məsələlərinin həllərində istifadə olunan yanaşmalar, üsullar və proqram təminatları arasında oxşar cəhətlər tapmaq mümkün olsa da hər dilin OCR məsələsi təsvirlərdə obyektlərin tanınması məsələlər sinfində geniş və yeni tədqiqat imkanları açır. Dissetasiya işində təklif olunan yanaşmalar və əlamətlər Azərbaycan əlifbasının böyük çap əlyazma hərflərinin tanınması məsələsi üzərində tədqiq olunmuşdur.

Optik simvol tanıma məsələləri simvol tanıma məsələlər sinfinin bir hissəsidir. Simvol tanıma məsələlərini şərti olaraq onlayn və oflayn olmaqla iki qrupa bölmək olar. Tanıma onlayn simvol tanımada istifadəçi hərfləri yazdığı anda, oflayn simvol tanımada isə istifadəçi mətni yazıb bitirdikdən sonra aparılır. Oflayn simvol tanıma da öz növbəsində növlərə ayrılır, ümumi təsnifat sxemi şəkil 2.1.1-də verilmişdir [35]:



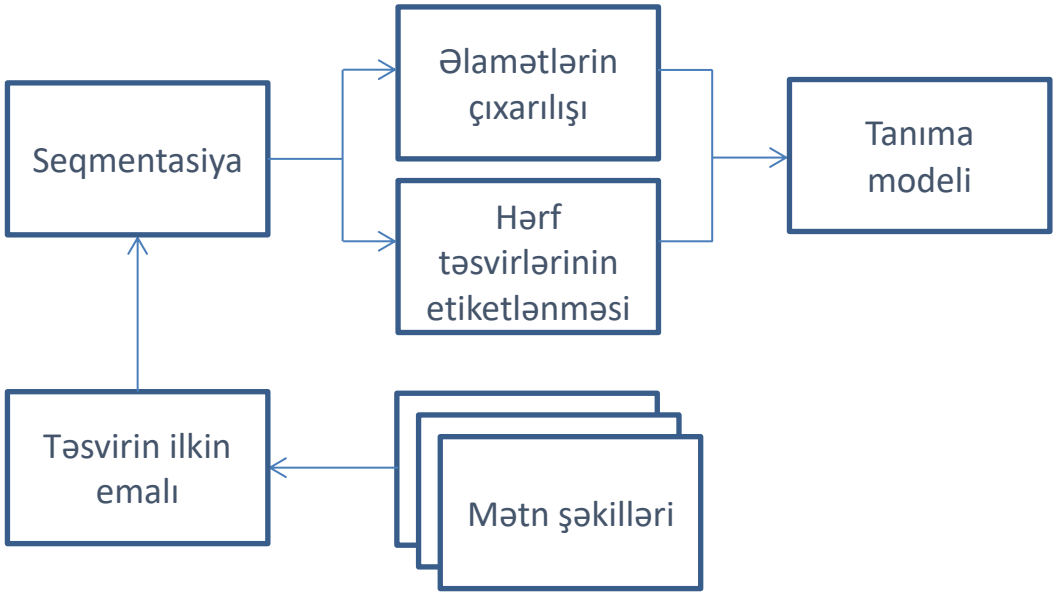
Şəkil 2.1.1. Simvol tanımanın növləri.

2.1.2. OCR sisteminin iş mərhələləri və yaradılmasının təsviri.

OCR sistemlərinin iş mərhələləri aşağıdakı kimidir:

1. Mətn təsvirinin əldə olunması. Bu mərhələdə mətnin təsviri mətnin yazıldığı fiziki səhifənin şəkli çəkilməklə və ya skan olunmaqla kompüterə daxil edilir.
2. Mətn təsvirinin ilkin emalı. Mətn təsvirinin üzərində morfoloji əməliyyatlar, filterləmə, təsvirin rəng formatının dəyişdirilməsi və s. aparıldıqdan sonra, mətnin əyriliyi düzəldilir.
3. Mətn təsvirinin seqmentasiyası. Mətndəki ayrı-ayrı hərflərin təsvirləri 2-ci mərhələdən alınmış mətn təsvirindən çıxarılır.
4. Əlamətlərin çıxarılışı. Mətn təsvirindən alınmış hər hərf şəklinin əlaməti çıxarılır.
5. Hərf şəkillərinin sinfinin təyini. Hər hərf şəklinin əlaməti ayrı-ayrılıqda tanıma modelinə ötürülür. Model hər hərf şəklinin sinfini, yəni hansı hərf olduğunu təyin etdikdən sonra hərf şəkilləri ilkin təsvirdəki ardıcıllıqla onların sinfini bildirən işarələrlə - rəqəmsal simvollarla əvəz olunur.

OCR sisteminin iş mərhələlərindən 5-cisinin işini təmin etmək üçün sinifləşdirmə aparacaq tanıma modeli hazırlanmalıdır. Modelin hazırlanması struktur və parametrlərin seçimi olmaqla iki mərhələdə aparılır. Struktur seçimi öyrətmə bazasını ümumiləşdirəcək modelin növünün seçilməsidir. Seçilmiş strukturdakı modelin parametrlərinin seçimi isə öyrətmə bazasının köməyi ilə aparılır. Burada maşın öyrənməsinin müəllimlə öyrənmə növü istifadə olunur. Ümumilikdə, OCR sistemində istifadə olunacaq tanıma modelinin parametrlərinin seçimi şəkil 2.1.2-dəki kimi aparılır [34]:



Şəkil 2.1.2. OCR sisteminin iş mərhələləri.

2.1.3. Məsələnin qoyuluşu.

Azərbaycan dilində çap əlyazma hərflərinin tanınması formaların tanınması məsələlərinin həllində geniş istifadə olunur. Dosent, r.ü.f.d. E.E. Mustafayev fəlsəfə doktoru üzrə dissertasiya işində formaların emalını tədqiq etmişdir, ədədi eksperimentlərin aparılması və formaların tanınması proqram təminatının hazırlanması üçün Azərbaycan əlifbasının 32 hərfindən 28-nin nümunələrindən ibarət baza yığılmış və istifadə olunmuşdur. Həmin hərflər aşağıdakılardır:

{ABCÇDEƏFGHXIJKQLMNOPRSŞTUVYZ}.

Azərbaycan əlifbasındakı {ĞİÖÜ} hərflərinin tanınması daha sadə üsullarla həyata keçirilə bildiyindən bazada bu hərflərə aid nümunələr mövcud deyil.

Dissertasiya işində həmin baza üzərində fərqli tanıma modeləri, fərqli yanaşmalarda, fərqli əlamətlərin istifadəsi ilə sınanmışdır.

2.2. Tanıma məsələlərinin həllinə təklif edilən yanaşmaların təsviri

2.2.1. Yanaşmaların ümumi təsviri.

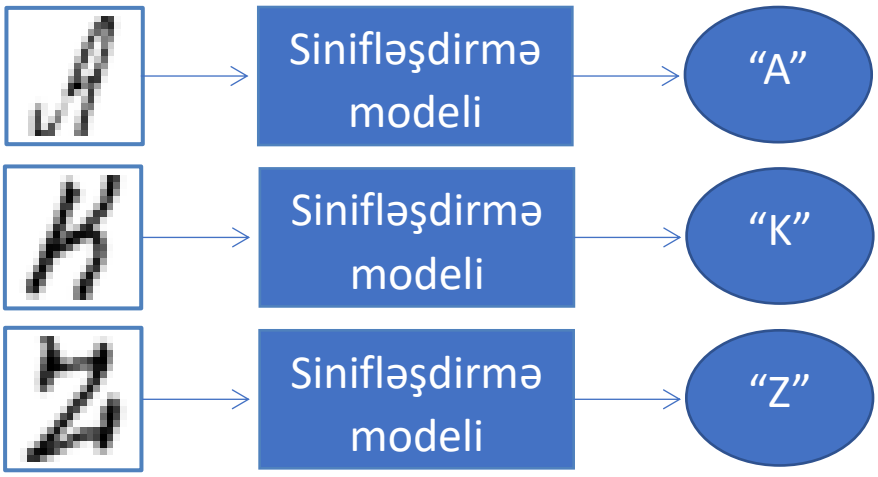
Dissertasiya işində təklif olunan yanaşmalar tanımanı aparan aparatın quruluşuna görə fərqlənir. Başqa sözlə, keçən yarımfəsildə OCR sisteminin iş mərhələlərindən axırıncısının seçiminə görə dörd yanaşma təklif oluna bilər, onlar aşağıdakılardır:

Dissertasiya işində təklif olunan yanaşmalar tanımanı aparan aparatın quruluşuna görə fərqlənir. Başqa sözlə, keçən yarımfəsildə OCR sisteminin iş mərhələlərindən axırıncısının seçiminə görə dörd yanaşma təklif oluna bilər. Bu yanaşmalardan ikisi sinifləşdirmədə dəqiqliyi artırmaq üçün klasterləmədən istifadə etməyi özündə ehtiva edir (Yanaşma 2 və 3). Digər iki yanaşmanın birində (Yanaşma 1) tanımanı sinifləşdirmə modeli aparır. Nəhayət son yanaşmada (Yanaşma 4) hər hərf üçün bir model qurulur, verilmiş hərf təsvirinin sinfi bütün bu modellərdə sınanır.

2.2.2. Birinci yanaşmanın təsviri.

Bu yanaşma məlum və tanıma məsələlərində susmaya görə istifadə olunan yanaşmadır. Tanıma üçün I fəslin 1-ci yarımfəslində izah olunan üsullardan biri istifadə olunur. Bu yanaşmada vahid tanıma modeli var, bu model bütün hərflərə aid nümunələrlə öyrədilir. Verilmiş hərf təsvirinin sinfi – hansı hərf olduğu həmin modelin köməyi ilə təyin olunur. Dissertasiya işində

bu yanaşma üçün tanımanı aparan aparat kimi Pixels və PDC əlamətləri ilə Süni Neyron Şəbəkə (SNŞ) və Bükülmə Neyron Şəbəkə (BNŞ) istifadə olunmuşdur. Yanaşma 1-in sxemi şəkil 2.2.1-də verilmişdir.



Şəkil 2.2.1. Yanaşma 1-in hazırlanma və iş sxemi.

2.2.3. İkinci yanaşmanın təsviri.

Yanaşma 2 və Yanaşma 3-də tanıma üçün istifadə olunan aparatın hazırlanmasında sadəcə müəllimlə öyrənmə yox, həm də müəllimsiz öyrənməyə müraciət olunur. Başqa sözlə, tanımada dəqiqliyi artırmaq üçün sinifləşdirmə ilə klasterləmə birlikdə istifadə olunur. Yanaşma 2-də tanıma üçün aparatın hazırlanması aşağıdakı addımlar atılır (şəkil 2.2.2):

1. Öyrətmə bazasındakı şəkillər verilmiş əlamətlərdən birinin köməyi ilə klasterlərə ayrılır.
2. Öyrətmə bazasındakı hər şəkil və onun hansı klasterə düşdüyünü bildirən qiymətdən ibarət cütlərlə bir SNŞ öyrədilir. (Bu SNŞ verilmiş hərf təsvirinin mövcud klasterlərdən hansına yaxın olması təyin olunması üçün istifadə olunacaq.)
3. Hər klaster üçün ayrı sinifləşdirmə modeli qurulur, həmin model həmin klasterə düşmüş verilənlərlə öyrədilir.

Yuxarıdakı addımlardan 2-cisində də qeyd olunduğu kimi verilmiş hərf təsvirinin hansı klasterə aid olması klasterləşdirilmiş öyrətmə verilənlərinin köməyi ilə öyrədilmiş model vasitəsi ilə təyin olunur. Verilmiş hərf şəklinə uyğun klasterinin təyin olunması üçün K-ortalar üsulundakı mərkəzlərə yaxınlıq əsas götürmək olar. Lakin ədədi eksperimentlər zamanı aşkar olundu ki, verilmiş hərf təsvirinin klasterləşdirilməsi mərkəzlərə yaxınlıq əsasında aparıldıqda test müşahidələrinin əksəriyyətində hərf şəkli elə klasterə daxil olur ki, o klasterdə həmin test müşahidəsinin sinfinə aid müşahidə yoxdur.

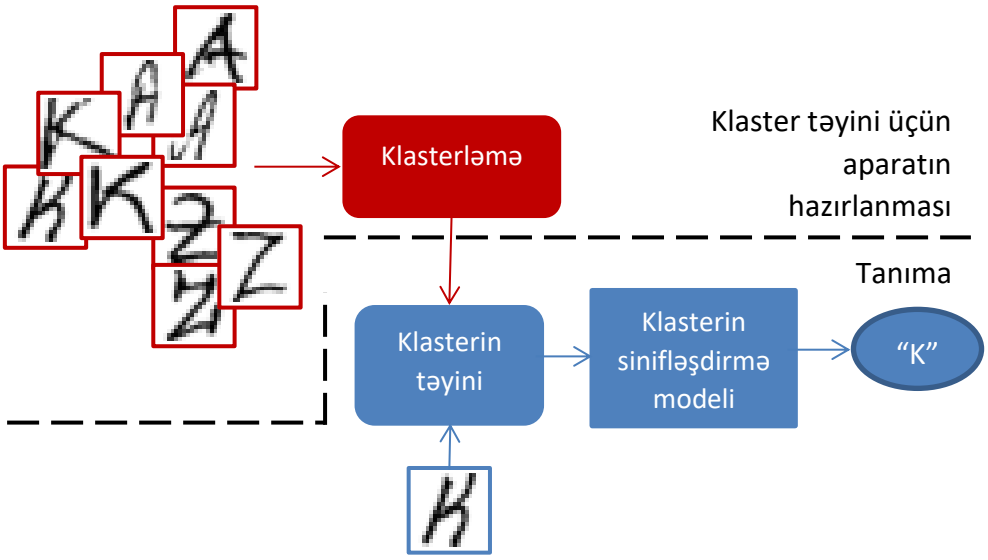
Klasterləşdirmə K-ortalar üsulu ilə aparılmış, 10 başlanğıc mərkəzdən ən yaxşı nəticə verəni seçilmiş, işlədilmişdir.

Bu yanaşmada klaster sayı kimi 0, 14000, 14001, 14002,... götürüldükdə klasterləşdirmənin həlli trivialdır. Yanaşma trivial olmayan klaster sayları arasından 2,...,50 götürülərək sınanmışdır.

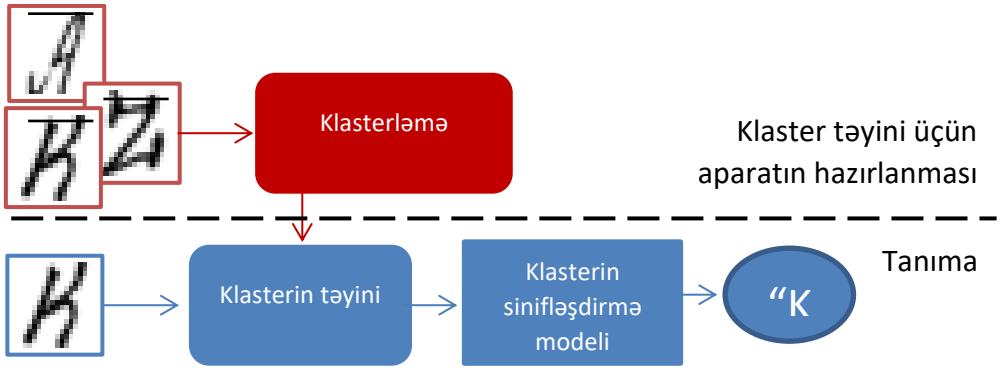
2.2.4. Üçüncü yanaşmanın təsviri.

Yanaşma 3-də tanımanı aparacaq aparatın hazırlanması üçün aşağıdakı mərhələləri yerinə yetirmək lazım gəlir (şəkil 2.2.3):

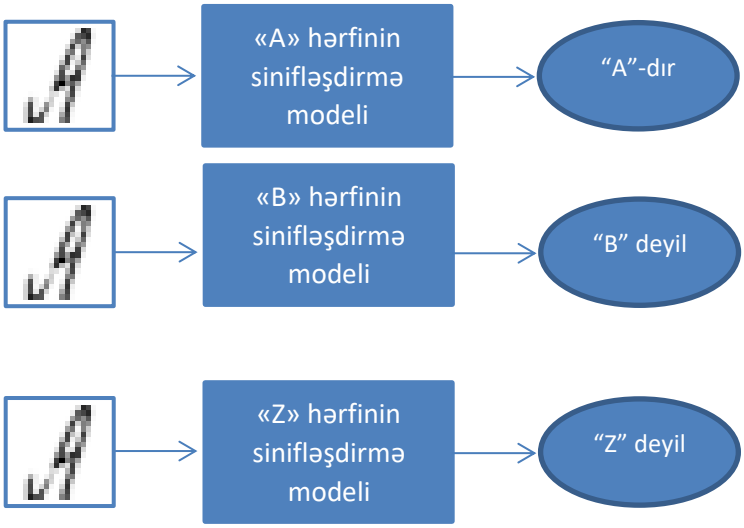
1. Hər hərfə - sinfə aid şəkillərin əlamətlərinin orta qiyməti hesablanır. Nəticədə hər hərfi xarakterizə edən bir nöqtə təyin olunur.
2. Mərhələ (1)-də alınmış 28 nöqtə klasterlərə ayrılır.
3. Öyrətmə bazasındakı hər şəkil və onun hansı klasterə düşdüyünü bildirən qiymətdən ibarət cütlərlə bir SNŞ öyrədilir. (Bu SNŞ verilmiş hərf təsvirinin mövcud klasterlərdən hansına yaxın olması təyin olunması üçün istifadə olunacaq.)
4. Hər klaster üçün ayrı sinifləşdirmə modeli qurulur, həmin model həmin klasterə düşmüş verilənlərlə öyrədilir.



