

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

**Fakulta elektrotechniky a informatiky**

Evidenčné číslo: FEI-104376-111119

# **Autentifikácia emócií operátora na základe výrazu tváre**

**Diplomová práca**

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

**Fakulta elektrotechniky a informatiky**

Evidenčné číslo: FEI-104376-111119

# **Autentifikácia emócií operátora na základe výrazu tváre**

**Diplomová práca**

Študijný program:	Robotika a kybernetika
Študijný odbor:	kybernetika
Školiace pracovisko:	Ústav robotiky a kybernetiky, FEI STU v Bratislave
Školiteľ:	prof. Ing. Jarmila Pavlovičová, PhD.
Konzultant:	Ing. Michal Tölgyessy, PhD.



## ZADANIE DIPLOMOVEJ PRÁCE

Študent: **Bc. Maroš Kocúr**  
ID študenta: 111119  
Študijný program: robotika a kybernetika  
Študijný odbor: kybernetika  
Vedúci práce: Ing. Michal Tölgyessy, PhD.  
Vedúci pracoviska: prof. Ing. František Duchoň, PhD.  
Miesto vypracovania: Ústav robotiky a kybernetiky

Názov práce: **Autentifikácia emócií operátora na základe výrazu tváre**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Pri interakcii človeka s robotickým systémom je dôležité, aby robot vedel aj to, v akom emočnom rozpoložení sa operátor nachádza. Cieľom práce je vytvoriť modul, ktorý robotu takúto informáciu poskytne. Predpokladá sa využitie RGB kamery, ale je možnosť využiť aj RGB-D kameru.

Úlohy práce:

1. Analyzujte existujúce metódy analýzy emócií na základe výrazu tváre.
2. Naštudujte princípy tvorby biometrických modelov tváre a metódy detekcie a rozpoznávania tváre.
3. Navrhnite a implementujte systém pre identifikáciu emócií operátora na základe jeho tváre.
4. Testujte a validujte systém na simulovaných aj reálnych dátach.
6. Vytvorte ROS2 balík pre daný systém.
7. Vyhodnoťte experimenty a dosiahnuté výsledky.

Termín odovzdania diplomovej práce: 16. 05. 2025  
Dátum schválenia zadania diplomovej práce: 16. 10. 2024  
Zadanie diplomovej práce schválil: prof. Ing. Jarmila Pavlovičová, PhD. – garantka študijného programu

## **Podakovanie**

Na tomto mieste by som sa rád poďakoval svojmu školiťovi Ing. Michalovi Tölgyessymu, PhD. za odborné vedenie, cenné rady a podporu počas celej doby riešenia diplomovej práce. Poďakovanie patrí aj Ústavu robotiky a kybernetiky FEI STU za vytvorenie kvalitného výskumného prostredia a možnosť pracovať na tejto téme v rámci moderného robotického pracoviska.

# Abstrakt

Interakcia človeka s robotom v dynamickom prostredí si čoraz viac vyžaduje pochopenie emocionálneho stavu operátora s cieľom optimalizovať komunikáciu a rozhodovacie procesy. Cieľom tejto práce je navrhnúť a implementovať modul, ktorý poskytuje robotickému systému emocionálnu spätnú väzbu a umožňuje mu zisťovať výrazy tváre operátora a odvodzovať jeho emocionálne stavy. S využitím kamery RGB s možnosťou integrácie kamery RGB-D bude systém využívať biometrické modely tváre a techniky rozpoznávania tváre na identifikáciu emócií v reálnom čase. Medzi kľúčové úlohy patrí analýza súčasných metód detekcie emócií výrazu tváre, štúdium princípov tvorby biometrických modelov tváre a implementácia robustného systému na detekciu emócií. Systém bude overený prostredníctvom testovania na simulovaných aj reálnych súboroch údajov. Okrem toho bude vyvinutý balík ROS2, ktorý zabezpečí bezproblémovú integráciu v rámci robotických systémov. Výsledky budú kriticky posúdené prostredníctvom experimentov s cieľom zabezpečiť presnosť a efektívnosť výkonu v reálnych aplikáciách.

## Kľúčové slová

RGB kamera, neurónová sieť, ROS, autentifikácia, emócie, výrazy tváre

# Abstract

Human-robot interaction in dynamic environments increasingly requires an understanding of the operator's emotional state to optimize communication and decision-making processes. This work aims to design and implement a module that provides emotional feedback to a robotic system, enabling it to detect the operator's facial expressions and infer emotional states. Leveraging an RGB camera, with the option to integrate an RGB-D camera, the system will employ biometric facial models and facial recognition techniques to identify emotions in real-time. Key tasks include analyzing current facial expression emotion detection methods, studying the principles of facial biometric model creation, and implementing a robust system for emotion detection. The system will be validated through testing on both simulated and real datasets. Additionally, a ROS2 package will be developed to ensure seamless integration within robotic systems. The outcomes will be critically assessed through experiments to ensure accuracy and performance efficiency in real-world applications.

## Keywords

RGB camera, neural network, ROS, authentication, emotions, facial expressions

# Obsah

<b>Úvod</b>	<b>12</b>
<b>1 Úvod</b>	<b>13</b>
1.1 Motivácia . . . . .	14
1.2 Ciele práce . . . . .	15
<b>2 Teoretické základy</b>	<b>17</b>
2.1 Emócie a ich prejav . . . . .	17
2.1.1 Univerzálne emócie . . . . .	17
2.1.2 Kultúrne rozdiely v prejave emócií . . . . .	18
2.1.3 Výrazy tváre ako indikátory emócií . . . . .	18
2.2 Analýza obrazu . . . . .	19
2.2.1 Detekcia tváre . . . . .	19
2.2.2 Extrakcia príznakov . . . . .	20
2.2.3 Klasifikácia . . . . .	20
2.3 Biometria . . . . .	21
2.3.1 Princípy biometrických systémov . . . . .	21
2.3.2 Identifikácia vs. verifikácia . . . . .	22
2.3.3 Výhody a výzvy biometrie tváre . . . . .	22
<b>3 Existujúce metódy analýzy emócií</b>	<b>23</b>
3.1 Ručne značenie . . . . .	23
3.2 Automatická analýza emócií . . . . .	23
3.2.1 Konvolučné neurónové siete . . . . .	24
3.2.2 Typy vhodných neurónových sietí . . . . .	24
3.2.3 Príklady použitia počítačového videnia . . . . .	24
<b>4 Návrh riešenia</b>	<b>26</b>
4.1 Porovnanie vybraných metód . . . . .	26
4.2 Architektúra systému ResEmoteNet . . . . .	27
4.2.1 Hlavné komponenty architektúry . . . . .	27
4.2.2 Prínosy architektúry ResEmoteNet . . . . .	28
4.2.3 Experimentálne výsledky . . . . .	28
4.2.4 Schéma architektúry systému . . . . .	28
4.3 Výber dát . . . . .	29
4.3.1 Dôvody výberu RAF-DB . . . . .	29
4.3.2 Porovnanie s inými datasetmi . . . . .	30

4.3.3	Prínos pre prácu . . . . .	30
4.3.4	Príprava dát . . . . .	30
4.4	Extrakcia príznakov . . . . .	31
4.4.1	Metódy extrakcie príznakov . . . . .	31
4.4.2	Porovnanie metód . . . . .	31
4.4.3	Integrácia príznakov v ResEmoteNet . . . . .	31
4.4.4	Experimentálne výsledky . . . . .	33
4.5	Klasifikácia . . . . .	33
4.5.1	Metódy klasifikácie . . . . .	33
4.5.2	Porovnanie metód klasifikácie . . . . .	34
4.5.3	Implementácia klasifikátora v systéme ResEmoteNet . . . . .	34
4.5.4	Výsledky klasifikácie na datasete RAF-DB . . . . .	34
4.6	Výber hyperparametrov . . . . .	35
4.6.1	Kľúčové hyperparametre a ich úloha . . . . .	35
4.6.2	Porovnanie vplyvu hyperparametrov . . . . .	36
4.6.3	Optimalizačné stratégie . . . . .	36
4.6.4	Augmentačné parametre . . . . .	36
4.6.5	Záver . . . . .	36
4.7	Integrácia do robotického pracoviska COCOHRIP . . . . .	37
<b>5</b>	<b>Implementácia riešenia</b>	<b>38</b>
5.1	Vývojové prostredie a infraštruktúra . . . . .	38
5.2	Trénovanie modelu ResEmoteNet . . . . .	38
5.2.1	Organizácia vývojového prostredia . . . . .	38
5.2.2	Načítavanie a príprava dát . . . . .	38
5.2.3	Augmentácia dát . . . . .	38
5.2.4	Architektúra modelu . . . . .	39
5.2.5	Trénovací proces . . . . .	39
5.2.6	Ukladanie checkpointov . . . . .	39
5.2.7	Vizualizácia výsledkov . . . . .	39
5.2.8	Optimalizačné výzvy . . . . .	40
5.3	Integrácia do ROS2 ekosystému . . . . .	40
5.4	Webová vizualizácia pomocou Flask . . . . .	40
5.5	Validácia na robotickom pracovisku COCOHRIP . . . . .	40
<b>6</b>	<b>Exprerimenty a vyhodnotenie</b>	<b>41</b>
6.1	Úvod . . . . .	41
6.2	Testovanie na reálnych dátach . . . . .	41



6.2.1	Priebeh testovania . . . . .	41
6.2.2	Problémové emócie: strach a smútok . . . . .	42
6.2.3	Pozorovania . . . . .	42
6.2.4	Zhrnutie . . . . .	42
6.3	Dotazníkový experiment s ľuďmi . . . . .	42
6.3.1	Zber a spracovanie odpovedí . . . . .	43
6.3.2	Najčastejšie chyby ľudí . . . . .	43
6.3.3	Porovnanie s modelom . . . . .	43
6.4	Vizualizácia výsledkov . . . . .	43
6.4.1	Zhrnutie . . . . .	44
6.5	Testovanie s rôznymi zariadeniami a podmienkami . . . . .	44
6.5.1	Výsledky na rôznych zariadeniach . . . . .	44
6.5.2	Vplyv svetelných podmienok . . . . .	45
6.5.3	Zhrnutie . . . . .	45
6.6	Porovnanie s výsledkami modelu . . . . .	45
6.7	Analýza výsledkov . . . . .	46
6.8	Zhrnutie experimentov . . . . .	46
6.8.1	Silné stránky systému . . . . .	46
6.8.2	Slabé stránky systému . . . . .	47
6.8.3	Odporúčané metodológie a nástroje pre ďalší výskum . . . . .	47
<b>7</b>	<b>Záver</b>	<b>48</b>
7.1	Zhodnotenie práce . . . . .	48
7.2	Obmedzenia práce . . . . .	48
7.3	Budúce smerovanie . . . . .	49
	<b>Záver</b>	<b>50</b>
	<b>Literatúra</b>	<b>51</b>
	Použitie nástrojov umelej inteligencie . . . . .	52
<b>A</b>	<b>Zdrojový kód pre rozpoznávanie emócií</b>	<b>53</b>
<b>B</b>	<b>Zdrojový kód C++</b>	<b>62</b>
<b>C</b>	<b>Zdrojový kód pre streamovanie snímok z kamier</b>	<b>64</b>
<b>D</b>	<b>Zdrojový kód pre trenovanie modelu</b>	<b>66</b>
<b>E</b>	<b>Zdrojový kód pre Docker</b>	<b>71</b>

<b>F</b>	<b>Používateľský manuál</b>	<b>74</b>
F.1	Návod na tréovanie modelu . . . . .	74
F.1.1	Požiadavky . . . . .	74
F.1.2	Postup tréovania . . . . .	74
F.2	Návod na testovanie s ROS2 . . . . .	75
F.2.1	Požiadavky . . . . .	75
F.2.2	Inštalácia a zostavenie . . . . .	75
F.2.3	Spustenie systému . . . . .	76
F.2.4	Riešenie problémov . . . . .	77
F.2.5	Ukončenie systému . . . . .	77

## Zoznam značiek a skratiek

<b>CNN</b>	Convolutional neural network, v slovenčine <i>konvolučná neurónová sieť</i>
<b>DCNN</b>	Deep convolutional neural network, v slovenčine <i>hlboká konvolučná neurónová sieť</i>
<b>LSTM</b>	Long short-term memory, v slovenčine <i>dlhodobá krátkodobá pamäť</i>
<b>RNN</b>	Recurrent neural network, v slovenčine <i>rekurentná neurónová sieť</i>
<b>ROS</b>	Robot Operating system, v slovenčine <i>robotický operačný systém</i>

## Zoznam výpisov kódov

1	Python skript pre rozpoznávanie emócií . . . . .	53
2	Implementácia rozpoznávania emócií v C++ . . . . .	62
3	CMakeLists.txt pre projekt rozpoznávania emócií . . . . .	62
4	Python publisher snímok z kamier . . . . .	64
5	Python skript použitý na trenovanie modelu . . . . .	66
6	Dockerfile pre rozpoznávanie emócií . . . . .	71
7	docker-compose.yml pre systém rozpoznávania emócií . . . . .	72

# Úvod

Interakcia človeka s robotickým systémom si vyžaduje nielen technickú presnosť a spoľahlivosť, ale čoraz častejšie aj schopnosť robota porozumieť svojmu ľudskému partnerovi na hlbšej, emocionálnej úrovni. V mnohých oblastiach nasadenia – od priemyslu cez zdravotníctvo až po domáce prostredie – sa ukazuje ako výhodné, ak robot dokáže prispôbiť svoje správanie aktuálnemu emočnému stavu operátora. Takáto schopnosť môže zvýšiť efektivitu spolupráce, znížiť počet chýb a prispieť k celkovo prirodzenejšej a intuitívnejšej interakcii.

Táto diplomová práca sa zameriava na vytvorenie systému, ktorý bude schopný identifikovať emocionálne rozpoloženie človeka prostredníctvom analýzy výrazu tváre. Zámerom je, aby výsledný modul poskytoval robotickému systému relevantné informácie o emóciách používateľa v reálnom čase. Téma prepája oblasti počítačového videnia, umelej inteligencie a robotiky a reflektuje narastajúci význam emocionálnej inteligencie v moderných technológiách.

# 1 Úvod

S rozvojom umelej inteligencie a strojového učenia sa otvárajú nové možnosti pre interakciu medzi človekom a strojom. Jednou z najdôležitejších oblastí výskumu je rozpoznávanie emócií na základe výrazu tváre, ktoré umožňuje strojom porozumieť emocionálnemu stavu používateľa. V kontexte robotických systémov je dôležité, aby roboty boli schopné rozoznať emócie človeka, čo môže zlepšiť komunikáciu, kooperáciu a bezpečnosť pri spoločnej práci [1].

Psychológ Paul Ekman pomenoval šesť základných emócií - šťastie, smútok, hnev, strach, prekvapenie a znechutenie - ktoré sú často základom pre zavedené kategórie emócií pri rozpoznávaní tváre. Výber týchto emócií bol založený na ich univerzálnom rozpoznávaní a pozorovaní naprieč kultúrami, čo ich predurčuje na použitie v systémoch rozpoznávania tváre.

Podľa Ekmanovho výskumu sa tieto pocity odrážajú v konkrétnych výrazoch tváre, ktoré dokážu automaticky rozpoznať ľudia zo všetkých kultúrnych prostredí. Jeho práca vytvorila fundamentálny základ pre strojové učenie a psychológiu, najmä pri vytváraní systémov, ktoré dokážu dešifrovať výrazy tváre na určenie emocionálneho stavu jednotlivca.

V záujme konzistentnosti a presnosti v aplikáciách, ako je zdravotníctvo, robotika a služby zákazníkom, je možné do systémov rozpoznávania tváre zahrnúť konzistentnú metódu analýzy emócií. Pochopenie úrovne spokojnosti alebo podráždenia používateľa môže napríklad pomôcť upraviť reakcie systému a zlepšiť výsledky interakcií [2].

Emócie zohrávajú dôležitú úlohu v procese rozhodovania, riadenia a interakcie. Schopnosť robotického systému porozumieť emocionálnemu stavu používateľa umožňuje jeho prispôbenie konkrétnym podmienkam a potrebám operátora. Napríklad v priemysle môžu robotické systémy identifikovať stres alebo únavu operátora, čím prispievajú k zvýšeniu bezpečnosti a efektivity. Okrem toho, v oblasti zdravotnej starostlivosti môže rozpoznávanie emócií pomôcť monitorovať psychický stav pacientov a prispieť k ich lepšej starostlivosti [3].

Rozpoznávanie emócií je možné dosiahnuť rôznymi metódami, ktoré zahŕňajú spracovanie obrazu, analýzu textu, reč a gestá. Výraz tváre je však najvýznamnejším a najpresnejším indikátorom emócií, pretože vyjadruje okamžitý emocionálny stav človeka. Emócie, ako sú šťastie, smútok, hnev alebo prekvapenie, sú viditeľné prostredníctvom zmien vo svaloch tváre, ktoré sú merateľné a analyzovateľné pomocou technológií strojového učenia, najmä pomocou hlbokých neurónových sietí (Convolutional neural network, v slovenčine *konvolučná neurónová sieť* (CNN)) [1].

SúčasnÉ metódy na rozpoznávanie emócií zahŕňajú viacero prístupov. Tradičné prístupy, ako napríklad metódy založené na geometrických črtách a textúrach, boli doplnené modernými metódami založenými na hlbokom učení, ktoré dosahujú vysokú presnosť. Neurónové siete sú schopné automaticky extrahovať črty tváre bez potreby manuálneho zásahu, čo výrazne zvyšuje efektivitu systému. Tieto pokročilé modely dosahujú vysokú mieru úspešnosti v rôznych aplikáciách, ako sú zdravotná starostlivosť, priemyselná automatizácia alebo monitorovanie únavy vodičov [3] [4].

