

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

**Návrh systému na rozpoznávanie znakov
posunkovej reči v reálnom čase s využitím
umelej inteligencie**

Diplomová práca

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

**Návrh systému na rozpoznávanie znakov
posunkovej reči v reálnom čase s využitím
umelej inteligencie**

Diplomová práca

Študijný program: Informatika
Študijný odbor: Informatika
Školiace pracovisko: Katedra počítačov a informatiky (KPI)
Školiteľ: Ing. Peter Gnip, PhD.

Košice 2026

Bc. Matúš Sobinovský

Abstrakt v SJ

Diplomová práca sa zaoberá problematikou rozpoznávania znakov posunkovej reči v reálnom čase využitím prístupov počítačového videnia. Na detekciu rúk a ich kľúčových bodov bola využitá architektúra *YOLO11n-pose*, ktorá dosiahla hodnotu Box mAP(50-95) 0.9790 a Pose mAP(50-95) 0.9487 s rýchlosťou 34.63 FPS na mobilnom zariadení. Pri klasifikácii znakov posunkovej reči dosiahla najlepšie výsledky architektúra *MobileNetV2* využívajúca informácie o kľúčových bodoch ruky s hodnotou AUC skóre 0.9850 a G-mean 0.9371 pri rýchlosti 69.29 FPS na mobilnom zariadení. Modely boli optimalizované pre nasadenie na mobilných zariadeniach. Systém bol doplnený o modul textovej korekcie využívajúci veľký jazykový model *Gemini 2.5 Flash*. Všetky moduly boli integrované do mobilnej aplikácie pre platformu Android, ktorá bola podrobená používateľskému testovaniu s dosiahnutým SUS skóre 88.75.

Kľúčové slová v SJ

posunková reč, počítačové videnie, Android, konvolučné neurónové siete, veľké jazykové modely

Abstrakt v AJ

The diploma thesis deals with the problem of real-time sign language recognition using computer vision approaches. For hand detection and keypoint extraction, the *YOLO11n-pose* architecture was employed, achieving a Box mAP(50-95) of 0.9790 and Pose mAP(50-95) of 0.9487 at 34.63 FPS on a mobile device. For sign language classification, the *MobileNetV2* architecture utilizing hand keypoints informations achieved the best results with an AUC score of 0.9850 and G-mean of 0.9371 at 69.29 FPS on a mobile device. The models were optimized for mobile deployment. The system was extended with a text correction module utilizing the large language model *Gemini 2.5 Flash*. All modules were integrated into an Android mobile application, which was subjected to user testing with an achieved SUS score of 88.75.

Kľúčové slová v AJ

sign language, computer vision, Android, convolutional neural networks, large language models

Bibliografická citácia

SOBINOVSKÝ, Matúš. *Návrh systému na rozpoznávanie znakov posunkovej reči v reálnom čase s využitím umelej inteligencie*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2026. 73s. Vedúci práce: Ing. Peter Gnip, PhD.

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

**ZADANIE
DIPLOMOVEJ PRÁCE**

Študijný odbor: **Informatika**

Študijný program: **Informatika**

Názov práce:

**Návrh systému na rozpoznávanie znakov posunkovej reči v reálnom čase s
využitím umelej inteligencie**

**Design of a Real-Time Sign Language Recognition System Using Artificial
Intelligence**

Študent: **Bc. Matúš Sobinovský**

Školiteľ: **Ing. Peter Gnip, PhD.**

Školiace pracovisko: **Katedra počítačov a informatiky**

Konzultant práce:

Pracovisko konzultanta:

Pokyny na vypracovanie diplomovej práce:

1. Spracovať teoretický prehľad problematiky rozpoznávania posunkovej reči so zameraním na metódy hlbokého učenia využiteľné pri spracovaní obrazu v reálnom čase.
2. Analyzovať súčasné prístupy na rozpoznávanie znakov posunkovej reči a zhodnotiť ich z hľadiska presnosti, výpočtovej efektívnosti a vhodnosti pre spracovanie v reálnom čase na mobilných zariadeniach.
3. Navrhnuť a implementovať model detekcie ruky v obraze a samostatný model klasifikácie vybraných znakov posunkovej reči a zabezpečiť ich optimalizáciu pre nasadenie na mobilnom zariadení.
4. Navrhnuť architektúru systému na rozpoznávanie znakov posunkovej reči v reálnom čase a implementovať ju vo forme mobilnej aplikácie vrátane integrácie vytvorených modelov.
5. Experimentálne overiť navrhnuté riešenie a vyhodnotiť jeho presnosť, výkonnosť a používateľskú použiteľnosť.
6. Vypracovať diplomovú prácu v súlade so štandardmi KPI vrátane používateľskej a systémovej dokumentácie.

Jazyk, v ktorom sa práca vypracuje: **slovenský**

Termín pre odovzdanie práce: **24.04.2026**

Dátum zadania diplomovej práce: **31.10.2025**



n.z. Prof. Ing. Liberios Vokorokos
prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval samostatne s použitím uvedenej odbornej literatúry, ako aj s využitím nástrojov generatívnej umelej inteligencie v súlade s Metodickým pokynom o záverečných a kvalifikačných prácach a Etickým kódexom študenta Technickej univerzity v Košiciach. Umelá inteligencia bola použitá ako podporný nástroj bez porušenia zásad akademickej etiky a autorskej integrity.

Podakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce za jeho čas a odborné vedenie počas riešenia mojej záverečnej práce. Rovnako by som sa rád poďakoval svojim rodičom a priateľom za ich podporu a povzbudzovanie počas celého môjho štúdia.

Obsah

1	Úvod	1
2	Teoretické východiská	4
2.1	Úvod do problematiky posunkovej reči	4
2.1.1	Definícia a význam posunkovej reči	4
2.1.2	Motivácia pre automatické rozpoznávanie posunkovej reči	5
2.2	Rozpoznávanie posunkovej reči	5
2.2.1	Automatické rozpoznávanie posunkovej reči	6
2.2.2	Typy prístupov k rozpoznávaniu posunkovej reči	6
2.2.3	Výzvy automatického rozpoznávania posunkovej reči	6
2.3	Existujúce prístupy rozpoznávania znakov posunkovej reči	7
2.3.1	Prístupy k rozpoznávaniu posunkovej reči	7
2.3.2	Zhrnutie a porovnanie existujúcich prístupov	8
2.4	Počítačové videnie pre detekciu a klasifikáciu rúk	9
2.4.1	Klasifikácia obrazových dát	10
2.4.2	Detekcia objektov v obraze	10
2.5	Konvolučné neurónové siete	11
2.5.1	Základné komponenty CNN	12
2.6	Architektúry CNN pre klasifikáciu a detekciu	15
2.6.1	ResNet50	15
2.6.2	MobileNetV2	16
2.6.3	Modely YOLO pre detekciu rúk	17
2.6.4	Modely YOLO s detekciou kľúčových bodov	19
2.7	Jazykové modely na korekciu textu	20
2.7.1	Transformerová architektúra	20
2.7.2	Využitie LLM na korekciu textu	21
3	Návrh systému na rozpoznávanie znakov posunkovej reči	22
3.1	Funkčné požiadavky	22

3.2	Nefunkčné požiadavky	22
3.3	Architektúra systému	23
3.4	Návrh jednotlivých modulov	23
3.4.1	Modul detekcie rúk	24
3.4.2	Modul klasifikácie znakov posunkovej reči	24
3.4.3	Modul textovej korekcie	25
3.4.4	Mobilná aplikácia	26
4	Charakteristika a spracovanie dát	27
4.1	Dáta z verejných zdrojov	27
4.2	Vlastné dátové sady	30
4.3	Príprava dát pre detekciu rúk	31
4.4	Augmentácia dát	32
4.5	Príprava dát pre klasifikáciu znakov	33
4.6	Rozdelenie dát	34
5	Implementácia systému na rozpoznávanie znakov posunkovej reči	38
5.1	Modul detekcie rúk	38
5.1.1	Hodnotiace metriky pre detekciu rúk a kľúčových bodov	38
5.1.2	Ladenie hyperparametrov	39
5.1.3	Trénovanie modelov s ich optimálnou konfiguráciou	41
5.1.4	Vyhodnotenie natrénovaných modelov	42
5.1.5	Optimalizácia a výber finálnych modelov	43
5.2	Modul klasifikácie znakov posunkovej reči	45
5.2.1	Hodnotiace metriky klasifikácie znakov posunkovej reči	45
5.2.2	Ladenie hyperparametrov	46
5.2.3	Trénovanie modelov s ich optimálnou konfiguráciou	48
5.2.4	Vyhodnotenie natrénovaných modelov	49
5.2.5	Optimalizácia a výber finálnych modelov	50
5.3	Modul textovej korekcie	53
5.3.1	Návrh promptu pre textovú korekciu	53
5.3.2	Vyhodnotenie promptu pre textovú korekciu	54
5.4	Mobilná aplikácia	55
5.4.1	Vývoj mobilnej aplikácie	56
5.4.2	Používateľské testovanie	58
6	Vyhodnotenie systému	60
6.1	Modul detekcie rúk	60

6.2	Modul klasifikácie znakov posunkovej reči	60
6.3	Modul textovej korekcie	61
6.4	Mobilná aplikácia	62
7	Záver	63
	Zoznam použitej literatúry	65
	Zoznam príloh	74
A	Používateľská príručka	75
A.1	Základná obrazovka	75
A.2	Obrazovka načítania modelov	76
A.3	Obrazovka prekladu znakov posunkovej reči do textu	77
A.4	Obrazovka prekladu textu do znakov posunkovej reči	78
B	Systémová príručka	80
B.1	Hardvérové požiadavky systému	80
B.2	Použité knižnice jazyka Dart	80
B.3	Popis jednotlivých tried	81
B.4	Popis integrovaných funkcií v aplikácii	82
B.5	Použité knižnice jazyka Python	83