Şəkil 2.2.2. Yanaşma 2-nin hazırlanma və iş sxemi.



Şəkil 2.2.3. Yanaşma 3-ün hazırlanma və iş sxemi.



Şəkil 2.2.4. Yanaşma 1-in hazırlanma və iş sxemi.

Bu yanaşmanın Yanaşma 2-dən əsas fərqi ondan ibarətdir ki, Yanaşma 2-də bir hərfə aid müşahidələrdən bəziləri bir klasterə, digərləri başqa bir klasterə düşə bilirdi. Lakin Yanaşma 3-də bir hərfə aid bütün müşahidələr eyni klasterə daxil olur.

Klasterləşdirmə K-ortalar üsulu ilə aparılmış, 10 başlanğıc mərkəzdən ən yaxşı nəticə verəni seçilmiş, işlədilmişdir.

Yanaşma klaster sayı 2,...,27 götürülərək sınanmışdır. Yəni trivial həll sayılmayan bütün klaster sayları yoxlanılmışdır.

2.2.5. Dördüncü yanaşmanın təsviri.

Yanaşma 4-də hər hərf – sinif üçün ayrı sinifləşdirmə modeli qurulur (şəkil 2.2.4). Hər hərf modeli verilmiş hərf şəklinin müvafiq hərf olub-olmadığını aydınlaşdırır. Verilmiş hərfin sinfi tapılmalı olduqda hərf şəklinin əlaməti bu modellərin hər birinə ayrı-ayrılıqda verilir. Bu halda nəticə aşağıdakılardan biridir:

1. Verilmiş hərf təsvirini hərf modellərindən sadəcə biri tanıyır, digərləri isə bu hərf təsviri bu hərfin təsviri olmadığını deyir.
2. Verilmiş hərf təsvirini birdən çox hərf modeli tanıyır. Məsələn, “B” hərfinin bir təsvirin əlaməti modellərə verildikdə, bu hərf təsvirini həm “B” hərfinin modeli, həm də “R” hərfinin modeli tanıyır.
3. Verilmiş hərf təsvirini heç bir model tanımır.

2.3. Təsvirlərdəki obyektlərin əlamətlərinin çıxarılışı alqoritmlərinin təhlili

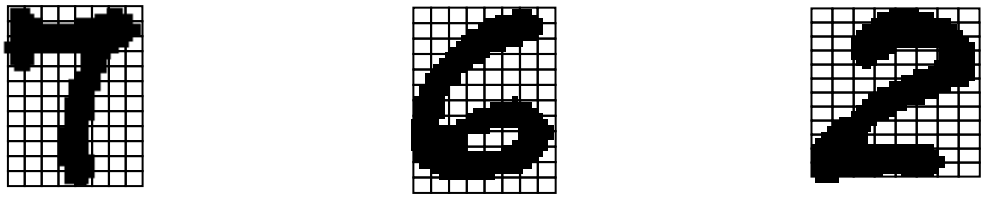
2.3.1. Bəzi məlum əlamətlərin təsviri.

Təsvirlərdən əlamət çıxarılışı təsvirlərdə tanıma məsələlərinin mühüm mərhələlərindəndir. İnformativ əlamət verilmiş surətin – burada təsvirin ifadəedicisi olmaqla sinifləşdirmənin effektivliyinə birbaşa təsvir edir, eyni sinifdəki fərqli nümunələrin ümumiləşdirilməsinə imkan verir. Aşağıda təsvirlərdən çıxarılan məlum informativ əlamətlərin təsnifatı verilmişdir (şəkil 2.3.1):

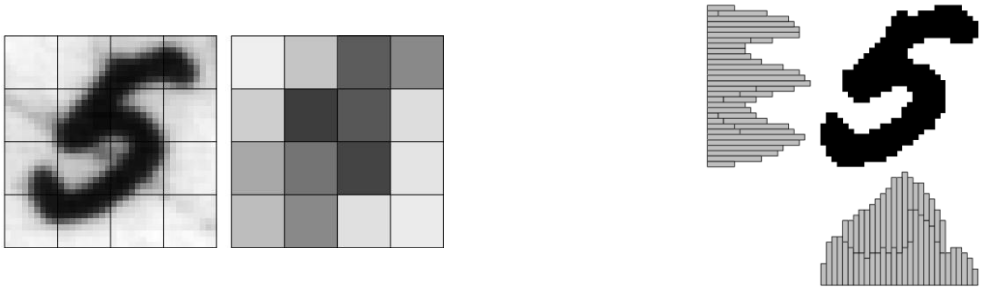
1. Qlobal əlamətlər. Bu cür əlamətlərdə təsvir olunmuş simvolun həndəsi forması və yaxud topoloji sturkturu nəzərə alınmadan təsvirinin ibarət olduğu piksellərin rəngi götürülür. Əlamətlərin bu qrupu küyə qarşı həssasdır. Eyni sinifdəki simvol təsvirlərinin ayrı-ayrı simvol siniflərində olmasına, ya da fərqli sinif simvol təsvirlərinin eyni sinifdə olmasına gətirib çıxara bilər.
2. Statistik əlamətlər. Bu qrup əlamətlərdə ağırlıq mərkəzinin xüsusiyyətləri, kəsişmə nöqtələri, ərazilərə görə paylanma və s. nəzərə alına bilər. Bu qrup əlamətlərdə müəyyən dinamik və topoloji informasiya mövcud ola bilir, ona görə də, qlobal əlamətlərə nəzərən küylərə qarşı daha az həssas ola bilər. Buna baxmayaraq, simvolların təsvir olunması variasiyaları artdıqca bu əlamətlər də eyni sinifdəki fərqli təsvirli simvolların eyni, fərqli sinifdəki oxşar təsvirli simvolların ayrı sinifdə təyin olunmasına nəzərdə yaxşı nəticə verməyə bilər.
3. Həndəsi və ya topoloji əlamətlər. Bu cür əlamətlər fərqli variasiyalardan, demək olar ki, asılı olmayaraq obyektin xarakterik xassələrini özündə ehtiva edir. Bu qrup əlamətlər təhriflər və küylərə qarşı daha az həssasdır.

Yuxarıdakı qrup əlamətlərin çıxarılışı üçün fərqli yanaşmalar mövcuddur. Məsələn, 3-cü qrup – həndəsi və topoloji əlamətlərin çıxarılışı üçün alqoritmlərindən biri ingilisdilli ədəbiyyatlarda “Counter direction” (CD)

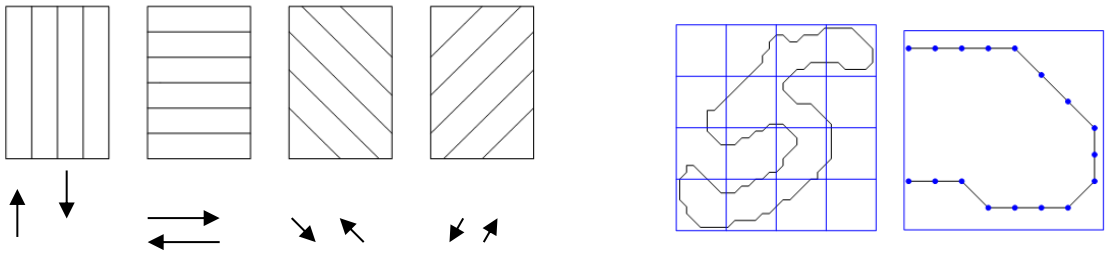
adlanır. Belə bir alqoritmlə obyektin konturu, təsvirindəki xəttlər və əyrilər nəzərə alına bilir.



a)



b)



c)

Şəkil 2.3.1. Əlamətlərin seçilməsi nümunələri

a) qlobal əlamətlər
b) statistik əlamətlər (ərazilərə görə paylanma, histoqram analizi)
c) topoloji

Digər bir əlamət çıxarılışı alqoritmi isə I fəslin 1-ci yarımfəslinin 3-cü bəndində izahatı verilən bükülmə (ing. convolution) əməliyyatıdır. Bu əməliyyatın köməyi ilə qlobal və topoloji əlamətlər çıxarıla bilər. Öyrətmə

mərhələsində bükülmə əməliyyatı/əməliyyatları üçün elə nüvə/nüvələr tapmaq olar ki, həmin nüvəli bükülmə əməliyyatı informativ əlamət çıxarılışı üçün istifadə oluna bilər.

2.3.2. Tanımada istifadə olunan əlamətlər.

Dissertasiya işində qlobal, statistik, topoloji əlamətlər və üç fərqli əlamət çıxarılışı alqoritmi istifadə olunmuşdur. Həmin əlamətlər aşağıdakılardır:

1. Əlamət – Pixels. Şəkil 20×20 ölçüsünə gətirilmiş, pikselləri bildirən ədədlər sətirlər ardıcıl olmaqla bir vektora yığılmışdır. Bu əlamət qlobal əlamətlər qrupuna daxildir. Verilmiş şəklin eni və uzunluğu $m = 20$ olduqda şəkil $\bar{X} = \left(\left(\bar{x}_{i,j}\right)\right)_{i,j=1}^{m} \in R^{m \times m}$ olarsa, əlamət vektoru $\bar{E} = (\bar{e}_1, \dots, \bar{e}_{m^2}) \in R^{m^2}$ olarsa, həmin vektorun elementlərinin qiymətləri aşağıdakı kimi tapılır:

$$\bar{e}_{(i-1)m+j} = \bar{x}_{i,j}, \qquad i = 1,2, \dots, m, \qquad j = 1,2, \dots, m.$$

2. Əlamət – Hist. Histoqramlar statistik əlamətlər qrupuna daxildir. Histoqram əlaməti vektoru üfüqi histoqram vektoru $\bar{E}^{b_1} = \left(\bar{e}_1^{b_1}, \dots, \bar{e}_m^{b_1}\right)$ və şaquli histoqram vektoru $\bar{E}^{b_2} = \left(\bar{e}_1^{b_2}, \dots, \bar{e}_m^{b_2}\right)$-nin birləşməsidir: $E^{b,p} = \left(\bar{e}_1^{b_1}, \dots, \bar{e}_m^{b_1}, \bar{e}_1^{b_2}, \dots, \bar{e}_m^{b_2}\right) \in R^{2m}$. m=32 enli və uzunluqlu şəkillərdən çıxarılan əlamət vektorunun elementlərinin sayı $2 \times 32 = 64$-dür, qiymətləri aşağıdakı kimi tapılır:

$$\bar{e}_i^{b_1} = \left|\left\{\bar{x}_{i,j}: \bar{x}_{i,j} > 0.5, \ j = 1,2, \dots, m\right\}\right|, \qquad i = 1,2, \dots, m,$$

$$\bar{e}_i^{b_2} = \left|\left\{\bar{x}_{i,j}: \bar{x}_{i,j} > 0.5, \ i = 1,2, \dots, m\right\}\right|, \qquad j = 1,2, \dots, m,$$

burada $|A|$ – A vektorunun elementlərinin sayıdır.

3. Əlamət – PDC (ing. Peripheral Directional Contributivity). Bu əlamət topoloji əlamətlər sırasına daxildir. Japon və Çin hərflərinin tanınmasında istifadə olunub. Qara piksellərin bir-biri ilə əlaqəli

yerləşmələrini təyin olunur, bunun üçün yuxarıda adı çəkilən CD qrup alqoritmlərdən biri aparılır, əlamət bu alqoritm vasitəsi ilə çıxarılır. Dissertasiyada istifadə olunan PDC əlamətinin çıxarılışı təsvirə fərqli istiqamətlərdə yaxınlaşaraq ilk və ikinci qara pikselin tapılıb, başlanğıc nöqtə təyin olunduqdan sonra bu iki nöqtənin hər birindən fərqli istiqamətlərə gedərək ilk qara piksellərə çatmaq üçün lazım olan addımlar sayılır. Dissertasiya işində istifadə olunan PDC əlamətinin çıxarılışı alqoritmi aşağıdakı kimidir:

1. Təsvirə 4 istiqamətdən yanaşılır: soldan sağa, sağdan sola, yuxarıdan aşağı və aşağıdan yuxarı,
2. Təsvirin dörd tərəfinin hər biri 8 seqmentə bölünür,
3. İlk 2 qara piksel təyin olunur, bu iki nöqtələr başlanğıc nöqtə adlandırılır,
4. Bu başlanğıc nöqtələrdən 4 istiqamətdə qara piksel axtarışına başlanılır, addımlar sayılır.

Nəticədə hər təsviri ifadə edən PDC əlaməti $4 \times 8 \times 2 \times 4 = 256$ elementli vektordur. Qeyd: Bu əlamət təsvirin ölçüsündən asılı deyil. Simvolun təsvirinin ölçüsü 20×20-dür, bu təsvir ağ rəngli piksellərlə əhatə olunaraq şəklin ümumi ölçüsü 32×32 olur.

4. Əlamət – Conv. Bu əlamət həm qlobal, həm də topoloji xarakteristikaları özündə saxlayır. Əlamətin çıxarılışı isə digər əlamət çıxarılışı alqoritmlərindən fərqlənir. Conv əlamət çıxarılışı BNŞ-lərdə fərqli laylarda fərqli saylardakı bükülmə əməliyyatları vasitəsi ilə həyata keçirilir. Dissertasiya işində LeNet-5 arxitekturalı BNŞ-lərdən istifadə olunub. Ənənəvi LeNet-5 arxitekturasında 2 bükülmə layı, bu laylarda isə uyğun olaraq 6, 16 bükülmə filteri var. Bu iki laydakı filterlərin sayını dəyişməklə həm faktiki olaraq fərqli əlamətlər alınır, həm də bu şəbəkələrin öyrədilməli parametr sayları bir-birindən fərqlənir.

Dissertasiya işində istifadə olunan BNŞ arxitekturalarının xarakteristikaları III fəsildə verilmişdir.

Qeyd: Giriş şəklinin və şəklin mərkəzindəki təsvirin ölçüsü PDC əlamətin çıxarılan şəkildəki kimi, uyğun olaraq, 20×20 və 32×32-dir.

III FƏSİL. KOMPÜTER EKSPERİMENTLƏRİNİN NƏTİCƏLƏRİ

3.1. Kompüter eksperimentlərinin təsviri

3.1.1. Yanaşmalarda istifadə olunan sinifləşdirmə modellərinin bəzi xüsusiyyətləri.

Yanaşma 1. “Pixels” və “PDC” əlamətləri ilə Süni Neyron Şəbəkə (SNŞ) istifadə olunmuşdur. Hər əlamət üçün iki fərqli struktur seçilmişdir (şəkil 3.1.1).

Cədvəl 3.1.1. “Pixels” və “PDC” əlamətləriylə işləyən modellərin strukturlarının təsvirləri.

N	Model	Layların sayı	Laylardakı neyron sayları	Parametrlərin sayı
1	Pixels_400_50_28	3	400–50–28	21478
2	Pixels_400_50_50_28	4	400–50–50–28	24028
3	PDC_256_50_28	3	256–50–28	14278
4	PDC_256_50_50_28	4	256–50–50–28	16828

Bu iki əlamətlə işləyən SNŞ-lərdən savayı LeNet-5 arxitekturalı Bükülmə Neyron Şəbəkə (BNŞ) istifadə olunmuş, ənənəvi LeNet-5 arxitekturasının ilk və ikinci bükülmə laylarındakı filterlərin sayını dəyişməklə fərqli BNŞ arxitekturaları alınmışdır. Həmin modellərin strukturlarının detalları cədvəl 3.1.2-də təsvir olunmuşdur:

Cədvəl 3.1.2. BNŞ modellərinin strukturlarının təsvirləri.

N	Model	1-ci bükülmə laydakı filterlərin sayı	2-ci bükülmə laydakı filterlərin sayı	Tam əlaqəli laylardakı neyron sayları	Parametrlərin sayı
1	LeNet5-filters-6-16	6	16	120–84–28	63236
2	LeNet5-filters-6-8	6	8	120–84–28	38028
3	LeNet5-filters-6-4	6	4	120–84–28	25424

BNŞ-lərdə pulinq laylarında orta və maks pulinq seçimlərinin hər ikisi sınanmışdır.

Modellər öyrətmə bazasında və öyrətmə bazasının həcmini 2, 3, 5 dəfə aldığımız bazalarda 5 ayrı başlanğıc nötqəsində sınanmışdır.

Fərqli xarakteristikalı modellərin sayı ümumilikdə 40-dır:

- SNŞ: {model strukturlarının sayı}x{öyrətmə bazasının sayı} = 4 x 4 = 16;
- BNŞ: {model strukturlarının sayı}x{öyrətmə bazasının sayı} x{pulinq seçimlərinin sayı} = 3 x 4 x 2 = 24.

Yanaşma 2. Həm klasterləmə, həm də sinifləşdirmədə əlamət kimi “Pixels”, “PDC” və Yanaşma 1-də ən yaxşı nəticə göstərən LeNet-5 arxitekturalı, öyrədilmiş BNŞ-nin çıxışa qədərki son çıxışı, daha dəqiq desək, son bükülmə layının çıxışından alınan 120 elementli vektor istifadə olunur ki, bu əlaməti II fəsildə “Conv” adlandırmışdıq. Deməli, Yanaşma 2-də sinifləşdirmə modeli kimi yalnız SNŞ istifadə edilmişdir.