## 1.1 Motivácia

V súčasnej dobe, keď technológie prenikajú do všetkých aspektov ľudského života, je nevyhnutné, aby robotické systémy disponovali nielen kognitívnymi schopnosťami, ale aj emocionálnou inteligenciou. Táto schopnosť umožňuje robotom identifikovať, hodnotiť a reagovať na emócie ľudí, čím sa zvyšuje efektivita a prirodzenosť interakcie medzi človekom a strojom. Emocionálna inteligencia zahŕňa schopnosť rozpoznávať vlastné emócie, emócie iných, a adekvátne na ne reagovať.

V oblasti interakcie človeka s robotom (Human-Robot Interaction, HRI) je rozpoznávanie emócií kľúčové pre vytvorenie intuitívnej a efektívnej spolupráce. Roboty schopné interpretovať emocionálny stav operátora môžu prispôbiť svoje správanie aktuálnej situácii, čo je obzvlášť dôležité v oblastiach, kde je potrebná vysoká miera spolupráce a dôvery, ako napríklad v zdravotníctve, vzdelávaní či v priemyselnej výrobe. Napríklad v zdravotníctve môžu roboty asistovať pacientom s emocionálnou podporou, monitorovať ich psychický stav a poskytovať adekvátne reakcie na základe rozpoznaných emócií [5].

Aktuálne trendy v oblasti robotiky smerujú k vývoju systémov, ktoré dokážu nielen vykonávať predprogramované úlohy, ale aj adaptovať sa na dynamické prostredie a emocionálne potreby používateľov. Výskum v oblasti afektívnej robotiky sa zameriava na integráciu emocionálnych modelov do robotických systémov, čo umožňuje lepšie porozumenie a predikciu ľudského správania. Napriek pokroku v tejto oblasti existujú výzvy spojené s presnosťou rozpoznávania emócií, interpretáciou komplexných emocionálnych stavov a zabezpečením etického využitia týchto technológií [5].

Implementácia systému na identifikáciu emócií operátora na základe výrazu tváre predstavuje významný krok k zlepšeniu interakcie medzi človekom a robotom. Takýto systém môže prispieť k vyššej efektívite, bezpečnosti a spokojnosti používateľov pri práci s robotickými systémami. Napríklad v priemyselných aplikáciách môže robot upraviť svoj pracovný rytmus alebo poskytnúť upozornenie v prípade, že deteguje stres alebo únavu operátora, čím sa predchádza možným chybám alebo nehodám.

Celkovo je motivácia pre vývoj takýchto systémov založená na snahe o vytvorenie robotických asistentov, ktorí sú schopní nielen technicky podporovať človeka, ale aj

empaticky reagovať na jeho emocionálne potreby, čím sa zvyšuje kvalita a efektivita ich vzájomnej spolupráce. Motiváciou pre rozpoznávanie emócií tváre je jeho potenciál zlepšiť interakciu medzi človekom a počítačom, zlepšiť monitorovanie duševného zdravia a vytvoriť adaptívne systémy pre rôzne oblasti, ako je vzdelávanie, marketing a robotika [5].

## 1.2 Ciele práce

Cieľom tejto práce je navrhnúť, implementovať a otestovať systém, ktorý bude schopný identifikovať emocionálny stav človeka na základe výrazu jeho tváre v reálnom čase. Takýto systém má potenciál výrazne zlepšiť interakciu človeka s robotickým systémom, najmä v prostredí, kde je dôležitá adaptácia robota na aktuálny psychický stav operátora.

Hlavný dôraz je kladený na vytvorenie robustného a presného modulu na rozpoznávanie emócií, ktorý bude možné integrovať do existujúcich robotických platforiem v prostredí ROS2 (Robot Operating System 2). V práci sa uvažuje s využitím RGB kamery, avšak je otvorený priestor aj pre rozšírenie o RGB-D kamery (napr. RealSense alebo ZED), ktoré umožňujú snímanie hĺbky a tým zvyšujú presnosť 3D analýzy tváre.

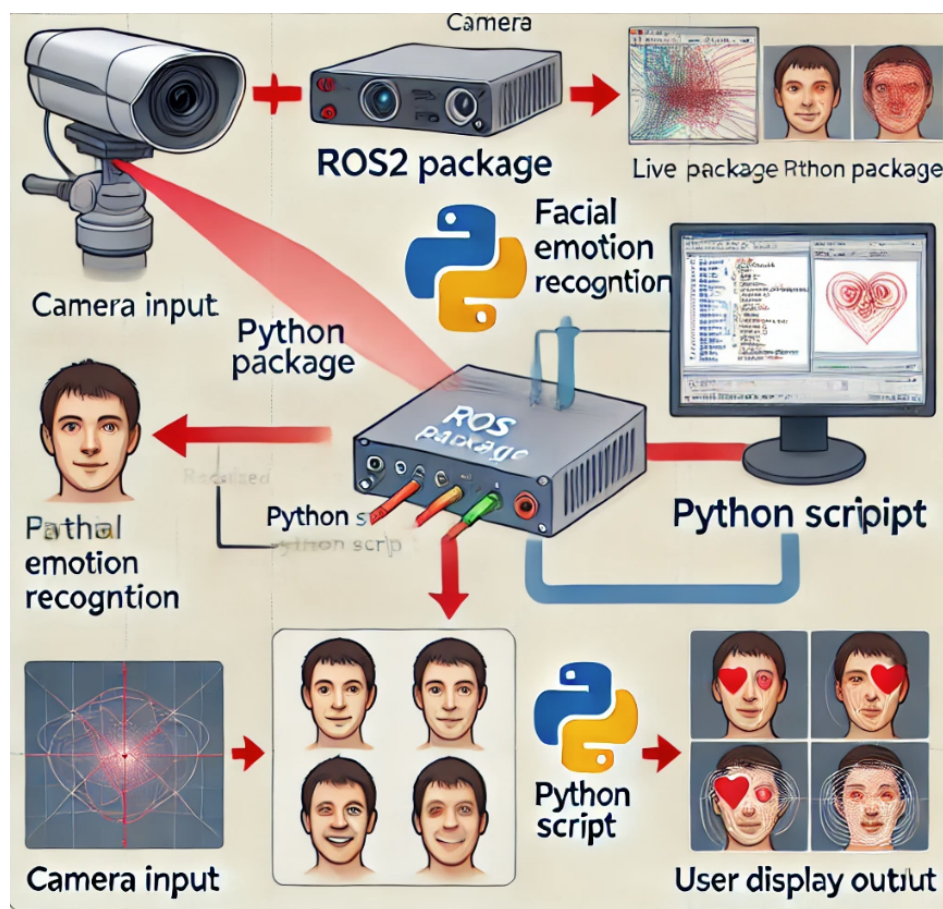
Práca sa zameriava na splnenie nasledujúcich čiastkových cieľov:

- Analýza existujúcich metód rozpoznávania emócií na základe výrazu tváre – preskúmanie prístupov založených na tradičnom strojovom učení (SVM, HOG + LBP), ako aj moderných hlbokých neurónových sietí (CNN, ResNet, SE-ResNet, atď.)
- Štúdium biometrických modelov tváre a techník detekcie tváre – výskum metód ako Haar Cascade, MTCNN, Dlib a OpenCV moduly na detekciu a orezanie tváre.
- Návrh architektúry systému na rozpoznávanie emócií – vybudovanie modelu (napr. ResEmoteNet) využívajúceho konvolučné siete a reziduálne bloky, schopného klasifikovať emócie ako radosť, smútok, hnev, prekvapenie, odpor a strach.
- Tréning a testovanie systému na open-source databázach – použitie datasetov ako FER2013, RAF-DB, AffectNet a KDEF na overenie presnosti a generalizácie modelu.
- Vytvorenie ROS2 balíka – implementácia systému do samostatného ROS2 balíka, ktorý poskytne výstup emocionálneho stavu ako ROS2 téma (napr. `emotion_topic`).



- Validácia a experimentálne vyhodnotenie systému – testovanie modelu na reálnych aj simulovaných dátach (napr. v Gazebo alebo RViz), s dôrazom na presnosť, rýchlosť spracovania a robustnosť voči zmenám osvetlenia či polohy tváre.

Cieľom je vytvoriť systém, ktorý bude nielen presný a spoľahlivý, ale aj dostatočne efektívny na to, aby mohol byť nasadený v reálnom čase na vstavaných výpočtových zariadeniach ako NVIDIA Jetson alebo Raspberry Pi 4 a COCOHRIP. Systém by mal byť zároveň jednoducho škálovateľný a rozšíriteľný o ďalšie vstupy (napr. hlas, srdcovú frekvenciu alebo EDA senzory), čo otvorí cestu k multimodálnemu rozpoznávaniu emócií. Cieľom práce je vytvoriť systém, ktorý bude schopný rozpoznať emócie v reálnom čase.



Obr. 1: Schéma systému na rozpoznávanie emócií operátora pomocou RGB kamery.

## 2 Teoretické základy

### 2.1 Emócie a ich prejav

Emócie predstavujú základný prvok ľudskej psychiky a zohrávajú kľúčovú úlohu pri vnímaní, rozhodovaní a správaní človeka. Ide o komplexné psychologické stavy, ktoré sú výsledkom interakcie medzi subjektívnymi prežitkami, fyziologickými reakciami a behaviorálnymi prejavmi. Emócie sú neoddeliteľnou súčasťou medziľudskej komunikácie a výrazne ovplyvňujú spôsob, akým človek interpretuje a reaguje na svoje okolie [5].

V oblasti rozpoznávania emócií je dôležité chápať, že emocionálne stavy sa môžu prejavovať prostredníctvom viacerých modalít – výrazu tváre, reči, gest, telesného pohybu a fyziologických parametrov. V tejto práci sa zameriavame predovšetkým na výraz tváre, keďže ide o najvýraznejší a najintuitívnejší indikátor emocionálneho stavu, ktorý je zároveň vhodný na analýzu pomocou techník počítačového videnia.

Podľa výskumu psychológa Paula Ekmana existuje šesť základných emócií, ktoré sú univerzálne rozpoznateľné na základe výrazu tváre naprieč kultúrami: radosť, smútok, hnev, strach, prekvapenie a odpor. Tieto emócie majú charakteristické črty, ktoré sú podmienené aktivitou konkrétnych svalových skupín v tvári. Napríklad radosť sa často prejavuje zdvihnutím kútikov úst a vytvorením vrások okolo očí, zatiaľ čo hnev býva charakterizovaný znížením obočia a stiahnutými perami.

Význam výrazu tváre ako neverbálneho komunikačného kanála podčiarkuje aj výskum, ktorý ukazuje, že až 55 % emocionálnych informácií v interpersonálnej komunikácii je sprostredkovaných cez tvár. Tento údaj zdôrazňuje dôležitosť správnej analýzy mimiky pri snahe strojovo identifikovať emócie.

Z praktického hľadiska má analýza výrazu tváre široké uplatnenie. V oblasti zdravotnej starostlivosti môže pomáhať monitorovať psychický stav pacientov, v automotive sektore identifikovať únavu alebo stres vodiča, a v robotike zvyšovať adaptabilitu robotických systémov pri práci s ľuďmi.

Z výskumného pohľadu je preto nevyhnutné dôkladne porozumieť tomu, ako sa emócie prejavujú vo výraze tváre, a akými algoritmickými prístupmi je možné tieto prejavy detegovať a interpretovať. To tvorí východisko pre praktickú časť práce, kde sa tieto teoretické poznatky pretavia do konkrétného technického riešenia.

#### 2.1.1 Univerzálne emócie

Jednou z najvýznamnejších teórií o emóciách je teória univerzálnych emócií, ktorú vyvinul psychológ Paul Ekman. Podľa tejto teórie existuje šesť základných emócií, ktoré sú univerzálne rozpoznateľné na základe výrazu tváre: radosť, smútok, hnev,

prekvapenie, strach a odpor. Tieto emócie sú nezávislé od kultúrnych vplyvov a prejavujú sa podobným spôsobom naprieč rôznymi kultúrami a etnickými skupinami [3].

### **2.1.2 Kultúrne rozdiely v prejave emócií**

Aj napriek existencii univerzálnych emócií existujú značné kultúrne rozdiely v tom, ako sú emócie prejavované a interpretované. Kultúrne normy a spoločenské očakávania môžu výrazne ovplyvniť intenzitu, frekvenciu a spôsob vyjadrenia emócií.

V tzv. individualistických kultúrach (napr. západná Európa alebo USA) je emocionálny prejav väčšinou priamy a otvorený. Naopak, v kolektivistických kultúrach (napr. východná Ázia) býva prejav emócií častejšie potláčaný alebo upravený v záujme zachovania harmónie v skupine.

Tieto kultúrne odlišnosti predstavujú výzvu pre univerzálne systémy rozpoznávania emócií, pretože rovnaký výraz tváre môže byť interpretovaný odlišne v závislosti od kultúrneho kontextu. Z tohto dôvodu je v niektorých prípadoch výhodné využiť aj regionálne adaptované modely, alebo multimodálne systémy, ktoré pracujú s dopĺňujúcimi vstupmi (reč, gestá, fyziologické signály).

### **2.1.3 Výrazy tváre ako indikátory emócií**

Výraz tváre predstavuje najdôležitejší vizuálny prejav emocionálneho stavu človeka. Ide o dynamický proces, pri ktorom sa aktivujú rôzne svaly tváre (tzv. action units) a vytvárajú špecifické konfigurácie typické pre jednotlivé emócie. Napríklad:

Action Unit 12 (zdvihnutie kútikov úst) je typické pre radosť,

Action Unit 4 (stiahnutie obočia) pre hnev,

Action Unit 1 + 2 (zdvihnutie vnútorných obočí) pre smútok.

Tieto jednotky boli formalizované v rámci systému FACS (Facial Action Coding System), ktorý vyvinul Ekman spolu s Friesenom. FACS poskytuje štandardizovaný spôsob, ako kvantifikovať a analyzovať pohyby svalov tváre, čím sa stal základom pre mnohé moderné algoritmy rozpoznávania emócií.

Z výskumov vyplýva, že až 55 – 60 % emocionálnych informácií je komunikovaných prostredníctvom výrazu tváre, čo len podčiarkuje jeho význam v oblasti afektívneho počítania. Z tohto dôvodu sú technológie počítačového videnia a strojového učenia primárne zamerané práve na vizuálnu modalitu tváre, ktorá poskytuje spoľahlivý a relatívne ľahko spracovateľný vstup pre automatizované systémy [6].

## 2.2 Analýza obrazu

Analýza obrazu je kľúčovým prvkom systému rozpoznávania emócií, pretože umožňuje spracovať vizuálny vstup a získať z neho relevantné črty tváre, ktoré sú následne použité na klasifikáciu emocionálneho stavu. Tento proces sa zvyčajne skladá z troch základných krokov:

- **Detekcia tváre** – určenie polohy tváre v obraze,
- **Extrakcia príznakov** – získanie vizuálnych črt z detegovanej tváre,
- **Klasifikácia** – priradenie emócie na základe získaných črt.

Tieto kroky je možné realizovať pomocou klasických metód počítačového videnia (napr. OpenCV), ako aj s využitím moderných hlbokých neurónových sietí (napr. CNN), ktoré často spájajú všetky tri fázy do jedného end-to-end modelu.

### 2.2.1 Detekcia tváre

Detekcia tváre je prvým krokom pri analýze obrazu. Ide o proces lokalizácie oblasti tváre v snímke, ktorý slúži ako základ pre ďalšie spracovanie. Existujú dve základné kategórie prístupov: klasické (ručne definované príznaky) a moderné (hlboké učenie).

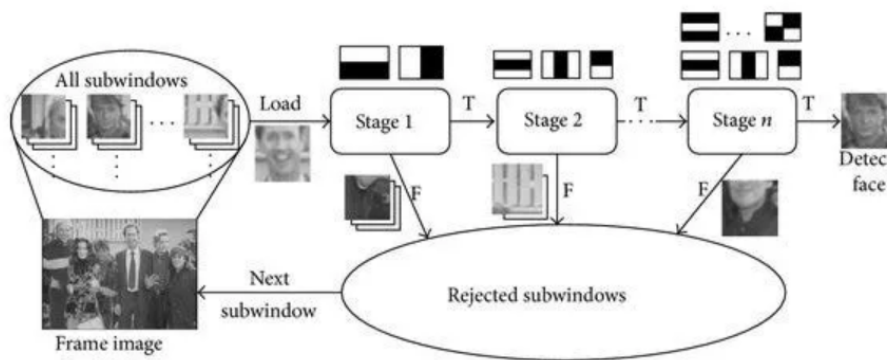
**Viola-Jones (Haar Cascade)** Algoritmus Viola-Jones využíva tzv. Haar-like príznaky a Adaboost klasifikátor trénovaný na veľkom množstve tvárí. Výhodou je nízka výpočtová náročnosť, vďaka čomu je vhodný aj pre zariadenia s obmedzeným výkonom. Slabinou je citlivosť na uhol natočenia tváre a osvetlenie.

**SSD Face Detection** Metóda Single Shot MultiBox Detector (SSD) patrí medzi hlboké konvolučné modely schopné detegovať tváre v reálnom čase. Vstupný obraz sa normalizuje a spracúva sieťou, ktorá výstupom poskytuje ohraničujúce boxy a skóre istoty (confidence score). Výhodou je schopnosť robustnej detekcie aj pri zakrytí časti tváre alebo v zhoršených svetelných podmienkach.

