CHUQUR O‘QITISHGA ASOSLANGAN REAL VAQTLI YUZNI TANISH TIZIMINING TIZIMLI TAHLILI

To‘raqulov Asfandiyor Jaxongirovich[1]

[1] Buxoro innovatsiyalar universiteti, Buxoro, O’zbekiston.
[1] E-mail: asfan.webdeloper@gmail.com
DOI: 10.61587/mmit.tiue.uz.v1i1.309

Annotatsiya. Maqolada videokuzatuv kameralari oqimlarida shaxsni real vaqtda aniqlash (face recognition) uchun chuqur o’qitishga asoslangan modulli tizimning to’liq arxitekturasi va har bir bosqichning ishlash mexanizmi batafsil yoritiladi. Tizim besh bosqichli ishlov berish quvuridan iborat: kameradan kadrlarni qabul qilish, yuzni aniqlash (RetinaFace/MTCNN), tekislash (alignment), embedding (FaceNet/ArcFace) va kosinus o’xshashligi orqali taqqoslash. Embedding vektorlari PostgreSQL + pgvector ombori orqali tezkor qidiriladi. Maqolaning amaliy qismida tizimning taqqoslash mantig’i 50 ta shaxsdan iborat sintetik embedding to’plamida baholandi: AUC = 1,000, teng xato darajasi (EER) = 0%, 0,6 chegarada aniqlik (accuracy) = 1,000 va F1-o’lchov = 0,995 natijalari olindi. Maqola texnik mutaxassisga tizimni to’liq tushunish va qayta tiklash imkonini beradigan tarzda, har bir komponent “nima uchun va qanday ishlashi” darajasida tushuntirilgan.

Kalit so’zlar: *yuzni tanish, chuqur o’qitish, real vaqt, RetinaFace, FaceNet, ArcFace, embedding, kosinus o’xshashlik, videokuzatuv, kompyuter ko’rishi, pgvector.*

SYSTEMATIC ANALYSIS OF A DEEP-LEARNING-BASED REAL-TIME FACE RECOGNITION SYSTEM

Toraqulov Asfandiyor Jaxongirovich[1]

[1] Bukhara University of Innovation, Bukhara, Uzbekistan.
[1] E-mail: asfan.webdeloper@gmail.com

Abstract. This paper provides a detailed account of the full architecture and the operating mechanism of each stage of a deep-learning-based modular system for real-time person identification (face recognition) in surveillance camera streams. The system consists of a five-stage pipeline: frame acquisition, face detection (RetinaFace/MTCNN), alignment, embedding (FaceNet/ArcFace), and comparison via cosine similarity. Embedding vectors are searched rapidly through a PostgreSQL + pgvector store. In the experimental part, the comparison logic was evaluated on a synthetic embedding set of 50 identities, yielding AUC = 1.000, an Equal Error Rate (EER) of 0%, an accuracy of 1.000 at a 0.6 threshold, and an F1-score of 0.995. The paper explains each component at a “why and how” level so that a technical specialist can fully understand and reproduce the system.

Keywords: *face recognition, deep learning, real-time, RetinaFace, FaceNet, ArcFace, embedding, cosine similarity, video surveillance, computer vision, pgvector.*

Kirish

So‘nggi yillarda sun’iy intellekt va kompyuter ko‘rish texnologiyalarining jadal rivojlanishi natijasida biometrik identifikatsiya tizimlari keng qo‘llanila boshladi. Ayniqsa, yuzni tanish texnologiyasi xavfsizlik tizimlari, videokuzatuv, bank sektori, ta’lim muassasalari, transport va aqlli shahar infratuzilmalarida muhim ahamiyat kasb etmoqda. Kamera oqimlarida shaxsni avtomatik aniqlash va identifikatsiya qilish jarayonini real vaqt rejimida amalga oshirish inson omiliga bo‘lgan ehtiyojni kamaytirish bilan birga tizimning aniqligi va tezkorligini oshiradi. An’anaviy yuzni tanish usullari yoritilish, yuz holati, masofa va tasvir sifati o‘zgarishlariga nisbatan sezgir bo‘lib, yuqori aniqlikni ta’minlashda ayrim cheklovlarga ega. Shu sababli chuqur o‘qitish (Deep Learning) asosidagi yondashuvlar yuzni aniqlash va tanish masalalarida samarali yechim sifatida rivojlanmoqda. Konvolyutsion neyron tarmoqlar