Klaster sayı 2, 3, ..., 50 götürülərək yoxlanılmışdır. Yanaşma 1-də auqmentasiya – öyrətmə bazasındakı verilənlərin artırılması nəticələrə ciddi təsir göstərmədiyindən Yanaşma 2-də yalnız orjinal baza istifadə olunmuşdur. O cümlədən yanaşma 1-də fərqli başlanğıc nöqtələri seçmənin bir əhəmiyyəti aşkar olunmadığından nəticələrin alınmasını və analizini sürətləndirmək üçün yalnız bir başlanğıc nöqtədə eksperimentlər aparılmışdır.

Verilmiş təsvirin klasterinin təyin olunması üçün öyrətmə bazası üzərində klasterə ayırma aparılır. Verilmiş təsvirin hansı mövcud klasterə yaxın olması əvvəlcə K-ortalar üsulundakı mərkəzlərə nəzərən aparılmışdır: təsvirin klasteri kimi əlamətinə ən yaxın mərkəzin aid olduğu klaster təyin edilir. Lakin bu üsulla test bazasındakı təsvirlərin əksəriyyəti elə həmin təsvirin sinfinə aid müşahidələr olmayan klasterlərə düşmüşdür, hansı ki, bu halda təsvirinin doğru sinfinin təyin edilməsi mümkün deyil. Ona görə də,

öyrətmə bazası klasterlərə ayrıldıqdan sonra, öyrətmə bazasındakı hər təsvir və müvafiq təsvirin hansı klasterə düşdüyünü bildirən qiymətdən ibarət cütlər hazırlanır, ayrıca bir SNŞ-yə müəllimlə öyrənmə vasitəsi ilə öyrədilir. Beləliklə, sinfinin tanınması tələb olunan təsvirin əlaməti bu xüsusi təyinatı olan SNŞ-yə verilir və hansı klasterə aid sayılmalı olduğu belə təyin olunur. Ardınca verilmiş təsvirin əlaməti təyin olunan klasterə düşmüş müşahidələrlə öyrədilmiş sinifləşdirmə modelinə verilə və təsvirin sinfi təyin oluna bilər.

Yanaşma 3. Bu yanaşmada Yanaşma 2-dəki əlamətlər istifadə olunmuşdur.

Klaster sayı 2, 3, ..., 27 götürülərək yoxlanılmışdır. Yanaşma 2-də verilmiş səbəblərə görə yalnız orjinal baza işlədilmişdir, eksperimentlər bir başlanğıc nöqtədə aparılmışdır.

Yanaşma 2-də olduğu kimi Yanaşma 3-də də verilmiş təsvirin hansı klasterə təyin olunması üçün K-ortalar üsuluyla təyin olunmuş klaster mərkəzləri ilə məsafənin istifadəsi yaxşı nəticə verməmişdir. Klasterləmə üçün Yanaşma 2-də olduğu kimi bir SNŞ qurulmuşdur.

Yanaşma 4. Bu yanaşmada da “Pixels” və “PDC” əlaməti istifadə olunmuşdur. “Conv” əlamətinin SNŞ-də istifadəsi əvəzinə isə BNŞ istifadə olunmuşdur. Əlamətlərə uyğun Yanaşma 1-də yaxşı nəticə göstərən model stukturları istifadə edilib. Eksperimentlər bir başlanğıc nöqtədə aparılmışdır. II fəslin 3-cü yarımfəslində Yanaşma 4-dəki 3 mümkün hala – verilmiş təsviri modellərin sadəcə birinin tanıması, birdən çoxunun tanıması və heç birinin tanımaması, uyğun nəticələrin sayı təhlil olunmuşdur.

3.1.2. Ümumi xarakteristikaların təsviri.

Dissertasiya işində qurulmuş modellərdə istifadə olunan aktivasiya funksiyalarının təsviri. Bütün yanaşmalardakı sinifləşdirmə modellərində axırıncı lay istisna olmaqla aktivasiya funksiyası kimi ReLU, axırıncı layda isə “One-Hot Encoding” formatına uyğun gələn Softmaks istifadə olunmuşdur.

Dissertasiya işində qurulmuş modellərin parametrlərinin seçimi üçün optimallaşdırmanın istifadəsinin bəzi detallarının təsviri. İstifadə olunan məqsəd funksiyası sinifləşdirmə modellərində çox rast gəlinən “Cross-entropy”-dir. Optimallaşdırma üsulu kimi Adam üsulu istifadə olunmuşdur. Adam üsulu minimallaşdırma istiqamətində dəyişməni nəzərə alaraq çoxekstremallı funksiyaların minimallaşdırılması üçün, xüsusilə, surətlərin tanınması məsələlərində geniş istifadə olunur.

3.1.3. Öyrətmə bazasının təsviri.

Bu bənddə əvvəlcə işdə istifadə olunan öyrətmə bazasının təsviri verilir, sonra verilənlərin sayının artırılması üçün istifadə olunan auqmentasiya təsvir olunur.

Dissertasiya işində istifadə olunan öyrətmə bazasının yığılmasının təsviri. Öyrətmə bazasındakı şəkillər ağ-qara (ing. binary) formatda saxlanılmışdır. Bu nümunələr E.E.Mustafayevin dissertanturada tələbələriylə doldurduğu formalardan yığılıb, hər hərf müəllifinin yazdığı ölçüsüylə çıxarılıb, ayrı-ayrı şəkillər halına salınıb (şəkil 3.1.1). Eksperimentlərin aparılması üçün

Şəkil 3.1.1. Çap əlyazma hərf təsvirləri nümunələri.

bazadakı şəkillər daxil edilərkən hər şəkil 20×20 ölçüyə gətirilib, beləliklə, ölçüsü dəyişdirilmiş – genişləndirilmiş, ya da daraldılmışdır, nəticədə şəkillərin rəng formatı boz çalarlı (ing. gray scale) formata gətirilmişdir. [6-9,11-14]

Baza öyrətmə və test üçün istifadə olunan iki fərqli alt bazadan ibarətdir. Öyrətmə bazasında hər hərfdən 500 müşahidə olmaqla, ümumilikdə 14 min nümunə var. Test bazasında isə hər hərfdən 100 müşahidə olmaqla, ümumilikdə 2 min 800 nümunə mövcuddur.

Dissertasiya işində verilənlərin artırılmasından (auqmentasiya) istifadənin təsviri. Çap əlyazma hərflərin tanınması üçün öyrətmə və test bazalarındakı şəkillər çoxölçülü əlamət fəzasında təyin olunur və ayrı-ayrı hərflərin şəkilləri bu əlamət fəzasında bir-birilərinə bir hərfə aid şəkillərin olduğundan daha yaxın ola bilərlər. Bu da verilmiş təsvirin əlaməti ilə çıxış qiyməti arasında əlaqənin ümumiləşməsi üçün daha mürəkkəb strukturlu model tələb edir. Başqa sözlə, təklif olunan modellərin öyrətmə bazasındakı müşahidələrin sayından daha çox öyrədilməli parametri olduğundan öyrətmə bazasının həcmini artırmaq lazım gəlir. Bu yanaşma tədqiqatlarda geniş istifadə olunur, hətta məlum BNŞ arxitekturalarının bəzilərində milyonlarla öyrədilməli parametr olur. Məsələn, 2012-ci ildə ILSVRC şəkillərin sinifləşdirilməsi yarışında birinci yerə layiq görülmüş AlexNet şəbəkəsinin 60 milyona yaxın parametri var, lakin uyğun öyrətmə bazasındakı cəmi şəkil sayı 1.2 milyon ətrafında idi.

Şəkillərdə öyrətmə bazasındakı müşahidələrin mövcud müşahidələr əsasında artırılması inglisdilli ədəbiyyatlarda “auqmenteyşn” – auqmentasiya adlanır. Auqmentasiya üçün fərqli üsullar mövcuddur. Bu üsulları ağ qutu və qara qutu adlanmaqla şərti olaraq iki hissəyə ayırırlar.[42] Qara qutu yanaşmaları öyrətmə bazası əsasında BNŞ-nin köməyi ilə adaptiv olaraq təyin olunur. Ağ qutu yanaşmasında isə auqmentasiya üsulları universaldır. Dissertasiya işində öyrətmə bazasının auqmentasiyası ağ qutu auqmentasiyasının aşağıdakı yanaşmalarını kombinə etməklə aparılmışdır:

- şəkildəki surətin təsvirinin fırladılması (ing. rotation range) - $[-10°, 10°]$ parçasında ixtiyari bir dərəcə ilə,

- şəkildəki surətin təsvirinin dartılması - $[-20\%, 20\%]$ parçasında ixtiyari bir dərəcəyə,

Cədvəl 3.2.1. Birinci yanaşmanın ədədi nəticələri.

N	Model	Bazanın həcmi	Parametrlərin sayı	Öyrətmə bazasını tanıma dəqiqliyi	Test bazasını tanıma dəqiqliyi
1	LeNet5-filters-6-16	14000	63236	99,64%	89,32%
2	LeNet5-filters-6-16	70000	63236	99,58%	89,14%
3	LeNet5-filters-6-16	42000	63236	99,23%	88,82%
4	LeNet5-filters-6-8	70000	38028	99,26%	88,75%
5	LeNet5-filters-6-8	14000	38028	99,57%	88,57%
6	LeNet5-filters-6-8	28000	38028	99,38%	88,43%
7	PDC-256-50-28	14000	14278	99,50%	88,21%
8	LeNet5-filters-6-4	70000	25424	99,02%	88,18%
9	LeNet5-filters-6-16	28000	63236	99,49%	88,18%
10	LeNet5-filters-6-4	28000	25424	99,21%	88,07%
11	LeNet5-filters-6-4	42000	25424	99,22%	87,96%
12	LeNet5-filters-6-8	42000	38028	99,53%	87,96%
13	PDC-256-50-28	70000	14278	97,79%	87,86%
14	PDC-256-50-50-28	42000	16828	98,58%	87,54%
15	PDC-256-50-50-28	70000	16828	97,97%	87,50%
16	LeNet5-filters-6-4	14000	25424	99,31%	87,43%
17	PDC-256-50-28	42000	14278	98,69%	87,32%
18	PDC-256-50-50-28	14000	16828	99,60%	87,32%
19	PDC-256-50-28	28000	14278	98,53%	87,29%
20	PDC-256-50-50-28	28000	16828	98,74%	87,29%
21	Pixels-400-50-50-28	70000	24028	98,30%	84,04%
22	Pixels-400-50-28	70000	21478	98,76%	83,96%
23	Pixels-400-50-50-28	42000	24028	98,91%	83,36%
24	Pixels-400-50-50-28	14000	24028	99,98%	83,25%
25	Pixels-400-50-28	42000	21478	99,20%	83,04%
26	Pixels-400-50-50-28	28000	24028	99,27%	82,43%
27	Pixels-400-50-28	14000	21478	99,94%	82,04%
28	Pixels-400-50-28	28000	21478	99,49%	81,46%

- şəkildəki surətin təsvirinin böyüdülməsi və ya kiçildilməsi (ing. zoom range) - $[-20\%, 20\%]$ parçasında ixtiyari bir nisbətlə,

burada müsbət və mənfi dərəcələr, uyğun olaraq, saat əqrəbi və saat əqrəbinin əksi istiqamətini bildirir.

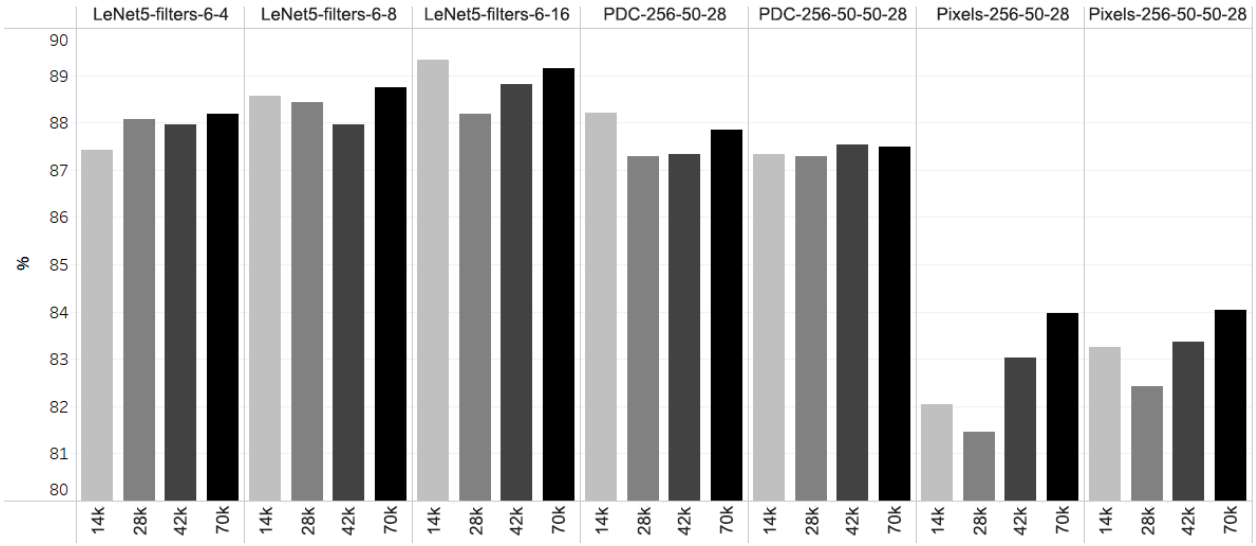
3.2. Kompüter eksperimentlərinin nəticələri və analizi

3.2.1. Birinci yanaşmanın realizasiyası zamanı alınmış ədədi nəticələr.

Test bazasındakı təsvirlərin sinfinin tanınması dəqiqlikləri cədvəl 3.2.1-də verilmişdir.

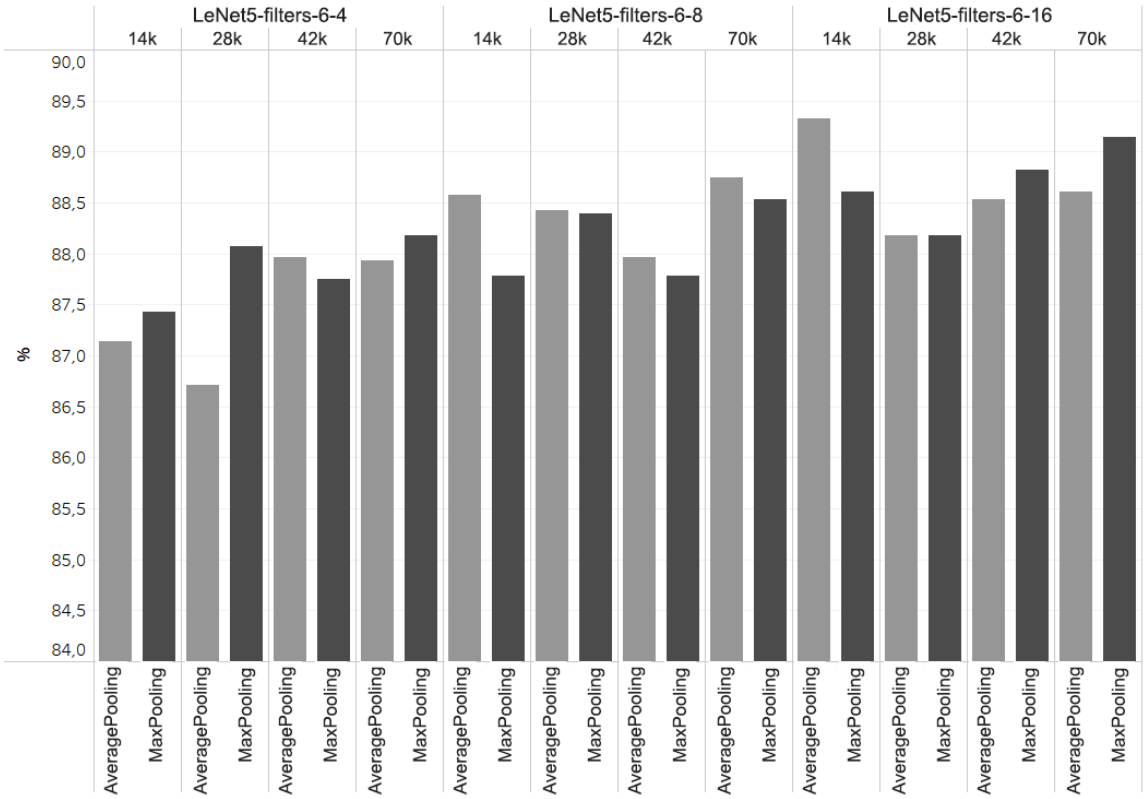
Cədvəl 3.2.1-də 5 başlanğıc nöqtə və BNŞ-lər üçün pulinq seçimləri arasından test bazasında ən yaxşı nəticə göstərənlər seçilib qeyd olunmuşdur.

Şəkil 3.2.2-də birinci yanaşmadakı modellərin test bazasındakı dəqiqlik nəticələri verilmişdir:



Şəkil 3.2.1. Birinci yanaşmanın qrafik şəklində nəticələri.