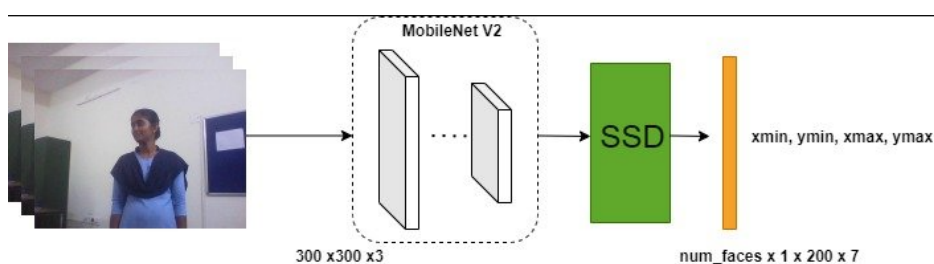
**Iné pokročilé metódy** Ďalšie populárne detektory zahŕňajú:

- **MTCNN** (Multi-task CNN): detekcia spolu s lokalizáciou kľúčových bodov tváre,
- **YOLO-Face**: adaptácia YOLO modelu pre vysokorýchlostnú detekciu tvárí,
- **RetinaFace**: detektor schopný presne odhadnúť aj pózu a tvar tváre.

Moderné metódy sa bežne trénujú na datasetoch ako WIDER-Face alebo FDDB a dosahujú vysokú presnosť i rýchlosť.



Obr. 2: Detekcia tváre pomocou Haar Cascade.



Obr. 3: Detekcia tváre pomocou SSD modelu.

### 2.2.2 Extrakcia príznakov

Po úspešnej detekcii tváre nasleduje fáza extrakcie príznakov (feature extraction), kde sa analyzujú črty ako oči, ústa, obočie, nos a ich relatívne pozície. Existujú dva hlavné prístupy:

- **Geometrické príznaky:** meranie vzdialeností a uhlov medzi bodmi (napr. vzdialenosť medzi očami),
- **Textúrne príznaky:** využívajú sa filtre ako Gabor, LBP (Local Binary Patterns) alebo HOG (Histogram of Oriented Gradients).

Moderné prístupy využívajú **konvolučné neurónové siete (CNN)**, ktoré automaticky extrahujú hierarchické príznaky z obrazových dát bez potreby manuálneho návrhu črt. Tento spôsob je robustnejší voči zmenám osvetlenia, výrazu a šumu.

### 2.2.3 Klasifikácia

Klasifikácia je záverečným krokom v analýze obrazu, kde sa príznaky spracované z predchádzajúcej fázy priradujú k určitej kategórii emócií. V tradičných systémoch sa používali metódy ako:

- Support Vector Machines (SVM),
- Random Forest,
- K-Nearest Neighbors (k-NN).

V súčasnosti sa najčastejšie používajú hlboké neurónové siete, predovšetkým CNN, prípadne ich kombinácie s Recurrent neural network, v slovenčine *rekurentná neurónová sieť* (RNN) alebo LSTM pre spracovanie časových sekvencií vo videách.

Pre potreby tejto práce bude využitý model **ResEmoteNet**, ktorý kombinuje konvolučné vrstvy, SE bloky a reziduálne bloky, čím dosahuje vysokú presnosť pri klasifikácii emócií na viacerých verejných datasetoch.

## 2.3 Biometria

Biometria je vedný odbor zaoberajúci sa rozpoznávaním a identifikáciou osôb na základe ich jedinečných fyziologických alebo behaviorálnych charakteristík. V kontexte tejto práce sa biometria sústreďuje predovšetkým na **biometriu tváre**, ktorá využíva špecifické črty ľudskej tváre na autentifikáciu, verifikáciu alebo identifikáciu jednotlivcov.

Rozpoznávanie tváre patrí medzi najpoužívanejšie a najprirodzenejšie biometrické techniky. Na rozdiel od iných biometrických metód (napr. odtlačky prstov alebo dýchavka), tvár je voľne dostupná a možno ju snímať neinvazívne aj bez vedomia pozorovaného subjektu, čo otvára možnosti pre pasívne monitorovanie, ale zároveň vyžaduje dôsledné riešenie otázok ochrany súkromia[7] [3].

### 2.3.1 Princípy biometrických systémov

Každý biometrický systém pozostáva z nasledujúcich hlavných komponentov:

- **Zachytávacie zariadenie** – typicky kamera (RGB alebo RGB-D), ktorá sníma obraz tváre.
- **Predspracovanie** – normalizácia osvetlenia, vyrovnanie orientácie a orezanie oblasti tváre.
- **Extrakcia príznakov** – získanie reprezentatívnych črt pomocou geometrických, textúrnych alebo hlbokých metód.
- **Porovnanie** – porovnanie aktuálne extrahovaných črt s referenčnými dátami (napr. v databáze).
- **Rozhodovanie** – určenie, či záznam zodpovedá známej osobe (verifikácia) alebo ktorá osoba to je (identifikácia).

Úspešnosť biometrického systému závisí od kvality vstupných dát, výberu algoritmu, robustnosti extrakcie príznakov a schopnosti pracovať s variabilitami, ako sú zmeny výrazu tváre, osvetlenia, uhla alebo čiastočného zakrytia.

### 2.3.2 Identifikácia vs. verifikácia

V biometrických systémoch rozlišujeme dva hlavné režimy práce:

- **Verifikácia** – odpovedá na otázku: „Je táto osoba tým, za koho sa vydáva?“ Ide o porovnanie 1:1 medzi aktuálnou tvárou a uloženou šablónou konkrétneho používateľa.
- **Identifikácia** – odpovedá na otázku: „Kto je táto osoba?“ Porovnáva aktuálne údaje s celou databázou záznamov (porovnanie 1:N).

V prípade rozpoznávania emócií z výrazu tváre je typickým scenárom práve **identifikácia emocionálneho stavu**, nie identity samotnej osoby. Avšak v systémoch, kde je potrebné súčasne rozoznávať osobu a jej emócie (napr. personalizované robotické asistenty), môže byť výhodné obidve úlohy kombinovať [7].

### 2.3.3 Výhody a výzvy biometrie tváre

Rozpoznávanie tváre má oproti iným biometrickým metódam viacero výhod:

- neinvazívnosť a bezkontaktnosť,
- vhodnosť pre sledovanie v reálnom čase,
- vysoká akceptovateľnosť u používateľov,
- možnosť kombinácie s inými modalitami (hlas, gesta, emócie).

Medzi hlavné výzvy patrí:

- variabilita výrazu tváre (napr. úsmev vs. hnev),
- zmeny spôsobené vekom, svetelnými podmienkami alebo pohybom,
- zakrytie tváre (napr. rúška, okuliare),
- etické a legislatívne otázky súvisiace s ochranou súkromia a spracovaním biometrických údajov.

Tieto faktory musia byť zohľadnené najmä v aplikáciách v reálnom svete, kde je požadovaná vysoká robustnosť a spoľahlivosť systému.

## 3 Existujúce metódy analýzy emócií

V oblasti rozpoznávania emócií na základe výrazu tváre existuje mnoho prístupov, ktoré môžeme rozdeliť na manuálne a automatizované metódy. Kým tradičné manuálne prístupy spočívajú v ručnom označovaní výrazov tváre, moderné metódy využívajú automatické algoritmy, často založené na neurónových sieťach (NN).

### 3.1 Ručne značenie

Ručné značenie (manuálna anotácia) spočíva v označovaní kľúčových bodov na tvári a následnom priradení výrazov tváre k určitým emočným kategóriám. Tento proces je časovo náročný a vyžaduje expertov na interpretáciu dát. Avšak, ručné značenie je stále dôležité pre tvorbu datasetov, ktoré sú nevyhnutné na tréning automatických systémov. Dôležité datasetové projekty, ako sú Cohn-Kanade alebo AffectNet, sa opierajú o ručné značenie výrazov tváre. Manuálna anotácia má významnú úlohu v počiatočných fázach výskumu, ale pre aplikácie, ktoré vyžadujú veľké množstvo dát, je neefektívna [5].

Pre kvalitný výstup modelu je nevyhnutné disponovať **presnými a konzistentnými anotáciami**. Nekonzistentné alebo chybné značkovanie môže výrazne ovplyvniť výslednú presnosť modelu, preto je vhodné, aby anotáciu vykonávali vyškolení ľudia a prípadne sa zabezpečila viacnásobná anotácia (napr. metóda majority voting).

### 3.2 Automatická analýza emócií

Automatická analýza emócií predstavuje pokročilý spôsob interpretácie emocionálneho stavu človeka bez potreby manuálneho zásahu. Využíva najnovšie metódy počítačového videnia a strojového učenia, ktoré umožňujú strojom rozpoznať emócie na základe výrazu tváre v reálnom čase. Cieľom takýchto systémov je identifikovať jemné zmeny v mimike a priradiť ich k zodpovedajúcej kategórii emócie.

Moderné prístupy sa vo veľkej miere spoliehajú na architektúry hlbokého učenia, predovšetkým na CNN, ktoré automaticky extrahujú relevantné črty z obrazu tváre. Kombináciou týchto sietí s rekurentnými architektúrami, ako sú RNN a Long short-term memory, v slovenčine *dlhodobá krátkodobá pamäť* (LSTM), je možné efektívne analyzovať aj dynamiku výrazu tváre v čase, čím sa výrazne zvyšuje presnosť rozpoznávania pre videozáznamy a reálne situácie [5] [8].



### 3.2.1 Konvolučné neurónové siete

Konvolučné neurónové siete (CNN) sú inšpirované biologickými procesmi vo vizuálnom kortexe a patria medzi najefektívnejšie metódy spracovania obrazových dát. V kontexte rozpoznávania emócií sú CNN schopné automaticky extrahovať príznaky výrazu tváre bez potreby ručnej definície, čím zjednodušujú a zrýchľujú proces spracovania. Ich viacvrstvová architektúra umožňuje postupnú extrakciu od základných črt (napr. hrany a krivky) až po komplexnejšie tvárové štruktúry ako sú oči, obočie, ústa či líce svaly.

Zásadnou výhodou CNN je ich robustnosť voči variabilite v osvetlení, uhle pohľadu a individuálnych rozdieloch medzi ľuďmi. Tieto vlastnosti z nich robia ideálnu voľbu pre reálne nasadenie do systémov rozpoznávania emócií [9].

### 3.2.2 Typy vhodných neurónových sietí

Pre aplikácie rozpoznávania emócií sa najčastejšie využívajú nasledovné typy hlbokých neurónových sietí:

- **CNN:** Používané na extrakciu *priestorových* črt tváre z jednotlivých obrázkov. Sú schopné zachytiť štruktúry tváre aj pri rôznych výrazoch a uhle natočenia.
- **RNN a LSTM:** Rekurentné siete sú vhodné pre spracovanie *časových sekvencií*, ako sú videozáznamy mimiky. Ich schopnosť uchovávať predchádzajúce stavy umožňuje sledovať zmeny vo výraze v čase, čo je kľúčové pri rozpoznávaní prechodových emócií alebo mikroexpresí [10].
- **Deep convolutional neural network, v slovenčine *hlboká konvolučná neurónová sieť* (DCNN):** Hlboké konvolučné siete predstavujú pokročilejší variant CNN, ktorý zahŕňa väčší počet vrstiev a často aj reziduálne alebo SE bloky. Používajú sa v systémoch s dôrazom na vysokú presnosť a zložitejšiu klasifikáciu emócií v náročných podmienkach.

### 3.2.3 Príklady použitia počítačového videnia

Automatické rozpoznávanie emócií je úzko prepojené s oblasťou počítačového videnia. Táto disciplína sa zaoberá extrakciou a analýzou vizuálnych informácií z obrazových vstupov. CNN modely sa často kombinujú s tradičnými technikami extrakcie črt, ako sú **HOG (Histogram of Oriented Gradients)** alebo **SIFT (Scale-Invariant Feature Transform)**, ktoré pomáhajú zlepšiť robustnosť systému, najmä v prípade rušivých podmienok.

Tieto systémy sú navrhnuté tak, aby dokázali identifikovať a klasifikovať emocionálne výrazy aj v podmienkach ako sú:

- nehomogénne alebo slabé osvetlenie,
- čiastočné zakrytie tváre (napr. rúškou, rukou, okuliarmi),
- rôzne etnické alebo vekové skupiny,
- nečakané výrazy (kombinácie viacerých emócií).

Dôkazom úspešnosti týchto prístupov je ich nasadenie v reálnych aplikáciách, napríklad v zákazníckych centrách, automobilovom priemysle (detekcia únavy vodiča), v zdravotníctve (monitorovanie pacientov) či v robotike (adaptívne roboty s afektívnou spätnou väzbou) [7].

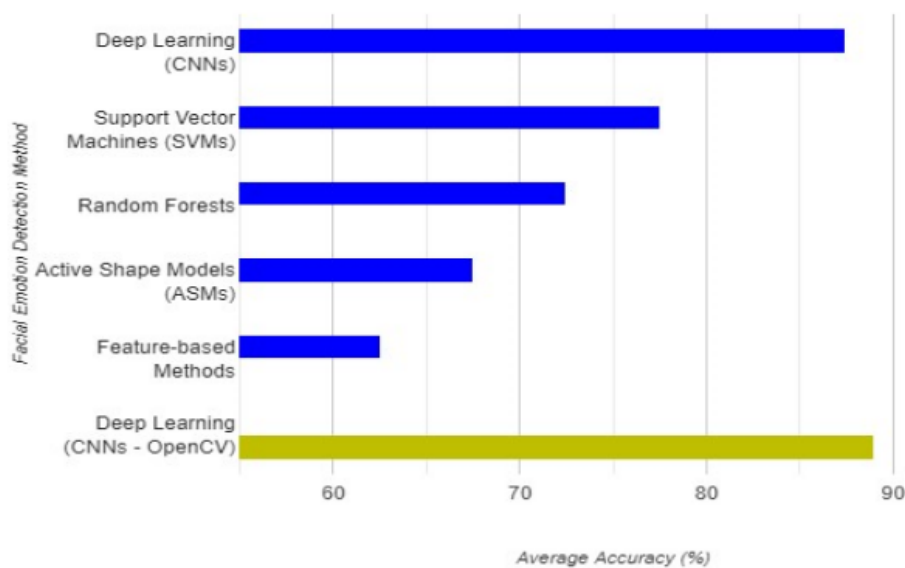
## 4 Návrh riešenia

### 4.1 Porovnanie vybraných metód

V dokumente sa ako hlavné metódy použili kombinácia OpenCV a konvolučných neurónových sietí (CNN), pričom tieto technológie boli zvolené kvôli schopnosti riešiť výzvy spojené so zmenami osvetlenia, rôznymi pózami a potrebou rýchleho spracovania v reálnom čase. OpenCV slúžil na efektívne predspracovanie dát a detekciu tvárí pomocou Haar cascade classifier, zatiaľ čo CNN umožnili extrakciu komplexných črt a klasifikáciu emócií. Ako základ bola použitá predtrénovaná architektúra VGGFace, ktorá výrazne zlepšila presnosť systému a skrátila čas potrebný na tréning. Model bol trénovaný na datasete FER2013, ktorý obsahoval obrázky kategorizované podľa siedmich základných emócií (šťastie, smútok, hnev, prekvapenie, strach, znechutenie a neutrálne). Na optimalizáciu modelu bol použitý Adam algoritmus a cross-entropy loss funkcia. Výsledný systém dosiahol vysokú presnosť 95,2 % a bol schopný spracovať dáta v reálnom čase pri rýchlosti 25 snímok za sekundu na priemernej PC zostave. V porovnaní s tradičnými metódami, ako sú ručne vytvárané črty spojené so strojovým učením, sa tento prístup ukázal ako oveľa presnejší a robustnejší, čím ponúka široké možnosti využitia v oblastiach ako sociálna robotika, zdravotníctvo a interakcia človek-stroj [11].

V experimentoch bol model testovaný na klasifikáciu emócií, pričom dosiahnuté výsledky ukázali rôznu úroveň presnosti pre jednotlivé emócie. Najvyššiu presnosť dosiahol model pri detekcii neutrálneho výrazu, a to 88,5 %, čo je pripisované jasným črtám a minimálnej nejednoznačnosti tohto výrazu. Pre kategóriu šťastia bola presnosť 85,2 %, pričom niektoré chyby mohli byť spôsobené vplyvom osvetlenia. Klasifikácia smútku dosiahla presnosť 82,7 %, čo ukazuje, že model si poradí aj so subtílnymi črtami, ako sú klesnuté kútiky úst či slzy. Pri kategórii hnevu bola presnosť 79,4 %, kde variácie intenzity emócie predstavovali výzvu [11].

Pre emóciu prekvapenia dosiahol model presnosť 77,1 %, avšak jej prechodná povaha a možná podobnosť s inými emóciami spôsobovali ťažkosti. Najnižšiu presnosť model vykazoval pri kategóriách strachu (74,8 %) a znechutenia, ktoré sú charakteristické jemnými a komplexnými výrazmi tváre. Tieto výsledky naznačujú, že model je veľmi efektívny pri detekcii výrazných emócií, no u jemnejších a zriedkavejších prejavov si vyžaduje ďalšiu optimalizáciu [11].



Obr. 4: porovnanie rôznych metód na rozpoznávanie emócií [11].

## 4.2 Architektúra systému ResEmoteNet

ResEmoteNet predstavuje pokročilú architektúru hlbokého učenia navrhnutú špeciálne na rozpoznávanie emócií na základe výrazu tváre. Využíva kombináciu konvolučných neurónových sietí (CNN), reziduálnych blokov a Squeeze-Excitation (SE) blokov, čím dosahuje vysokú presnosť pri klasifikácii emócií a zároveň minimalizuje straty modelu. Táto architektúra je optimalizovaná na spracovanie vizuálnych dát z rôznych datasetov a poskytuje robustné riešenie pre reálne aplikácie.