(CNN), FaceNet, DeepFace, ArcFace va YOLO kabi modellar katta hajmdagi ma'lumotlar asosida murakkab biometrik belgilarni ajratib olish imkonini beradi. Real vaqtli yuzni tanish tizimlari bir nechta asosiy bosqichlardan iborat bo'lib, ular tasvirni qabul qilish, yuzni aniqlash, yuz belgilarini ajratish, identifikatsiya va natijani qayta ishlash jarayonlarini o'z ichiga oladi. Ushbu bosqichlarning samarali integratsiyasi tizimning ishlash tezligi va aniqligini belgilaydi. Ayniqsa, video oqimlar bilan ishlashda hisoblash tezligi va apparat resurslaridan optimal foydalanish muhim omillardan hisoblanadi.

Yuzni aniqlash va tanish (face detection and recognition) kompyuter ko'rishining (computer vision) markaziy yo'nalishlaridan biri hisoblanadi. U kirishni nazorat qilish (access control), videokuzatuv, davomatni avtomatik qayd etish, to'lov tizimlari va xavfsizlik sohalarida keng qo'llaniladi [8]. Mazkur texnologiyaning maqsadi oddiy: kamera ko'rgan odamning kim ekanligini avtomatik aniqlash. Biroq bu oddiy maqsad ortida bir necha murakkab texnik bosqichlardan iborat. Xususan:

Yuzni aniqlash (face detection) — bu kadrda yuz bor-yo'qligini va u qayerda joylashganligini topish. Yuzni tanish (face recognition) — bu topilgan yuzning aynan kimga tegishli ekanligini aniqlash. Birinchisi "bu yerda yuz bor" deydi, ikkinchisi "bu yuz — Aliyev" deydi. To'liq tizim avval aniqlashni, so'ng tanishni bajaradi.

An'anaviy usullar (Haar-like kaskadlari, AdaBoost, shablonga moslash) yuzning geometrik xususiyatlari va shablonlardan foydalanadi. Ular tez ishlaydi, lekin yoritish o'zgarishi, bosh burilishi yoki yuzning qisman to'silishi (occlusion) sharoitida aniqligi keskin pasayadi. Chuqur o'qitish (deep learning) ushbu cheklovlarni bartaraf etib, yuqori aniqlik va barqarorlikni ta'minladi [5, 8].

Yuqorida muhokama qilingan ilmiy ishlargadan kelib chiqib mazkur maqolada chuqur o'qitishga asoslangan real vaqtli yuzni tanish tizimining arxitekturasi, ishlash prinsipi va asosiy algoritmlari tizimli ravishda tahlil qilinadi. Shuningdek, zamonaviy neyron tarmoq modellari asosida yuzni aniqlash va identifikatsiya qilish usullarining samaradorligi hamda amaliy qo'llanilish jihatlari ko'rib chiqiladi.

TIZIMNING UMUMIY ARXITEKTURASI.

Yuzni tanish — bu yagona algoritm emas, balki ketma-ket bog'langan bosqichlar zanjiri. Bu zanjir "ishlov berish quvuri" (pipeline) deb ataladi. Kameradan kelgan har bir videokadr quvurning boshidan oxirigacha o'tadi va natijada "bu kim?" degan savolga javob chiqadi. Tizimning umumiy tuzilishi va ma'lumotlar oqimi 1-rasmda keltirilgan.