Şəkil 3.2.2 isə BNŞ-lərdə pulinq seçimlərinin tanımanın effektivliyinə təsirlərinin müqayisəsinə həsr olunmuşdur:



Şəkil 3.2.2. Birinci yanaşmada BNŞ-lərdə fərqli pulinq seçimlərinin istifadəsinin təhlili üçün nəticələr.

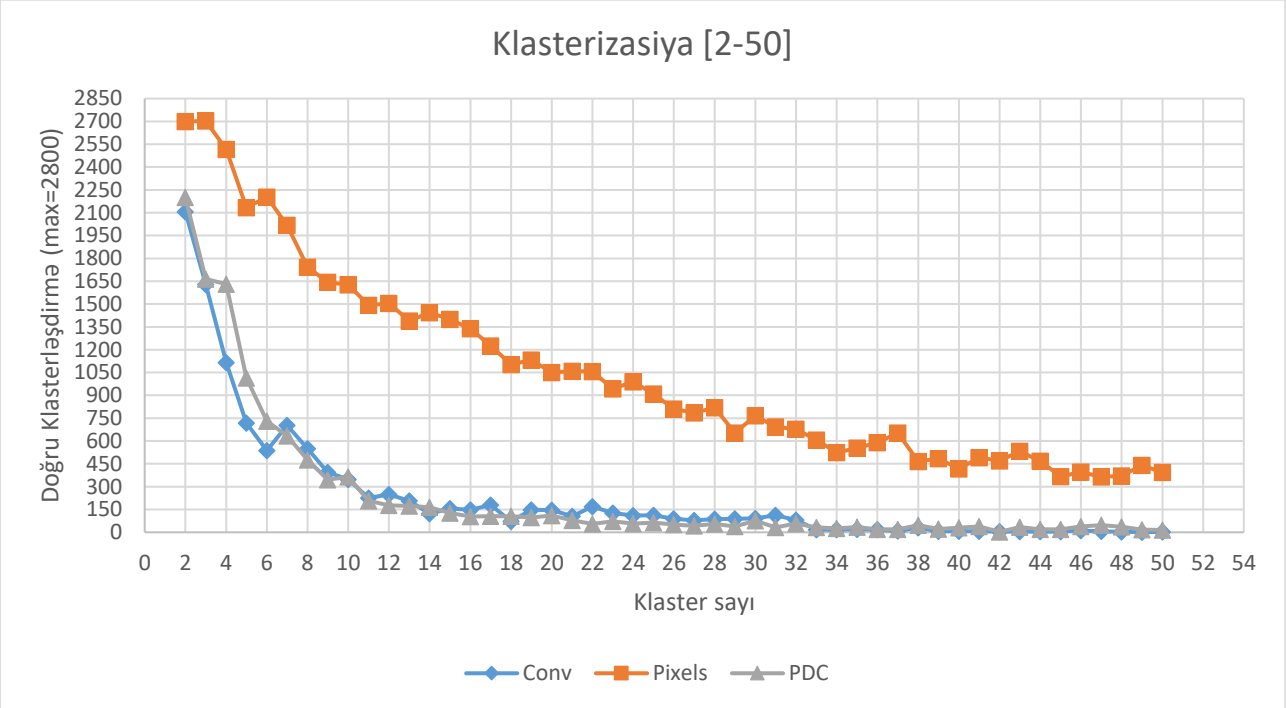
Birinci yanaşma üzərində aparılmış ədədi eksperimentdən alınan nəticələrin analizindən çıxan mülahizələr aşağıdakılardır:

- İlk və ikinci laylarda uyğun olaraq 6, 16 filteri olan BNŞ digər strukturlu BNŞ-lərdən daha yaxşı nəticə göstərmişdir.
- Əlamət çıxarılışı üçün bükülmə əməliyyatı və ya “PDC” çıxarılış alqoritminin istifadə edilməsindən asılı olmayaraq “Pixels” əlaməti - təsvirin piksellərinin qiymətləri digər iki əlamətlə müqayisədə zəif nəticə göstərmişdir.
- Öz əlamətini özü təyin edən və əlamət çıxarılışını sadəcə ilk mərhələdə yox, sonrakı mərhələlərdə də əvvəlki mərhələlərin çıxışı üzərində həyata keçirən BNŞ-lər, adətən, “PDC” və “Pixels” əlamətlərinin qiymətləri ilə işləyən SNŞ-lərdən daha yaxşı nəticə göstərmişdir.

- “Pixels” əlamətinin qiymətləri ilə işləyən SNŞ-ləri çıxmaqla digər modellər üçün auqmentasiyanın modellərin təsvirlərin siniflərini tanıması dəqiqliyinin artımına ciddi təsiri aşkarlanmamışdır.
- Şəkil 17-dən başa düşüldüyü kimi pulinq seçimlərindən hər hansı birinin digəri üzərində bütün hallarda üstünlüyünə dair bir aşkar bir münasibət yoxdur.

3.2.2. İkinci yanaşmanın realizasiyası zamanı alınmış ədədi nəticələr.

Klasterləmənin nəticələri. Verilən təsvirin klasterinin təyin olunması üçün K-ortalar üsulundakı mərkəzlər istifadə olunan halda test bazasındakı təsvirlərin neçəsinin siniflərinin tanına biləcəyi klasterə düşmə sayı şəkil 3.2.3-də verilmişdir.



Şəkil 3.2.3. İkinci yanaşmada verilmiş surətin klasterləşdirilməsi K-ortalar üsulu ilə aparıldıqda test müşahidələrinin tanına biləcəyi klasterə düşmə tezlikləri.

Verilən təsvirin klasterinin təyin olunması üçün öyrətmə bazasındakı təsvirlər və onların klaster məlumatları əsasında öyrədilən SNŞ istifadə olunan halda test bazasındakı təsvirlərin neçəsinin siniflərinin tanına biləcəyi klasterə düşmə sayı şəkil 3.2.4-də verilmişdir.

Sinifləşdirmənin nəticələri. “Pixels”, “PDC” və “Conv” əlamətindən istifadə edən modellərin test bazasındakı təsvirləri tanıması dəqiqlikləri şəkil 3.2.5-də təsvir olunub.

3.2.3. Üçüncü yanaşmanın realizasiyası zamanı alınmış ədədi nəticələr.

Klasterləmənin nəticələri. Verilən təsvirin klasterinin təyin olunması üçün K-ortalar üsulundakı mərkəzlər istifadə olunan halda test bazasındakı təsvirlərin neçəsinin siniflərinin tanına biləcəyi klasterə düşmə sayı şəkil 3.2.6-da verilmişdir.

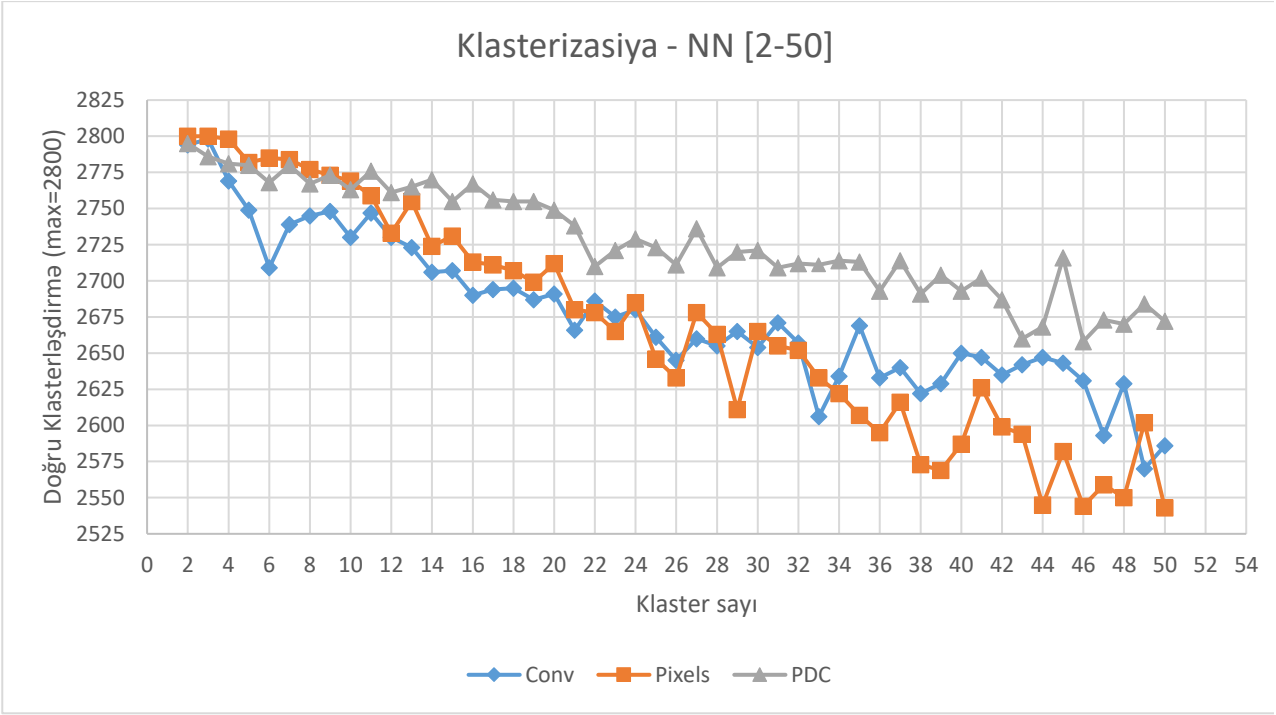
Verilən təsvirin klasterinin təyin olunması üçün öyrətmə bazasındakı təsvirlər və onların klaster məlumatları əsasında öyrədilən SNŞ istifadə olunan halda test bazasındakı təsvirlərin neçəsinin siniflərinin tanına biləcəyi klasterə düşmə sayı şəkil 3.2.7-də verilmişdir.

Sinifləşdirmənin nəticələri. “Pixels”, “PDC” və “Conv” əlamətindən istifadə edən modellərin test bazasındakı təsvirləri tanıması dəqiqlikləri şəkil 3.2.8-də təsvir olunub.

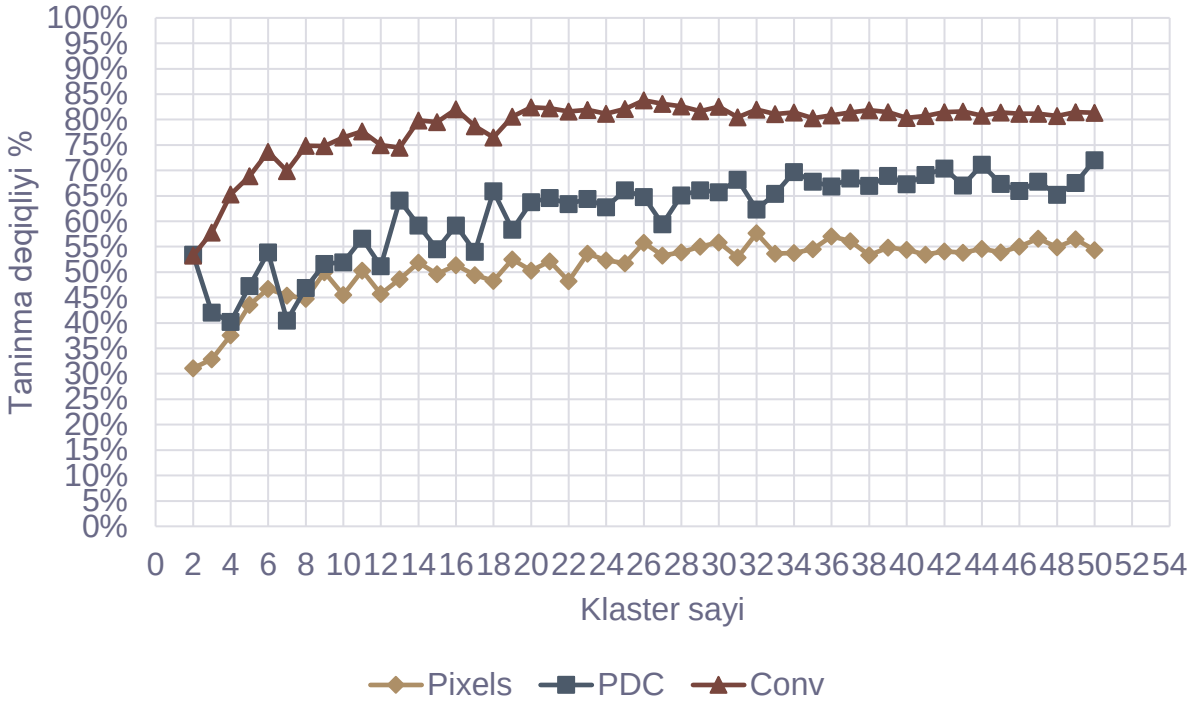
3.2.4. Dördüncü yanaşmanın realizasiyası zamanı alınmış ədədi nəticələr.

Test bazasındakı təsvirlərdən müxtəlif əlamətlərə görə birdən çox modelin tanıdığı və ya heç bir modelin tanımadığı müşahidələrin sayları aşağıdakı cədvəldə verilmişdir:

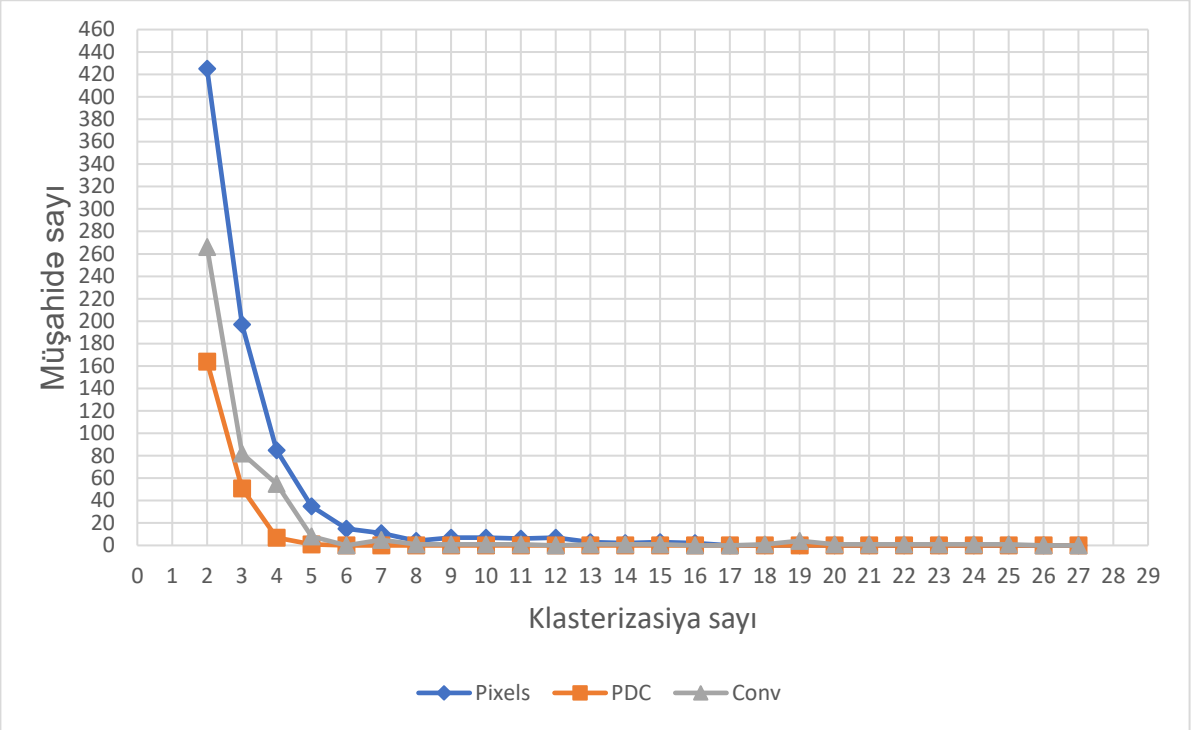
Test bazasındakı təsvirlərin neçə faizinin doğru tanınması və Yanaşma 4-lə tanına bilən test bazasındakı təsvirlərin neçə faizinin doğru tanınması aşağıda, uyğun olaraq, “Ümumi” və “Xüsusi” sütunlarında verilmişdir.



Şəkil 3.2.4. İkinci yanaşmada verilmiş surətin klasterləşdirilməsi SNŞ ilə aparıldıqda test müşahidələrinin tanına biləcəyi klasterə düşmə tezlikləri.



Şəkil 3.2.5. İkinci yanaşmada test müşahidələrinin tanına dəqiqlikləri.



Şəkil 3.2.6. Üçüncü yanaşmada verilmiş surətin klasterləşdirilməsi K-ortalar üsulu ilə aparıldıqda test müşahidələrinin tanına biləcəyi klasterə düşmə tezlikləri.

Cədvəl 3.2.2. Dördüncü yanaşmayla tanına bilməyən test müşahidələri sayları.