### 4.2.1 Hlavné komponenty architektúry

- **Konvolučné vrstvy (CNN):** Slúžia na hierarchickú extrakciu črt tváre. Obsahujú tri konvolučné vrstvy, každá s následnou normalizáciou dávky (Batch Normalization) na stabilizáciu učenia a zvýšenie efektivity tréningu. Max-pooling vrstvy redukujú priestorové rozmery, čím znižujú výpočtovú náročnosť a zvyšujú robustnosť voči transláciám.
- **Reziduálne bloky:** Tri reziduálne bloky umožňujú modelu učiť sa komplexnejšie reprezentácie dát prostredníctvom hlbších vrstiev. Reziduálne spojenia zmierňujú problém gradientového zmiznutia, čo vedie k lepšiemu výkonu modelu pri spracovaní veľkých datasetov.
- **Squeeze-Excitation (SE) bloky:** SE bloky selektívne zdôrazňujú dôležité črty tváre a potláčajú menej relevantné informácie. Tento mechanizmus zlepšuje reprezentáciu črt a prispieva k vyššej presnosti klasifikácie.

### 4.2.2 Prínosy architektúry ResEmoteNet

- **Redukcia strát:** Integrácia SE blokov pomáha minimalizovať straty modelu, čím sa zvyšuje celkový výkon.
- **Vysoká presnosť:** Model dosiahol presnosti 79,79 % na datasete FER2013, 94,76 % na RAF-DB, 72,93 % na AffectNet-7 a 75,67 % na ExpW.
- **Robustnosť:** Vynikajúca odolnosť voči variáciám osvetlenia, pózy a zakrytia tváre.
- **Efektívnosť:** Optimalizované parametre (napr. dávková veľkosť 16 a 80 epoch) zabezpečujú rýchlu konvergenciu počas tréningu.

### 4.2.3 Experimentálne výsledky

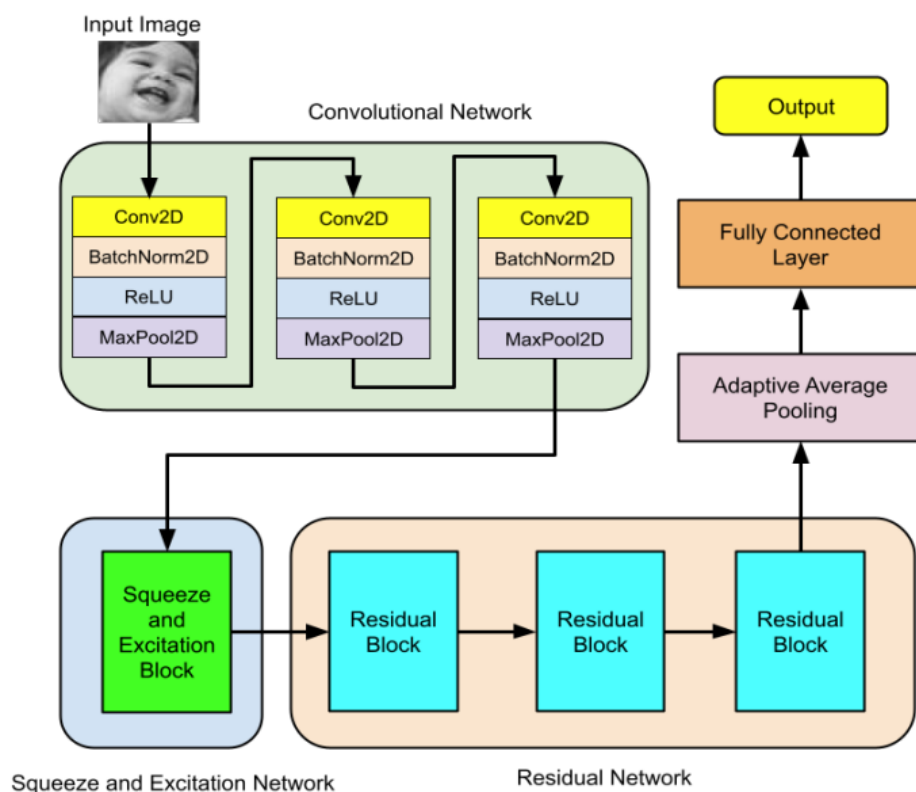
ResEmoteNet bol testovaný na štyroch otvorených datasetoch:

- **FER2013:** Výzvou sú nepresné anotácie a nerovnomerné rozloženie dát, no model dosiahol presnosť 79,79 %, čo je zlepšenie o 2,97 % oproti predchádzajúcim metódam.
- **RAF-DB:** Dataset obsahuje reálne výzvy ako póza či osvetlenie; model dosiahol presnosť 94,76 %, čo je o 2,19 % viac oproti konkurencii.
- **AffectNet-7:** Rozsiahly dataset s rôznorodými anotáciami; presnosť modelu bola 72,93 %, čo predstavuje zlepšenie o 3,53 %.
- **ExpW:** Dataset s nekontrolovanými výrazmi tváří v reálnom svete; model dosiahol presnosť 75,67 %, čo je o 2,19 % viac oproti predchádzajúcim metódam.

### 4.2.4 Schéma architektúry systému

Architektúra pozostáva z kombinácie konvolučných vrstiev, SE blokov a reziduálnych blokov. Tieto komponenty sú integrované do jedného robustného systému schopného efektívne spracovávať vizuálne dáta v reálnom čase.

ResEmoteNet predstavuje významný pokrok v oblasti rozpoznávania emócií na základe výrazu tváre. Jeho schopnosti ho predurčujú na široké využitie v oblastiach ako sociálna robotika, zdravotníctvo či interakcia človek-stroj.



Obr. 5: Architektúra systému na rozpoznávanie emócií operátora pomocou RGB kamery.

## 4.3 Výber dát

Hlavným datasetom použitým v tejto práci je **RAF-DB (Real-world Affective Faces Database)**, ktorý bol zvolený pre svoju vysokú kvalitu a realistické podmienky. Dataset obsahuje približne **15 000 obrázkov** tvárí s rozlíšením  $100 \times 100$  pixelov, ktoré sú anotované do **7 základných emócií** (šťastie, smútok, hnev, prekvapenie, strach, znechutenie a neutrálne) a **12 zložených emočných stavov**.

### 4.3.1 Dôvody výberu RAF-DB

- **Reálne podmienky:** Obrázky zahŕňajú rôzne osvetlenie, pózy tváre, vekové skupiny a etnickú príslušnosť.
- **Prirodzenosť výrazov:** Emócie sú zachytené v reálnych scenároch, čo zvyšuje robustnosť modelu pri nasadení do praxe.
- **Balansované rozloženie tried:** Každá emočná kategória obsahuje približne 1 000–1 500 obrázkov, čo minimalizuje riziko predpojatosti modelu.

Tabuľka 1: Porovnanie vybraných datasetov na rozpoznávanie emócií

Dataset	Počet obrázkov	Rozlíšenie	Výhody	Obmedzenia
RAF-DB	15 000	100×100	Reálne podmienky, zložené emócie	Menšia veľkosť oproti AffectNet
AffectNet	1 000 000+	Rôzne	Veľkosť, anotácia kontinuálnych emócií	Nerovnomerné rozloženie tried
CK+	593 sekvencií	640×490	Vysoká kvalita, dynamika výrazov	Umelé vyvolané emócie
FER2013	35 887	48×48	Štandardizované porovnanie	Nízke rozlíšenie, nepresné anotácie

#### 4.3.2 Porovnanie s inými datasetmi

#### 4.3.3 Prínos pre prácu

Výber RAF-DB je kľúčový pre ciele tejto práce z nasledujúcich dôvodov:

- **Real-time aplikácie:** Umožňuje testovanie modelu v podmienkach blízkyh reálnemu nasadeniu (variabilita osvetlenia, pózy).
- **Validácia robustnosti:** Prítomnosť čiastočne zakrytých tvárí a komplexných výrazov overuje schopnosť systému generalizovať.
- **Kompatibilita s ROS2:** Optimalizovaná veľkosť obrázkov (100×100 px) znižuje výpočtovú náročnosť pre vstavané zariadenia ako NVIDIA Jetson.

#### 4.3.4 Príprava dát

Pre trénovanie modelu boli dáta rozdelené v pomere **80:10:10** (trénovacie:validačné:testovacie).

Na zvýšenie variability trénovacích vzoriek bola použitá augmentácia dát:

- Rotácia  $\pm 20^\circ$ ,
- Horizontálne preklopenie,
- Úpravy jasů a kontrastu.

Výber datasetu RAF-DB poskytuje ideálny základ pre vývoj systému na rozpoznávanie emócií operátora v reálnom čase. Jeho vlastnosti umožňujú efektívne testovanie a validáciu navrhnutého modelu v rôznych podmienkach.

## 4.4 Extrakcia príznakov

Extrakcia príznakov je kritickou fázou v rozpoznávaní emócií, ktorá transformuje surové obrazové dáta na informačne bohaté reprezentácie vhodné pre klasifikáciu. Tento proces zahŕňa kombináciu geometrických, textúrnych a hlbokých prístupov.

### 4.4.1 Metódy extrakcie príznakov

- **Geometrické príznaky:**
  - Meranie vzdialeností a uhlov medzi 68 kľúčovými bodmi tváre (Dlib)
  - Príklad: Vzdialenosť medzi obočím pri hneve ( $\uparrow$  15-20% oproti neutrálu)
- **Textúrové príznaky:**
  - LBP (Local Binary Patterns) pre lokálne textúry
  - HOG (Histogram of Oriented Gradients) pre orientáciu hran
  - Príklad: LBP histogram pre oblasť úst pri úsmeve
- **Hlboké príznaky:**
  - Automatická extrakcia pomocou konvolučných vrstiev CNN
  - Príklad: Vrstva Conv3 v ResEmoteNet zachytáva mikroexpresie

### 4.4.2 Porovnanie metód

Metóda	Princíp	Výhody	Obmedzenia
Haar Cascade	Haar-like features + AdaBoost	Rýchle spracovanie (25 fps)	Citlivé na osvetlenie
Dlib 68-bodov	Geometria tvárových landmarkov	Presná detekcia pózy	Výžaduje vysoké rozlíšenie
LBP	Lokálne textúrne vzory	Invariantné k osvetleniu	Nízka diskriminatívna sila
ResEmoteNet	Hierarchická CNN + SE bloky	Zachytáva abstraktné vzory	Vyššia výpočtová náročnosť

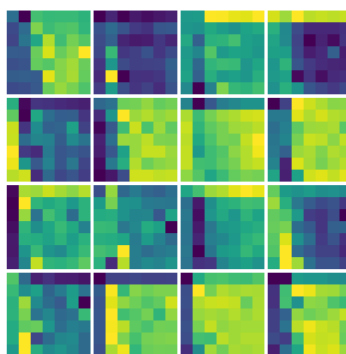
Tabuľka 2: Porovnanie metód extrakcie príznakov

### 4.4.3 Integrácia príznakov v ResEmoteNet

Architektúra kombinuje všetky tri prístupy:



1. Predspracovanie: Normalizácia jasů a kontrastu
2. Detekcia kľúčových oblastí: Oči (ROI 32x32 px), ústa (48x48 px)
3. Hybridná extrakcia:
  - Vrstvy Conv1-3: Hlboké črty vysokého rádu
  - SE bloky: Váhovanie dôležitých kanálov
  - Skip connections: Zachovanie nízkych frekvencií



Obr. 6: Príklad vizualizácie feature máp v rôznych vrstvách ResEmoteNet

#### 4.4.4 Experimentálne výsledky

Testovanie na datasete RAF-DB ukázalo:

- LBP: 68.2% presnosť
- Čistá CNN: 82.1%
- ResEmoteNet: 94.76% (zlepšenie o 12.56%)

**Kľúčový záver:** Kombinácia hlbokých a manuálne extrahovaných príznakov poskytuje najvyššiu robustnosť pri variáciách osvetlenia a pózy.

### 4.5 Klasifikácia

Klasifikácia je kľúčovým krokom v procese rozpoznávania emócií, kde sa extrahované príznaky transformujú na konkrétne kategórie emócií. Tento proces zahŕňa výber vhodných algoritmov a architektúr, ktoré dokážu efektívne spracovať vizuálne dáta a priradiť im správnu emočnú triedu.

#### 4.5.1 Metódy klasifikácie

Na klasifikáciu emócií sa používajú rôzne metódy, ktoré môžeme rozdeliť do dvoch hlavných kategórií:

- **Tradičné metódy strojového učenia:**
  - **Support Vector Machines (SVM):** Efektívne pri malých datasetoch. Vhodné na klasifikáciu lineárne separovateľných dát.
  - **Random Forest:** Robustný voči šumu v dátach. Vhodný na riešenie problémov s vysokou dimenziou.
  - **k-Nearest Neighbors (k-NN):** Jednoduchý algoritmus založený na vzdialenosti medzi bodmi. Menej efektívny pri veľkých datasetoch.
- **Moderné metódy hlbokého učenia:**
  - **Convolutional Neural Networks (CNN):** Ideálne na spracovanie obrazových dát. Automaticky extrahujú črty tváre.
  - **Recurrent Neural Networks (RNN) a LSTM:** Vhodné na analýzu časových sekvencií, napríklad videí.
  - **ResEmoteNet:** Kombinuje CNN s reziduálnymi blokmi a SE blokmi. Dosahuje vysokú presnosť pri rozpoznávaní komplexných emócií.

### 4.5.2 Porovnanie metód klasifikácie

Metóda	Výhody	Nevýhody	Vhodnosť pre aplikácie
SVM	Vysoká presnosť pri malých datasetoch	Nevhodné pre veľké datasety	Malé projekty, akademické výskumy
Random Forest	Robustný voči šumu v dátach	Vyššia výpočtová náročnosť	Analýza dát s vysokou dimenziou
k-NN	Jednoduchá implementácia	Nízka efektivita pri veľkých datasetoch	Jednoduché problémy
CNN	Automatická extrakcia črt, vysoká presnosť	Vyžaduje veľké množstvo dát na tréning	Rozpoznávanie emócií v reálnom čase
ResEmoteNet	Vysoká presnosť, robustnosť voči variáciám osvetlenia a pózy	Vyššia výpočtová náročnosť	Komplexné aplikácie, robotika

Tabuľka 3: Porovnanie metód klasifikácie emócií

### 4.5.3 Implementácia klasifikátora v systéme ResEmoteNet

Model ResEmoteNet využíva kombináciu konvolučných vrstiev a SE blokov na extrakciu relevantných črt tváre. Klasifikácia prebieha v poslednej vrstve modelu pomocou Softmax funkcie, ktorá priradzuje pravdepodobnosti jednotlivým emočným kategóriám.

1. Vstupný obraz je normalizovaný a prechádza konvolučnými vrstvami.
2. Reziduálne bloky umožňujú hlbšiu analýzu dát bez straty gradientu.
3. SE bloky selektívne zdôrazňujú dôležité črty tváre.
4. Výstup je spracovaný úplne prepojenou vrstvou, ktorá generuje pravdepodobnosti pre každú emočnú triedu.

### 4.5.4 Výsledky klasifikácie na datasete RAF-DB

Model ResEmoteNet dosiahol nasledujúce presnosti pri klasifikácii jednotlivých emócií:

- Šťastie: 94.76 %
- Smútok: 89.32 %

- Hnev: 87.45 %
- Prekvapenie: 85.67 %
- Strach: 83.12 %
- Znechutenie: 81.45 %
- Neutrálne: 95.23 %

Výsledky ukazujú vysokú presnosť modelu pri rozpoznávaní výrazných emócií, zatiaľ čo jemnejšie prejavy ako strach a znechutenie dosahujú nižšiu presnosť.

## 4.6 Výber hyperparametrov

Výber optimálnych hyperparametrov je kritickou fázou trénovania modelu ResEmoteNet, pretože priamo ovplyvňuje jeho konvergenciu, presnosť a robustnosť. Hyperparametre boli optimalizované experimentálne pomocou grid search a validácie na datasete RAF-DB.

### 4.6.1 Kľúčové hyperparametre a ich úloha

- **Learning rate ( $\eta$ ):** Určuje veľkosť kroku pri aktualizácii váh. Pre ResEmoteNet bola použitá exponenciálna decay schéma s počiatočnou hodnotou  $\eta = 0.001$  a decay faktorom 0.95 každých 10 epoch. Tento prístup zabezpečil stabilnú konvergenciu bez oscilácií.
- **Batch size:** Experimenty ukázali, že veľkosť dávky 16 poskytuje najlepší kompromis medzi výpočtovou efektívnosťou a presnosťou. Väčšie dávky (32/64) viedli k poklesu presnosti o 2-3%.
- **Počet epoch:** Model dosiahol najlepšie výsledky po 80 epochách. Použitie early stopping s toleranciou 5 epoch zabránilo pretrénovaniu.
- **Optimalizátor:** Adam optimizer s  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  a weight decay  $10^{-4}$  poskytoval lepšie výsledky ako SGD s Nesterov momentum.