Zoznam obrázkov

2.1	Štyri základné úlohy počítačového videnia [9]	9
2.2	Princíp detekcie objektov v obraze [9]	10
2.3	Základná architektúra konvolučnej neurónovej siete [18]	11
2.4	Princíp procesu konvolúcie [20]	12
2.5	Princíp techniky dopĺňania nulami [20]	13
2.6	Združujúce operácie máp vlastností	13
2.7	Plne prepojená vrstva pre binárnu klasifikáciu [20]	14
2.8	Architektúra ResNet50 [32]	16
2.9	Architektúra MobileNetV2 [36]	17
2.10	Architektúra YOLOv8n [39]	18
2.11	Architektúra YOLO11n [42]	19
3.1	Architektúra systému na rozpoznávanie znakov posunkovej reči . .	24
4.1	Ukážky vzoriek verejných dátových sád - YOLO formát	28
4.2	Ukážky vzoriek verejných dátových sád - YOLO pose formát	29
4.3	Ukážky vzoriek dátovej sady American Sign Language	29
4.4	Ukážky vzoriek dátových sád CHD Dataset a CHK Dataset	30
4.5	Ukážky vzoriek dátových sád CHDT Dataset a CHKT Dataset	31
4.6	Ukážky augmentovaných vzoriek verejných dátových sád	33
4.7	Ukážky augmentovaných vzoriek vlastných dátových sád	33
4.8	Ukážky vzoriek pred a po procese predspracovania	35
5.1	Hodnoty metriky FPS pre jednotlivé detekčné architektúry po optimalizácii	44
5.2	Box mAP(50-95) skóre pre architektúry určené na detekciu rúk bez kľúčových bodov po optimalizácii	44
5.3	Box mAP(50-95) a Pose mAP(50-95) skóre pre architektúry určené na detekciu rúk a ich kľúčových bodov po optimalizácii	45

5.4	Hodnoty metriky FPS pre jednotlivé klasifikačné architektúry po optimalizácii	51
5.5	AUC skóre a G-mean pre architektúry využívajúce len informácie o polohe ruky po optimalizácii	52
5.6	AUC skóre a G-mean pre architektúry využívajúce aj informácie o kľúčových bodoch ruky po optimalizácii	52
5.7	Úvodná obrazovka mobilnej aplikácie	56
5.8	Obrazovka kamery mobilnej aplikácie	57
5.9	Obrazovka reverzného prekladu	58
A.1	Základná obrazovka mobilnej aplikácie	75
A.2	Obrazovka načítania modelov	76
A.3	Obrazovka prekladu znakov posunkovej reči do textovej podoby . .	77
A.4	Obrazovka prekladu textu do grafickej podoby znakov posunkovej reči	78

Zoznam tabuliek

4.1	Prehľad použitých verejne dostupných dátových sád	27
4.2	Dátové sady použité v rámci dátovej sady HKT Dataset	36
4.3	Dátové sady použité v rámci dátovej sady HDT Dataset	36
4.4	Dátové sady použité v rámci dátovej sady HDCT Dataset	37
4.5	Dátové sady použité v rámci dátovej sady HKCT Dataset	37
5.1	Výsledné Box mAP(50-95) skóre pre architektúry určené na detekciu rúk bez kľúčových bodov po ladení hyperparametrov	40
5.2	Výsledné Box mAP(50-95) a Pose mAP(50-95) skóre pre architektúry určené na detekciu rúk a ich kľúčových bodov po ladení hyperparametrov	40
5.3	Výsledné Box mAP(50-95) skóre pre architektúry určené na detekciu rúk bez kľúčových bodov po trénovaní a testovaní	41
5.4	Výsledné Box mAP(50-95) a Pose mAP(50-95) skóre pre architektúry určené na detekciu rúk a ich kľúčových bodov po trénovaní a testovaní	42
5.5	Výsledné Box mAP(50-95) a FPS skóre pre architektúry určené na detekciu rúk bez kľúčových bodov po finálnom testovaní	42
5.6	Výsledné Box mAP(50-95), Pose mAP(50-95) a FPS skóre pre architektúry určené na detekciu rúk a ich kľúčových bodov po finálnom testovaní	43
5.7	Výsledné AUC skóre a G-mean hodnoty pre architektúry využívajúce len informácie o polohe ruky po ladení hyperparametrov	47
5.8	Výsledné AUC skóre a G-mean hodnoty pre architektúry využívajúce aj informácie o kľúčových bodoch ruky po ladení hyperparametrov	48
5.9	Výsledné AUC skóre a G-mean hodnoty pre architektúry využívajúce len informácie o polohe ruky po trénovaní a testovaní	49

5.10	Výsledné AUC skóre a G-mean hodnoty pre architektúry využívajúce aj informácie o kľúčových bodoch ruky po trénovaní a testovaní	49
5.11	Výsledné AUC skóre, G-mean a FPS hodnoty pre architektúry využívajúce len informácie o polohe ruky po finálnom testovaní . . .	50
5.12	Výsledné AUC skóre, G-mean a FPS hodnoty pre architektúry využívajúce aj informácie o kľúčových bodoch ruky po finálnom testovaní	50
5.13	Výsledky testovania bezpečného promptu	55
5.14	Výsledky používateľského testovania pomocou metodiky SUS . . .	59

Zoznam výpisov

5.1 Bezpečný prompt pre textovú korekciu	53
--	----

1 Úvod

Sluch je jeden zo základných ľudských zmyslov, ktorý umožňuje komunikáciu medzi ľuďmi prostredníctvom reči. Podľa autorov Zhang et al. [1] trpelo v roku 2021 sluchovým postihnutím približne 1,56 miliardy ľudí na celom svete, pričom sa odhaduje, že do roku 2050 by toto číslo mohlo narásť až na 2,45 miliardy ľudí. Pre ľudí s čiastočnou stratou sluchu dnes existuje viacero možností komunikácie, napríklad prostredníctvom načúvacích prístrojov alebo čítania z pier. Naopak, ľudia s úplnou stratou sluchu alebo tí, ktorí sa narodili nepočujúci, sú odkázaní na komunikáciu prostredníctvom posunkovej reči. Napriek tomu zostáva komunikácia medzi nepočujúcimi a počujúcimi ľuďmi výrazne obmedzená, keďže väčšina počujúcich ľudí posunkovú reč neovláda, čo negatívne ovplyvňuje integráciu nepočujúcich ľudí do spoločnosti.