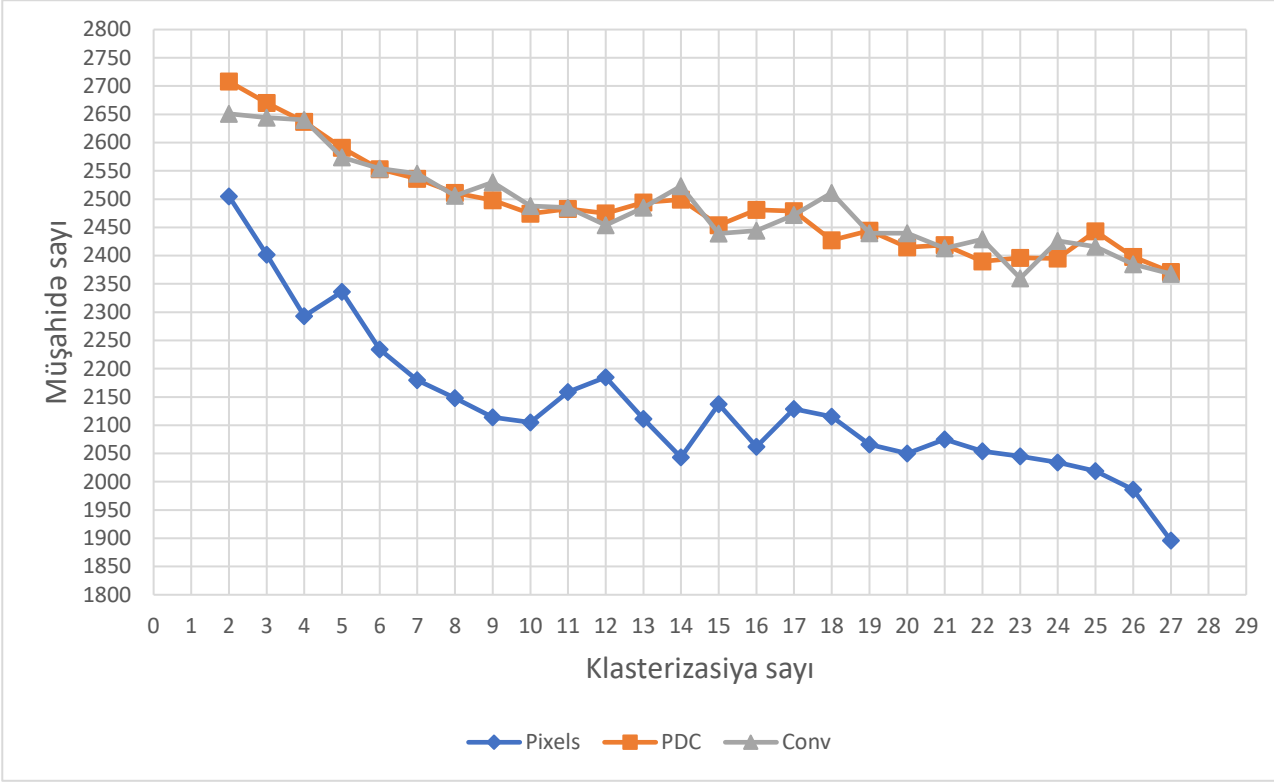
	Birdən çox modelin tanıdığı	Heç bir modelin tanımadığı
Pixels	268	383
PDC	681	99
Conv	351	155

Cədvəl 3.2.3. Dördüncü yanaşmayla tanına bilən test müşahidələri sayları.

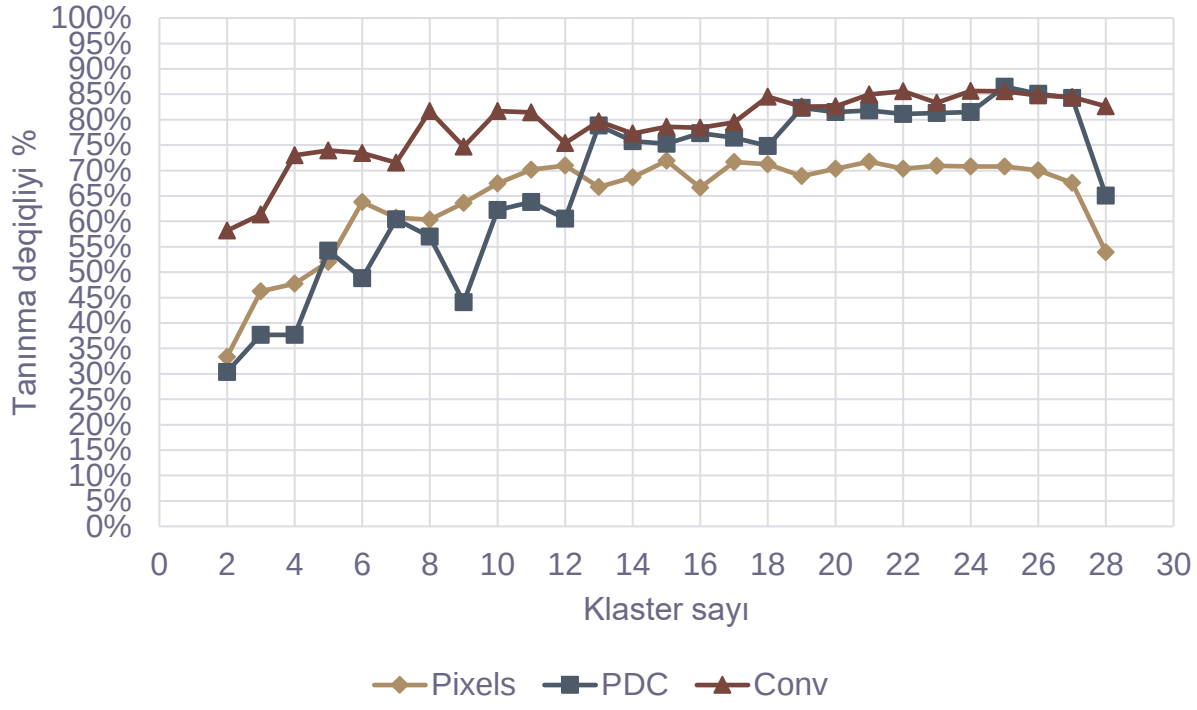
	Tanına bilənlər
Pixels	2149
PDC	2020
Conv	2294

Cədvəl 3.2.4. Dördüncü yanaşmanın test müşahidələrini tanıma dəqiqlikləri.

	Ümumi	Xüsusi
Pixels	70.03%	91.25%
PDC	67.21%	93.16%
Conv	77.39%	94.46%



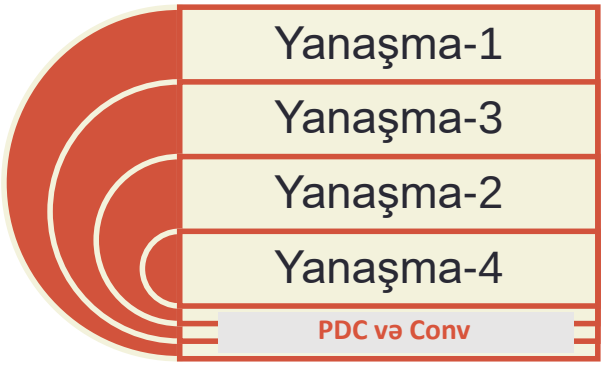
Şəkil 3.2.7. Üçüncü yanaşmada verilmiş surətin klasterləşdirilməsi SNŞ ilə aparıldıqda test müşahidələrinin tanına biləcəyi klasterə düşmə tezlikləri.



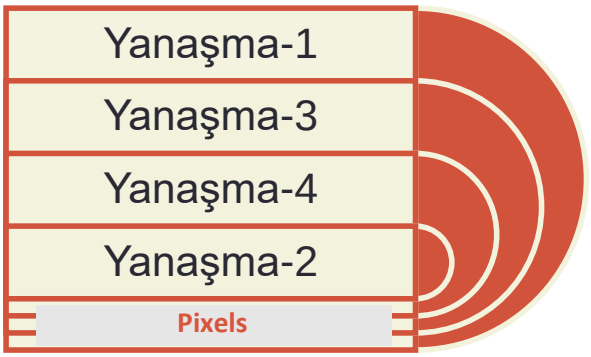
Şəkil 3.2.8. Üçüncü yanaşmada test müşahidələrinin tanına dəqiqlikləri.

3.2.5. Yanaşmaların müqayisəli təhlili.

Yanaşmaların tanıma dəqiqliyinə görə sıralaması PDC və Conv əlamətlərində eyni, Pixels əlamətində isə onlardan fərqlidir.



Şəkil 3.2.9. PDC və Conv əlamətlərinin istifadəsində yanaşmaların tanıma dəqiqliklərinin azalma sırası ilə düzülüşü.



Şəkil 3.2.10. Pixels əlamətinin istifadəsində yanaşmaların tanıma dəqiqliklərinin azalma sırası ilə düzülüşü.

Aşağidakı cədvəldə tanıma dəqiqlikləri üzrə nəticələr verilmişdir:

Cədvəl 3.2.5. Yanaşmaların fərqli əlamətlər üzrə tanıma dəqiqliyi nəticələri.

	Pixels	PDC	Conv
Yanaşma-1	84.04%	88.21%	89.32%
Yanaşma-2	57.64%	72.03%	83.79%
Yanaşma-3	71.92%	86.50%	85.64%
Yanaşma-4	70.03%	67.21%	77.39%

Aşağıdakı cədvəldə isə klasterləmə istifadə olunan yanaşmalarda – yanaşma 2 və yanaşma 3-də ən yüksək tanıma dəqiqliyinin alındığı klaster sayları qeyd olunmuşdur:

Cədvəl 3.2.6. İkinci və üçüncü yanaşmada yüksək tanıma dəqiqliyi vermiş klaster sayları.

	Pixels	PDC	Conv
Yanaşma-2	32	50	26
Yanaşma-3	15	25	24

NƏTİCƏ

Dissertasiya işində aşağıdakı nəticələr alınmışdır:

1. Azərbaycan əlifbasının çap əlyazma hərflərinin tanınmasında istifadəsi üçün fərqli sinif əlamətlərin çıxarılışı alqoritmləri və proqram təminatı işlənib hazırlanmışdır.
2. Tədqiq olunan əlamət qruplarının tanımada effektivliyi tanımaya dörd fərqli yanaşmada tədqiq olunmuşdur.
3. Öyrədilmiş Bükülmə Neyron Şəbəkə klasterləmədə istifadə oluna bilən əlamətlərin çıxarılışı mexanizmi kimi istifadə olunmuşdur.
4. Baxılan tanıma yanaşmalarının kompüter eksperimentlərinin nəticələri əsasında müqayisəli təhlili aparılmışdır.
5. Kompüter eksperimentlərinin aparılması üçün Python alqoritmik dilində proqram təminatı hazırlanmışdır.

ƏDƏBİYYAT

1. Əliquliyev, Rasim & Aliguliyev, Ramiz & Abdullayeva, Fargana. (2017). Bulud infrastrukturunun keyfiyyət göstəricilərində anomaliyaların real zamanda aşkarlanması metodu. 30-36. 10.25045/NCSoftEng.2017.05.

2. Əzimov R.B.. (2021). “Riyaziyyatın tətbiqi məsələləri və yeni in-formasiya texnologiyaları” adlı IV Respub-lika elmi konfrans materialları, 9-10 Dekabr 2021, № 9, səh.79-84.

3. Əzimov R.B.. (2022). Azərbaycan çapəlyazma əlifbasının tanınmasına yanaşmaların müqayisəli təhlili. “Proseslərin avtomatlaş-dırılması və informasiya təhlükəsizliyi-2022” adlı “Tələbə və Gənc Tədqi-qatçıların III Beynəlxalq Elmi Konfransları, 26-27 aprel.

4. Abdalla, Sevgi & Erdoğmuş, Şenol. (2014). Destek vektör makineleriyle sınıflandırma problemlerinin çözümü için çekirdek fonksiyonu seçimi. Eskişehir osmangazi üniversitesi iktisadi idari bilimler fakültesi dergisi. 9. 175-198.

5. Üyesi, Öğr & Metlek, Sedat & Kayaalp, Kıyas. (2021). MAKİNE ÖĞRENMESİNDE, TEORİDEN ÖRNEK MATLAB UYGULAMALARINA KADAR DESTEK VEKTÖR MAKİNELERİ.

6. Айда-заде К.Р., Мустафаев Э.Э. Интеллектуальная автоматизированная система обработки и оценивания результатов тестовых экзаменов / Прикладная математика и фундаментальная информатика, Омск, 2018, том 5, №2, стр 51-60.

7. Айда-заде К.Р., Мустафаев Э.Э. Интеллектуальная система распознавания рукопечатных форм азербайджанского языка / Труды Республиканской научной конференции «Современные проблемы информатизации, кибернетики и информационных технологий», Баку, 2006, том III, стр 85-88.

8. Айда-заде К.Р., Мустафаев Э.Э. Об одной иерархической системе распознавания рукописных форм на основе нейронных

сетей // Известия НАН Азербайджана, серия ф.т. и м.н., №2-3, 2002, с. 94-98.

9. Айда-заде К.Р., Мустафаев Э.Э., Гасанов Дж.З. Об использовании баз знаний для повышения интеллектуальности систем распознавания / Доклады 11-й Всероссийской конференции «Математические методы распознавания образов», Москва, 2003, с. 6-8.

10. Мустафаев Э.Э. Методы распознавания рукопечатных текстов. Баку, Издательство «Фуюзат», 2020, 189 стр.

11. Мустафаев Э.Э. Многоуровневая иерархическая система распознавания рукописных форм // Материалы научной конференции «Современные проблемы прикладной математики», Баку, 2002, с. 154-157.

12. Мустафаев Э.Э., Азимов Р.Б., Ахмедлы Н.А.. (2021). Сравнительный Анализ Применения Нейронных и Сверточных Нейронных Сетей Распознаванания Рукопечатных Букв Азербайджанского Алфавита. “Информационный бюллетень Омского научно-образователь-ного центра ОмГТУ и ИМ СО РАН в области математики и инфор-матики», 2021, т.5, № 1. с.94-95.

13. Мустафаев Э.Э., Азимов Р.Б.. (2022). Использование многослой-ных и сверточных нейрон-ных сетей для распознавания рукопечатных букв на примере азербайджанского алфавита. Прикладная математи-ка и фундаментальная информатика. т.8, №2. с.38-45.

14. Aida-zade K.R., Mustafayev E.E. Intelligent handwritten form recognition system based on artificial neural networks / Proceedings of the Intern. Conf. on Modeling and Simulation, 2006, 28-30 August, Konya, Turkey, pp. 609-613.

15. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks.

Commun. ACM 60, 6 (June 2017), 84–90. DOI:https://doi.org/10.1145/3065386

16. Andina, Diego & Pham, D. & Andina, D. & Vega-Corona, Antonio & Seijas, Juan & Torres-Garcìa, J.. (2007). Neural Networks Historical Review. 10.1007/0-387-37452-3_2.

17. Basheer, Imad & Hajmeer, M.N.. (2001). Artificial Neural Networks: Fundamentals, Computing, Design, and Application. Journal of microbiological methods. 43. 3-31. 10.1016/S0167-7012(00)00201-3.

18. Chapelle, O., Schölkopf, B., & Zien, A. (2006). Semi-supervised Learning. Ανακτήθηκε από https://books.google.az/books?id=kfqvQgAACAAJ

19. Cios, Krzysztof. (2018). Deep Neural Networks—A Brief History. 10.1007/978-3-319-67946-4_7.

20. Dushkoff, Michael and Raymond W. Ptucha. “Adaptive Activation Functions for Deep Networks.” Computational Imaging (2016).

21. Dustin Boswell. (2002 – 6 aug). Introduction to Support Vector Machines.

22. E.E.Mustafayev, R.B.Azimov. (2021). Comparative analysis of the application of multilayer and convolutional neural networks for recognition of handwritten letters of the Azerbaijani alphabet. Kiev. Cybernetics and Computer Technologies, No.3.

23. F. Rosenblatt, Principles of Neurodynamics, Cornell Aeronautıcal Laboratory, Inc., New York, 15 march 1961.

24. Gholamalinejad, Hossein & Khosravi, Hossein. (2020). Pooling Methods in Deep Neural Networks, a Review.

25. Ghosh, Anirudha & Sufian, A. & Sultana, Farhana & Chakrabarti, Amlan & De, Debashis. (2020). Fundamental Concepts of Convolutional Neural Network. 10.1007/978-3-030-32644-9_36.

26. Gruzdeva T.V., Ushakov A.V. (2021) K-Means Clustering via a Nonconvex Optimization Approach. In: Pardalos P., Khachay M.,

Kazakov A. (eds) Mathematical Optimization Theory and Operations Research. MOTOR 2021. Lecture Notes in Computer Science, vol 12755. Springer, Cham. https://doi.org/10.1007/978-3-030-77876-7_31

27. Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow by Aurélien Géron (O'Reilly). Copyright 2019 Aurélien Géron, 978-1-492-03264-9.

28. http://sdu-sdtk.edu.az/wp-content/uploads/2020/04/Ehtimal-n%C9%99z%C9%99riyy%C9%99si.pdf

29. https://home.work.caltech.edu/~boswell/IntroToSVM.pdf

30. https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

31. https://towardsdatascience.com/so-why-the-heck-are-they-called-support-vector-machines-52fc72c990a1.

32. https://www.geeksforgeeks.org/naive-bayes-classifiers/

33. https://www.ibm.com/cloud/learn/unsupervised-learning

34. https://www.ijert.org/research/a-review-of-optical-character-recognition-IJERTV2IS120772.pdf

35. https://www.irjet.net/archives/V7/i4/IRJET-V7I4583.pdf

36. https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html

37. https://www.ripublication.com/ijcir17/ijcirv13n2_13.pdf

38. Kaviani, Pouria & Dhotre, Sunita. (2017). Short Survey on Naive Bayes Algorithm. International Journal of Advance Research in Computer Science and Management. 04.