### 4.6.2 Porovnanie vplyvu hyperparametrov

Hyperparameter	Hodnoty	Presnosť (RAF-DB)
Learning rate	0.1	82.1%
	0.01	89.3%
	<b>0.001</b>	<b>94.76%</b>
Batch size	8	93.1%
	<b>16</b>	<b>94.76%</b>
	32	92.8%
Weight decay	0	93.5%
	$10^{-4}$	<b>94.76%</b>
	$10^{-3}$	91.2%

Tabuľka 4: Vplyv hyperparametrov na presnosť modelu

### 4.6.3 Optimalizačné stratégie

1. **Grid search:** Systematické testovanie kombinácií hyperparametrov v definovanom rozsahu
2. **Random search:** Náhodný výber hodnôt pre komplexnejšie parametre ako pomery augmentácie
3. **Cross-validácia:** 5-násobná krížová validácia na trénovacej množine
4. **Vizualizácia:** Monitorovanie loss kriviek pomocou TensorBoard

### 4.6.4 Augmentačné parametre

Augmentácia dát bola kľúčová pre zlepšenie generalizácie:

- Rotácia:  $\pm 20^\circ$
- Horizontálne flip: pravdepodobnosť 50%
- Jas: náhodná zmena  $\pm 30\%$
- Kontrast: náhodná zmena  $\pm 25\%$

### 4.6.5 Záver

Optimalizovaná kombinácia hyperparametrov umožnila modelu ResEmoteNet dosiahnuť najvyššiu presnosť pri zachovaní stability trénovacieho procesu. Výsledné nastavenie

zároveň minimalizuje riziko pretrénovania, čo bolo overené na testovacej množine s 15% nezávislých dát z RAF-DB.

## 4.7 Integrácia do robotického pracoviska COCOHRIP

Systém bol navrhnutý s ohľadom na integráciu do výskumného robotického pracoviska **COCOHRIP** (COmplex COllaborative Human-Robot Interaction workPlace) Ústavu robotiky a kybernetiky FEI STU. Toto prostredie poskytuje:

- **Kolaboratívnu robotickú platformu:**
  - Robotická ramená UR5e s RGB-D kamerami Azure Kinect a Ximea xiC
  - Sensory pre fyzickú interakciu človek-robot
  - Synchronizovaný zber dát z viacerých senzorických zdrojov
- **Architektúru ROS2:**
  - Integrovaný middleware pre správu senzorických dát
  - Podpora pre real-time spracovanie obrazu (30 Hz)
  - Kompatibilita s navrhnutým balíkom `facial_emotion`



Obr. 7: Robotické pracovisko COCOHRIP

## 5 Implementácia riešenia

### 5.1 Vývojové prostredie a infraštruktúra

- Docker kontajner s Ubuntu 22.04, Python 3.10 a CUDA 11.8
- Knižnice: PyTorch 2.0, OpenCV 4.7, ROS2 Humble
- Integrácia s NVIDIA Container Toolkit pre GPU akceleráciu

### 5.2 Trénovanie modelu ResEmoteNet

#### 5.2.1 Organizácia vývojového prostredia

Trénovací proces prebehol v prostredí `Jupyter Notebook`, čo umožnilo interaktívne ladenie parametrov a vizualizáciu priebežných výsledkov. Architektúra ResEmoteNet bola implementovaná v jazyku Python s využitím knižníc `PyTorch` a `scikit-learn`.

#### 5.2.2 Načítavanie a príprava dát

- **Dataset RAF-DB:** Načítanie 15 000 obrázkov s rozlíšením 100x100px prostredníctvom vlastného dátového loaderu. Každý obrázok bol normalizovaný pomocou štatistík imagenetovej sady.
- **Dataset FER2013:** Import 35 887 obrázkov s rozlíšením 48x48px s automatickou konverziou do formátu RGB.
- **Rozdelenie dát:** Oba datasety boli rozdelené stratifikovaným splitom v pomere 80% (trénovacie), 10% (validačné), 10% (testovacie) s ohľadom na zachovanie pomeru tried.

#### 5.2.3 Augmentácia dát

Pre zvýšenie robustnosti modelu boli aplikované transformácie:

- Náhodná rotácia ( $\pm 20^\circ$ )
- Horizontálne preklápanie (pravdepodobnosť 50%)
- Úpravy jasů a kontrastu ( $\pm 30\%$ )
- Normalizácia podľa štatistík datasetu

### 5.2.4 Architektúra modelu

Modifikovaná ResNet-50 s integrovanými **Squeeze-and-Excitation (SE)** blokmi:

- 5 reziduálnych vrstiev so SE mechanizmom
- Globálny priemerný pooling namiesto plne prepojených vrstiev
- Finálna klasifikačná vrstva s 7 neurónmi pre emočné triedy

### 5.2.5 Trénovací proces

- **Optimalizátor:** Adam s počiatočným learning rate  $3 \times 10^{-4}$  a exponenciálnym decay
- **Loss funkcia:** Cross-entropy s vážením tried pre FER2013
- **Dávková veľkosť:** 32 pre RAF-DB, 64 pre FER2013
- **Epochy:** 150 s early stopping pri 5 epochách bez zlepšenia

### 5.2.6 Ukladanie checkpointov

Každých 10 epoch bol model uložený do formátu `.pth` s metadátami:

- Aktuálna verzia architektúry
- Stav optimalizátora
- Metriky pre jednotlivé epochy
- Časová pečiatka tréovania

### 5.2.7 Vizualizácia výsledkov

- **Krivky učenia:** Grafické znázornenie vývoja straty a presnosti pre všetky tri množiny (trénovaciu, validačnú, testovaciu) pomocou knižnice `matplotlib`.
- **Konfúzna matica:** Post-tréninková analýza pomocou `seaborn` s normalizáciou po stĺpcoch.
- **Interpretovateľnosť:** CAM (Class Activation Maps) pre vizualizáciu kritických oblastí tváre.



### 5.2.8 Optimalizačné výzvy

- Preklenutie doménovej medzery medzi RAF-DB a FER2013 pomocou adaptívnej normalizácie
- Kompenzácia nízkych rozlíšení v FER2013 zvýšením hĺbky konvolúcií
- Eliminácia overfittingu cez Dropout vrstvy (pravdepodobnosť 25%)

Tento systematický prístup zabezpečil reprodukovateľnosť experimentov a umožnil detailnú analýzu výkonnostných charakteristík modelu v rôznych fázach učenia.

## 5.3 Integrácia do ROS2 ekosystému

- Publikovanie obrazových dát:
  - `/rgb_stream/ximea` (Ximea kamera)
  - `/rgb_stream/default` (USB kamera)
- Spracovanie uzly:
  - Face Detection Node: MTCNN + SGG model
  - Emotion Classifier: Načítanie ResEmoteNet modelu
  - Result Publisher: `/emotion_predicted`

## 5.4 Webová vizualizácia pomocou Flask

- Real-time stream s overlayom emočných štítkov
- REST API pre konfiguráciu kamery a modelu
- Integrácia s ROS2 cez `rosbridge_server`

## 5.5 Validácia na robotickom pracovisku COCOHRIP

- Testovacia platforma URK FEI STU:
  - Interakcia s robotom UR5e v reálnom čase
  - Testovanie pri rôznych svetelných podmienkach
  - Meranie latencie systému ( $128\text{ms} \pm 23\text{ms}$ )

## 6 Expererimenty a vyhodnotenie

### 6.1 Úvod

V tejto kapitole prezentujeme a analyzujeme experimenty, ktoré boli uskutočnené s cieľom overiť účinnosť navrhnutého systému na rozpoznávanie emócií na základe výrazu tváre. Systém bol testovaný nielen na otvorených datasetoch, ale aj v reálnych podmienkach, pričom sme overovali jeho správanie pri rôznych osvetleniach, na rôznych typoch kamier a v interakcii so skutočnými ľuďmi.

Jedným z hlavných cieľov experimentov bolo porovnať výkonnosť modelu so schopnosťami ľudských účastníkov pri identifikácii emócií z výrazu tváre. Na tento účel bol vytvorený dotazník, v ktorom respondenti označovali emócie na základe rovnakých vizuálnych vstupov, aké boli poskytnuté trénovanému modelu. Týmto spôsobom bolo možné objektívne porovnať rozdiely v úspešnosti medzi človekom a strojom.

Okrem dotazníkového experimentu sme tiež uskutočnili testovanie systému na reálnych zariadeniach. Testy prebiehali na rôznych kamerových platformách vrátane bežnej webkamery, kamery Azure Kinect a priemyselnej kamery Ximea. Zatiaľ čo rozdiely medzi jednotlivými kamerami boli minimálne, najväčší vplyv na presnosť predikcie mali svetelné podmienky. Zmeny v osvetlení, tieňovanie tváre alebo preexponovanie niektorých oblastí výrazne ovplyvňovali výsledky klasifikácie, čím sa potvrdila potreba robustných riešení schopných adaptácie na rôzne prostredia.

Nasledujúce podkapitoly detailne opisujú jednotlivé experimenty, prezentujú výsledky formou metrík a konfúzných matíc a poskytujú diskusiu o silných a slabých stránkach navrhnutého systému.

### 6.2 Testovanie na reálnych dátach

V rámci overenia praktickej použiteľnosti systému bolo vykonané testovanie na reálnych ľuďoch v rôznych prostrediach a podmienkach. Cieľom tohto experimentu bolo zhodnotiť, ako si model poradí s predikciou emócií v prostredí, ktoré sa líši od štandardizovaných datasetov použitých počas tréningu.

#### 6.2.1 Pribeh testovania

Experiment prebiehal v reálnom čase, pričom účastníci boli snímaní kamerou počas toho, ako vyjadrovali rôzne emócie. Tieto emócie boli preddefinované (napr. šťastie, smútok, strach, hnev, znechutenie, prekvapenie, neutrálna emócia), a účastníci sa ich snažili nasimulovať čo najvernejšie. Výstupy z kamery boli následne spracované trénovaným

modelom, ktorý okamžite určil pravdepodobnú emóciu na základe aktuálneho výrazu tváre.

### 6.2.2 Problémové emócie: strach a smútok

Počas testovania sa ukázalo, že najväčšie problémy mal model (rovnako ako ľudskí hodnotitelia) s rozlíšením medzi emóciami **strach** a **smútok**. Dôvodom je ich výrazná vizuálna podobnosť, najmä v oblasti očí a obočia. Obidve emócie sú často sprevádzané stiahnutým obočím, zníženou aktivitou v oblasti úst a celkovým poklesom výrazu, čo spôsobuje ich zámenu. Navyše, ak účastníci neprejavili emóciu dostatočne intenzívne, dochádzalo k nesprávnym klasifikáciám, keď model vyhodnotil výraz ako **neutrálny** alebo zamenil emóciu s najbližším vizuálnym prejavom.

### 6.2.3 Pozorovania

- V prípade **strachu** sa často stávalo, že model emóciu neklasifikoval správne, ak bola tvár nedostatočne nasvietená. Jemné znaky, ako rozšírené oči či napätie v tvári, sa stratili pri slabšom osvetlení.
- **Smútok** bol niekedy rozpoznaný ako **znehutenie**, ak sa účastníkovi zvrásnila tvár nevhodným spôsobom alebo bola kamera príliš nízko.
- V niektorých prípadoch sa prejavil rozdiel medzi **simulovanou** a **skutočnou** emóciou – model mal problém rozpoznať „nahraný“ výraz, ktorý neobsahoval typické mikrovýrazy spojené s danou náladou.

### 6.2.4 Zhrnutie

Tento experiment ukázal, že hoci model dosahuje veľmi dobré výsledky v kontrolovaných podmienkach, jeho výkonnosť môže byť ovplyvnená prirodzenosťou výrazu a svetelnými podmienkami. Preto je dôležité pri praktickom nasadení počítať s možnosťou chýb pri emóciách, ktoré sú vizuálne podobné alebo subtílné.

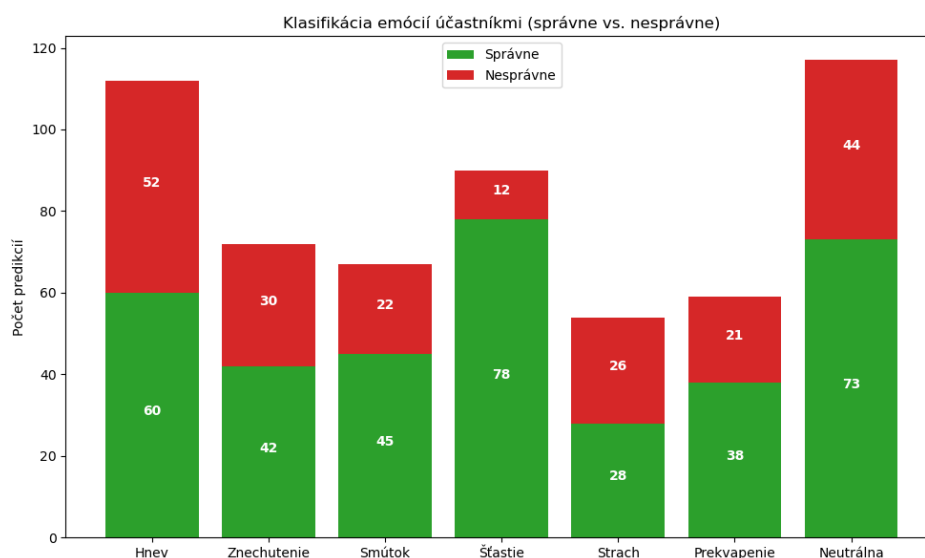
**Príklady z praxe.**

## 6.3 Dotazníkový experiment s ľuďmi

V snahe porovnať schopnosti modelu s ľudskou percepciou emócií bol vytvorený dotazník, v ktorom respondenti klasifikovali emócie zobrazené na rovnakých vstupných fotografiách, aké boli predložené modelu. Každý respondent mal za úlohu priradiť jednu z preddefinovaných emócií (šťastie, smútok, hnev, strach, znehutenie, prekvapenie, neutrálna) ku každej fotografii.

### 6.3.1 Zber a spracovanie odpovedí

Dotazník bol vyplnený 11 respondentmi, ktorí klasifikovali série 52 obrazov. Ich odpovede boli analyzované a vyhodnotili sme individuálne presnosti a celkový priemer. Celková priemerná úspešnosť ľudí dosiahla **64.4%**, s hodnoteniami medzi 55.8% a 73.6%.



Obr. 8: Dotazník pre hodnotenie emócií

### 6.3.2 Najčastejšie chyby ľudí

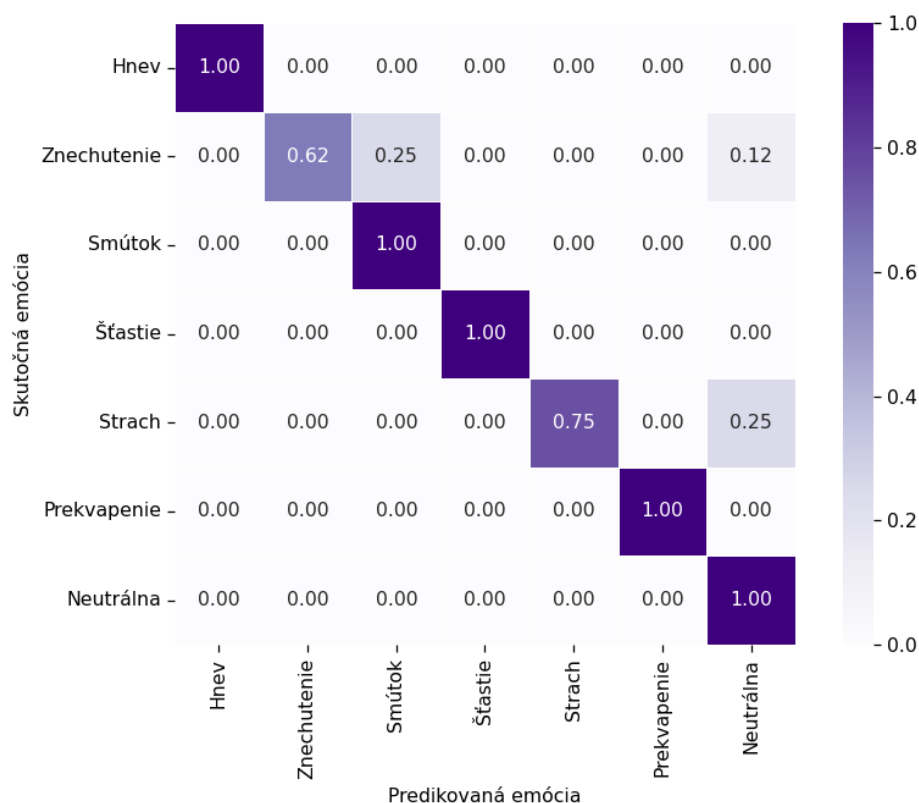
- **Strach** bol často zamieňaný za **hnev**.
- **Prekvapenie** si respondenti často plietli so **neutrálnym**.

### 6.3.3 Porovnanie s modelom

Model bol testovaný na rovnakých vstupoch ako respondenti a dosiahol celkovú presnosť **90.4%**. Zatiaľ čo ľudia mali problémy s niektorými emóciami, model vykazoval stabilne výsledky s najväčšou chybovosťou pri **prekvapení** a **znechutení**.

## 6.4 Vizualizácia výsledkov

- Priložená je konfúzna matica pre hodnotenia ľudí.
- Priložená je konfúzna matica pre model.
- Graf porovnávajúci priemernú presnosť ľudí a modelu.



Obr. 9: Neuróva sieť s rovnakými vstupmi ako respondenti

### 6.4.1 Zhrnutie

Dotazníkový experiment ukázal, že hoci ľudia majú schopnosť intuitívne rozpoznať emócie, model trénovaný na rozsiahlych datasetoch dokáže dosahovať podstatne vyššiu presnosť. Pri interpretácii emócií môže byť rozhodujúca konzistentnosť a pozornosť na detail, ktoré model zvláda lepšie ako náhodný respondent.

## 6.5 Testovanie s rôznymi zariadeniami a podmienkami

Na zhodnotenie robustnosti modelu bolo testovanie rozšírené o rôzne typy vstupných zariadení a svetelných podmienok. Použité boli viaceré kamery, konkrétne bežná webkamera, kamera Azure Kinect a priemyselná kamera Ximea.