1-rasm. Real vaqtli yuzni tanish tizimining besh bosqichli ishlov berish quvuri

Yuqoridagi arxitektura bo'yicha tizimni har bir qadamini algrotirmlarini boshqichma bosqich qurib chiqiladi.

Birinchi bosqich: YUZNI ANIQLASH (DETECTION)

Tizim ishga tushganda u butun videokadrni ko'radi — unda devor, mebel, bir necha odam bo'lishi mumkin. Birinchi vazifa — ushbu katta rasmda yuzlar qayerdaligini aniqlash. Bu vazifani yuzni aniqlash modeli bajaradi va har bir topilgan yuz uchun ikki narsa qaytaradi: (a) chegaralovchi to'rtburchak (bounding box, qisqacha bbox) — yuzni o'rab turuvchi to'rtburchak koordinatalari; (b) asosiy nuqtalar (landmarks) — odatda beshta nuqta: ikki ko'z, burun uchi va og'izning ikki burchagi.

RetinaFace — hozirgi kunda eng aniq modellardan biri. U "single-shot" (bir o'tishli) va "anchor-free" arxitekturaga ega bo'lib, yuzni va beshta nuqtani bir vaqtning o'zida aniqlaydi. To'silish va turli burchaklarda ham yaxshi ishlaydi [3].

MTCNN — uch bosqichli kaskadli neyron tarmoq. RetinaFace'dan yengilroq, CPU'da ham ishlay oladi, lekin biroz sekinroq [4]. Tanlov vaziyatga bog'liq: yuqori aniqlik kerak bo'lsa RetinaFace, cheklangan resurs bo'lsa MTCNN.

1-listing. Yuzni aniqlash va landmark olish (DeepFace)

```
from deepface import DeepFace
import cv2

frame = cv2.imread('frame.jpg')    # kameradan kadr
faces = DeepFace.extract_faces(
    img_path=frame,
    detector_backend='retinaface',  # aniqlash modeli
    align=True)                     # landmark asosida tekislash

print('Topilgan yuzlar soni:', len(faces))
```

Ikkinchi bosqich: YUZNI TEKISLASH (ALIGNMENT)

Kamera odamni turli burchaklarda ko'radi: bosh chapga yoki o'ngga egilgan, biroz qiya bo'lishi mumkin. Agar bu "qiyshiq" yuzlarni to'g'ridan-to'g'ri keyingi bosqichga uzatsak, tizim bir odamning ikki rasmini turli deb hisoblashi mumkin. Shuning uchun tekislash (alignment) bosqichi qo'llaniladi.

Tekislashning mohiyati sodda: oldingi bosqichda topilgan ko'zlar koordinatasidan foydalanib, yuz shunday buriladiki, ikki ko'z bir gorizontal chiziqda turadi. Natijada barcha yuzlar bir xil standart holatga keladi va odatda 160×160 yoki 112×112 piksel o'lchamiga qirqiladi (crop). Bu qadam embedding sifatini sezilarli oshiradi — chunki model endi har doim "to'g'ri qaragan" yuzni ko'radi.

Soddalashtirib aytganda: tekislash — bu rasmga olishdan oldin odamni "to'g'ri o'tir, kameraga qara" deyishga o'xshaydi. U keyingi taqqoslashni adolatli va aniq qiladi.

Uchinchi bosqich: EMBEDDING — YUZNING RAQAMLI BARMOQ IZI

Bu bosqichda— butun tizimning eng muhim va eng "aqlli" qismi. Kompyuter rasmni piksellar to'plami sifatida ko'radi, lekin ikki rasm bir odamra tegishli ekanligini piksellarni taqqoslab aniqlab bo'lmaydi (yoritish, soya, burchak hammasini o'zgartiradi). Yechim — har bir yuzni o'zgarmas o'lchamli sonlar ro'yxatiga, ya'ni vektorga aylantirish. Bu vektor embedding deb ataladi.