V posledných rokoch otvorili moderné prístupy strojového učenia v oblasti počítačového videnia nové možnosti v automatickom rozpoznávaní znakov posunkovej reči. Tieto technológie umožňujú vytvárať systémy schopné automaticky prekladať posunkovú reč do textovej podoby v reálnom čase, čo by mohlo výrazne uľahčiť každodennú komunikáciu medzi nepočujúcimi a počujúcimi ľuďmi a prispieť k ich lepšej integrácii do spoločnosti. V tejto práci sa zameriavame na rozpoznávanie znakov posunkovej reči, konkrétne jednotlivých písmen abecedy, ktoré predstavujú základný stavebný kameň posunkovej reči.

Hlavným cieľom tejto práce je navrhnúť a implementovať systém na rozpoznávanie znakov posunkovej reči v reálnom čase s využitím metód strojového učenia a počítačového videnia. Navrhnutý systém pozostáva z troch modulov. Prvým modulom je modul detekcie rúk, ktorý zabezpečuje lokalizáciu ruky používateľa a extrakciu jej kľúčových bodov z kamerového vstupu. Výstup tohto modulu slúži ako vstup pre druhý modul, ktorým je modul klasifikácie znakov posunkovej reči, ktorý na základe získaných informácií klasifikuje zobrazovaný znak. Treťou je doplnkový modul textovej korekcie, ktorý využíva veľký jazykový model na zlepšenie gramatickej správnosti generovaného textu. Všetky moduly sú integrované do mobilnej aplikácie na báze operačného systému Android vyvinutej

pomocou frameworku Flutter.

Táto práca tak predstavuje konkrétny krok k zníženiu komunikačnej bariéry medzi nepočujúcimi a počujúcimi ľuďmi prostredníctvom využitia moderných metód umelej inteligencie. Navrhnutý systém kombinuje prístupy počítačového videnia na detekciu a klasifikáciu znakov posunkovej reči s využitím veľkých jazykových modelov na zlepšenie kvality generovaného textu. Vďaka optimalizácii modelov pre mobilné zariadenia a integrácii do mobilnej aplikácie je systém dostupný širokej skupine používateľov bez potreby špeciálneho hardvéru. Veríme, že výsledný systém môže poskytnúť nepočujúcim ľuďom užitočný a efektívny nástroj na každodennú komunikáciu s počujúcimi ľuďmi a prispieť k ich lepšej integrácii do spoločnosti.

Formulácia úlohy

Cieľom tejto diplomovej práce je vytvoriť a natrénovať modely strojového učenia, ktoré budú schopné v reálnom čase detekovať ruku používateľa zo vstupu z kamery mobilného zariadenia a klasifikovať symboly posunkovej reči, ktoré používateľ ukazuje. Tieto modely sú následne integrované do mobilnej aplikácie na báze operačného systému Android, ktorá je prepojená s cloudovou službou Firebase a používateľovi umožňuje komunikáciu s nepočujúcimi ľuďmi prostredníctvom týchto modelov strojového učenia. V práci je potrebné dodržať nasledujúce požiadavky:

- analyzovať existujúce prístupy a riešenia problematiky rozpoznávania znakov posunkovej reči,
- analyzovať a spracovať vhodné dátové sady pre tréning modelov strojového učenia na detekciu rúk a klasifikáciu znakov posunkovej reči,
- využitím techník strojového učenia a počítačového videnia navrhnuť a implementovať model detekcie ruky v obraze a samostatný model klasifikácie vybraných znakov posunkovej reči,
- zabezpečiť optimalizáciu jednotlivých modelov pre nasadenie na mobilných zariadeniach,
- navrhnuť architektúru systému na rozpoznávanie znakov posunkovej reči v reálnom čase a implementovať ju vo forme mobilnej aplikácie vrátane integrácie vytvorených modelov,
- experimentálne overiť navrhnuté riešenie a vyhodnotiť jeho presnosť, výkonnosť a používateľskú použiteľnosť.

2 Teoretické východiská

Rozpoznávanie znakov posunkovej reči v reálnom čase je komplexný problém, ktorý si vyžaduje kombináciu poznatkov z viacerých oblastí. V tejto kapitole predstavujeme teoretické základy, na ktorých je postavený navrhovaný systém. Najprv uvádzame úvod do problematiky posunkovej reči a jej automatického rozpoznávania, po ktorom nasleduje prehľad existujúcich prístupov. Následne sa venujeme metódam počítačového videnia (z angl. *Computer Vision*), konvolučným neurónovým sieťam (z angl. *Convolutional Neural Networks*), skrátene CNN, a ich vybraným architektúram, ktoré sú kľúčové pre detekciu a klasifikáciu znakov posunkovej reči. Na záver predstavujeme veľké jazykové modely (z angl. *Large Language Models*), skrátene LLM, využívané na korekciu generovaného textu.

2.1 Úvod do problematiky posunkovej reči

Posunková reč predstavuje prirodzený a plnohodnotný jazykový systém s vlastnou gramatikou a slovnou zásobou, ktorý je primárnym komunikačným prostriedkom nepočujúcich ľudí. Na rozdiel od hovoreného jazyka využíva na vyjadrenie významu pohyby rúk, mimiku tváre a pohyby tela. V nasledujúcich častiach bližšie charakterizujeme posunkovú reč, jej globálny dosah a motiváciu pre vývoj systémov na jej automatické rozpoznávanie.