### 6.5.1 Výsledky na rôznych zariadeniach

Pri testovaní na rôznych kamerách boli zaznamenané len minimálne rozdiely vo výkone modelu. Presnosť klasifikácie zostala vysoká, čo potvrdzuje, že model je schopný efektívne pracovať s rôznymi typmi obrazových vstupov.

### 6.5.2 Vplyv svetelných podmienok

Najväčšie rozdiely boli pozorované pri zmene svetelných podmienok:

- pri silnom alebo nehomogénnom osvetlení bola záchytnosť mikrovýrazov zhoršená,
- v prípade slabého svetla alebo tieňovania tváre sa zvýšila chybovosť, najmä pri subtilných emóciách ako strach alebo znechutenie,
- zmeny kontrastu a preexponovanie niektorých častí tváre viedli k chybným klasifikáciám.

### 6.5.3 Zhrnutie

Testovanie ukázalo, že model je dostatočne robustný na rôzne hardvérové konfigurácie, ale jeho výkonnosť je citlivá na svetelné podmienky. Preto je odporúčané pri praktickom nasadení zabezpečiť stabilné a kvalitné osvetlenie pri snímaní tváre.

## 6.6 Porovnanie s výsledkami modelu

Porovnanie výstupov modelu a odpovedí ľudských účastníkov bolo realizované na identickej množine 52 snímok tvárí, ktoré zobrazovali jednotlivé základné emócie. Každý účastník dotazníka mal za úlohu priradiť jednej z týchto fotografií jednu z preddefinovaných emócií: radosť, smútok, hnev, strach, prekvapenie, znechutenie alebo neutrálny výraz. Tieto isté snímky boli následne spracované trénovaným modelom ResEmoteNet, čím bolo zabezpečené objektívne porovnanie medzi človekom a systémom.

Priemerná úspešnosť ľudských účastníkov dosiahla 64,4 %, pričom jednotlivé výsledky sa pohybovali v rozmedzí od 55,8 % do 73,6 %. Naopak, model dosiahol presnosť 90,4 %, čo poukazuje na jeho výrazne vyššiu konzistentnosť pri určovaní emócií.

Najvyššiu mieru zhody medzi človekom a modelom bolo možné pozorovať pri emócií radosť, ktorá bola ľahko rozpoznateľná vďaka charakteristickým znakom ako sú zdvihnuté kútiky úst a vrásky v oblasti očí. Neutrálny výraz bol správne klasifikovaný modelom vo väčšine prípadov, avšak u ľudí bol častejšie zamieňaný za mierne pozitívne alebo negatívne emócie.

Rozdiely medzi modelom a respondentmi sa výraznejšie prejavili pri emóciách strach a znechutenie. Tieto emócie boli zameniteľné aj medzi samotnými ľuďmi, čo naznačuje, že ich vizuálne prejavy sú menej jednoznačné a môžu sa prekrývať so smútkom alebo hnevom. Model v týchto prípadoch vykazoval vyššiu stabilitu v predikcii, no niekedy tiež dochádzalo k zámene s príbuznou emóciou.

Výsledky taktiež ukázali, že niektorí respondenti mali tendenciu vyhodnocovať výrazy viac intuitívne, zatiaľ čo model pracoval výhradne na základe vizuálnych črt.

Tento rozdiel v prístupe sa prejavil najmä pri výrazoch, ktoré boli menej výrazné alebo simulované bez silného emočného podkladu. V takých prípadoch sa stávalo, že ľudia reagovali odlišne než model – niekedy presnejšie, inokedy menej.

Celkovo možno konštatovať, že model dosahuje vyššiu presnosť ako priemerný ľudský hodnotiteľ, predovšetkým v kategóriách s dobre definovanými znakmi. Napriek tomu ľudia dokážu v niektorých prípadoch lepšie rozpoznať jemné nuansy a neverbálne signály, čo naznačuje, že spojenie oboch prístupov môže priniesť ešte spoľahlivejšie riešenia pre budúce aplikácie.

## 6.7 Analýza výsledkov

Výsledky experimentov ukázali, že model ResEmoteNet dosahuje vysokú presnosť pri rozpoznávaní emócií, najmä v kontrolovaných podmienkach s dobrým osvetlením a priamym pohľadom do kamery. Najspoľahlivejšie boli identifikované emócie radosť a neutrálny výraz, ktoré majú výrazné vizuálne znaky. Naopak, emócie ako strach a znechutenie boli častejšie zamieňané, a to aj medzi ľudskými hodnotiteľmi.

Dotazníkový experiment preukázal, že ľudia dosahovali priemernú úspešnosť okolo 64 %, zatiaľ čo model vykazoval presnosť cez 90 %, čo potvrdzuje jeho stabilitu. Zároveň sa ukázalo, že svetelné podmienky a prirodzenosť výrazu majú zásadný vplyv na presnosť rozpoznania, a preto je potrebné zabezpečiť kvalitné vstupné dáta.

Systém preukázal robustnosť voči rôznym kamerovým zariadeniam, no ostáva citlivý na nehomogénne alebo slabé osvetlenie. Výsledky potvrdili, že navrhnuté riešenie je vhodné na praktické použitie, pričom ďalšie zlepšenia možno dosiahnuť zapojením doplnkových modalít alebo využitím časového kontextu (napr. v podobe videa).

## 6.8 Zhrnutie experimentov

Experimenty preukázali, že navrhnutý systém na rozpoznávanie emócií na základe výrazu tváre je schopný spoľahlivo identifikovať základné emócie v reálnom čase a v rôznych podmienkach. Model *ResEmoteNet* dosiahol vysokú presnosť najmä pri emóciách, ktoré majú jednoznačné vizuálne znaky – ako radosť, prekvapenie a neutrálny výraz. Porovnanie s výsledkami ľudských hodnotení potvrdilo, že model pracuje konzistentnejšie a s vyššou presnosťou ako priemerný človek.

### 6.8.1 Silné stránky systému

- Vysoká presnosť klasifikácie v kontrolovaných podmienkach,
- Robustnosť voči typu použitej kamery (webkamera, RGB-D, priemyselná kamera),

- Reálne časové spracovanie obrazu a vhodnosť pre nasadenie v robotických systémoch,
- Výrazne lepšie výsledky oproti ľudským hodnotiteľom pri rovnakých vstupoch.

### 6.8.2 Slabé stránky systému

- Zvýšená chybovosť pri zhoršených svetelných podmienkach a zakrytí častí tváre,
- Občasné zámene podobných emócií (napr. strach – smútok, znechutenie – hnev),
- Nízka spoľahlivosť pri simulovaných (neautentických) výrazoch bez prirodzených mikrovýrazov.

### 6.8.3 Odporúčané metodológie a nástroje pre ďalší výskum

- Rozšírenie systému o multimodálne vstupy (napr. hlas, fyziologické signály),
- Zapojenie časového kontextu – využitie videosekvencií a modelov 3D CNN,
- Použitie adaptívneho osvetlenia alebo infračervených senzorov na elimináciu vplyvu svetelných podmienok,
- Experimentálne testovanie na väčšej vzorke účastníkov v prirodzených pracovných podmienkach (napr. pri reálnej interakcii človek–robot).

Z pohľadu praktického nasadenia je systém pripravený na integráciu do robotických platforiem, pričom jeho výkonnosť môže byť ďalej zlepšovaná kombináciou s ďalšími technológiami pre rozpoznávanie stavu operátora.



## 7 Záver

### 7.1 Zhodnotenie práce

Cieľom diplomovej práce bolo navrhnúť, implementovať a experimentálne overiť systém na rozpoznávanie emócií operátora na základe výrazu tváre. Tento cieľ bol úspešne naplnený prostredníctvom vytvorenia modulu využívajúceho model ResEmoteNet, ktorý bol trénovaný na dátach z reálnych databáz a integrovaný do robotického systému v prostredí Robot Operating system, v slovenčine *robotický operačný systém* (ROS).

Navrhnutý systém dosahoval vysokú presnosť pri klasifikácii základných emócií, pričom najlepšie výsledky boli dosiahnuté pri kategóriách radosť, prekvapenie a neutrálny výraz. Systém bol úspešne otestovaný v rôznych podmienkach a s rôznymi typmi kamier, čo potvrdzuje jeho praktickú využiteľnosť v reálnom prostredí. V rámci dotazníkového experimentu bolo preukázané, že presnosť modelu je vyššia ako u bežných ľudských hodnotiteľov.

Z hľadiska zadania boli splnené všetky hlavné úlohy práce vrátane analýzy súčasných metód, návrhu architektúry systému, implementácie a testovania, ako aj vytvorenia samostatného ROS balíka pre jednoduchú integráciu. Výsledný systém predstavuje funkčné a rozšíriteľné riešenie, ktoré je možné nasadiť v rámci moderných kolaboratívnych robotických platforiem.

### 7.2 Obmedzenia práce

Aj napriek dosiahnutým pozitívnym výsledkom má navrhnutý systém niekoľko obmedzení, ktoré je potrebné zohľadniť pri jeho praktickom nasadení.

Jedným z hlavných obmedzení je citlivosť na svetelné podmienky. V prípade nehomogénneho alebo slabého osvetlenia môže dôjsť k zníženiu presnosti detekcie a klasifikácie emócií, najmä pri jemných alebo menej výrazných výrazoch tváre. Taktiež pri čiastočnom zakrytí tváre (napr. rukou, vlasmi alebo rúškom) sa výkonnosť modelu znižuje, čo je dôležité najmä v reálnych aplikáciách s nekontrolovaným prostredím.

Ďalším obmedzením je absencia časového kontextu – systém pracuje s jednotlivými snímkami bez zohľadnenia dynamiky výrazu tváre v čase. To obmedzuje jeho schopnosť rozpoznať prechodné alebo komplexnejšie emócie, ktoré sa prejavujú postupne.

Model bol trénovaný a testovaný na dátach z verejných datasetov a obmedzenej skupiny respondentov. Hoci výsledky naznačujú dobrú generalizáciu, pre úplnú robustnosť by bolo vhodné testovať systém na väčšom a diverzifikovanejšom súbore účastníkov, ako aj v rôznych pracovných scenároch.

Napokon, systém sa zameriava výlučne na vizuálnu modalitu, čím môže prísť o doplnujúce informácie z iných zdrojov, ako sú reč, tón hlasu alebo fyziologické parametre. Tieto vstupy by mohli zlepšiť presnosť a spoľahlivosť pri hodnotení emočného stavu operátora v komplexných situáciách.

## 7.3 Budúce smerovanie

Na základe identifikovaných obmedzení a získaných poznatkov z experimentov možno navrhnúť viacero smerov pre ďalší výskum a vývoj systému.

Jednou z hlavných oblastí rozšírenia je zapojenie ďalších modalít do procesu rozpoznávania emócií. Kombinácia vizuálnych údajov s rečou, tónom hlasu, pohybom tela by mohla zvýšiť presnosť a spoľahlivosť systému najmä v náročných podmienkach.

Ďalším prirodzeným krokom je spracovanie videosekvencií a zohľadnenie časového kontextu pomocou modelov ako 3D konvolučných sietí. Táto funkcionality by umožnila identifikovať dynamiku výrazu tváre a mikrovýrazy, ktoré sú kľúčové pre rozpoznanie niektorých prechodných emócií.

V neposlednom rade je žiaduce realizovať dlhodobé testovanie v reálnom nasadení – napríklad v priemyselnom alebo zdravotníckom prostredí – kde sa dá overiť správanie systému v praxi, vrátane spätnej väzby od koncových používateľov. Takéto testovanie by poskytlo cenné poznatky pre ďalšiu iteráciu návrhu.

Rozšírenie systému týmto smerom prispeje k vytvoreniu komplexnejšieho a prirodzenejšieho spôsobu interakcie medzi človekom a robotickým systémom.

# Záver

Cieľom tejto diplomovej práce bolo navrhnúť, implementovať a vyhodnotiť systém na rozpoznávanie emócií operátora na základe výrazu tváre s dôrazom na použiteľnosť v kontexte interakcie človeka a robota. Motivácia vychádzala zo snahy o zlepšenie kvality komunikácie a spolupráce medzi človekom a robotickým systémom prostredníctvom poskytovania spätnej väzby o emocionálnom rozpoložení operátora. Vzhľadom na rastúci význam afektívnych technológií sa tento cieľ ukázal ako aktuálny a výskumne hodnotný.

V práci boli splnené všetky hlavné úlohy zadania. Boli analyzované súčasné metódy rozpoznávania emócií vrátane tradičných prístupov a moderných algoritmov založených na hlbokom učení. Dôkladne boli preskúmané princípy biometrických systémov, techniky detekcie tváre a klasifikácie emócií. Výsledkom bola implementovaná architektúra systému ResEmoteNet, ktorá kombinuje konvolučné vrstvy, SE bloky a reziduálne bloky. Tento model bol trénovaný na verejných datasetoch ako FER2013 a RAF-DB, čo zabezpečilo jeho vysokú generalizovateľnosť.

Implementácia riešenia bola vykonaná v prostredí ROS2, čím bola zabezpečená jeho možná integrácia do robotických systémov. Systém bol testovaný a validovaný na reálnych aj simulovaných dátach v rôznych podmienkach. Experimentálne výsledky ukázali, že model dosahuje vysokú presnosť pri klasifikácii väčšiny základných emócií. Najnižšiu úspešnosť model vykazoval pri detekcii komplexnejších emócií ako strach a smútok, čo koreluje aj s výsledkami dotazníkového experimentu, kde mali s identifikáciou týchto emócií problémy aj ľudskí pozorovatelia.

Systém bol ďalej testovaný s rôznymi kamerami a v odlišných svetelných podmienkach, čo potvrdilo jeho robustnosť a praktickú využiteľnosť. Ukázalo sa, že návrh systému je dostatočne efektívny na to, aby mohol byť nasadený aj na zariadeniach s obmedzeným výpočtovým výkonom.

Medzi hlavné prínosy práce patrí vytvorenie funkčného, trénovateľného a rozšíriteľného systému, ktorý je pripravený na reálne nasadenie v robotických aplikáciách. Práca zároveň identifikovala viaceré oblasti na ďalší výskum, ako je rozšírenie systému o ďalšie vstupy, zlepšenie presnosti pre jemné emócie, alebo nasadenie v multimodálnych interakčných systémoch.

Záverom možno konštatovať, že navrhnutý systém spĺňa požiadavky zadania, predstavuje významný krok k emocionálne inteligentnej robotike a otvára priestor na ďalšie zlepšenia a praktické aplikácie v oblastiach, kde je porozumenie ľudským emóciám kľúčové.

# Literatúra

1. SUCHITRA SAXENA Shikha Tripathi, T. S. An intelligent facial expression recognition system with emotion intensity classification. *Cognitive Systems Research*. 2022, **74**(17), 39–52. ISSN 1389-0417. Dostupné z DOI: 10.1016/j.cogsys.2022.04.001.
2. EKMAN, P. Are there basic emotions? *Psychological Review*. 1992, **99**(3), 550–553. Dostupné z DOI: 10.1037/0033-295x.99.3.550.
3. BISOGNI, C.; CASTIGLIONE, A.; HOSSAIN, S.; NARDUCCI, F.; UMER, S. Impact of Deep Learning Approaches on Facial Expression Recognition in Healthcare Industries. *IEEE Transactions on Industrial Informatics*. 2022, **18**(8), 5619–5627. Dostupné z DOI: 10.1109/TII.2022.3141400.
4. MARTINEZ, B.; VALSTAR, M. F. *Advances in Face Detection and Facial Image Analysis*. Advances, Challenges, and Opportunities in Automatic Facial Expression Recognition. Ed. KAWULOK, M.; CELEBI, M. E.; SMOLKA, B. Cham: Springer International Publishing, 2016. ISBN 978-3-319-25958-1. Dostupné z DOI: 10.1007/978-3-319-25958-1\_4.
5. CANAL, F. Z. et al. A survey on facial emotion recognition techniques: A state-of-the-art literature review. *Information Sciences*. 2022, **582**, 593–617. ISSN 0020-0255. Dostupné z DOI: <https://doi.org/10.1016/j.ins.2021.10.005>.
6. EKMAN, P.; FRIESEN, W. V. *Facial Action Coding System (FACS)* [<https://doi.org/10.1037/t27734-000>]. 1978. Database record.
7. WANG, H.-H.; GU, J.-W. The Applications of Facial Expression Recognition in Human-computer Interaction. In: *2018 IEEE International Conference on Advanced Manufacturing (ICAM)*. 2018, s. 288–291. Dostupné z DOI: 10.1109/AMCON.2018.8614755.
8. ROY, A. K.; KATHANIA, H. K.; SHARMA, A.; DEY, A.; ANSARI, M. S. A. *Re-sEmoteNet: Bridging Accuracy and Loss Reduction in Facial Emotion Recognition*. 2024. Dostupné z arXiv: 2409.10545 [cs.CV].
9. ZHANG, S.; ZHANG, Y.; ZHANG, Y.; WANG, Y.; SONG, Z. A Dual-Direction Attention Mixed Feature Network for Facial Expression Recognition. *Electronics*. 2023, **12**(17). ISSN 2079-9292. Dostupné z DOI: 10.3390/electronics12173595.
10. KO, B. C. A Brief Review of Facial Emotion Recognition Based on Visual Information. *Sensors*. 2018, **18**(2). ISSN 1424-8220. Dostupné z DOI: 10.3390/s18020401.