Embedding — bu yuzning raqamli "barmoq izi". Masalan, FaceNet har bir yuzni 128 ta sondan iborat vektorga, ArcFace esa odatda 512 ta songa aylantiradi. Bu sonlar yuzning o'ziga xos xususiyatlarini (ko'zlar orasidagi masofa, yuz shakli va h.k. — lekin model buni o'zi o'rganadi) kodlaydi.

Asosiy g'oya — sehrli, lekin sodda: model shunday o'qitilganki,

1) bir odamning turli rasmlari yaqin vektorlar beradi (bir-biriga o'xshash sonlar);

2) turli odamlarning rasmlari uzoq vektorlar beradi (farqli sonlar).

Aynan shu xususiyat tufayli ikki yuzni taqqoslash — bu ikki vektor orasidagi masofani o'lchashga aylanadi. ArcFace bu ajratishni yanada kuchaytirish uchun maxsus "qo'shimcha burchak chekkasi yo'qotish funksiyasi" (Additive Angular Margin Loss) dan foydalanadi — u bir shaxs vektorlarini yanada zichlashtirib, turli shaxslarni yanada uzoqlashtiradi [2].

2-listing. Yuzdan embedding olish va normallashtirish

```
import numpy as np

# Tekislangan yuzdan 128-o'lchovli vektor olish
emb = DeepFace.represent(
    img_path=face_img,
    model_name='Facenet',
    detector_backend='skip')[0]['embedding']
```

Oxirgi qatordagi normallashtirish (L2-normalization) muhim: u vektorning uzunligini 1 ga tenglashtiradi, shunda taqqoslashda faqat vektor yo'nalishi muhim bo'ladi, uzunligi emas. Bu kosinus o'xshashlikni barqaror qiladi.

To'rtinchi bosqich: TAQQOSLASH VA QAROR

Kosinus o'xshashlik:

Endi yangi yuzning vektori va bazada saqlangan ma'lum shaxslarning vektorlari bor. Savol: yangi vektor qaysi bazadagi vektorga eng yaqin? Yaqinlikni o'lchash uchun kosinus o'xshashlik (cosine similarity) ishlatiladi. U ikki vektor orasidagi burchakni o'lchaydi: ikki vektor bir tomonga qaragan bo'lsa, o'xshashlik 1 ga yaqin; perpendikulyar bo'lsa, 0; qarama-qarshi bo'lsa, −1. Formula:

$$\text{sim}(a, b) = (a \cdot b) / (\|a\| \times \|b\|)$$

bu yerda $a \cdot b$ — ikki vektorning skalyar ko'paytmasi, $\|a\|$ va $\|b\|$ — ularning uzunliklari. Natija [−1, 1] oralig'ida bo'ladi. Yuz tanish kontekstida: o'xshashlik yuqori bo'lsa — bir odam, past bo'lsa — turli odamlar.

Chegara (threshold) va qaror

O'xshashlik sonini olganimizdan keyin qaror qabul qilish kerak: "bu bir odammi yoki yo'qmi?" Buning uchun chegara (threshold) belgilanadi. Agar o'xshashlik chegaradan yuqori bo'lsa — "ha, bir odam", past bo'lsa — "yo'q". Amaliyotda chegara odatda 0,6 atrofida tanlanadi [6]. Chegara tanlash muhim muvozanat: juda past chegara begona odamlarni "tanigan" deb xato qiladi (xavfsizlik xatari), juda yuqori chegara esa o'z odamini "tanimadi" deb rad etadi (noqulaylik).

3-listing. Kosinus o'xshashlik va qaror funksiyasi

```
from numpy import dot
from numpy.linalg import norm

def verify(a, b, threshold=0.6):
    sim = dot(a, b) / (norm(a) * norm(b))
    is_same = sim >= threshold
    return is_same, sim
```

Tezkor qidiruv: pgvector ombori

Bazada minglab shaxs bo'lsa, yangi vektorni har biri bilan birma-bir solishtirish sekin kechadi. Buni hal qilish uchun embedding vektorlari PostgreSQL ma'lumotlar bazasiga pgvector kengaytmasi yordamida saqlanadi. pgvector maxsus indeks (ivfflat) yordamida "eng yaqin qo'shni" (nearest neighbour) ni millisekundlarda topadi — hatto 100 mingdan ortiq yuz orasida ham. Quyidagi SQL so'rovi berilgan vektorga eng o'xshash beshta shaxsni qaytaradi.

