SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

AIS ID: 92320

HĽADANIE OBJEKTOV NA OBRAZE: DETEKCIA TVÁRE

ZÁPOČTOVÉ ZADANIE

Predmet:	I-BIOM – Biometria
Prednášajúci:	prof. Dr. Ing. Miloš Oravec
Cvičiaci:	Ing. Marián Šebeňa

Bratislava 2025 **Bc. Lukáš Patrnčiak**

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek

AR	Rozšírená realita
CSP	Cross Stage Partial
FERET	Face Recognition Technology
FN	False Negatives
FP	False Positives
FPN	Feature Pyramid Network
IoU	Intersection over Union
KNN	Konvolučná neurónová sieť
PAN	Path Aggregation Network
SSD	Single Shot Detector
TN	True Negatives
TP	True Positives
YOLO	You Only Look Once

Úvod

Cieľom tohto zadania je vo vybranom programovacom jazyku implementovať program, ktorý dokáže nájsť tváre na obrázku. V rámci tohto zadania máme k dispozícii databázu fotografii a k ním prislúchajúce YOLO anotácie.

Úlohou je spracovať tieto fotografie (a YOLO anotácie) tak, aby vedeli vykresliť anotácie (je potrebné spraviť koláž 6 fotografii s anotáciami) a taktiež vyskúšať vybranú metódu predspracovania obrázkov.

Ďalší krok bude spočívať v detekovaní tvárii na týchto obrázkov prostredníctvom dvoch natrénovaných detektorov (Haar Cascade a MediaPipe), pričom bude vypočítané taktieť Precision, Recall a F1 skóre pre IoU > 0.7. Na záver kapitoly prebehne analýza výsledkov.

Nasledujúci krok spočíva v natrénovaní, resp. dotrénovaní vlastného detektora, pričom bude popísaná architektúra daného detektora a taktiež budú vypočítané metriky Precision, Recall a F1 skóre pre IoU > 0.7, vrátane vykreslenia konfúznej matice (TP, TN, FP, FN).

Posledná kapitola bude venovaná porovnaniu týchto dvoch metód a ich úspešnosti - porovnanie výsledkov detekovania tvári v prípade natrénovaných detektorov a natrénovaného vlastného detektora.

1 Spracovanie dát

Spracovanie dát a implementácia jednotlivých častí zadania bude prebiehať v programovacom jazyku Python[1] s použitím nasledovným knižníc: [1]

1. **OpenCV[2]:** najpoužívanejšia knižnica na spracovanie obrazu a počítačové videnie
2. **MediaPipe[3]:** knižnica od Google na detekciu tvárí, rúk, tela atď. pomocou strojového učenia
3. **MatPlotLib.PyPlot[4]:** vytváranie vizualizácii a grafov
4. **OS[5]:** štandardná knižnica Pythona na prácu so súbormi a adresármi
5. **NumPy[6]:** knižnica na rýchlu prácu s veľkými poľami (polia, matice) a numerické výpočty.
6. **Seaborn[7]:** knižnica pre tvorbu štatistických a vizualizačných grafov založená na Matplotlibe, ktorá uľahčuje tvorbu krásnych grafov
7. **Ultralytics[8]:** moderná knižnica pre jednoduché používanie a tréning YOLO modelov na detekciu objektov

Najprv bolo potrebné obrázky načítať obrázky a YOLO anotácie, ktorých cesta je "Dataset/images/nazov_obrazku.png, jpg, jpeg", resp. "Dataset/labels/nazov_obrazku.txt". Súbor obsahujúci YOLO anotácie má tvar:

```
<class_id> <x_center> <y_center> <width> <height>
```

pričom popis jednotlivých prvkov tohto súboru jenasledovný:

- **<class_id>**: Číselný identifikátor triedy objektu (0 = tvár)
- **<x_center>**: Vodorovná súradnica stredu objektu (relatívna: 0.0 – 1.0)
- **<y_center>**: Zvislá súradnica stredu objektu (relatívna: 0.0 – 1.0)
- **<width>**: Šírka objektu (relatívna k šírke obrázka)
- **<height>**: Výška objektu (relatívna k výške obrázka)