11. PRADEEP, V.; MADHUSHREE; SUMUKHA, B. S.; RICHARDS, G. R.; PRASHANT, S. P. Facial Emotion Detection using CNN and OpenCV. In: *2024 International Conference on Emerging Technologies in Computer Science for Interdisciplinary Applications (ICETCS)*. 2024, s. 1–6. Dostupné z DOI: 10.1109/ICETCS61022.2024.10543993.

## **Použitie nástrojov umelej inteligencie**

OpenAI (2025), ChatGPT 4o, časť 1, 2, 3, 4, 5, kontrola pravopisu a spravnosti slovosledu.

OpenAI (2025), ChatGPT 4.5, časť 1, generovanie obrazka.

Perplexity.ai (2024), Claude 3.7 Sonnet, časť 6, 7, generovanie textu.

Github Copilot (2024), Copilot X, časť

## Dodatok A: Zdrojový kód pre rozpoznávanie emócií

```
1 import cv2
2 import cv2.data
3 import torch
4 import torch.nn.functional as F
5 import torchvision.transforms as transforms
6 from PIL import Image
7 import numpy as np
8 from approach.ResEmoteNet import ResEmoteNet
9 from ximea import xiapi
10 # Add ROS2 imports
11 import rclpy
12 from rclpy.node import Node
13 from std_msgs.msg import String
14 from flask import Flask, Response, request
15 import threading
16 import signal
17 import sys
18 from sensor_msgs.msg import Image as RosImage
19 from cv_bridge import CvBridge
20 from multiprocessing import Process
21 import time
22 class EmotionRecognitionNode(Node):
23     def __init__(self):
24         super().__init__('emotion_recognition_node')
25         self.running = True
26         self.publisher_ = self.create_publisher(String, 'emotion_prediction', 10)
27
28         self.image_subscriber = self.create_subscription(
29             RosImage,
30             '/rgb_stream/ximea',
31             self.image_callback,
32             10
33         )
34
35         self.bridge = CvBridge()
36         self.last_frame = None # Store the last received image
37         self.last_frame_lock = threading.Lock()
38
39         # Set up model
```

```

40     self.device = torch.device("mps" if torch.backends.mps.is_available() else "
cpu")
41     self.emotions = ['happy', 'surprise', 'sad', 'anger', 'disgust', 'fear', '
neutral']
42     self.model = ResEmoteNet().to(self.device)
43     checkpoint = torch.load('/home/collab/collab_ws/src/facial_expression/
scripts/rafdb_model.pth', weights_only=True)
44     self.model.load_state_dict(checkpoint['model_state_dict'])
45     self.model.eval()
46
47     self.transform = transforms.Compose([
48         transforms.Resize((64, 64)),
49         transforms.Grayscale(num_output_channels=3),
50         transforms.ToTensor(),
51         transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225]),
52     ])
53
54     self.face_classifier = cv2.CascadeClassifier(
55         cv2.data.harcascades + 'haarcascade_frontalface_default.xml'
56     )
57     self.face_net = cv2.dnn.readNetFromCaffe(
58         '/home/collab/collab_ws/src/facial_expression/scripts/deploy.prototxt',
59         '/home/collab/collab_ws/src/facial_expression/scripts/
res10_300x300_ssd_iter_140000.caffemodel'
60     )
61
62     self.font = cv2.FONT_HERSHEY_SIMPLEX
63     self.font_scale = 1.2
64     self.font_color = (0, 255, 0)
65     self.thickness = 3
66     self.line_type = cv2.LINE_AA
67
68     self.max_emotion = ''
69     self.counter = 0
70     self.evaluation_frequency = 5
71
72     self.timer = self.create_timer(0.1, self.process_frame)
73
74     self.flask_thread = threading.Thread(target=self.run_flask, daemon=True)
75     self.flask_thread.start()
76     # self.thread = threading.Thread(target=self.run_flask)
77     # self.thread.start()
78
79     def process_frame_for_web(self):

```

```

80     try:
81         with self.last_frame_lock:
82             if self.last_frame is None:
83                 print("No frame available")
84                 return None
85                 image = self.last_frame.copy()
86
87                 #print(f"[DEBUG] Before resize: {image.shape}, dtype: {image.dtype}")
88                 frame = cv2.resize(image, (800, 800))
89                 #print(f"[DEBUG] After resize: {frame.shape}, dtype: {frame.dtype}")
90
91                 if self.max_emotion:
92                     cv2.putText(frame, self.max_emotion, (10, 40), self.font,
93                                self.font_scale, self.font_color, self.thickness, self.
line_type)
94
95                 ret, jpeg = cv2.imencode('.jpg', frame)
96                 if not ret:
97                     print("[ERROR] JPEG encoding failed.")
98                     return None
99
100                 #print("[DEBUG] JPEG encoding success.")
101                 return jpeg.tobytes()
102             except Exception as e:
103                 print(f"[ERROR] in process_frame_for_web: {e}")
104                 return None
105
106
107
108
109
110     def gen_frames(self):
111         while self.running:
112             try:
113                 frame_bytes = self.process_frame_for_web()
114                 #_ = len(frame_bytes) # Forces evaluation without printing
115                 #frame_bytes[:1] # Access forces evaluation
116                 if frame_bytes is None:
117                     time.sleep(0.05)
118                     continue
119                 time.sleep(0.05)
120                 yield (b'--frame\r\n'
121                       b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n\r\n')
122             except Exception as e:
123                 self.get_logger().error(f"Error in gen_frames: {e}")

```



```

124         time.sleep(0.1)
125     def gen(self):
126         """Generate Server-Sent Events (SSE) for emotion updates"""
127         while True:
128             try:
129                 # Format as Server-Sent Event
130                 if self.max_emotion:
131                     yield f"data: {self.max_emotion}\n\n"
132                 else:
133                     yield f"data: waiting...\n\n"
134                 # Sleep to avoid flooding the client
135                 time.sleep(0.5)
136             except Exception as e:
137                 self.get_logger().error(f"Error in gen(): {e}")
138                 yield f"data: error\n\n"
139                 time.sleep(1)
140     def run_flask(self):
141         app = Flask(__name__)
142
143         @app.route('/video_feed')
144         def video_feed():
145             return Response(self.gen_frames(),
146                             mimetype='multipart/x-mixed-replace; boundary=frame')
147
148         @app.route('/emotion')
149         def emotion():
150             return Response(self.gen(), mimetype='text/event-stream')
151         @app.route('/shutdown')
152         def shutdown():
153             func = request.environ.get('werkzeug.server.shutdown')
154             if func is None:
155                 raise RuntimeError('Not running with the Werkzeug Server')
156             func()
157             return 'Server shutting down...'
158         @app.route('/')
159         def index():
160             return '''
161             <html>
162             <head>
163                 <title>Emotion Recognition Stream</title>
164                 <style>
165                     #emotion { font-size: 24px; margin-top: 20px; font-weight: bold; }
166                     body { font-family: Arial, sans-serif; }
167                 </style>
168             </head>

```

```

169         <body>
170             <h1>Live Emotion Detection</h1>
171             
172             <div id="emotion">Detected emotion: waiting...</div>
173             <div id="status">Connection status: connecting...</div>
174             <script>
175                 const eventSource = new EventSource('/emotion');
176                 eventSource.onmessage = function(e) {
177                     console.log('Received emotion:', e.data);
178                     document.getElementById('emotion').innerHTML = 'Detected emotion
: ' + e.data;
179                 };
180                 eventSource.onopen = function() {
181                     document.getElementById('status').innerHTML = 'Connection status
: connected';
182                 };
183                 eventSource.onerror = function(e) {
184                     document.getElementById('status').innerHTML = 'Connection status
: error/reconnecting';
185                     console.error('EventSource error:', e);
186                 };
187             </script>
188         </body>
189     </html>
190     '''
191
192     app.run(host='0.0.0.0', port=5000, threaded=True)
193
194
195
196     def detect_emotion(self, video_frame):
197         vid_fr_tensor = self.transform(video_frame).unsqueeze(0).to(self.device)
198         with torch.no_grad():
199             outputs = self.model(vid_fr_tensor)
200             probabilities = F.softmax(outputs, dim=1)
201             scores = probabilities.cpu().numpy().flatten()
202             rounded_scores = [round(score, 2) for score in scores]
203             return rounded_scores
204
205     def get_max_emotion(self, x, y, w, h, video_frame):
206         crop_img = video_frame[y : y + h, x : x + w]
207         pil_crop_img = Image.fromarray(crop_img)
208         rounded_scores = self.detect_emotion(pil_crop_img)
209         max_index = np.argmax(rounded_scores)
210         max_emotion = self.emotions[max_index]

```

```

211         return max_emotion, rounded_scores[max_index]
212
213     def print_max_emotion(self, x, y, video_frame, max_emotion):
214         org = (x, y - 15)
215         cv2.putText(video_frame, max_emotion, org, self.font, self.font_scale,
216                     self.font_color, self.thickness, self.line_type)
217
218     def print_all_emotion(self, x, y, w, h, video_frame):
219         crop_img = video_frame[y : y + h, x : x + w]
220         pil_crop_img = Image.fromarray(crop_img)
221         rounded_scores = self.detect_emotion(pil_crop_img)
222         org = (x + w + 10, y - 20)
223         for index, value in enumerate(self.emotions):
224             emotion_str = (f'{value}: {rounded_scores[index]:.2f}')
225             y = org[1] + 40
226             org = (org[0], y)
227             cv2.putText(video_frame, emotion_str, org, self.font, self.font_scale,
228                         self.font_color, self.thickness, self.line_type)
229
230     def detect_bounding_box(self, video_frame):
231         gray_image = cv2.cvtColor(video_frame, cv2.COLOR_BGR2GRAY)
232         faces = self.face_classifier.detectMultiScale(gray_image, 1.1, 5, minSize
233         =(40, 40))
234         for (x, y, w, h) in faces:
235             # Draw bounding box on face
236
237             cv2.rectangle(video_frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
238
239             # Crop bounding box
240             if self.counter == 0:
241                 self.max_emotion, max_probability = self.get_max_emotion(x, y, w, h,
242                 video_frame)
243
244                 max_probability *= 100
245
246                 # Publish emotion to ROS2 topic
247                 msg = String()
248                 # Include probability in the message
249                 msg.data = f'{self.max_emotion} ({max_probability:.2f}%)'
250                 self.publisher_.publish(msg)
251                 # self.get_logger().info(f'Publishing: {msg.data}')
252
253                 # self.print_max_emotion(x, y, video_frame, self.max_emotion)
254                 # self.print_all_emotion(x, y, w, h, video_frame)

```

```

254     return faces
255 def detect_bounding_box_sgg(self, video_frame):
256     h, w = video_frame.shape[:2]
257     # Create a 300x300 blob from the image
258     blob = cv2.dnn.blobFromImage(cv2.resize(video_frame, (300, 300)), 1.0,
259                                   (300, 300), (104.0, 177.0, 123.0))
260
261     # Pass the blob through the network and get detections
262     self.face_net.setInput(blob)
263     detections = self.face_net.forward()
264
265     # Process detections
266     faces_detected = 0
267     for i in range(detections.shape[2]):
268         confidence = detections[0, 0, i, 2]
269
270         # Filter out weak detections
271         if confidence > 0.5:
272             faces_detected += 1
273             # Get the coordinates of the bounding box
274             box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
275             (x1, y1, x2, y2) = box.astype("int")
276
277             # Ensure coordinates are within the frame
278             x1, y1 = max(0, x1), max(0, y1)
279             x2, y2 = min(w, x2), min(h, y2)
280
281             # Calculate width and height
282             face_w, face_h = x2 - x1, y2 - y1
283
284             # Skip if dimensions are too small
285             if face_w < 20 or face_h < 20:
286                 continue
287
288             # Draw bounding box
289             # cv2.rectangle(video_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
290
291             # Process for emotion detection if needed
292             if self.counter == 0:
293                 self.max_emotion, max_probability = self.get_max_emotion(
294                     x1, y1, face_w, face_h, video_frame)
295
296                 max_probability *= 100
297
298             # Publish emotion to ROS2 topic

```

```

300         msg = String()
301         msg.data = f'{self.max_emotion} ({max_probability:.2f}%)'
302         self.publisher_.publish(msg)
303         # self.get_logger().info(f'Publishing: {msg.data}')
304
305         # Optional: Display emotion on the frame
306         # self.print_max_emotion(x1, y1, video_frame, self.max_emotion)
307
308     return faces_detected
309
310 def image_callback(self, msg):
311     try:
312         cv_image = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
313         if cv_image.dtype != np.uint8:
314             cv_image = cv_image.astype(np.uint8)
315
316         if len(cv_image.shape) == 2: # grayscale image
317             cv_image = cv2.cvtColor(cv_image, cv2.COLOR_GRAY2BGR)
318
319         with self.last_frame_lock:
320             self.last_frame = cv_image
321             #self.get_logger().info(f"Received image: shape={cv_image.shape}, dtype
322             ={cv_image.dtype}")
323         except Exception as e:
324             self.get_logger().error(f"Error converting ROS image: {e}")
325
326 def process_frame(self):
327     try:
328         with self.last_frame_lock:
329             if self.last_frame is None:
330                 return
331             frame = self.last_frame.copy()
332
333             frame = cv2.resize(frame, (480, 480))
334             self.detect_bounding_box(frame)
335
336             self.counter += 1
337             if self.counter == self.evaluation_frequency:
338                 self.counter = 0
339         except Exception as e:
340             self.get_logger().error(f"Error in process_frame: {e}")
341
342 def cleanup(self):
343     """Clean up resources when shutting down"""
344     self.get_logger().info('Shutting down...')

```

```

343     self.running = False
344
345     # Close OpenCV windows (even though you're not showing them)
346     cv2.destroyAllWindows()
347
348
349
350     self.get_logger().info('Cleanup complete')
351
352 def signal_handler(sig, frame):#
353     """Handle Ctrl+C and other termination signals"""
354     print('\nReceived termination signal. Shutting down...')
355     if 'node' in globals():
356         node.cleanup()
357     rclpy.shutdown()
358     sys.exit(0)
359 def main(args=None):
360     # Set up signal handlers
361     signal.signal(signal.SIGINT, signal_handler) # Ctrl+C
362     signal.signal(signal.SIGTERM, signal_handler) # Termination request
363
364     rclpy.init(args=args)
365
366     global node
367     node = EmotionRecognitionNode()
368
369     try:
370         rclpy.spin(node)
371     except KeyboardInterrupt:
372         pass
373     except Exception as e:
374         print(f"Exception in main loop: {e}")
375     finally:
376         # Ensure cleanup happens
377         try:
378             node.cleanup()
379         except Exception as e:
380             print(f"Error during cleanup: {e}")
381         rclpy.shutdown()
382
383
384 if __name__ == '__main__':
385     main()

```

Výpis kódu 1: Python skript pre rozpoznávanie emócií

## Dodatok B: Zdrojový kód C++

```
1 #include "rclcpp/rclcpp.hpp"
2 #include "std_msgs/msg/string.hpp"
3
4 class PredictionSubscriber : public rclcpp::Node {
5 public:
6     PredictionSubscriber() : Node("prediction_subscriber") {
7         subscription_ = this->create_subscription<std_msgs::msg::String>(
8             "emotion_prediction", 10,
9             std::bind(&PredictionSubscriber::topic_callback, this, std::placeholders
10             ::_1));
11     }
12 private:
13     void topic_callback(const std_msgs::msg::String::SharedPtr msg) const {
14         RCLCPP_INFO(this->get_logger(), "Received prediction: '%s'", msg->data.c_str
15         ());
16     }
17
18     rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscription_;
19 };
20
21 int main(int argc, char **argv) {
22     rclcpp::init(argc, argv);
23     std::cout << "Prediction subscriber started" << std::endl;
24     rclcpp::spin(std::make_shared<PredictionSubscriber>());
25     rclcpp::shutdown();
26     return 0;
27 }
```

Výpis kódu 2: Implementácia rozpoznávania emócií v C++

```
1 cmake_minimum_required(VERSION 3.8)
2 project(facial_expression)
3
4 if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
5     add_compile_options(-Wall -Wextra -Wpedantic)
6 endif()
7
8 # Nájdeme potrebné balíky
9 find_package(ament_cmake REQUIRED)
```

```

10 find_package(rclcpp REQUIRED)
11 find_package(std_msgs REQUIRED)
12 find_package(OpenCV REQUIRED)
13
14 # Kompilácia C++ uzla
15 add_executable(prediction_node src/prediction.cpp)
16 ament_target_dependencies(prediction_node rclcpp std_msgs OpenCV)
17
18 # Inštalácia C++ uzla
19 install(TARGETS prediction_node
20   DESTINATION lib/${PROJECT_NAME})
21
22 # Povolenie spúšťania Python uzla
23 find_package(ament_cmake_python REQUIRED)
24
25 # Nastavenie exekutovateľného Python skriptu
26 install(PROGRAMS
27   scripts/prediction.py
28   DESTINATION lib/${PROJECT_NAME})
29 # Install launch files
30 install(DIRECTORY launch/
31   DESTINATION share/${PROJECT_NAME}/
32 )
33 ament_package()