4-listing. pgvector orqali eng yaqin yuzni qidirish (SQL)

```
SELECT person_name,
       1 - (embedding <=> :query) AS similarity
FROM faces
ORDER BY embedding <=> :query   -- eng yaqindan boshlab
LIMIT 5;
```

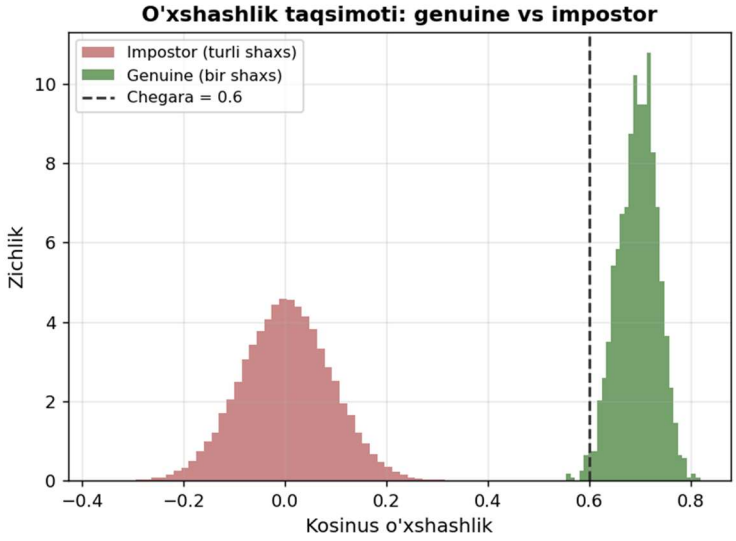
Bu yerda "<=>" operatori pgvector'ning kosinus masofa operatori bo'lib, "1 − masofa" o'xshashlikni beradi.

TIZIMNI BAHOLASH VA EKSPERIMENT NATIJALARI

Tizimning yuragi — taqqoslash bosqichi qanchalik ishonchli ekanligini o'lchash uchun nazorat eksperimenti o'tkazildi. 50 ta shaxsdan iborat, har biri 8 ta tasvirga ega sintetik embedding to'plami yaratildi (jami 400 ta 128-o'lchovli vektor). Sintetik ma'lumot real FaceNet embeddinglarining xulqini taqlid qiladi: bir shaxs vektorlari yaqin, turli shaxslarniki uzoq. So'ngra barcha mumkin bo'lgan juftliklar ikki guruhga ajratildi: genuine (bir shaxsning ikki rasmi) va impostor (turli shaxslar).