2.1.1 Definícia a význam posunkovej reči

Posunková reč je vizuálny jazyk, ktorý využíva pohyby rúk, mimiku tváre a pohyby tela na vyjadrenie myšlienok a gramatických vzťahov [2]. Nejde o univerzálny jazyk, každá krajina má spravidla vlastnú posunkovú reč, napríklad americkú posunkovú reč (z angl. *American Sign Language*), skrátene ASL, alebo britskú posunkovú reč (z angl. *British Sign Language*), skrátene BSL.

Podľa autorov Zhang et al. [1] trpelo v roku 2021 sluchovým postihnutím približne 1,56 miliardy ľudí na celom svete, pričom sa odhaduje, že do roku 2050

by toto číslo mohlo narásť až na 2,45 miliardy ľudí. Pre ľudí s čiastočnou stratou sluchu existuje viacero možností komunikácie, napríklad načúvacie prístroje alebo čítanie z pier. Ľudia s úplnou stratou sluchu sú však odkázaní práve na posunkovú reč ako svoj primárny komunikačný prostriedok.

Napriek rozšírenosti sluchového postihnutia väčšina počujúcej populácie posunkovú reč neovláda. Táto skutočnosť vytvára výraznú komunikačnú bariéru, ktorá obmedzuje možnosti nepočujúcich ľudí v každodennom živote, napríklad pri bežnej komunikácii, v oblasti vzdelávania, alebo na pracovnom trhu, a tým negatívne ovplyvňuje ich integráciu do spoločnosti.

2.1.2 Motivácia pre automatické rozpoznávanie posunkovej reči

Prekonanie komunikačnej bariéry medzi nepočujúcimi a počujúcimi ľuďmi si tradične vyžadovalo prítomnosť tlmočníka posunkovej reči, čo je riešenie, ktoré je finančne nákladné a nie vždy dostupné. Moderné technológie v oblasti strojového učenia a počítačového videnia otvárajú nové možnosti automatického rozpoznávania posunkovej reči, ktoré by mohli túto bariéru výrazne znížiť.

Automatické systémy na rozpoznávanie posunkovej reči môžu nepočujúcim ľuďom poskytnúť väčšiu mieru samostatnosti v každodennom živote. Na rozdiel od tradičného tlmočenia sú tieto systémy dostupné kedykoľvek a kdekoľvek, napríklad vo forme mobilnej aplikácie, bez potreby ďalšej osoby. Zároveň môžu výrazne uľahčiť komunikáciu v situáciách, kde je prítomnosť tlmočníka náročná alebo nemožná, ako napríklad pri návšteve lekára či vybavovaní úradných záležitostí.

Práve tieto skutočnosti nás motivovali k vývoju systému prezentovaného v tejto práci, ktorého cieľom je poskytnúť nepočujúcim ľuďom dostupný nástroj na komunikáciu s počujúcimi ľuďmi prostredníctvom automatického rozpoznávania znakov posunkovej reči v reálnom čase.

2.2 Rozpoznávanie posunkovej reči

Rozpoznávanie posunkovej reči predstavuje komplexnú úlohu, ktorá si vyžaduje nielen porozumenie vizuálnym gestám, ale aj schopnosť spracovávať ich v reálnom čase. V závislosti od charakteru vstupných dát a požadovaného výstupu možno túto úlohu rozdeliť do viacerých kategórií, pričom každá z nich prináša špecifické výzvy. V nasledujúcich častiach bližšie definujeme problém automatického rozpoznávania posunkovej reči, predstavujeme jeho základné typy a popisujeme hlavné výzvy, ktorým musia moderné systémy čeliť.

2.2.1 Automatické rozpoznávanie posunkovej reči

Automatické rozpoznávanie posunkovej reči (z angl. *Sign Language Recognition*), skrátené SLR, je úloha, pri ktorej systém na základe vizuálneho vstupu, najčastejšie obrazu z kamery, identifikuje a prekladá gestá posunkovej reči do textovej alebo hlasovej podoby [3]. Vstupom systému môže byť statický obrázok, sekvencia snímok alebo video, pričom výstupom je zodpovedajúci znak, slovo alebo veta v prirodzenom jazyku.

Na rozdiel od rozpoznávania hovorenej reči, kde je vstupom akustický signál, SLR pracuje s vizuálnymi dátami zachytenými kamerou. To vytvára špecifické požiadavky na spracovanie obrazu, keďže systém musí byť schopný spoľahlivo lokalizovať ruku v obraze, extrahovať z nej relevantné príznaky a na ich základe klasifikovať zobrazovaný znak posunkovej reči.

2.2.2 Typy prístupov k rozpoznávaniu posunkovej reči

Z hľadiska charakteru vstupných dát rozlišujeme podľa autorov Rastgoo et al. [3] rozpoznávanie statických a dynamických znakov posunkovej reči. Statické znaky sú tvorené jednou pevnou polohou ruky a vstupom pre systém je jediný obrázok, zatiaľ čo dynamické znaky zahŕňajú pohyb ruky v čase a vyžadujú spracovanie sekvencie snímok.