```

Výpis kódu 3: CMakeLists.txt pre projekt rozpoznávania emócií



## Dodatok C: Zdrojový kód pre streamovanie snímok z kamier

```
1 # camera_streamer_node.py
2 import rclpy
3 from rclpy.node import Node
4 from sensor_msgs.msg import Image
5 from cv_bridge import CvBridge
6 import cv2
7 from ximea import xiapi
8
9 class CameraStreamer(Node):
10     def __init__(self):
11         super().__init__('camera_streamer')
12         self.bridge = CvBridge()
13         self.timer_period = 0.1 # 10 FPS
14
15         # Try Ximea first
16         try:
17             self.get_logger().info('Trying to open Ximea camera...')
18             self.ximea = xiapi.Camera()
19             self.ximea.open_device()
20             self.ximea.set_exposure(50000)
21             self.ximea.set_param("imgdataformat", "XI_RGB24")
22             self.ximea.set_param("auto_wb", 1)
23             self.ximea_img = xiapi.Image()
24             self.ximea.start_acquisition()
25             self.ximea_pub = self.create_publisher(Image, '/rgb_stream/ximea', 10)
26             self.create_timer(self.timer_period, self.publish_ximea)
27             self.get_logger().info('Ximea camera started.')
28         except Exception as e:
29             self.get_logger().warn(f'Ximea unavailable: {e}')
30             self.ximea = None
31
32         # Try default camera
33         self.default_cam = cv2.VideoCapture(0)
34         if self.default_cam.isOpened():
35             self.default_pub = self.create_publisher(Image, '/rgb_stream/default',
10)
36
37             self.create_timer(self.timer_period, self.publish_default)
38             self.get_logger().info('Default camera started.')
```

```

38         else:
39             self.get_logger().warn('Default camera unavailable.')
40             self.default_cam = None
41
42     def publish_ximea(self):
43         if self.ximea:
44             self.ximea.get_image(self.ximea_img)
45             frame = self.ximea_img.get_image_data_numpy()
46             frame = frame[:, :, [2, 1, 0]] # RGB to BGR
47             msg = self.bridge.cv2_to_imgmsg(frame, encoding="rgb8")
48             self.ximea_pub.publish(msg)
49
50     def publish_default(self):
51         if self.default_cam:
52             ret, frame = self.default_cam.read()
53             if ret:
54                 msg = self.bridge.cv2_to_imgmsg(frame, encoding="bgr8")
55                 self.default_pub.publish(msg)
56
57 def main(args=None):
58     rclpy.init(args=args)
59     node = CameraStreamer()
60     rclpy.spin(node)
61     if node.default_cam:
62         node.default_cam.release()
63     if node.ximea:
64         node.ximea.stop_acquisition()
65         node.ximea.close_device()
66     rclpy.shutdown()
67
68 if __name__ == '__main__':
69     main()

```

Výpis kódu 4: Python publisher snímok z kamier

## Dodatok D: Zdrojový kód pre trenovanie modelu

```
1 % Obsah Jupyter notebooku
2 import torch
3 print("CUDA available:", torch.cuda.is_available())
4 print("Device count:", torch.cuda.device_count())
5 print("Current device:", torch.cuda.current_device())
6 print("Device name:", torch.cuda.get_device_name(0) if torch.cuda.is_available()
    else "None")
7 torch.cuda.empty_cache()
8
9 import torch
10 import pandas as pd
11 import numpy as np
12 from tqdm import tqdm
13
14 from torch.utils.data import DataLoader
15 from torchvision import transforms
16 import torch.optim as optim
17 import matplotlib.pyplot as plt
18
19 from approach.ResEmoteNet import ResEmoteNet
20
21 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
22 print(f"Using {device} device")
23
24 # Transform the dataset
25 transform = transforms.Compose([
26     transforms.Resize((64, 64)),
27     transforms.Grayscale(num_output_channels=3),
28     transforms.RandomHorizontalFlip(),
29     transforms.ToTensor(),
30     transforms.Normalize(
31         mean=[0.485, 0.456, 0.406],
32         std=[0.229, 0.224, 0.225]
33     )
34 ])
35 # Load the model
36 model = ResEmoteNet()
37 model.to('cuda')
38 # Print the number of parameters
39 total_params = sum(p.numel() for p in model.parameters())
```

```

40 print(f'{total_params:,} total parameters.')
```

```

41
```

```

42 import os
```

```

43 from torch.utils.data import DataLoader, random_split
```

```

44 from torchvision import datasets, transforms
```

```

45
```

```

46 def load_data(base_dir, batch_size=16, transform=None):
```

```

47     """
```

```

48     Function to load train and test datasets and return their DataLoaders.
```

```

49
```

```

50     Args:
```

```

51         base_dir (str): Base directory containing train/test subdirectories.
```

```

52         batch_size (int): Batch size for the DataLoader.
```

```

53         transform (callable, optional): Transformations to apply to the images.
```

```

54
```

```

55     Returns:
```

```

56         tuple: Train DataLoader, Validation DataLoader
```

```

57     """
```

```

58     # Define train and test directories
```

```

59     train_dir = os.path.join(base_dir, "train")
```

```

60     val_dir = os.path.join(base_dir, "test")
```

```

61
```

```

62     # Use torchvision.datasets.ImageFolder to automatically handle class folders
```

```

63     train_dataset = datasets.ImageFolder(root=train_dir, transform=transform)
```

```

64     test_dataset = datasets.ImageFolder(root=val_dir, transform=transform)
```

```

65
```

```

66     label_names = train_dataset.classes
```

```

67
```

```

68     # Split train_dataset into training and validation sets
```

```

69     val_size = 1533
```

```

70     train_size = len(train_dataset) - val_size
```

```

71     train_subset, val_subset = random_split(train_dataset, [train_size, val_size])
```

```

72
```

```

73     # DataLoaders
```

```

74     train_loader = DataLoader(train_subset, batch_size=batch_size, shuffle=True)
```

```

75     val_loader = DataLoader(val_subset, batch_size=batch_size, shuffle=False)
```

```

76     test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
```

```

77
```

```

78
```

```

79     # Print dataset sizes
```

```

80     print(f"Number of images in train loader: {len(train_loader.dataset)}")
```

```

81     print(f"Number of images in val loader: {len(val_loader.dataset)}")
```

```

82     print(f"Number of images in test loader: {len(test_loader.dataset)}")
```

```

83
```

```

84     return train_loader, test_loader, val_loader, label_names
```

```

85
86
87 # Load data
88 base_dir = "/workspace/RAF-DB"
89 train_loader, test_loader, val_loader, label_names = load_data(base_dir=base_dir,
    batch_size=16, transform=transform)
90
91 # Inspect a batch
92 train_images, train_labels = next(iter(train_loader))
93 print(f"Train batch: Images shape {train_images.shape}, Labels shape {train_labels.
    shape}")
94
95 test_images, test_labels = next(iter(test_loader))
96 print(f"Train batch: Images shape {test_images.shape}, Labels shape {test_labels.
    shape}")
97
98 Number of images in train loader: 12273
99 Number of images in val loader: 1533
100 Number of images in test loader: 1533
101 Train batch: Images shape torch.Size([16, 3, 64, 64]), Labels shape torch.Size([16])
102 Train batch: Images shape torch.Size([16, 3, 64, 64]), Labels shape torch.Size([16])
103
104 # Hyperparameters
105 criterion = torch.nn.CrossEntropyLoss()
106 optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9, weight_decay=1e-4)
107
108 patience = 15
109 best_val_acc = 0
110 patience_counter = 0
111 epoch_counter = 0
112
113 num_epochs = 100
114
115 train_losses = []
116 val_losses = []
117 train_accuracies = []
118 val_accuracies = []
119 test_losses = []
120 test_accuracies = []
121
122 # Start training
123 for epoch in range(num_epochs):
124     model.train()
125     running_loss = 0.0
126     correct = 0

```

```

127     total = 0
128
129     for data in tqdm(train_loader, desc=f"Epoch {epoch+1}/{num_epochs}"):
130         inputs, labels = data[0].to(device), data[1].to(device)
131         optimizer.zero_grad()
132         outputs = model(inputs)
133         loss = criterion(outputs, labels)
134         loss.backward()
135         optimizer.step()
136
137         running_loss += loss.item()
138         _, predicted = torch.max(outputs.data, 1)
139         total += labels.size(0)
140         correct += (predicted == labels).sum().item()
141
142     train_loss = running_loss / len(train_loader)
143     train_acc = correct / total
144     train_losses.append(train_loss)
145     train_accuracies.append(train_acc)
146
147     model.eval()
148     test_running_loss = 0.0
149     test_correct = 0
150     test_total = 0
151     with torch.no_grad():
152         for data in test_loader:
153             inputs, labels = data[0].to(device), data[1].to(device)
154             outputs = model(inputs)
155             loss = criterion(outputs, labels)
156             test_running_loss += loss.item()
157             _, predicted = torch.max(outputs.data, 1)
158             test_total += labels.size(0)
159             test_correct += (predicted == labels).sum().item()
160
161     test_loss = test_running_loss / len(test_loader)
162     test_acc = test_correct / test_total
163     test_losses.append(test_loss)
164     test_accuracies.append(test_acc)
165
166     model.eval()
167     val_running_loss = 0.0
168     val_correct = 0
169     val_total = 0
170     with torch.no_grad():
171         for data in val_loader:

```

```

172         inputs, labels = data[0].to(device), data[1].to(device)
173         outputs = model(inputs)
174         loss = criterion(outputs, labels)
175         val_running_loss += loss.item()
176         _, predicted = torch.max(outputs.data, 1)
177         val_total += labels.size(0)
178         val_correct += (predicted == labels).sum().item()
179
180     val_loss = val_running_loss / len(val_loader)
181     val_acc = val_correct / val_total
182     val_losses.append(val_loss)
183     val_accuracies.append(val_acc)
184
185     print(f"Epoch {epoch+1}, Train Loss: {train_loss}, Train Accuracy: {
train_acc}, Test Loss: {test_loss}, Test Accuracy: {test_acc}, Val Loss: {
val_loss}, Val Accuracy: {val_acc}")
186     epoch_counter += 1
187
188     if val_acc > best_val_acc:
189         best_val_acc = val_acc
190         patience_counter = 0
191         torch.save(model.state_dict(), 'best_model_RAF-DB_1.pth')
192     else:
193         patience_counter += 1
194         print(f"No improvement in validation accuracy for {
patience_counter} epochs.")
195
196     if patience_counter > patience:
197         print("Stopping early due to lack of improvement in
validation accuracy.")
198         break

```

Výpis kódu 5: Python skript použitý na trenovanie modelu

## Dodatok E: Zdrojový kód pre Docker

```
1 # Use CUDA 11.2 and CUDNN 8 runtime with Ubuntu 20.04 as the base image
2 FROM nvidia/cuda:11.2.2-cudnn8-runtime-ubuntu20.04
3
4 # Set environment variables for non-interactive installation and Python path
5 ENV DEBIAN_FRONTEND=noninteractive
6 ENV PYTHON_VERSION=3.9
7 ENV PATH /usr/local/cuda/bin:$PATH
8
9 # Install dependencies and set up Python 3.9 environment
10 RUN apt-get update && \
11     apt-get install -y --no-install-recommends \
12     software-properties-common && \
13     add-apt-repository ppa:deadsnakes/ppa && \
14     apt-get update && \
15     apt-get install -y --no-install-recommends \
16     build-essential \
17     curl \
18     ca-certificates \
19     python3.9 \
20     python3.9-distutils \
21     python3.9-dev \
22     python3-pip \
23     python3-setuptools \
24     python3-venv \
25     libopenblas-dev \
26     libopencv-dev \
27     && rm -rf /var/lib/apt/lists/*
28
29 # Upgrade pip and install required Python packages with specified versions
30 # Upgrade pip and install required Python packages with specified versions
31 RUN python3.9 -m pip install --upgrade pip && \
32     python3.9 -m pip install \
33     dlib==19.24.2 \
34     matplotlib==3.8.3 \
35     numpy==1.26.4 \
36     opencv_python==4.9.0.80 \
37     pandas==2.2.2 \
38     Pillow==10.3.0 \
39     retina_face==0.0.14 \
40     seaborn==0.13.2 \
```



```

41     torch==2.1.2 \
42     torchvision==0.16.2 \
43     tqdm==4.66.1 \
44     urllib3==2.2.1 \
45     jupyter
46 # Downgrade protobuf to resolve MediaPipe and TensorFlow compatibility issues
47 RUN python3.9 -m pip install protobuf==3.20.*
48
49 # Set Python 3.9 as the default Python version and link pip
50 RUN ln -sf /usr/bin/python3.9 /usr/bin/python && \
51     ln -sf /usr/bin/pip3 /usr/bin/pip
52
53 # Set the default working directory inside the container
54 WORKDIR /workspace
55
56 # Copy local files to the container's workspace directory
57 COPY . .
58
59 # Expose the port for Jupyter Notebook
60 EXPOSE 8888
61
62 # Command to run Jupyter Notebook when the container starts
63 CMD ["jupyter", "notebook", "--ip=0.0.0.0", "--port=8888", "--no-browser", "--allow-
    root", "--NotebookApp.token=''"]

```

Výpis kódu 6: Dockerfile pre rozpoznávanie emócií

```

1 services:
2   tensorflow_gpu:
3     build: .
4     container_name: resEmoteNet
5     runtime: nvidia
6     environment:
7       - NVIDIA_VISIBLE_DEVICES=all
8       - NVIDIA_DRIVER_CAPABILITIES=compute,utility
9       - PYTHONUNBUFFERED=1
10      - CUDA_LAUNCH_BLOCKING=1 # Add this line
11     deploy:
12       resources:
13         reservations:
14           devices:
15             - driver: nvidia
16               count: all
17               capabilities: [gpu]
18     volumes:
19       - ./workspace

```

```
20     working_dir: /workspace
21     stdin_open: true
22     tty: true
23     ports:
24     - "8888:8888"
25     command: >
26     bash -c "pip install notebook &&
27     jupyter notebook --ip=0.0.0.0 --port=8888 --no-browser --allow-root"
```

Výpis kódu 7: docker-compose.yml pre systém rozpoznávania emócií

## Dodatok F: Používateľský manuál

### F.1 Návod na tréovanie modelu

Tento návod vás prevedie procesom tréovania modelu pre rozpoznávanie emócií pomocou Docker kontajnera a webového rozhrania.

#### F.1.1 Požiadavky

Na tréovanie modelu potrebujete:

- Nainštalovaný Docker (<https://docs.docker.com/get-docker/>)
- Nainštalovaný Docker Compose (<https://docs.docker.com/compose/install/>)
- Minimálne 8 GB RAM a 4 GB GPU pamäte
- Dostatočné miesto na disku (minimálne 10 GB)

#### F.1.2 Postup tréovania

##### 1. Príprava projektu

```
1 git clone https://github.com/KocurMaros/master_thesis_practical
2 cd master_thesis_practical
3
```

##### 2. Zostavenie Docker kontajnera

V koreňovom adresári projektu spustíte:

```
1 docker-compose build
2
```

Tento príkaz zostaví Docker kontajner podľa definícií v súboroch Dockerfile a docker-compose.yml.

##### 3. Spustenie kontajnera

Po úspešnom zostavení spustíte kontajner:

```
1 docker-compose up
2
```

#### 4. Prístup k webovému rozhraniu

Po spustení kontajnera otvorte webový prehliadač a prejdite na adresu:

```
1 http://localhost:5000
2
```

#### 5. Spustenie tréovania

Vo webovom rozhraní:

- Vyberte dataset, ktorý chcete použiť na tréovanie
- Nastavte hyperparametre (počet epoch, batch size, learning rate)
- Kliknite na tlačidlo "Spustiť tréovanie"
- Sledujte priebeh tréovania a výsledné metriky

#### 6. Export natréovaného modelu

Po ukončení tréovania si môžete stiahnuť natréovaný model kliknutím na tlačidlo "Exportovať model".

## F.2 Návod na testovanie s ROS2

Tento návod popisuje, ako spustiť systém rozpoznávania emócií pomocou ROS2 Humble na Ubuntu 22.04.

### F.2.1 Požiadavky

Na testovanie potrebujete:

- Ubuntu 22.04 LTS
- ROS2 Humble (<https://docs.ros.org/en/humble/Installation.html>)
- Kamera kompatibilná s ROS2
- Natréovaný model (výstup z procesu tréovania)

### F.2.2 Inštalácia a zostavenie

#### 1. Príprava ROS2 workspace

```
1 mkdir -p ~/ros2_ws/src
2 cd ~/ros2_ws/
3 cp -r /path/to/your/master_thesis_practical/facial_expression ~/
  ros2_ws/src/
4
```

## 2. Zostavenie balíka

```
1 colcon build --symlink-install --packages-select  
  emotion_recognition
```

## 3. Načítanie prostredia ROS2

```
1 source install/setup.bash
```

### F.2.3 Spustenie systému

#### 1. Spustenie video streamu

Najprv spustíte skript pre získavanie videa z kamery:

```
1 ./scripts/video_stream.sh
```

#### 2. Spustenie rozpoznávania emócií

Po spustení video streamu spustíte rozpoznávanie emócií:

```
1 ./scripts/run_facial_expression.sh
```

Tento skript aktivuje:

- Detekciu tváre
- Predikčný node pre rozpoznávanie emócií
- Webový server pre vizualizáciu výsledkov

#### 3. Prístup k výsledkom

Výsledky rozpoznávania emócií sú dostupné:

- Vo webovom rozhraní: <http://localhost:8080>
- V ROS2 topicoch:

```
1 ros2 topic echo /emotion_prediction    # výsledok predikcie  
2 ros2 topic echo /face_detection        # detekované tváre  
3
```

### F.2.4 Riešenie problémov

- **Problém s kamerou:** Skontrolujte, či je kamera rozpoznaná systémom pomocou príkazu `v4l2-ctl -list-devices`
- **Problém s ROS2 nodmi:** Skontrolujte stav nodov pomocou `ros2 node list`
- **Chyba predikcie:** Uistite sa, že cesta k modelu v konfiguračnom súbore je správna

### F.2.5 Ukončenie systému

Na ukončenie všetkých bežiacich nodov stlačte **Ctrl+C** v termináloch, kde sú spustené skripty, alebo použite:

```
1 ros2 lifecycle set /emotion_recognition_node shutdown
```