O'xshashlik taqsimoti

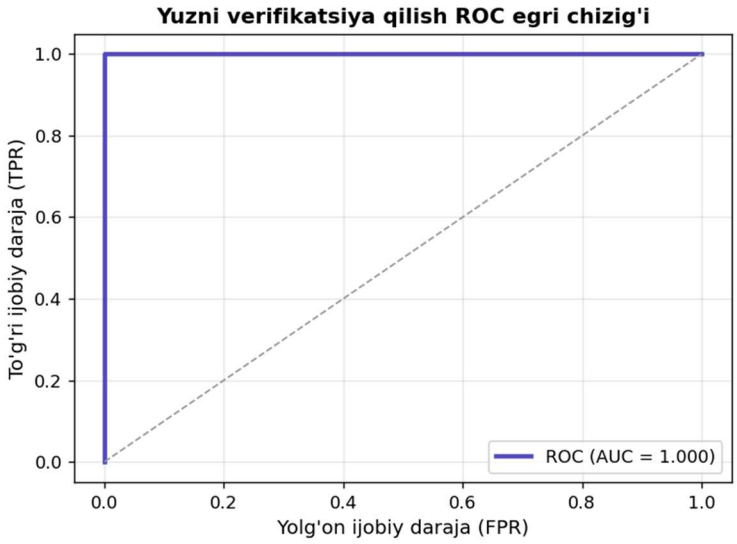
2-rasmda genuine va impostor juftliklarining kosinus o'xshashlik taqsimoti ko'rsatilgan. Ko'rinib turibdiki, impostor juftliklar (turli odamlar) 0 atrofida, genuine juftliklar (bir odam) esa 0,7 atrofida to'plangan. Ikki taqsimot deyarli to'liq ajralgan — bu 0,6 chegara ularni ishonchli ajrata olishini bildiradi.



2-rasm. Kosinus o'xshashlik taqsimoti: genuine (bir shaxs) va impostor (turli shaxs)

ROC egri chizig'i va ko'rsatkichlar

Tizim sifatini umumiy baholash uchun ROC (Receiver Operating Characteristic) egri chizig'i qurildi (3-rasm). U turli chegaralarda to'g'ri va yolg'on aniqlash darajalari o'rtasidagi munosabatni ko'rsatadi. Egri chiziq ostidagi maydon (AUC) qancha 1 ga yaqin bo'lsa, tizim shuncha mukammal. Bizning sintetik sinovda AUC = 1,000 ga teng bo'ldi.



3-rasm. Yuzni verifikatsiya qilish ROC egri chizig'i (AUC = 1,000)

To'liq ko'rsatkichlar 2-jadvalda keltirilgan. Teng xato darajasi (EER) — yolg'on qabul va yolg'on rad teng bo'lgan nuqtadagi xato; u qancha past bo'lsa shuncha yaxshi.

2-jadval. Sintetik embedding to'plamidagi verifikatsiya ko'rsatkichlari

Ko'rsatkich	Qiymat
AUC (ROC ostidagi maydon)	1,000
Teng xato darajasi (EER)	0%
Aniqlik / Accuracy (t=0,6)	1,000
F1-o'lchov (t=0,6)	0,995
Genuine o'rtacha o'xshashlik	0,695
Impostor o'rtacha o'xshashlik	0,003

Cheklovlar. Ushbu ko'rsatkichlar sintetik embedding to'plamiga asoslangani sababli tizimning taqqoslash mantig'i ishchanligini namoyish etadi, biroq real kamera sharoitidagi (yoritish, burchak, harakat xiralashuvi, to'silish) samaradorligini to'liq aks ettirmaydi. Real ma'lumotlarda EER va aniqlik ko'rsatkichlari pasayishi tabiiy. Keyingi tadqiqot bosqichida tizim LFW va CASIA-WebFace kabi ochiq haqiqiy ma'lumotlar to'plamlarida hamda jonli kamera oqimida (FPS, kechikish o'lchovlari bilan) sinovdan o'tkazilishi rejalashtirilgan.

TEXNOLOGIK STEK VA REAL VAQTLI JORIY ETISH

Tizim Python tilida amalga oshiriladi. Asosiy komponentlar: yuzni aniqlash va embedding uchun DeepFace yoki InsightFace kutubxonalari hamda OpenCV; vektorlarni saqlash va qidirish uchun PostgreSQL + pgvector; tizimni veb-xizmat sifatida taqdim etish uchun FastAPI; frontend (kamera oqimini ko'rsatish) uchun React.

Real vaqtli ishlash uchun yuk taqsimlanadi: yuzni aniqlash kabi tez-tez bajariladigan, lekin yengil vazifa chekka qurilmada (edge — masalan, kamera yonidagi Jetson Nano), embedding va baza qidiruvi esa kuchliroq serverda (cloud) bajariladi. Bunday edge-cloud taqsimot tarmoq yukini va kechikishni kamaytiradi: faqat qirqilgan kichik yuz tasviri serverga yuboriladi, butun videokadr emas [16]. Bu yondashuv kadrlar tezligini (FPS) sezilarli oshiradi.