Z hľadiska spôsobu spracovania vstupnej sekvencie možno rozlišovať dva hlavné prístupy. Prvým je rozpoznávanie izolovaných znakov (z angl. *Isolated Sign Language Recognition*), skrátené ISLR, pri ktorom systém spracováva jednotlivé znaky posunkovej reči s explicitnými hranicami medzi nimi. Druhým je rozpoznávanie súvislej posunkovej reči (z angl. *Continuous Sign Language Recognition*), skrátené CSLR, pri ktorom systém spracováva plynulý súvislý tok bez explicitných hraníc medzi jednotlivými znakmi. Ako uvádzajú autori Rastgoo et al. [3], CSLR predstavuje výrazne náročnejšiu úlohu, keďže systém musí zároveň určovať hranice aj klasifikovať znaky v reálnom čase.

Systém prezentovaný v tejto práci patrí do kategórie ISLR, konkrétne sa zameriava na rozpoznávanie statických znakov americkej posunkovej reči predstavujúcich jednotlivé písmená abecedy.

2.2.3 Výzvy automatického rozpoznávania posunkovej reči

Napriek výraznému pokroku v oblasti strojového učenia a počítačového videnia zostáva automatické rozpoznávanie posunkovej reči náročnou úlohou. Ako uvádzajú autori Rastgoo et al. [3], medzi hlavné výzvy patrí predovšetkým varia-

bilita rúk a používateľov, keďže tvar, veľkosť a farba rúk sa výrazne líšia medzi jednotlivými používateľmi a rovnaký znak posunkovej reči môže byť predvedený rôznymi spôsobmi v závislosti od individuálneho prejavu používateľa.

Ďalšou výzvou sú podmienky prostredia, najmä osvetlenie a pozadie. Nedostatočné alebo nerovnomerné osvetlenie môže sťažiť detekciu ruky, zatiaľ čo komplexné pozadie môže spôsobovať falošné detekcie. V prípade dynamických znakov predstavuje výzvu aj časová dimenzia, keďže systém musí modelovať pohyb ruky v čase a zachytávať závislosti medzi jednotlivými snímkami.

Osobitnou výzvou je spracovanie v reálnom čase, ktoré je nevyhnutnou podmienkou praktického nasadenia systému. To kladie vysoké nároky na výpočtovú efektivitu modelov, najmä pri nasadení na mobilných zariadeniach s obmedzeným výpočtovým výkonom.

2.3 Existujúce prístupy rozpoznávania znakov posunkovej reči

Problematike rozpoznávania znakov posunkovej reči s využitím umelej inteligencie sa v súčasnosti venuje množstvo odborných prác. V tejto kapitole predstavujeme prehľad niektorých existujúcich riešení, pričom sa zameriavame na použité prístupy, modely a ich vzájomné porovnanie. Cieľom tejto analýzy je identifikovať silné stránky a obmedzenia existujúcich riešení, ktoré nám poslúžia ako základ pri návrhu nášho vlastného systému.

2.3.1 Prístupy k rozpoznávaniu posunkovej reči

V analyzovaných riešeniach boli využité dva hlavné prístupy k rozpoznávaniu znakov posunkovej reči. Patria sem prístupy založené na klasifikácii a prístupy založené na detekcii objektov.

Autori Alsharif et al. [4] zvolili klasifikačný prístup a zamerali sa na klasifikáciu znakov americkej posunkovej reči pomocou piatich architektúr hlbokých neurónových sietí, konkrétne *AlexNet*, *ConvNeXt*, *EfficientNet*, *ResNet50* a *Vision-Transformer*. Na trénovanie a testovanie využili verejne dostupnú dátovú sadu obsahujúcu 87000 obrázkov rúk rozdelených do 29 tried, pričom všetky modely boli trénované s využitím 5-násobnej krížovej validácie. Najlepšie výsledky dosiahla architektúra *ResNet50* s presnosťou 0.9998. Nevýhodou tohto riešenia je však skutočnosť, že systém nebol schopný pracovať v reálnom čase a neobsahoval modul na korekciu textu.

Podobnou problematikou sa zaoberali aj autori Alalwany et al. [5], ktorí zvolili detekčný prístup a navrhli systém kombinujúci model *YOLO11* na detekciu gest s knižnicou *MediaPipe* na sledovanie kľúčových bodov ruky. Na trénovanie využili vlastnú dátovú sadu, ktorá obsahovala 130000 obrázkov spolu s anotáciami pre 21 kľúčových bodov ruky, pričom systém dosiahol *mAP50* skóre 0.9820, s rýchlosťou inferencie 1.3 ms na snímku. Nevýhodou tohto riešenia naďalej zostáva absencia nasadenia na mobilnom zariadení.

Autori Rahim et al. [6] zvolili klasifikačný prístup, pričom pred samotnou klasifikáciou využili segmentačné techniky na oddelenie ruky od pozadia. Konkrétne aplikovali Gaussovo rozostrenie a prístup Otsu na predspracovanie vstupných obrazov v grayscale formáte, po ktorom nasledovala klasifikácia pomocou CNN s klasifikátorom *Softmax*. Systém bol testovaný v reálnom čase a dosiahol presnosť 0.9659. Nevýhodou tohto riešenia je, že dátová sada bola vytvorená v kontrolovanom prostredí bez variability pozadia a osvetlenia, čo mohlo pozitívne ovplyvniť dosiahnuté výsledky.