39. Kecman, Vojislav. (2005). Support Vector Machines – An Introduction. 10.1007/10984697_1.

40. Kiran Bhowmick , Meera Narvekar , Mohammed Aqid Khatkhatay, 2019, A Comprehensive Study and Analysis of Semi Supervised Learning Techniques, INTERNATIONAL JOURNAL OF ENGINEERING

RESEARCH & TECHNOLOGY (IJERT) Volume 08, Issue 11 (November 2019).

41. Lee, Chen-Yu & Gallagher, Patrick & Tu, Zhuowen. (2015). Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree.

42. Mikołajczyk, Agnieszka & Grochowski, Michał. (2018). Data augmentation for improving deep learning in image classification problem. 117-122. 10.1109/IIPHDW.2018.8388338.

43. Nwankpa, Chigozie, Winifred L. Ijomah, Anthony Gachagan and Stephen Marshall. “Activation Functions: Comparison of trends in Practice and Research for Deep Learning.” ArXiv abs/1811.03378 (2018): n. pag.

44. Omran, Mahamed & Engelbrecht, Andries & Salman, Ayed. (2007). An overview of clustering methods. Intell. Data Anal.. 11. 583-605. 10.3233/IDA-2007-11602.

45. Park, D. (2016). Image Classification Using Naïve Bayes Classifier.

46. R.B.Azimov. (2022). Comparison of Artificial and Convolutional Neural Networks in recognition of handwritten letters of Azerbaijani alphabet. 8thWorld Conference of Soft Computing (8thWConSC’21), february 3-5.

47. Sharma, Siddharth, S Sharma and Anidhya Athaiya. “ACTIVATION FUNCTIONS IN NEURAL NETWORKS.” (2020).

48. Tian, Yingjie & Shi, Yong & Liu, Xiaohui. (2012). Recent advances on support vector machines research. Technological and Economic Development of Economy. 18. 10.3846/20294913.2012.661205.

49. Wu, Xindong & Kumar, Vipin & Quinlan, Ross & Ghosh, Joydeep & Yang, Qiang & Motoda, Hiroshi & Mclachlan, G. & Ng, Shu Kay Angus & Liu, Bing & Yu, Philip & Zhou, Zhi-Hua & Steinbach, Michael & Hand, David & Steinberg, Dan. (2007). Top 10 algorithms in data

mining. Knowledge and Information Systems. 14. 10.1007/s10115-007-0114-2.

50. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

51. Yu, Dingjun & Wang, Hanli & Chen, Peiqiu & Wei, Zhihua. (2014). Mixed Pooling for Convolutional Neural Networks. 364-375. 10.1007/978-3-319-11740-9_34.

52. Zhang, Jiawei. “Basic Neural Units of the Brain: Neurons, Synapses and Action Potential.” arXiv: Neurons and Cognition (2019).

Əlavə

1.1. Kitabxanalardan istifadə etmək üçün import əmrləri

```
# Süni Neyron Şəbəkələrlə bağlı kitabxanaların import əmrləri
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import mnist
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow import keras
from tensorflow.random import set_seed
from tensorflow.keras import regularizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.regularizers import l2
from tensorflow.keras import layers
from tensorflow import random
import tensorflow as tf
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.layers import Dropout
from keras.preprocessing.image import ImageDataGenerator

# K-ortalar üsulu və bəzi lazımlı modullar üçün kitabxanaların import əmrləri
from sklearn.metrics import confusion_matrix
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.utils.class_weight import compute_class_weight
from sklearn.utils import class_weight
from sklearn.model_selection import train_test_split
from skimage.transform import rescale
from sklearn.cluster import KMeans
from sklearn.preprocessing import OneHotEncoder
from sklearn.datasets import load_digits

# Vizuallaşdırma və məlumatların təmizlənməsi modulları yerləşən kitabxanaların import əmrləri
from matplotlib.image import imread
import matplotlib.pyplot as plt
from math import ceil
import pandas as pd
```

```
import numpy as np
from numpy import expand_dims
from pathlib import Path
from seaborn import heatmap
import csv, os, sys, cv2
```

1.2. Təsvirlərin yüklənilməsi, redaktəsi

```
# Təsvirin ağ piksellərlə əhatələnməsi
def padding(image, shape):
    def seperate_pad_width(width):
        return (width // 2, ceil(width / 2))
    padding_axis = [seperate_pad_width(abs(image.shape[0] - shape[0])),
                    seperate_pad_width(abs(image.shape[1] - shape[1]))]
    image = np.pad(image, (padding_axis[0], padding_axis[1], (0,0)), constant_values = 0)
    return image
# Təsvirin ölçüsünün dəyişdirilməsi
def resample(image, shape_read_file, shape_with_padding):
    ratio = min(shape_read_file[0] / image.shape[0], shape_read_file[1] / image.shape[1])
    image = rescale(image, ratio, order = 1, preserve_range = True, multichannel = True)
    image = padding(image, shape_with_padding)
    return image
# Bazadakı təsvirlərin yerləşdiyi faylın oxunması
def read_file(path, shape_read_file, shape_with_padding, is_gray=False, is_gonna_be_resampled=True):
    X, Y = [], []
    file = open(path, 'r')
    file.readline()
    for line in file:
        temp = line[ :-1].split('#')
        image_height = int(temp[3])
        image_width = int(temp[4])
```

```
            if is_gray:
                image = temp[-1].split(';')
            else:
                image = [(ord(i) & 1) * 255 for i in temp[-1]]
            image = np.array(image, dtype = 'uint8')
            image.resize(image_height, image_width, 1)
            if is_gonna_be_resampled:
                image = resample(image, shape_read_file, shape_with_padding)
            label = int(temp[0])
            X.append(image)
            Y.append(label)
        image_count = len(Y)
        if not is_gonna_be_resampled:
            width = max(len(im[0]) for im in X)
            height = max(len(im) for im in X)
            for i in range(len(X)):
                X[i] = resample(X[i], (width, height, 1), (width, height, 1))
        X = np.array(X, dtype = 'uint8')
        Y = to_categorical(Y, dtype = 'uint8') # num_classes = 10,  *
        return X,Y
    # Baza faylındakı təsvirlərin oxunulması
    def read_preapare_db(path, shape_read_file, shape_with_padding):
        is_gray = False #'gray' in str(path)
        X, Y = read_file(path, shape_read_file, shape_with_padding, is_gray)
        #X = vpadding(X, shape)
        X = X.astype('float32')
        X /= 255
        return X,Y
    # Baza faylındakı təsvirlərin öz ölçüsündə oxunulması (verilənlərin artırılmasında istifadə üçün)
```

```
def read_preapare_db_original_size(path, shape_read_file=(20,20,1), shape_with_padding=(32,32,1)):
    is_gray = False #'gray' in str(path)
    X, Y = read_file(path, shape_read_file, shape_with_padding, is_gray, is_gonna_be_resampled=False)
    #X = vpadding(X, shape)
    X = X.astype('float32')
    X /= 255
    return X,Y
# Təsvirlərin PDC əlamətlərinin qiymətlərinin oxunulması
def read_prepare_db_pdc(path, shape_read_file, shape_with_padding, is_gray=False):
    X, Y = [], []
    file = open(path, 'r')
    file.readline()
    for line in file:
        temp = line[ :-1].split('#')
        image_height = int(temp[3])
        image_width = int(temp[4])
        image = temp[-1].split(';')
        try:
            image = np.array(image, dtype = 'float32')
        except ValueError as e:
            print(temp[-1])
        label = int(temp[0])
        X.append(image)
        Y.append(label)
    X = np.array(X, dtype = 'float32')
    Y = to_categorical(Y, dtype = 'uint8') # num_classes = 10,
    return X,Y
# Təsvirlər siyahısındakı bütün təsvirlərinin ölçülərinin dəyişdirilməsi (verilənlərin artırılması üçün)
def extract_spec_size(x_train_original, shape_read_file, shape_with_padding):
```

```
    x_train_new = list()
    for i in range(len(x_train_original)):
        x_train_new.append(resample(x_train_original[i], shape_read_file, shape_with_padding))
    x_train_new = np.array(x_train_new, dtype = 'float32')
    return x_train_new
```

1.3. Auqmentasiya

```
# Verilənlərin sayının artırılması
def augment(x_train_original, y_train_original, n, is_shuffled = True):
    datagen = ImageDataGenerator(featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=10,
      width_shift_range=0,
      height_shift_range=0,
      shear_range=0.2,
      zoom_range=0.2,
      fill_mode='nearest')
    it = datagen.flow(x_train_original, y_train_original, batch_size=14000, shuffle=is_shuffled, seed=0)
#x_train_32x32 #.reshape(14000, 32, 32, 1)
    x, y = x_train_original, y_train_original #it.next()
    for i in range(n-1):
        temp_x, temp_y = it.next()
        x = np.concatenate((x, temp_x), axis=0)
        y = np.concatenate((y, temp_y), axis=0)
    return x, y
```

1.4. Bəzi xarakteristikalar

Auqmentasiya olunmuş təsvirlərdən çıxarılmış PDC əlamətlərinin qiymətləri fayllarının adları (1,2,3,5 ilkin bazadakı verilənlərin sayının neçə misli qədər şəkil əldə olunduğunu göstərir)

```
aug_switch = {
    2:'datasetsforfinalresults/GENERATED_HandChars_28k_32x32_Train_PDC4428_saved.txt',
    3:'datasetsforfinalresults/GENERATED_HandChars_42k_32x32_Train_PDC4428_saved.txt',
    5:'datasetsforfinalresults/GENERATED_HandChars_70k_32x32_Train_PDC4428_saved.txt'
}
# Modellərin öyrədilməsi zamanı aparılan optimallaşdırmada dayanma şərti
EPOCHS = 200
callback = EarlyStopping(monitor='loss', min_delta=0.01, patience=5)
```

1.5. Yanaşma 1: Tanıma dəqiqliyi nəticələrinin alınması

```
for aug_size in (5, 2, 3):
    results_df = pd.DataFrame(columns=['Seed', 'Pooling2D', 'Model', 'Param-Count',
                                       'Accuracy-Train', 'Loss-Train', 'Accuracy-Test', 'Loss-Test'])
    print('AUGMENTATION STARTING...')
    x_train_original, y_train_original =
read_preapare_db_original_size(Path('datasetsforfinalresults/HandChars_500_woI_Train.txt').absolute())
    print('DATA READ!')
    x_train_original, y_train_original = augment(x_train_original, y_train_original, n=aug_size, is_shuffled =
False) # n * 14000
    print('AUGMENTATION DONE', str(x_train_original.shape[0]))
    x_train_pdc, y_train_pdc = read_prepare_db_pdc(Path(aug_switch.get(aug_size, '')).absolute(), (0, 256,
1), (0, 256, 1))
    print('PDC READ')
    y_train = y_train_original
    x_train_32x32 = extract_spec_size(x_train_original, (20, 20, 1), (32, 32, 1))
    x_train_14x14 = extract_spec_size(x_train_original, (10, 10, 1), (14, 14, 1))
    x_train_20x20 = extract_spec_size(x_train_original, (20, 20, 1), (20, 20, 1))
    x_train_20x20 = x_train_20x20.reshape(x_train_20x20.shape[0], 400)
    print('SIZES DONE!')
    for seed in range(5):
```

```
        for Pooling2D in (layers.AveragePooling2D,layers.MaxPooling2D):   #layers.MaxPooling2D,
            set_seed(seed)
            # conv 1
            keras.backend.clear_session()
            model_LeNet5 = Sequential(name = 'LeNet5')
            model_LeNet5.add(layers.Conv2D(filters=6,  kernel_size=(5,5), padding='valid', activation='relu',
input_shape=(32, 32, 1)))
            model_LeNet5.add(Pooling2D(pool_size = (2,2)))
            model_LeNet5.add(layers.Conv2D(filters=16, kernel_size=(5,5), padding='valid', activation='relu'))
            model_LeNet5.add(Pooling2D(pool_size=(2,2)))
            model_LeNet5.add(layers.Conv2D(filters=120, kernel_size=(5,5), padding='valid', activation='relu'))
            model_LeNet5.add(layers.Flatten())
            model_LeNet5.add(layers.Dense(84, activation='relu'))
            model_LeNet5.add(layers.Dense(28, activation="softmax"))
            model_LeNet5.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])
            model_LeNet5.fit(x_train_32x32, y_train, epochs = EPOCHS, shuffle = True, callbacks=[callback],
verbose=0)
            loss_train_LeNet5, acc_train_LeNet5 = model_LeNet5.evaluate(x_train_32x32, y_train, verbose=0)
            loss_test_LeNet5, acc_test_LeNet5 = model_LeNet5.evaluate(x_test_32x32, y_test, verbose=0)
            results_df.loc[len(results_df)] = [str(seed), str(Pooling2D)[47:-2], str(model_LeNet5.name),
                                               str(model_LeNet5.count_params()),
                                               acc_train_LeNet5, loss_train_LeNet5,
                                               acc_test_LeNet5, loss_test_LeNet5]
            print(results_df.loc[len(results_df)-1,:].tolist())

            # conv 2
            keras.backend.clear_session()
            model_LeNet5 = Sequential(name = 'Conv-14x14')
            model_LeNet5.add(layers.Conv2D(filters=16, kernel_size=(5,5), padding='valid', activation='relu',
input_shape=(14, 14, 1)))
```

```
        model_LeNet5.add(Pooling2D(pool_size=(2,2)))
        model_LeNet5.add(layers.Conv2D(filters=120, kernel_size=(5,5), padding='valid', activation='relu'))
        model_LeNet5.add(layers.Flatten())
        model_LeNet5.add(layers.Dense(84, activation='relu'))
        model_LeNet5.add(layers.Dense(28, activation="softmax"))
        model_LeNet5.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])
        model_LeNet5.fit(x_train_14x14, y_train, epochs = EPOCHS, shuffle = True, callbacks=[callback],
verbose=0) #callbacks=[callback],
        loss_train_LeNet5, acc_train_LeNet5 = model_LeNet5.evaluate(x_train_14x14, y_train, verbose=0)
        loss_test_LeNet5, acc_test_LeNet5 = model_LeNet5.evaluate(x_test_14x14, y_test, verbose=0)

        results_df.loc[len(results_df)] = [str(seed), str(Pooling2D)[47:-2], str(model_LeNet5.name),
                                           str(model_LeNet5.count_params()),
                                           acc_train_LeNet5, loss_train_LeNet5,
                                           acc_test_LeNet5, loss_test_LeNet5]
        print(results_df.loc[len(results_df)-1,:].tolist())

        # conv 3
        keras.backend.clear_session()
        model_LeNet5 = Sequential(name = 'LeNet5-filters_6-8')
        model_LeNet5.add(layers.Conv2D(filters=6,  kernel_size=(5,5), padding='valid', activation='relu',
input_shape=(32, 32, 1)))
        model_LeNet5.add(Pooling2D(pool_size = (2,2)))
        model_LeNet5.add(layers.Conv2D(filters=8, kernel_size=(5,5), padding='valid', activation='relu'))
        model_LeNet5.add(Pooling2D(pool_size=(2,2)))
        model_LeNet5.add(layers.Conv2D(filters=120, kernel_size=(5,5), padding='valid', activation='relu'))
        model_LeNet5.add(layers.Flatten())
        model_LeNet5.add(layers.Dense(84, activation='relu'))
        model_LeNet5.add(layers.Dense(28, activation="softmax"))
```

```
        model_LeNet5.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])

        model_LeNet5.fit(x_train_32x32, y_train, epochs = EPOCHS, shuffle = True, callbacks=[callback], verbose=0) #callbacks=[callback],

        loss_train_LeNet5, acc_train_LeNet5 = model_LeNet5.evaluate(x_train_32x32, y_train, verbose=0)

        loss_test_LeNet5, acc_test_LeNet5 = model_LeNet5.evaluate(x_test_32x32, y_test, verbose=0)

        results_df.loc[len(results_df)] = [str(seed), str(Pooling2D)[47:-2], str(model_LeNet5.name),
                                           str(model_LeNet5.count_params()),
                                           acc_train_LeNet5, loss_train_LeNet5,
                                           acc_test_LeNet5, loss_test_LeNet5]

        print(results_df.loc[len(results_df)-1,:].tolist())


        # conv 4

        keras.backend.clear_session()

        model_LeNet5 = Sequential(name = 'LeNet5-filters_6-4')

        model_LeNet5.add(layers.Conv2D(filters=6,  kernel_size=(5,5), padding='valid', activation='relu', input_shape=(32, 32, 1)))

        model_LeNet5.add(Pooling2D(pool_size = (2,2)))

        model_LeNet5.add(layers.Conv2D(filters=4, kernel_size=(5,5), padding='valid', activation='relu'))

        model_LeNet5.add(Pooling2D(pool_size=(2,2)))

        model_LeNet5.add(layers.Conv2D(filters=120, kernel_size=(5,5), padding='valid', activation='relu'))

        model_LeNet5.add(layers.Flatten())

        model_LeNet5.add(layers.Dense(84, activation='relu'))

        model_LeNet5.add(layers.Dense(28, activation="softmax"))

        model_LeNet5.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])

        model_LeNet5.fit(x_train_32x32, y_train, epochs = EPOCHS, shuffle = True, callbacks=[callback], verbose=0) #callbacks=[callback],

        loss_train_LeNet5, acc_train_LeNet5 = model_LeNet5.evaluate(x_train_32x32, y_train, verbose=0)

        loss_test_LeNet5, acc_test_LeNet5 = model_LeNet5.evaluate(x_test_32x32, y_test, verbose=0)

        results_df.loc[len(results_df)] = [str(seed), str(Pooling2D)[47:-2], str(model_LeNet5.name),
                                           str(model_LeNet5.count_params()),
```

```
                                    acc_train_LeNet5, loss_train_LeNet5,
                                    acc_test_LeNet5, loss_test_LeNet5]
        print(results_df.loc[len(results_df)-1,:].tolist())
    # pixels 1
    keras.backend.clear_session()
    model_pixels_50_50 = Sequential(name = 'Pixels-400-50-50-28')
    model_pixels_50_50.add(layers.Dense(50, input_shape = (400,), activation='relu'))
    model_pixels_50_50.add(layers.Dense(50, activation='relu'))
    model_pixels_50_50.add(layers.Dense(28, activation='softmax'))
    model_pixels_50_50.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics =
['accuracy'])
    model_pixels_50_50.fit(x_train_20x20, y_train, epochs = EPOCHS, shuffle = True, callbacks=[callback],
verbose=0) #callbacks=[callback],
    loss_train_pixels_50_50, acc_train_pixels_50_50 = model_pixels_50_50.evaluate(x_train_20x20,
y_train, verbose=0)
    loss_test_pixels_50_50, acc_test_pixels_50_50 = model_pixels_50_50.evaluate(x_test_20x20, y_test,
verbose=0)
    results_df.loc[len(results_df)] = [str(seed), str('-'), str(model_pixels_50_50.name),
                                    str(model_pixels_50_50.count_params()),
                                    acc_train_pixels_50_50, loss_train_pixels_50_50,
                                    acc_test_pixels_50_50, loss_test_pixels_50_50]
    print(results_df.loc[len(results_df)-1,:].tolist())

    #pixels 2
    keras.backend.clear_session()
    model_pixels_50 = Sequential(name = 'Pixels-400-50-28')
    model_pixels_50.add(layers.Dense(50, input_shape = (400,), activation='relu'))
    model_pixels_50.add(layers.Dense(28, activation='softmax'))
    model_pixels_50.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])
```

```
model_pixels_50.fit(x_train_20x20, y_train, epochs = EPOCHS, shuffle = True, callbacks=[callback], verbose=0) #callbacks=[callback],

loss_train_pixels_50_50, acc_train_pixels_50_50 = model_pixels_50.evaluate(x_train_20x20, y_train, verbose=0)

loss_test_pixels_50_50, acc_test_pixels_50_50 = model_pixels_50.evaluate(x_test_20x20, y_test, verbose=0)

results_df.loc[len(results_df)] = [str(seed), str('-'), str(model_pixels_50.name),

                                   str(model_pixels_50.count_params()),

                                   acc_train_pixels_50_50, loss_train_pixels_50_50,

                                   acc_test_pixels_50_50, loss_test_pixels_50_50]

print(results_df.loc[len(results_df)-1,:].tolist())


#pdc 1

keras.backend.clear_session()

model_pdc_50_50 = Sequential(name = 'PDC-256-50-50-28')

model_pdc_50_50.add(layers.Dense(50, input_shape = (256,), activation='relu'))

model_pdc_50_50.add(layers.Dense(50, activation='relu'))

model_pdc_50_50.add(layers.Dense(28, activation='softmax'))

model_pdc_50_50.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])

model_pdc_50_50.fit(x_train_pdc, y_train_pdc, epochs = EPOCHS, shuffle = True, callbacks=[callback], verbose=0) #callbacks=[callback],

loss_train_pdc_50_50, acc_train_pdc_50_50 = model_pdc_50_50.evaluate(x_train_pdc, y_train_pdc, verbose=0)

loss_test_pdc_50_50, acc_test_pdc_50_50 = model_pdc_50_50.evaluate(x_test_pdc, y_test_pdc, verbose=0)

results_df.loc[len(results_df)] = [str(seed), str('-'), str(model_pdc_50_50.name),

                                   str(model_pdc_50_50.count_params()),

                                   acc_train_pdc_50_50, loss_train_pdc_50_50,

                                   acc_test_pdc_50_50, loss_test_pdc_50_50]

print(results_df.loc[len(results_df)-1,:].tolist())
```

```
#pdc 2

keras.backend.clear_session()

model_pdc_50 = Sequential(name = 'PDC-256-50-28')

model_pdc_50.add(layers.Dense(50, input_shape = (256,), activation='relu'))

model_pdc_50.add(layers.Dense(28, activation='softmax'))

model_pdc_50.compile(optimizer = 'Adam' , loss = ''categorical_crossentropy'', metrics = ['accuracy'])

model_pdc_50.fit(x_train_pdc, y_train_pdc, epochs = EPOCHS, shuffle = True, callbacks=[callback], verbose=0) #callbacks=[callback],

loss_train_pdc_50, acc_train_pdc_50 = model_pdc_50.evaluate(x_train_pdc, y_train_pdc, verbose=0)

loss_test_pdc_50, acc_test_pdc_50 = model_pdc_50.evaluate(x_test_pdc, y_test_pdc, verbose=0)

results_df.loc[len(results_df)] = [str(seed), str('-'), str(model_pdc_50.name),

                                   str(model_pdc_50.count_params()),

                                   acc_train_pdc_50, loss_train_pdc_50,

                                   acc_test_pdc_50, loss_test_pdc_50]

print(results_df.loc[len(results_df)-1,:].tolist())

results_df.to_csv(r'Results_train-' + str(aug_size * 14) +'k_all-conv-pixels-pdc-added.csv', index = False)
```