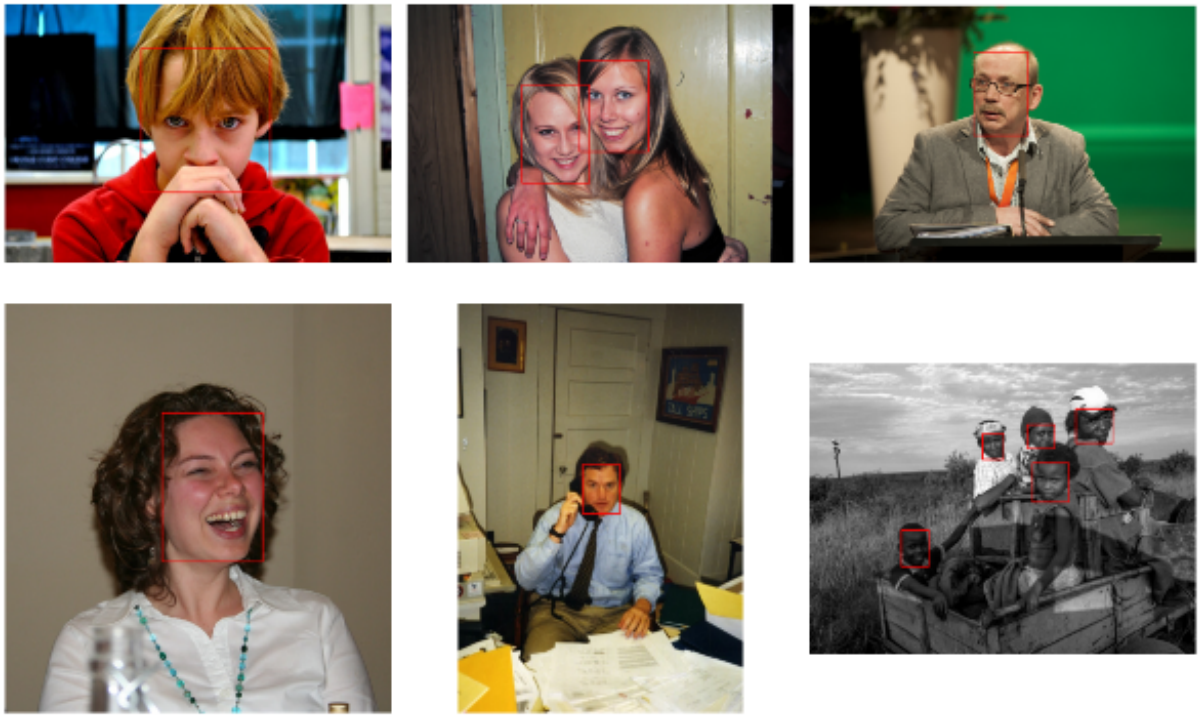
To, že sú tieto hodnoty v relatívnom formáte znamená, že vyjadrené ako zlomky šírky a výšky obrázka (napr. hodnota 0.5 znamená stred obrázka).

[1] https://python.org
[2] https://opencv.org/
[3] https://pypi.org/project/mediapipe/
[4] https://matplotlib.org/
[5] https://docs.python.org/3/library/os.html
[6] https://numpy.org/
[7] https://seaborn.pydata.org/
[8] https://www.ultralytics.com/

Následne prebehlo vykreslenie koláže 6 obrázkov, pričom pre každú tvár na jednotlivých obrázkov bol vykreslený rám podľa YOLO anotácii z prislušeného súboru z priečinka "labels". Pre vykreslenie koláže fotiek bola implementovaná funkcia *create_collage()* a pre vykreslenie jednotlivých rámov v obrázkoch bola implementovaná funkcia *draw_yolo_annotation()*, ktorej úlohou je načítať tieto anotácie, previesť ich zo relatívnych súradníc na pixely a následne ich vykresliť do vstupného obrázka vo forme červených rámčekov (bounding boxov). Detailnejší popis funkcie je nasledovný:

- Funkcia začína kontrolou, či súbor s anotáciami na zadanej ceste *labels_path* existuje. Ak súbor neexistuje, funkcia okamžite vracia pôvodný obrázok bez akýchkoľvek zmien. V opačnom prípade súbor otvorí a spracováva v ňom riadok anotácii.
- Ak riadok neobsahuje presne päť hodnôt, je považovaný za neplatný a preskočí sa. Pre platné riadky sa hodnoty načítajú a prevedú na typ float. Následne sa získajú rozmery vstupného obrázka, teda jeho šírka a výška. Pomocou týchto rozmerov sa relatívne hodnoty prevedú na absolútne hodnoty v pixeloch – napríklad $x_center \cdot width_image$ alebo $height \cdot height_image$.
- Po tomto prepočte sa vypočítajú rohové súradnice obdlžníka, ktorý reprezentuje anotovaný objekt. Výpočet vychádza zo stredu objektu a jeho šírky a výšky. Výsledné súradnice rohov (x_1, y_1, x_2, y_2) sa potom použijú na vykreslenie obdlžníka do obrázka pomocou funkcie *cv2.rectangle*, pričom rámček je červený a má hrúbku 2 pixely.
- Po spracovaní všetkých riadkov funkcia vráti obrázok s vykreslenými anotáciami.

Je potrebné poznamenať, že na podobnom princípe pracuje aj funkcia *get_yolo_annotation()*, avšak tá nevykresľuje rámčeky, ale iba zisťuje súradnice (x_1, y_1, x_2, y_2) týchto rámčekov.