Autori Rivera-Acosta et al. [7] zvolili detekčný prístup, ktorý kombinuje detekciu ruky v obraze s následnou klasifikáciou znaku posunkovej reči. Na detekciu a rozpoznávanie znakov použili architektúry *YOLOv3-tiny* a *YOLOv3*. Okrem samotnej detekcie znakov autori implementovali aj modul na korekciu gramatiky využívajúci rekurentnú neurónovú sieť *LSTM* trénovanú na približne 370 slovách bežnej anglickej komunikácie. Systém dosiahol *mAP50* skóre 0.8174. Výhodou tohto prístupu je schopnosť lokalizovať ruku v obraze bez ohľadu na pozadie, čo zvyšuje robustnosť systému v reálnych podmienkach. Nižšia presnosť v porovnaní s klasifikačnými prístupmi môže byť spôsobená práve testovaním na reálnych vzorkách s rôznorodým pozadím.

Autori Rouabhi et al. [8] zvolili klasifikačný prístup zameraný priamo na nasadenie systému na mobilných zariadeniach. Navrhli mobilnú aplikáciu na rozpoznávanie znakov posunkovej reči v reálnom čase pomocou predtrénovaných architektúr CNN. Spomedzi testovaných architektúr dosiahla najlepšie výsledky architektúra *EfficientNet B7* s presnosťou 0.9924. Nevýhodou tohto riešenia je, že systém nebol testovaný v reálnych podmienkach, napríklad s rôznorodým pozadím.

2.3.2 Zhrnutie a porovnanie existujúcich prístupov

Spoločným obmedzením väčšiny analyzovaných riešení je absencia podpory dynamických znakov posunkovej reči zahŕňajúcich pohyb rúk. Klasifikačné prístupy dosahujú vyššiu presnosť, no sú testované prevažne na kontrolovaných dá-

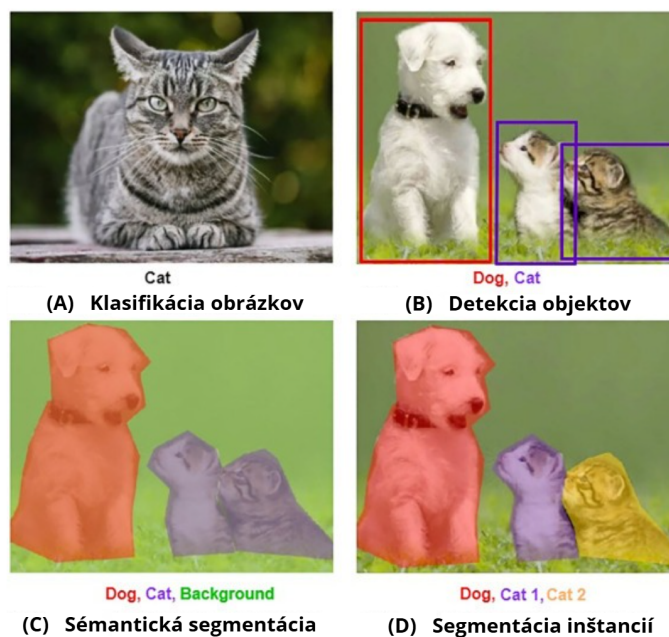
tových sadách s jednotným pozadím. Detekčné prístupy sú robustnejšie v reálnych podmienkach, no dosahujú nižšiu presnosť. Zároveň väčšina riešení nebola primárne navrhnutá pre nasadenie na mobilných zariadeniach, pričom výnimku predstavuje práca [8].

Tieto skutočnosti boli dôležitými faktormi pri návrhu nášho vlastného systému. Keďže cieľom tejto práce je vytvoriť systém, ktorý bude nasadený priamo na mobilných zariadeniach, existujúce riešenia neposkytovali dostatočný základ pre ich priame využitie. Z tohto dôvodu sme pristúpili k návrhu vlastného riešenia.

2.4 Počítačové videnie pre detekciu a klasifikáciu rúk

Počítačové videnie je oblasť umelej inteligencie, ktorá sa zaoberá analýzou a interpretáciou vizuálnych dát, ako sú obrázky a videá [9]. Počítačové videnie sa v súčasnosti využíva v mnohých oblastiach, napríklad pri rozpoznávaní ľudských tvári [10], sledovaní objektov pomocou kamier [11], monitorovaní dopravy [12] alebo detekcii objektov [13], [14]. V kontexte tejto práce sa počítačové videnie využíva predovšetkým na detekciu rúk a klasifikáciu znakov posunkovej reči zo vstupu kamery mobilného zariadenia.

Autori Huang a Le [9] uvádzajú, že počítačové videnie slúži najmä na realizáciu štyroch základných úloh. Medzi ne patria klasifikácia obrázkov, detekcia objektov, sémantická segmentácia a segmentácia inštancií, ktorých základný princíp je zobrazený na obrázku 2.1.



Obrázok 2.1: Štyri základné úlohy počítačového videnia [9]

2.4.1 Klasifikácia obrazových dát

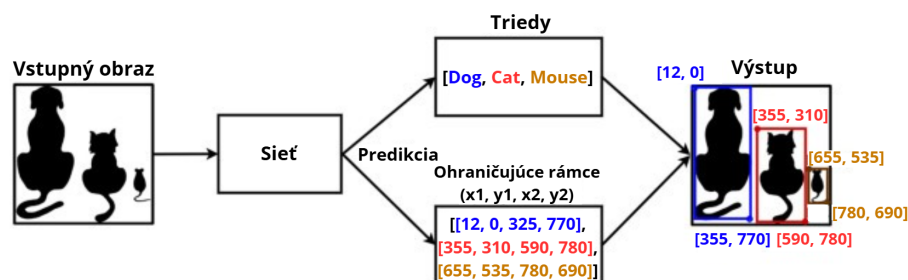
Základnou myšlienkou klasifikácie obrazových dát je priradiť vstupnému obrázku príslušnú triedu, alebo odhadnúť pravdepodobnosť príslušnosti obrázku k jednotlivým triedam [15]. Pri klasifikácii sa analyzujú príznaky obrazu, ktoré sú následne využité na priradenie obrázka do príslušnej kategórie. V priebehu rokov bolo vyvinutých množstvo klasifikačných metód, medzi ktoré patria napríklad umelé neurónové siete (z angl. *Artificial Neural Networks*), skrátene ANN. Pre úlohy klasifikácie obrazových dát sa ako výrazný pokrok ukázalo využitie hlbokých CNN, ktoré sa na rozdiel od tradičných prístupov dokážu automaticky učiť hierarchické reprezentácie príznakov priamo z obrazových dát [16], čo im umožňuje dosahovať vysokú presnosť klasifikácie pri rôznych úlohách strojového učenia.