1.6. Yanaşma 2 və 3-dəki Conv əlaməti çıxarılışı üçün modelin öyrədilməsi

```
Pooling2D = layers.AveragePooling2D

keras.backend.clear_session()

model_LeNet5 = Sequential(name = 'LeNet5')

model_LeNet5.add(layers.Conv2D(filters=6,  kernel_size=(5,5), padding='valid', activation='relu', input_shape=(32, 32, 1)))

model_LeNet5.add(Pooling2D(pool_size = (2,2)))

model_LeNet5.add(layers.Conv2D(filters=16, kernel_size=(5,5), padding='valid', activation='relu'))

model_LeNet5.add(Pooling2D(pool_size=(2,2)))

model_LeNet5.add(layers.Conv2D(filters=120, kernel_size=(5,5), padding='valid', activation='relu'))

model_LeNet5.add(layers.Flatten(name="last_conv"))
```

```
model_LeNet5.add(layers.Dense(84, activation='relu'))

model_LeNet5.add(layers.Dense(28, activation="softmax"))

x_train_32x32 = x_train_32x32.reshape(-1, 32, 32, 1)

x_test_32x32 = x_test_32x32.reshape(-1, 32, 32, 1)

model_LeNet5.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])

model_LeNet5.fit(x_train_32x32, y_train, epochs = EPOCHS, shuffle = True, callbacks=[callback],
verbose=0) #callbacks=[callback],

loss_train_LeNet5, acc_train_LeNet5 = model_LeNet5.evaluate(x_train_32x32, y_train, verbose=0)

loss_test_LeNet5, acc_test_LeNet5 = model_LeNet5.evaluate(x_test_32x32, y_test, verbose=0)

results_df.loc[len(results_df)] = [str(seed), str(Pooling2D)[47:-2], str(model_LeNet5.name),

                                   str(model_LeNet5.count_params()),

                                   acc_train_LeNet5, loss_train_LeNet5,

                                   acc_test_LeNet5, loss_test_LeNet5]

print(results_df.loc[len(results_df)-1,:].tolist())

initial_model = model_LeNet5

feature_extractor = keras.Model(

    inputs=initial_model.inputs,

    outputs=initial_model.get_layer(name="last_conv").output,

)
```

2. Yanaşma 2

2.1. Köməkçi metodlar

```
# Verilmiş klaster üzrə ixtisaslaşan modelin istifadə etdiyi öyrətmə bazasından klasterə düşməyən siniflərin
qiymətlərinin çıxarılması (hər biri 0-dır)

def drop_not_given_classes(Y, is_shape_printed = True):

    non_empty_indexes = find_non_empty_indexes(Y)

    new_Ys = []

    for non_empy_i in non_empty_indexes:

        y_ints = np.argmax(Y, axis = 1)

        temp_list = y_ints[y_ints == non_empy_i].copy()
```

```
        for i in range(temp_list.shape[0]):
            new_Ys.append(temp_list[i])
    new_Ys = np.array(new_Ys)
    onehotencoder = OneHotEncoder()
    new_Ys = onehotencoder.fit_transform(new_Ys.reshape(-1,1)).toarray()
    if is_shape_printed:
        print(new_Ys.shape)
    return new_Ys
# Klasterə düşən siniflərin indekslərinin tapılması
def find_non_empty_indexes(Y):
    y_ints = np.argmax(Y, axis = 1)
    return [i for i in range(28) if len(y_ints[y_ints == i]) != 0]
# Klasterin modelinin proqnozlaşdırdığı sinif əsasında ilkin sinfinin indeksinin tapılması
def find_existing_class_indices_in_clusters(clusters_Y_list):
    existing_class_indices_in_clusters_list = []
    for i in range(clusters_Y_list.shape[0]):
        existing_class_indices_in_clusters_list.append(find_non_empty_indexes(clusters_Y_list[i]))
    return existing_class_indices_in_clusters_list
```

2.2. Klaster təyini k ortalar ilə aparıldıqda doğru klasterə düşmə tezliklərinin çıxarılışı

```
# Pixels əlaməti üçün (digər 2 əlamət analoji realizasiya olunur)
recognition_results_df = pd.DataFrame(columns=['Cl.Count', 'Correct Cl. #'])
print(['Cl.Count', 'Correct Cl. #'])
for cluster_count in range(2, 51):#range(2,20) #2, 19, 29, 45, 47
    x_train_32x32 = x_train_32x32.reshape(x_train_32x32.shape[0], -1)
    x_train_32x32_feat = x_train_32x32
    x_train_32x32_kmeans = np.copy(x_train_32x32_feat)
    kmeans = KMeans(n_clusters = cluster_count, random_state=42)
    kmeans.fit_transform(x_train_32x32_kmeans)
```

```
    clusters_X_list = []
    clusters_Y_list = []
    for i in range(cluster_count):
        clusters_X_list.append(x_train_32x32_feat[kmeans.labels_ == i])
        clusters_Y_list.append(y_train[kmeans.labels_ == i])
    clusters_X_list = np.array(clusters_X_list)
    clusters_Y_list = np.array(clusters_Y_list)
    existing_class_indices_in_clusters = []
    for i in range(clusters_Y_list.shape[0]):
        existing_class_indices_in_clusters.append(find_non_empty_indexes(clusters_Y_list[i]))
        clusters_Y_list[i] = drop_not_given_classes(clusters_Y_list[i], is_shape_printed = False)
    x_test_32x32 = x_test_32x32.reshape(x_test_32x32.shape[0], -1)
    x_test_32x32_feat = x_test_32x32
    cluster_indices_test = np.argmax(kmeans.transform(x_test_32x32_feat), axis = 1)
    correct_clusterization_count = 0
    for index in range(len(x_test_32x32_feat)):
        y_real = y_test[index]
        index_in_real_output = np.argmax(y_real, axis = 0)
        if index_in_real_output in existing_class_indices_in_clusters[cluster_indices_test[index]]:
            correct_clusterization_count += 1
    recognition_results_df.loc[len(recognition_results_df)] = [str(cluster_count),
str(correct_clusterization_count)]
    print(recognition_results_df.loc[len(recognition_results_df)-1,:].tolist())
```

2.3. Klaster təyini klasterləmənin nəticələri əsasında öyrədilən Süni Neyron Şəbəkə ilə aparıldıqda doğru klasterə düşmə tezliklərinin çıxarılışı

```
# Pixels əlaməti üçün (digər 2 əlamət analoji realizasiya olunur)
recognition_results_df = pd.DataFrame(columns=['Cl.Count', 'Correct Cl. #'])
print(['Cl.Count', 'Correct Cl. #'])
training_results_df = pd.DataFrame(columns=['Seed', 'Pooling2D', 'Model', 'Param-Count',
```

```
                                'Accuracy-Train', 'Loss-Train', 'Accuracy-Test', 'Loss-Test'])
for cluster_count in range(2, 51):#range(2,20) #2, 19, 29, 45, 47
    x_train_32x32 = x_train_32x32.reshape(x_train_32x32.shape[0], -1)
    x_train_32x32_feat = x_train_32x32
    x_train_32x32_kmeans = np.copy(x_train_32x32_feat)
    kmeans = KMeans(n_clusters = cluster_count, random_state=42)
    kmeans.fit_transform(x_train_32x32_kmeans)
    clusters_X_list = []
    clusters_Y_list = []
    for i in range(cluster_count):
        clusters_X_list.append(x_train_32x32_feat[kmeans.labels_ == i])
        clusters_Y_list.append(y_train[kmeans.labels_ == i])
    clusters_X_list = np.array(clusters_X_list, dtype = "object")
    clusters_Y_list = np.array(clusters_Y_list, dtype = "object")
    existing_class_indices_in_clusters = []
    for i in range(clusters_Y_list.shape[0]):
        existing_class_indices_in_clusters.append(find_non_empty_indexes(clusters_Y_list[i]))
        clusters_Y_list[i] = drop_not_given_classes(clusters_Y_list[i], is_shape_printed = False)
    onehotencoder = OneHotEncoder()
    kmeans_labels_one_hot = onehotencoder.fit_transform(kmeans.labels_.reshape(-1, 1)).toarray()
    seed = 42
    keras.backend.clear_session()
    set_seed(seed)
    model_LeNet5 = Sequential(name = 'LeNet5Cl' + str(cluster_count))
    model_LeNet5.add(layers.Dense(8, activation='relu', input_shape = (1024,)))
    model_LeNet5.add(layers.Dense(64, activation='relu'))
    model_LeNet5.add(layers.Dense(16, activation='relu'))
    model_LeNet5.add(layers.Dense(16, activation='relu'))
    model_LeNet5.add(layers.Dense(64, activation='relu'))
```

```
    model_LeNet5.add(layers.Dense(16, activation='relu'))
    model_LeNet5.add(layers.Dense(cluster_count, activation="softmax"))
    d_class_weights = find_class_weights(kmeans_labels_one_hot)
    model_LeNet5.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])
    model_LeNet5.fit(x_train_32x32_feat, kmeans_labels_one_hot, epochs = EPOCHS, shuffle = True,
                     callbacks=[callback], verbose=0, class_weight=d_class_weights) #callbacks=[callback],
    model = model_LeNet5
    loss_train, acc_train = model.evaluate(x_train_32x32_feat, kmeans_labels_one_hot, verbose=0)
    loss_test, acc_test = '-', '-'
    training_results_df.loc[len(training_results_df)] = [str(seed), str(Pooling2D)[47:-2], str(model.name),
                                str(model.count_params()),
                                str(acc_train), str(loss_train),
                                str(acc_test), str(loss_test)]
    training_results_df.to_csv(OUTPUT_PATH + 'ssl-cl-acc\\' + 'Results_train-cl-pixels-nn-for-
clustering_cw.csv', index = False)
    x_test_32x32 = x_test_32x32.reshape(x_test_32x32.shape[0], -1)
    x_test_32x32_feat = x_test_32x32
    cluster_indices_test = np.argmax(model.predict(x_test_32x32_feat), axis = 1)
    correct_clusterization_count = 0
    for index in range(len(x_test_32x32_feat)):
        y_real = y_test[index]
        index_in_real_output = np.argmax(y_real, axis = 0)
        if index_in_real_output in existing_class_indices_in_clusters[cluster_indices_test[index]]:
            correct_clusterization_count += 1
    recognition_results_df.loc[len(recognition_results_df)] = [str(cluster_count),
str(correct_clusterization_count)]
    print(recognition_results_df.loc[len(recognition_results_df)-1,:].tolist())
    recognition_results_df.to_csv(OUTPUT_PATH + 'ssl-cl-acc\\' + 'cl-acc-pixels-nn-for-clustering_cw.csv',
index = False)
```