Obrázok 1: Koláž 6 obrázkov so zobrazenými rámčekmi

1.1 Predspracovanie obrázkov

V kapitolách nižšie budeme pracovať taktiež aj s predspracovaním obrázkov, aby sme zvýšili ich kvalitu. požutá metóda na predspracovanie obrázkov sa nazýva Histogramové vyrovnávanie jasu (Histogram Equalization) a pre tento účel bola vytvorená funkcia *preprocess_image()*, ktorá sa skladá z nasledovných krokov:

1. **Prevod obrázka z BGR do YCrCb farebného priestoru**:OpenCV používa predvolene farebný formát BGR a YCrCb farebný priestor rozdeľuje obraz na:

   - *Y (luminancia)* – jas
   - *Cr* – rozdiel medzi červenou a jasom
   - *Cb* – rozdiel medzi modrou a jasom

   Týmto sa oddelí svetlosť od farby, čo je výhodné pre kontrastné úpravy bez skreslenia farieb.

2. **Vyrovnanie histogramu (equalizácia) jasovej zložky**: Aplikuje sa histogramové vyrovnanie len na Y kanál (0. kanál). Výsledkom je zvýšenie kontrastu – tmavé oblasti sa zosvetlia, svetlé stmavia, čo zlepší viditeľnosť detailov. Tento krok pomáha detektorom lepšie identifikovať kontúry a štruktúry, hlavne pri zle nasvietených obrázkoch.

3. **Prevod späť do BGR**: Po úprave sa obraz konvertuje späť do BGR, aby ho bolo možné použiť v ďalších OpenCV funkciách, ktoré očakávajú BGR formát.

2 Natrénované detektory

V tejto kapitole budú popísané použité už natrénované detektory, ktoré boli použité pre nájdenie tvári na databáze obrázkov. [2, 3, 4, 5]

2.1 Haar Cascade

Haar Cascade je klasický algoritmus (nazývaný aj Viola-Jones algoritmus) založený na strojovom učení, ktorý funguje na základe Haarových príznakov, čo sú jednoduché čiernobiele šablóny, ktoré zachytávajú kontrast medzi susednými oblasťami obrázka. Architektúra detektora využíva:

1. **Haarové príznaky** – extrahované z rôznych častí obrázka
2. **Integrálny obrázok** – dátová štruktúra, ktorá umožňuje rýchly výpočet Haarových príznakov
3. **AdaBoost** – algoritmus strojového učenia, ktorý vyberá najrelevantnejšie príznaky
4. **Kaskádový klasifikátor** – séria jednoduchých klasifikátorov zoradených tak, aby rýchlo odmietli nepravdepodobné oblasti a sústredili sa na tie s vyššou pravdepodobnosťou výskytu objektu

Detekcia prebieha tak, že sa 'okienko' pohybuje po obrázku a testuje sa, či obsahuje objekt (resp. tvár), pomocou vyššie uvedených klasifikátorov. OpenCV poskytuje predtrénované XML súbory (haarcascade_frontalface_default.xml), ktoré sú výsledkom tohto tréningu, ktorý prebiehal na databázach ako:

- **BioID Face Database**[9] - obsahuje obrázky s jednou aj viacerými tvárami v rôznych podmienkach
- **MIT-CMU Face Dataset**[10]
- **FERET Database**[11] - Veľká databáza tvárí, ktorá obsahuje tisíce obrázkov tvárí od rôznych osôb, vrátane variácií v osvetlení, výraze, pohľade
- **Extended Yale Face Database**[12] - obsahuje obrázky s rôznym osvetlením a výrazmi

[9] https://www.bioid.com/About/BioID-Face-Database
[10] https://vismod.media.mit.edu/vismod/classes/mas622-00/demos/facedetect.html
[11] https://www.nist.gov/programs-projects/face-recognition-feret
[12] http://vision.ucsd.edu/content/yale-face-database

Tieto databázy obsahujú stovky až tisíce obrázkov tvárí s rôznymi pózami, výrazmi a osvetlením.

Medzi výhody použitia tohto detektora môžeme zahrnúť to, že je rýchli a nenáročný, čiže funguje aj na slabšom hardvéri, čo má však za následok to, že má nízku presnosť (napr. pri zlom osvetlení alebo zložitom pozadí). Taktiež nie je robustný voči rotáciam alebo skresleniam tváre

2.2 MediaPipe

MediaPipe Face Detection je moderný detektor tvárí vyvinutý Googlom. Používa hlboké neurónové siete optimalizované pre rýchlosť a presnosť na mobilných aj desktopových zariadeniach. V prípade MediaPipe sa uvažuje taktiež aj minimálny prah (confidence threshold, v tomto prípade nastavené na 0.5) aby sa rozhodlo, či je daná detekcia platná alebo nie (vyššia hodnota = presnejšie, ale viac vynechaných tvári). Architektúra je založená na:

1. **BlazeFace** – ultraľahká KNN navrhnutá pre detekciu tvárí v reálnom čase (pôvodne pre Google MediaPipe v rámci Android/AR aplikácií)
2. **Single Shot Detector (SSD)** – architektúra typu öne-pass", kde sa detekcia aj regresia bounding boxov robí v jednom kroku
3. **Kľúčové body na tvári:** Po detekcii tváre model predikuje 6 kľúčových bodov (napr. oči, nos, ústa), čo umožňuje presnejšie sledovanie a ďalšie aplikácie.
4. **TensorFlow Lite** - optimalizácia pre výkon na mobilných zariadeniach