5-listing. To'liq identifikatsiya endpointi (FastAPI)

```
from fastapi import FastAPI, UploadFile
app = FastAPI()

@app.post('/identify')
async def identify(file: UploadFile):
    img   = read_image(await file.read())   # kadrni o'qish
    faces = detect_and_align(img)           # 1-2 bosqich
    out = []
    for f in faces:
        emb  = get_embedding(f)             # 3-bosqich
        name, sim = search_pgvector(emb)    # 4-bosqich
        ok = sim >= 0.6                     # 5-bosqich
        out.append({'name': name if ok else 'Noma\'lum'.
```

XULOSA

Maqolada kamera oqimlarida shaxsni real vaqtda aniqlash uchun chuqur o'qitishga asoslangan yuzni tanish tizimining to'liq texnik arxitekturasi va har bir bosqichning ishlash mexanizmi batafsil yoritildi. Tizim besh bosqichli pipeline shaklida tashkil etilgan: aniqlash (RetinaFace/MTCNN), tekislash, embedding (FaceNet/ArcFace), kosinus o'xshashlik bo'yicha taqqoslash va chegara asosidagi qaror. Embedding qidiruvi pgvector orqali tezlashtiriladi, real vaqtli ishlash esa edge-cloud taqsimot orqali ta'minlanadi.Tizimning taqqoslash mantig'i sintetik embedding to'plamida baholanib, AUC = 1,000, EER = 0% va F1 = 0,995 ko'rsatkichlari olindi; genuine va impostor taqsimotlari to'liq ajralganligi tasdiqlandi. Maqola har bir komponentni "nima uchun va qanday ishlashi" darajasida tushuntirgani bois, texnik mutaxassis tizimni to'liq tushunib, qayta tiklay oladi.

Foydalanilgan adabiyotlar

1. Schroff F., Kalenichenko D., Philbin J. FaceNet: A unified embedding for face recognition and clustering // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015. P. 815–823.
2. Deng J., Guo J., Yang J., Xue N., Kotsia I., Zafeiriou S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022. Vol. 44, No. 10. P. 5962–5979. DOI: 10.1109/TPAMI.2021.3087709.
3. Deng J., Guo J., Ververas E., Kotsia I., Zafeiriou S. RetinaFace: Single-shot multi-level face localisation in the wild // IEEE/CVF CVPR. 2020. P. 5202–5211. DOI: 10.1109/CVPR42600.2020.00525.
4. Zhang K., Zhang Z., Li Z., Qiao Y. Joint face detection and alignment using multitask cascaded convolutional networks (MTCNN) // IEEE Signal Processing Letters. 2016. Vol. 23, No. 10. P. 1499–1503.
5. Abid M.M., Mahmood T., Ashraf R. et al. Computationally intelligent real-time security surveillance system in the education sector using deep learning // PLOS ONE. 2024. Vol. 19, No. 7. e0301908. DOI: 10.1371/journal.pone.0301908.
6. Firmansyah A., Kusumasari T.F., Alam E.N. Comparison of face recognition accuracy of ArcFace, FaceNet and FaceNet512 models on DeepFace framework // ICCoSITE. 2023. P. 535–539. DOI: 10.1109/ICCoSITE57641.2023.10127799.
7. Iype J.K., Sebastian S. Comparative analysis of state-of-the-art face recognition models: FaceNet, ArcFace, and OpenFace // ICCSST 2023, CCIS, vol. 1973. Springer, 2024. DOI: 10.1007/978-3-031-50993-3_18.
8. Optimizing face recognition inference with a collaborative edge–cloud network // Sensors (MDPI). 2022. Vol. 22, No. 21. Art. 8371.