2.4. Klaster təyini klasterləmənin nəticələri əsasında öyrədilən Süni Neyron Şəbəkə ilə aparıldıqda doğru klasterə düşmə tezliklərinin və tanıma dəqiqliklərinin çıxarılışı

```
# Pixels əlaməti üçün (digər 2 əlamət analoji realizasiya olunur)
folder_name = 'pixels\\'
recognition_results_df = pd.DataFrame(columns=['Cl.Count', 'Correct Cl. #', 'Correct Recog. #'])
training_results_df_cl_test = pd.DataFrame(columns=['Seed', 'Pooling2D', 'Model', 'Param-Count',
                                                    'Accuracy-Train', 'Loss-Train', 'Accuracy-Test', 'Loss-Test'])
for cluster_count in range(2, 51):
    model_per_class_structure_df = pd.DataFrame(columns=['Cl', 'Unique Class', 'Elements cl', 'Parameters',
'Layers', 'Neurons'])
    print(['Cl', 'Unique Class', 'Elements cl', 'Parameters', 'Layers', 'Neurons'])
    x_train_32x32 = x_train_32x32.reshape(x_train_32x32.shape[0], -1)
    x_train_32x32_feat = x_train_32x32
    x_train_32x32_kmeans = np.copy(x_train_32x32_feat)
    kmeans = KMeans(n_clusters = cluster_count, random_state=42)
    kmeans.fit_transform(x_train_32x32_kmeans)
    clusters_X_list = []
    clusters_Y_list = []
    for i in range(cluster_count):
        clusters_X_list.append(x_train_32x32_feat[kmeans.labels_ == i])
        clusters_Y_list.append(y_train[kmeans.labels_ == i])
    clusters_X_list = np.array(clusters_X_list, dtype="object")
    clusters_Y_list = np.array(clusters_Y_list, dtype="object")
    existing_class_indices_in_clusters = []
    for i in range(clusters_Y_list.shape[0]):
        existing_class_indices_in_clusters.append(find_non_empty_indexes(clusters_Y_list[i]))
        clusters_Y_list[i] = drop_not_given_classes(clusters_Y_list[i], is_shape_printed = False)
    # Create Recognition Model for Each Cluster
    model_list = []
```

```
    for cluster_index in range(cluster_count):
        unique_class_count = len(existing_class_indices_in_clusters[cluster_index])
        i = 4
        while unique_class_count // i <= 0:
            i -= 1
        L = unique_class_count // i # nn layers count
        q = clusters_X_list[cluster_index].shape[0] # elements count in the cluster
        k_star = round((-L + (L**2+4*L*q) ** 0.5) / (2*L)) # nn neurons count in each layer
        if q <= 1024 * 2:
            L = 0
        set_seed(seed)
        keras.backend.clear_session()
        model_LeNet5 = Sequential(name = 'RecogModelClIndex' + str(cluster_index+1))
        model_LeNet5.add(layers.Dense(2, activation='relu', input_shape=(1024,)))
        model_LeNet5.add(layers.Dense(5, activation='relu'))
        for l in range(L-2):
            model_LeNet5.add(layers.Dense(k_star, activation='relu'))
        model_LeNet5.add(layers.Dense(5, activation='relu'))
        model_LeNet5.add(layers.Dense(clusters_Y_list[cluster_index].shape[1], activation="softmax"))
        model_LeNet5.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])
        model_list.append(model_LeNet5)
        model_LeNet5.fit(clusters_X_list[cluster_index], clusters_Y_list[cluster_index], epochs = EPOCHS,
                         shuffle = True, callbacks=[callback], verbose=0) #callbacks=[callback],
        model_per_class_structure_df.loc[len(model_per_class_structure_df)] = [str(cluster_count),
str(unique_class_count),
                                                                               str(q), str(model_LeNet5.count_params()),
                                                                               str(L), str(k_star)]
        print(model_per_class_structure_df.loc[len(model_per_class_structure_df)-1,:].tolist())
        model_per_class_structure_df.to_csv(OUTPUT_PATH + folder_name + 'all-models-strs\\' +
```

```
                                'model-str-cl-' + str(cluster_count) + '.csv', index = False)
training_results_df = pd.DataFrame(columns=['Seed', 'Pooling2D', 'Model', 'Param-Count',
                                'Accuracy-Train', 'Loss-Train', 'Accuracy-Test', 'Loss-Test'])
print()
# Log training performance of models
for i in range(len(model_list)):
    model = model_list[i]
    loss_train, acc_train = model.evaluate(clusters_X_list[i], clusters_Y_list[i], verbose=0)
    loss_test, acc_test = '-', '-'
    # str(Pooling2D)[47:-2]
    training_results_df.loc[len(training_results_df)] = [str(seed), '-', str(model.name),
                                str(model.count_params()),
                                str(acc_train), str(loss_train),
                                str(acc_test), str(loss_test)]
    training_results_df.to_csv(OUTPUT_PATH + folder_name + 'all-models-trainings\\' +
                                'training-models-cl-' + str(cluster_count) + '.csv', index = False)
    print(training_results_df.loc[len(training_results_df)-1,:].tolist())
print()
onehotencoder = OneHotEncoder()
kmeans_labels_one_hot = onehotencoder.fit_transform(kmeans.labels_.reshape(-1, 1)).toarray()
seed = 42
keras.backend.clear_session()
set_seed(seed)
model = Sequential(name = 'ClModelClCount' + str(cluster_count))
model.add(layers.Dense(64, activation='relu', input_shape = (1024,)))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
```

```
    model.add(layers.Dense(16, activation='relu'))
    model.add(layers.Dense(cluster_count, activation="softmax"))
    model.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])
    model.fit(x_train_32x32_feat, kmeans_labels_one_hot, epochs = EPOCHS, shuffle = True,
            callbacks=[callback], verbose=0) #callbacks=[callback],
    loss_train, acc_train = model.evaluate(x_train_32x32_feat, kmeans_labels_one_hot, verbose=0)
    loss_test, acc_test = '-', '-'
    training_results_df_cl_test.loc[len(training_results_df_cl_test)] = [str(seed), '-', str(model.name),
                                str(model.count_params()),
                                str(acc_train), str(loss_train),
                                str(acc_test), str(loss_test)]
    training_results_df_cl_test.to_csv(OUTPUT_PATH + folder_name +
                                    'cl-models-trainings.csv', index = False)
    x_test_32x32 = x_test_32x32.reshape(x_test_32x32.shape[0], -1)
    x_test_32x32_feat = x_test_32x32
    #y_test_temp = np.argmax(kmeans.transform(x_test_32x32_feat), axis = 1)
    cluster_indices_test = np.argmax(model.predict(x_test_32x32_feat), axis = 1)
    #print(cluster_indices_test)
    correct_clusterization_count = 0
    correct_recognition_count = 0
    for index in range(len(x_test_32x32_feat)):
        y_real = y_test[index]
        index_in_real_output = np.argmax(y_real, axis = 0)
        y_predicted = model_list[cluster_indices_test[index]].predict(x_test_32x32_feat[index].reshape(1,
1024))
        index_in_model_output = np.argmax(y_predicted, axis = 1)
        sample_predicted_class_index_in_all_classes =
existing_class_indices_in_clusters[cluster_indices_test[index]][int(index_in_model_output)]
```

```
        if index_in_real_output in existing_class_indices_in_clusters[cluster_indices_test[index]]:
            correct_clusterization_count += 1
        if index_in_real_output == sample_predicted_class_index_in_all_classes:
            correct_recognition_count +=1
    recognition_results_df.loc[len(recognition_results_df)] = [str(cluster_count),
                                    str(correct_clusterization_count), str(correct_recognition_count)]
    print(recognition_results_df.loc[len(recognition_results_df)-1,:].tolist())
    recognition_results_df.to_csv(OUTPUT_PATH + folder_name +
                                  'cl-models-recognition.csv', index = False)
    print('----------------------------------------------------------------')
    print('\n')
```

3. Yanaşma 3

3.1. Köməkçi metodlar

```
# Əlamətlərin vektorizasiyası
def prep_data_X(X):
    return np.copy(X).reshape(X.shape[0],-1)
# Hər sinif üçün o sinfi ifadə edən orta əlamətin tapılması
def find_x_means(X, Y):
    X = prep_data_X(X)
    Y_ints = np.argmax(Y, axis=1)
    X_letters_list = []
    X_means = []
    for class_i in range(Y.shape[1]):
        temp = X[Y_ints == class_i]
        X_letters_list.append(temp)
        X_means.append(np.mean(temp, axis=0))
    X_means = np.array(X_means)
    return X_means
```

3.2. Klaster təyini k ortalar ilə aparıldıqda doğru klasterə düşmə tezliklərinin çıxarılışı

2.2.-dəki kodlarla analojidir. Sadəcə klasterləşdirmədə verilənlərin əlamətləri əsasında find_x_means(X, Y) modulunun köməyi ilə tapılmış orta qiymətlər istifadə olunur.

3.3. Klaster təyini klasterləmənin nəticələri əsasında öyrədilən Süni Neyron Şəbəkə ilə aparıldıqda doğru klasterə düşmə tezliklərinin çıxarılışı

2.3.-dəki kodlarla analojidir. Sadəcə klasterləşdirmədə verilənlərin əlamətləri əsasında find_x_means(X, Y) modulunun köməyi ilə tapılmış orta qiymətlər istifadə olunur.

3.4. Klaster təyini klasterləmənin nəticələri əsasında öyrədilən Süni Neyron Şəbəkə ilə aparıldıqda doğru klasterə düşmə tezliklərinin və tanıma dəqiqliklərinin çıxarılışı

2.4.-dəki kodlarla analojidir. Sadəcə klasterləşdirmədə verilənlərin əlamətləri əsasında find_x_means(X, Y) modulunun köməyi ilə tapılmış orta qiymətlər istifadə olunur.

4. Yanaşma 4

4.1. Köməkçi metodlar

Bazanın modellərin hər biri üçün çevrilərək yanaşma 4-dəki modellər üçün hazırlanması

```
def split_each_class(X_train, Y_train):
    new_X_train_list = []
    new_Y_train_list = []
    for index_of_class in range(Y_train.shape[1]):
        new_X_train = []
        new_Y_train = []
        for flag in (True, False):
            temp_X = X_train[(y_ints_train == index_of_class) == flag]
            value = 1 if flag else 0
            count = len([el for el in (y_ints_train == index_of_class) == flag if el])
            temp_Y = [[value, 1 - value]] * count
            for i in range(temp_X.shape[0]):
                new_X_train.append(temp_X[i])
```

```
                new_Y_train.append(temp_Y[i])
        new_X_train_list.append(new_X_train)
        new_Y_train_list.append(new_Y_train)
    new_X_train_list = np.array(new_X_train_list, dtype='object')
    new_Y_train_list = np.array(new_Y_train_list, dtype='object')
    print(new_X_train_list.shape)
    print(new_Y_train_list.shape)
    return new_X_train_list, new_Y_train_list
# Bazadan sinif indeksi verilən verilənlərin çıxarılışı
def find_class_samples_by_index(X_train, Y_train, index_of_class):
    new_X_train = []
    new_Y_train = []
    for flag in (True, False):
        temp_X = X_train[(y_ints_train == index_of_class) == flag]
        #temp_Y = Y_train[(y_ints_train == index_of_class) == flag]
        value = 1 if flag else 0
        count = len([el for el in (y_ints_train == index_of_class) == flag if el])
        temp_Y = [[value, 1 - value]] * count
        for i in range(temp_X.shape[0]):
            new_X_train.append(temp_X[i])
            new_Y_train.append(temp_Y[i])
    new_X_train = np.array(new_X_train, dtype='object')
    new_Y_train = np.array(new_Y_train, dtype='object')
    return new_X_train, new_Y_train
# Öyrətmədə balans yaratmaq üçün müşahidələrə məxsus olduğu sinifdəki müşahidə sayına tərs mütənasib
çəki verilməsi
def find_class_weights(y):
    y_integers = np.argmax(y, axis=1)
```

```
    class_weights = compute_class_weight(class_weight = 'balanced',classes = np.unique(y_integers), y = y_integers)
    d_class_weights = dict(enumerate(class_weights))
    return d_class_weights
# Tanıma nəticələrinin yadda saxlanılması
def save_a_dict(dictionary, file_name):
    with open(file_name, 'w', encoding="utf-8") as csvfile:
        csvwriter = csv.writer(csvfile)
        for key, val in dictionary.items():
            csvwriter.writerow([key, val])
# Verilmiş təsvirin əlamətlərinin qiymətlərinin bütün modellərdə sınanaraq yekun tanıma nəticəsi çıxarılışı
def predict_class(x, model_list):
    predictions = []
    for i in range(len(model_list)):
        prediction = np.argmax(model_list[i].predict(x), axis = 1)
        predictions.append(prediction)
    predicted_count = len([pred for pred in predictions if pred == 0])
    unpredicted_count = len([pred for pred in predictions if pred == 1])
    if predicted_count > 1 or unpredicted_count == 28:
        return -1, predicted_count, unpredicted_count
    prediction = [i for i in range(len(predictions)) if predictions[i] == 0][0]
    return prediction, predicted_count, unpredicted_count
# Testləşdirmə bazasındakı təsvirlərin hansı hərflərin təsvirləri olduğunun yadda saxlanılması üçün verilənlər sturkturunun yaradılması
def gen_dict_letters():
    wrong_dict = {}
    for i in range(len(label_map)):
        wrong_dict[label_map[i]] = 0
    return wrong_dict
```

4.2. Tanıma modellərinin öyrədilməsi

```
EPOCHS = 200
callback = EarlyStopping(monitor='loss', min_delta=0.01, patience=5)
seed = 42
Pooling2D = layers.AveragePooling2D
results_df = pd.DataFrame(columns=['Seed', 'Pooling2D', 'Model', 'Param-Count', 'Epochs',
                                   'Accuracy-Train', 'Loss-Train'])
print(['Seed', 'Pooling2D', 'Model', 'Param-Count', 'Epochs',
                                   'Accuracy-Train', 'Loss-Train'])
model_list = []
#for X_train, Y_train in zip(X_train_list, Y_train_list):
for i in range(28):
    new_X_train, new_Y_train = find_class_samples_by_index(X_train, Y_train, i)
    new_X_train = new_X_train.reshape(-1, 32, 32, 1)
    new_X_train = np.asarray(new_X_train).astype(np.float32)
    new_Y_train = np.asarray(new_Y_train).astype(np.int32)
    set_seed(seed)
    keras.backend.clear_session()
    model_LeNet5 = Sequential(name = 'LeNet5Letter' + str(i+1))
    model_LeNet5.add(layers.Conv2D(filters=6,  kernel_size=(5,5), padding='valid', activation='relu',
input_shape=(32, 32, 1)))
    model_LeNet5.add(Pooling2D(pool_size = (2,2)))
    model_LeNet5.add(layers.Conv2D(filters=16, kernel_size=(5,5), padding='valid', activation='relu'))
    model_LeNet5.add(Pooling2D(pool_size=(2,2)))
    model_LeNet5.add(layers.Conv2D(filters=120, kernel_size=(5,5), padding='valid', activation='relu'))
    model_LeNet5.add(layers.Flatten())
    model_LeNet5.add(layers.Dense(84, activation='relu'))
    model_LeNet5.add(layers.Dense(2, activation="softmax"))
```

```
    model_LeNet5.compile(optimizer = 'Adam' , loss = "categorical_crossentropy", metrics = ['accuracy'])
    d_class_weights = find_class_weights(new_Y_train)
    history = model_LeNet5.fit(new_X_train, new_Y_train, epochs = EPOCHS, shuffle = True,
callbacks=[callback], verbose=0,
                               class_weight=d_class_weights) #callbacks=[callback],

    loss_train_LeNet5, acc_train_LeNet5 = model_LeNet5.evaluate(new_X_train, new_Y_train, verbose=0)

    results_df.loc[len(results_df)] = [str(seed), str(Pooling2D)[47:-2], str(model_LeNet5.name),
                                       str(model_LeNet5.count_params()), str(len(history.history['loss'])),
                                       acc_train_LeNet5, loss_train_LeNet5]
    print(results_df.loc[len(results_df)-1,:].tolist())
    results_df.to_csv('output/Results_train-pixels32x32.csv', index = False)
    model_list.append(model_LeNet5)
```

4.3. Tanıma nəticələrinin alınması

```
predicted_counts = []
unpredicted_counts = []
correct_recognition_count = 0
correct_recognition_indices = []
predicted_indices = []
unpredicted_indices = []
wrong_predicted_dict = gen_dict_letters()
wrong_unpredicted_dict = gen_dict_letters()
wrong_all_dict = gen_dict_letters()
for i in range(X_test.shape[0]): #X_test.shape[0]
    sample = X_test[i].reshape(1, 256)
    prediction, predicted_count, unpredicted_count = predict_class(sample, model_list)
    real_class = y_ints_test[i]
```

```
    if prediction != -1:
        print(label_map[prediction], end = '')
        if real_class == prediction:
            correct_recognition_indices.append(i)
            correct_recognition_count += 1
    else:
        wrong_all_dict[label_map[real_class]] += 1
        if predicted_count != 0:
            predicted_indices.append(i)
            predicted_counts.append(predicted_count)
            wrong_predicted_dict[label_map[real_class]] += 1
        else:
            unpredicted_indices.append(i)
            unpredicted_counts.append(unpredicted_count)
            wrong_unpredicted_dict[label_map[real_class]] += 1
print()
print('Recog:', correct_recognition_count)
print('Predicted Ones:', len(predicted_counts))
print('Unpredicted Ones:', len(unpredicted_counts))
save_a_dict(wrong_predicted_dict, 'output/pred.csv')
save_a_dict(wrong_unpredicted_dict, 'output/unpred.csv')
save_a_dict(wrong_all_dict, 'output/all-pred.csv')
```

Резюме

В диссертационной работе разработаны алгоритмы и программное обеспечение для извлечения признаков из различных классов для использования в распознавание рукопечатных букв азербайджанского алфавита. Эффективность рассматриваемых признаков при распознавании изучалась в четырех различных подходах к распознаванию. В качестве механизма для извлечения признаков, которые можно использовать при кластеризации использовалась о бученная свертечная нейронная сеть. Проведен сравнительный анализ рассмотренных подходов распознавания.

Summary

In the master thesis algorithms and software have been developed for extracting features of various classes in order to use in Azerbaijani handwritten letters recognition. The effectiveness of the considered features used for recognition was studied in the four different approaches to recognition. A trained convolutional neural network was used as a feature extraction mechanism that can be used in clustering. A comparative analysis of the considered recognition approaches has been carried out.