Tréningová databáza BlazeFace a MediaPipe boli trénované na kombinácii veľkých open-source databázach, ako napríklad:

- **WIDER FACE[13]** – extrémne variabilná databáza s viac ako 32 000 obrázkami a 393 703 anotovanými tvárami
- **Interné dátové sady Google** - neverejné

Model MediaPipe Face Detection poskytuje dve varianty:

- *model_selection=0:* pre krátke vzdialenosti (napr. selfie)
- *model_selection=1:* pre dlhší dosah a rôzne uhly

[13]http://shuoyang1213.me/WIDERFACE/

Výhodou tohto detektora je vysoká presnoť a aj to, že je veľmi rýchli aj na mobilných zariadeniach (vďaka optimalizátorom). Taktiež je robustný voči osvetleniu, rotácii a vzdialenosti tváre na obrázku. Nevýhodou môže byť prevažne to, že poterbuje viac výpočtových zdrojov, na rozdiel od Haar.

2.3 Analýza výsledkov

V tejto podkapitole popíšeme vypočítané jednotlivé metriky:

- **Presnosť** - aká časť detekcií bola správna
- **Recall** - koľko skutočných objektov bolo nájdených
- **F1-skóre** - harmonický priemer presnosti a recallu – vyvážený pohľad

pre IoU. Tieto metriky budú vypočítané pre oba detektory (Haar Cascade a MediaPipe) a taktiež budú počítané zvlásť pre databázu predspracovaných a nepredspracovaných obrázkoch. Tieto výpočty budú realizované postupom uvedeným v [6, 7].

Intersection over Union (IoU) je metrika pre výpočet miery prekrývania medzi dvoma ohraničujúcimi rámčekmi – predpokladaným bounding boxom a ground truth bounding boxom. IoU je definované ako podiel oblasti prieniku k oblasti spojenia. Tento výpočet realizuje funkcia *iou()*, ktorá dostane na vstup dva bounding boxy - A, B, pričom súradnice x_1, y_1 reprezentujú ľavý horný roh a súradnice x_2, y_2 reprezentujú pravý dolný roh. Hranica pre správnu detekciu bola stanovená na 0.7 musí platiť (IoU > 0.7). Popis výpočtu: [6]

1. Majme 2 bounding boxy A, B so súradnicami (x_1, y_1, x_2, y_2).
2. Je potrebné nájsť:
   - najväčšiu x-súradnicu z ľavých hrán $\rightarrow$ ľavý okraj prieniku: $x_A = max(x_{1_A}, x_{1_B})$
   - najväčšiu y-súradnica z horných hrán $\rightarrow$ horný okraj prieniku: $y_A = max(x_{2_A}, x_{2_B})$
   - najmenšiu x-súradnica z pravých hrán $\rightarrow$ pravý okraj prieniku: $x_B = min(x_{3_A}, x_{3_B})$
   - najmenšiu y-súradnica zo spodných hrán $\rightarrow$ spodný okraj prieniku: $y_B = min(x_{4_A}, x_{4_B})$
3. Výpočet veľkosti jednotlivých plôch:

$$|A| = max((x_2 - x_1) \cdot (y_2 - y_1)),$$
$$|B| = max((x_2 - x_1) \cdot (y_2 - y_1))$$

4. Výpočet prieniku a zjednotenia:

$$A \cap B = max((x_B - x_A) \cdot (y_B - y_A)),$$
$$A \cup B = |A| + |B| - A \cap B$$

5. Výpočet $IoU = \frac{A \cap B}{A \cup B + 1e-6}$ Ak sa boxy neprekrývajú, výsledná plocha bude nulová, preto sa tam používa aj max(0, ...). Výraz $1e-6$ reprezentuje veľmi malé číslo, ktoré bolo pridané, aby sa predišlo deleniu nulou.

Pre vyhodnotenie presnosti detektorov tvári porovnaním ich predikcií s referenčnými anotáciami vo formáte YOLO slúži funkcia *evaluate_detector()*. Výsledkom sú hodnoty metrík Presnosť, Recall a F1-skóre, ktoré kvantitatívne vyjadrujú, ako dobre detektor funguje. Vo funkcii je možné aktivovať/deaktivovať predspracovanie obrázkov (popísané v kapitole 1.1). Priebeh hodnotenia:

1. Načítanie YOLO anotácií - získa sa príslušný .txt súbor s anotáciami (ground truth)
2. Detekcia predikovaných bounding boxov jednotlivými detektormi
3. Porovnanie predikcií s anotáciami: Porovnáva sa každý predikovaný box s anotovanými (YOLO). Ak IoU > 0.7:
   - Detekcia sa považuje za správnu (TP)
   - Zapamätá sa, že tento ground truth bounding box box bol už spárovaný (aby sa nespároval znova)
   - Ak sa predikcia nespáruje s ničím, ide o False Positive (FP)
4. Spočíta, koľko anotovaných tvárí sa nepodarilo detektorom nájsť, a pripočíta ich k počtu False Negatives (FN)
5. Výpočet metrík: Tento výpočet prebieha v percentách, resp. v rozsahu od 0 po 1.
   - **Presnosť**: (koľko z detekovaných kruhov bolo správnych).
   $$\text{Presnosť} = \frac{TP}{TP + FP}$$
   - **Recall**: (koľko správnych kruhov sa podarilo detekovať).
   $$\text{Recall} = \frac{TP}{TP + FN}$$
   - **F1-skóre**: (harmonický priemer precision a recall).
   $$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Nepredspracované obrázky	Presnosť	Recall	F-1 Skóre
Haar Cascade	0.292	0.214	0.247
Mediapipe	0.351	0.151	0.211

Tabuľka 2: Tabuľka vypočítaných metrík Presnosť, Recall a F1-skóre pre detektory s nepredspracovanými obrázkami

Predspracované obrázky	Presnosť	Recall	F-1 Skóre
Haar Cascade	0.263	0.208	0.232
Mediapipe	0.353	0.146	0.206

Tabuľka 3: Tabuľka vypočítaných metrík Presnosť, Recall a F1-skóre pre detektory s predspracovanými obrázkami

Vyhodnotenie výsledkov (prebiehalo na základe najvyššieho F1-skóre):

1. *Najúspešnejší model*: Haar Cascade bez predspracovania obrázkov dosiahol F-skóre: 0.247

2. *Účinok predspracovania obrázkov*:
   - Haar Cascade: predspracovanie zhoršilo výkon ($\Delta F1 = -0.015$)
   - MediaPipe: predspracovanie zhoršilo výkon ($\Delta F1 = -0.005$)

3. *Typy obrázkov, na ktorých bolo najťažšie nájsť tvár*: Ide o obrázky, na ktorých je veľké množstvo osôb, resp. tvári a zároveň nie sú tváre v dostatočne blízkej vzdialenosti a taktiež tie, kde je tvár príliš otočená, prípadne má obrázok horšiu kvalitu

Obrázok 2: Koláž 6 obrázkov s najhoršími detekciami

3 Trénovanie detektora

Na trénovanie vlastného detektora bol použitý You Only Look Once (YOLO) detektor. Ide o rýchly a jeden z najpoužívanejších objektových detektorov, ktorý detekuje objekty v obrázku v jednom prechode siete (na rozdiel od klasických dvojfázových detektorov ako R-CNN). Architektúra YOLOv8 (yolo11n.pt v tomto prípade) je najnovšia generácia. Informácie je možné nájsť v [8]. Jeho hlavná filozofia je:

- Naraz (v jednom prechode siete) predpovedá kde sú objekty aj čo sú zač (klasifikácia + lokalizácia).
- Funguje ako single-shot detektor: žiadne dvojité prehľadávanie obrázku (ako pri RCNN), ale všetko spraví v jednom kroku.

Základné kroky YOLO detektora sú:

1. Rozdelí obrázok na mriežku (grid cells), napr. 7x7 alebo 13x13.
2. Každá bunka predpovedá:
    - Kde by sa mohol nachádzať objekt (bounding box: súradnice x, y, šírka w, výška h),
    - Pravdepodobnosť, že v tejto bunke je objekt (confidence score),
    - Klasifikáciu objektu (class probabilities, napr. "tvár"vs. "nič").
3. Výstup siete je teda matica so súradnicami + pravdepodobnosťami pre každú bunku.

Popis konkrétnej použitej architektúry v tomto zadaní (YOLO v8):

1. **Backbone (extrakcia príznakov):**
    - CNN sieť ako CSPDarknet[14] alebo iný ľahký model (yolov8n je veľmi malý a rýchly model).
    - Extrahuje vlastnosti ako hrany, tvary, štruktúry z obrázka.
2. **Neck (kombinácia viacerých mierok):**
    - Feature Pyramid Network (FPN)[15] alebo Path Aggregation Network (PAN)[16].
    - Umožňuje detekovať objekty rôznych veľkostí (malé aj veľké tváre).

[14] Cieľom je znížiť redukciu informácií a zrýchliť tréning + znížiť náročnosť siete

[15] Kombinuje príznaky (features) z rôznych hĺbok siete a vyťahuje informácie o objektoch rôznych veľkostí.

[16] Vylepšenie FPN, nejde len z hora nadol (ako FPN), ale aj zdola nahor – lepšie prenáša informácie.

3. **Head (predikcia výsledkov):**
    - Predikcia bounding boxov, confidence score, a tried.
    - Namiesto pevného počtu boxov ako v YOLOv1/2 sa používajú anchor-free predikcie (v novších verziách).

Hlavnými výhodami tohto modelu sú predovšetkým to, že je extrémne rýchli, naraz sa učí klasifikovať aj lokalizovať (end-to-end učenie) a jeho novšie verzie majú lepšie presnosti a optimalizácie.

Popis implementácie tohto modelu je nasledovný:

1. **Použitie funkcií:** (funkcie *iou(), get_yolo_annotation() a evaluate_model()*, ktoré boli popísané v kapitolách 1 a 2.3, pričom v prípade *evaluate_model()* nastali drobné zmeny - pridanie predikcie z YOLO modelu, odstránenie možnosti predspracovania obrázkov a pridanie TN(žiadny reálna tvár + žiadna predikcia tváre)):
    - *split_dataset()*: táto funkcia rozdeľuje dataset na množiny (podadresáre) pre trénovanie, testovanie a validáciu a zároveň správne kopíruje aj obrázky aj YOLO anotácie k ním.
    - *train_ratio=0.7*: 70 % všetkých obrázkov a labelov ide na tréning modelu (na učenie).
    - *val_ratio=0.15*: 15 % obrázkov a labelov ide na validáciu modelu (na kontrolu počas tréningu).
    - *test_ratio*: (automaticky dopočítané), zvyšok (15 %) na testovanie (hodnotenie úplne na konci).
2. **Vytvorenie súboru** *yolo_data.yaml*, ktorý obsahuje nasledovné nastavenia pre YOLO detektor:
    - *path: YOLO*
    - *train: images/train* - Hlavná cesta k datasetu (kde sú uložené podpriečinky images/train, labels/train, atď.). V tomto prípade je to zložka YOLO/
    - *val: images/val* - Cesta k validačným obrázkom (YOLO/images/val).
    - *test: images/test* - Cesta k testovacím obrázkom (YOLO/images/test).
    - *names:* - Definícia názvov tried.
    - *0: face* - Trieda číslo 0 sa volá face (teda model sa učí detegovať tváre).

3. **Parametre pre trénovanie modelu:**
    - Vytvorenie inštancie YOLO modelu ("yolo11n.pt"je predtrénovaný checkpoint modelu, pričom ide o zmenšenú verziu (n = nano verzia YOLO), rýchly, ale menej presný – vhodný napr. na testovanie alebo rýchle experiment). Taktiež je potrebné zadať platnú cestu k datasetu a názvom tried (pomocou cesty k YAML súboru, ktorý bol predtým vytvorený).
    - *epochs=10:* Model prejde celý tréningový dataset 10-krát (10 epôch).
    - *imgsz=640:* Veľkosť obrázkov na vstupe do siete bude 640×640 pixelov.
    - *batch=32:* Na každom kroku tréningu použiješ 32 obrázkov naraz (Batch size 32).
4. **Vyhodnotenie modelu:** V tejto časti prebieha vyhodnotenie modelu pomocou funkcie *evaluate_detector()*, pričom sú vypísané aj jednotlivé metriky (Presnosť, Recall, F1-skóre) a aj konfúzna matica obsahujúca (TP, FP, TN, FN). V tejto časti sa taktiež vykresľuje aj graf úspešnosti jednotlivých detektorov (Haar Cascade, MediaPipe, YOLO) bez predspracovania obrázkov, pričom dáta pre Haar Cascade a MediaPipe sú vložené z predošlých výpočtov pre tieto modely (kapitola 2.3).

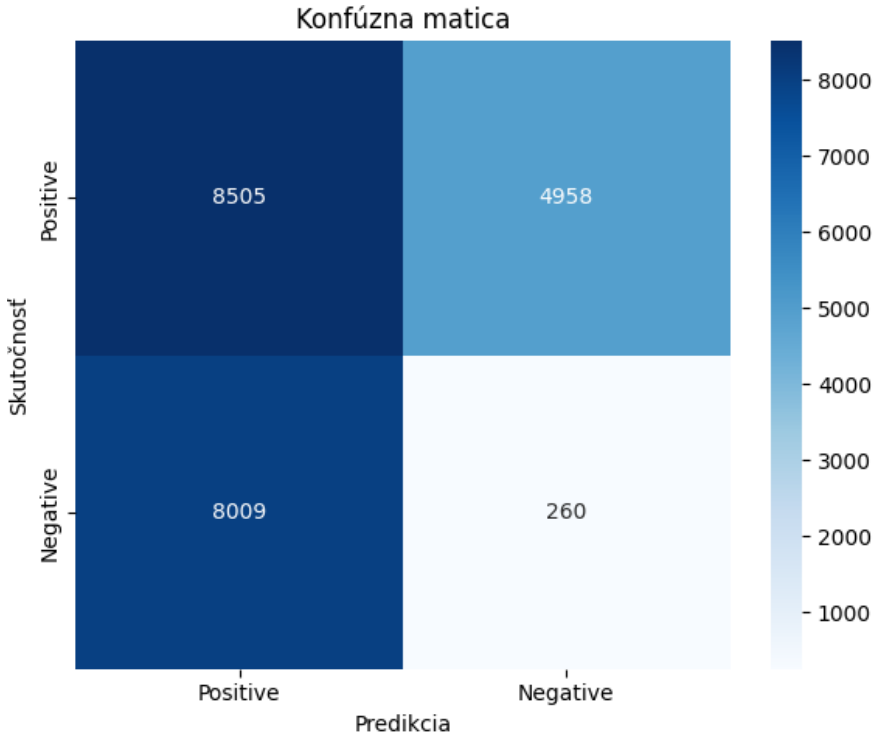
Taktiež je potrebné poznamenať, že kroky 2 - 4 sa vykonávajú vo funkcii *main()*, aby bolo možné na OS Windows naštartovať multiprocessing, pretože:

- Na Windows operačnom systéme funguje multiprocessing (viacvláknové spúšťanie) inak ako na Linuxe.
- Pri spostení viac procesov (čo robí napríklad YOLO počas tréningu), Windows potrebuje špeciálne vedieť, ktorú časť kódu môže spustiť v novom procese.
- Bez freeze_support() by vznikli chyby alebo nekonečné cykli (procesy by sa znova a znova spúšťali).
- multiprocessing.freeze_support() zabezpečí, že:
    - Python bezpečne odštartuje nové procesy,
    - program sa nebude cykliť alebo padnúť,
    - program bude fungovať aj keď ho neskôr prekompiluješ do .exe súboru.

4 Vyhodnotenie a porovnanie výsledkov

Pre tento model boli vypočítané pomocou vyššie popísaných funkcií nasledovné metriky, pričom uvažujeme Iou prahovú hodnotu 0.7:

- **Presnosť:** 0.515
- **Recall:** 0.632
- **F1-skóre**: 0.567



Obrázok 3: Konfúzna matica YOLO modelu

Na základe konfúznej matice na obrázku 3 je možné povedať, že obsahuje:

- Veľa TP: Model dokázal správne nájsť veľa tvárí (8505 prípadov).
- Veľa FP: Ale zároveň aj často falošne detegoval tam, kde nemal (8009 prípadov).
- Veľa FN: Model prehliadol 4958 skutočných tvárí (neodhalil ich).
- Málo TN: Správne ignorovaných obrázkov bez tvárí je veľmi málo (len 260 prípadov).

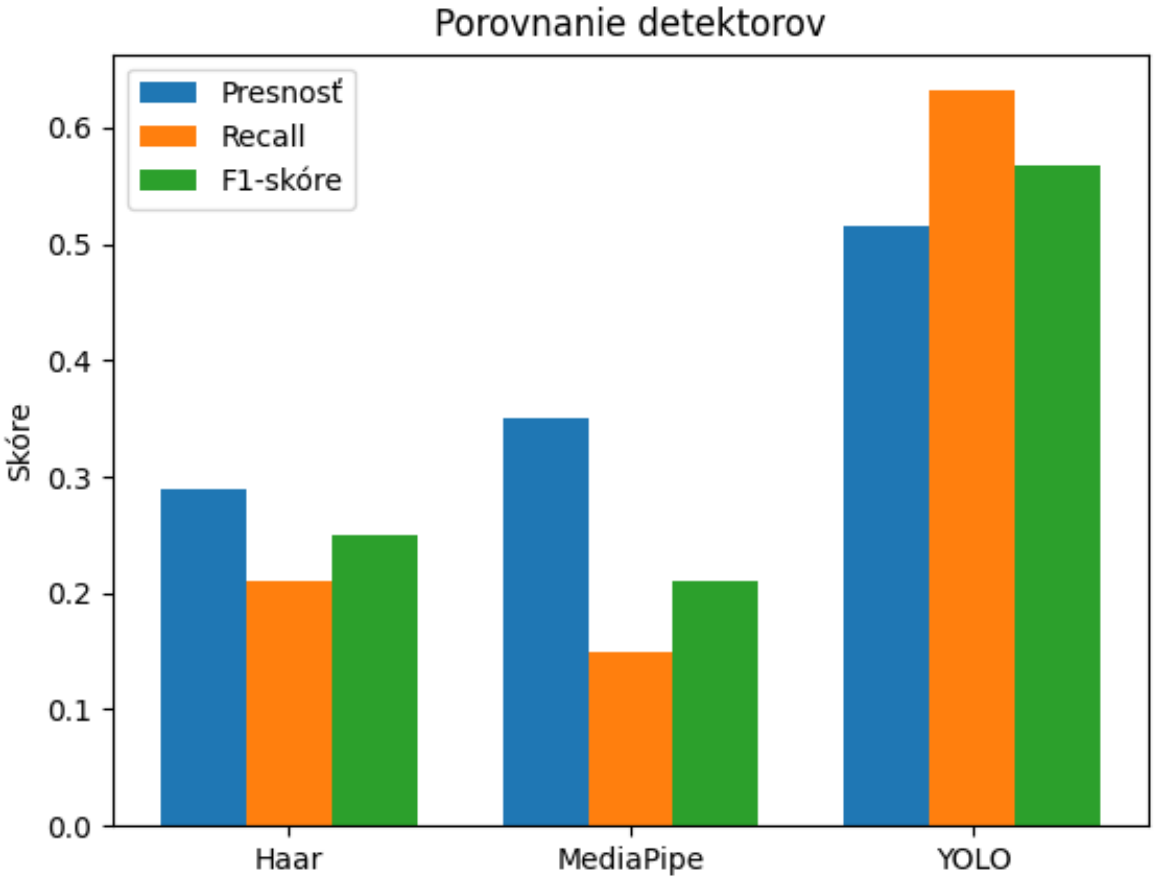
Model je veľmi aktívny v detekcii, avšak s často sa však mýli - obsahuje veľa falošných detekcií (FP) aj veľa prehliadnutých (FN).

Vzhľadom na to, že v kapitole 2.3 je možné vidieť, že nepredspracované modely dosahujú vyššie F1-skóre, nižšie uvedieme tabuľku pre porovnanie modelov Haar Cascade, Mediapipe a YOLO (bez predspracovania obrázkov).

	Presnosť	Recall	F1-skóre
Haar Cascade	0.292	0.214	0.247
Mediapipe	0.351	0.151	0.211
YOLO	0.515	0.632	0.567

Tabuľka 4: Porovnanie vypočítaných metrík Presnosť, Recall a F1-skóre pre všetky 3 detektory bez predspracovania obrázkov

Na tabuľke 4 je možné vidieť, že najlepšie výsledky pre jednotlivé metriky (Presnosť, Recall a F1-skóre) dosahuje natrénovaný model YOLO a grafickú vizualizáciu tejto tabuľky je možné vidieť na obrázku 4.



Obrázok 4: Grafické znázornenie úspešnosti jednotlivých modelov

Záver

Cieľom zadania bolo implementovať program v programovacom jazyku Python, ktorý detekuje tváre na obrázkoch, pričom boli použité dva natrénovaný modely (Haar Cascade a MediaPipe), kde boli vypočítané jednotlivé metriky (F1-skóre, Presnosť a Recall) pre obrázky s aplikovaným predspracovaním a bez predspracovania (lepšie F1-skóre bolo dosiahnuté pri obrázkov bez predspracovania). Taktiež bola vytvorená koláž 6 obrázkov podľa súborov s YOLO anotáciami a koláž 6 najhorších obrázkov obsahujpúcich najväčšie nepresnosti. Taktiež bolo vyhodnotené, že model Haar Cascade bez predspracovaných obrázkov dosiahol najlepšie F1-skóre a bolo vypočítané, o koľko sa F1-skóre zmenilo pri použití predspracovania obrázkov.

V ďalšej časti zadania bol natrénovaný vlastný detektor (YOLO), pri ktorom bolo možné po výpočte jednotlivých metrík usúdiť, že tento model dosahuje presnejšie výsledky (na základe porovnania predošlých dvoch modelov - Haar Cascade a MediaPipe). Bola taktiež vykreslená konfúzna matica pre TP, FP, TN, FN a taktiež aj porovnanie výsledkov pre jednotlivé metriky (vo forme tabuľky aj grafu).

Zoznam použitej literatúry

1. HALVORSEN, H. P. *Python Programming*. Dostupné tiež z: https://www.halvorsen.blog/documents/programming/python/resources/Python%20Programming.pdf.
2. GOOGLE. *Face detection guide*. Dostupné tiež z: https://ai.google.dev/edge/mediapipe/solutions/vision/face_detector.
3. BAZAREVSKY, V., KARTYNNIK, Y., VAKUNOV, A., RAVEENDRAN, K. a GRUNDMANN, M. *BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs*. arXiv. Dostupné tiež z: https://arxiv.org/pdf/1907.05047v1.
4. JAISWAL, Abhishek. *Face Detection using Haar-Cascade using Python*. Analytics Vidhya, 2024. Dostupné tiež z: https://www.analyticsvidhya.com/blog/2022/10/face-detection-using-haar-cascade-using-python/.
5. VIOLA, P. a JONES, M. *Rapid Object Detection using a Boosted Cascade of Simple Features*. Pittsburgh: Carnegie Mellon University, 2001. Dostupné tiež z: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf.
6. SHAH, D. *ntersection over Union (IoU): Definition, Calculation, Code*. 7Labs. Dostupné tiež z: https://www.v7labs.com/blog/intersection-over-union-guide.
7. KASHYAP, P. *Understanding Precision, Recall, and F1 Score Metrics*. Medium, 2024. Dostupné tiež z: https://medium.com/@piyushkashyap045/understanding-precision-recall-and-f1-score-metrics-ea219b908093.
8. *Ultralytics YOLO Documentation*. ULTRALYTICS, 2023. Dostupné tiež z: https://docs.ultralytics.com.