Systémy klasifikácie obrazových dát typicky pozostávajú z niekoľkých kľúčových komponentov. Vstupná vrstva prijíma nespracované obrazové dáta, po ktorých nasledujú vrstvy extrakcie príznakov, kde konvolučné operácie identifikujú vzory s rastúcou úrovňou zložitosti. Operácie združovania (z angl. *pooling*) redukujú dimenzionalitu pri zachovaní dôležitých informácií, aktivačné funkcie zavádzajú nelinearitu a plne prepojené vrstvy (z angl. *fully connected layers*) zabezpečujú finálnu klasifikáciu.

V tejto práci sa prístupy na klasifikáciu obrazových dát využívajú na rozpoznávanie znakov posunkovej reči na základe oblasti ruky extrahovanej z obrazu.

2.4.2 Detekcia objektov v obraze

Detekcia objektov v obraze je úloha počítačového videnia, ktorá na rozdiel od klasifikácie obrázkov nielenže určuje triedy objektov prítomných na obrázku, ale zároveň lokalizuje ich polohu v rámci obrázka pomocou ohraničujúcich rámcov (z angl. *bounding boxes*) [9]. Základný princíp detekcie objektov v obraze je zobrazený na obrázku 2.2.



Obrázok 2.2: Princíp detekcie objektov v obraze [9]

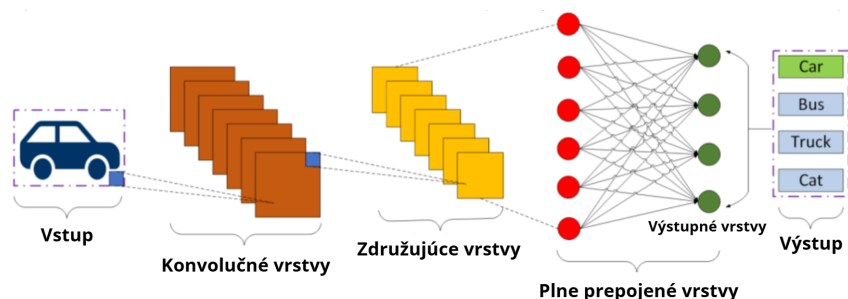
Podľa autorov Huang a Le [9], možno prístupy k detekcii objektov založené na CNN rozdeliť do dvoch hlavných kategórií. Prvou kategóriou je prístup založený na návrhu oblasti (z angl. *Region Proposal-Based Approach*), ktorý detekciu rozdeľuje do dvoch samostatných fáz. V prvej fáze sa identifikujú oblasti záujmu (z angl. *Regions of Interest*), napríklad pomocou algoritmu selektívneho vyhľadávania (z angl. *Selective Search*), a v druhej fáze sú následne tieto oblasti klasifikované [17]. Tento prístup dosahuje vysokú presnosť, no je výpočtovo náročnejší. Patria sem architektúry ako R-CNN, Fast R-CNN a Faster R-CNN [9].

Druhou kategóriou je prístup založený na klasifikácii alebo regresii (z angl. *Classification or Regression-Based Approach*), ktorý detekciu vykonáva v jednej fáze. Tento prístup mapuje celý vstupný obrázok priamo na ohraničujúce rámce a pravdepodobnosti tried. Tieto metódy sú spravidla rýchlejšie, a preto sú vhodné pre aplikácie v reálnom čase, hoci často dosahujú nižšiu presnosť. Medzi najznámejšie architektúry tohto prístupu patria napríklad SSD (z angl. *Single Shot MultiBox Detector*) architektúra a YOLO (z angl. *You Only Look Once*) architektúry [9].

V tejto práci sa prístupy k detekcii objektov v obraze využívajú na lokalizáciu ruky používateľa z kamerového vstupu mobilného zariadenia, pričom práve architektúry z rodiny YOLO boli zvolené ako základ pre modul detekcie rúk.

2.5 Konvolučné neurónové siete

CNN sú typom ANN, navrhnuté na efektívne spracovanie obrazových dát a získavanie priestorových charakteristík obrazu pomocou mechanizmu spätného šírenia (z angl. *backpropagation*) [18]. Architektúra CNN vo všeobecnosti pozostáva z troch hlavných typov vrstiev, a to z konvolučných vrstiev (z angl. *convolutional layers*), združujúcich vrstiev (z angl. *pooling layers*) a plne prepojených vrstiev (z angl. *fully connected layers*). Dôležitú úlohu v CNN zohrávajú aj aktivačné funkcie, ktoré zavádzajú nelinearitu do siete [19]. Základná architektúra CNN je zobrazená na obrázku 2.3.



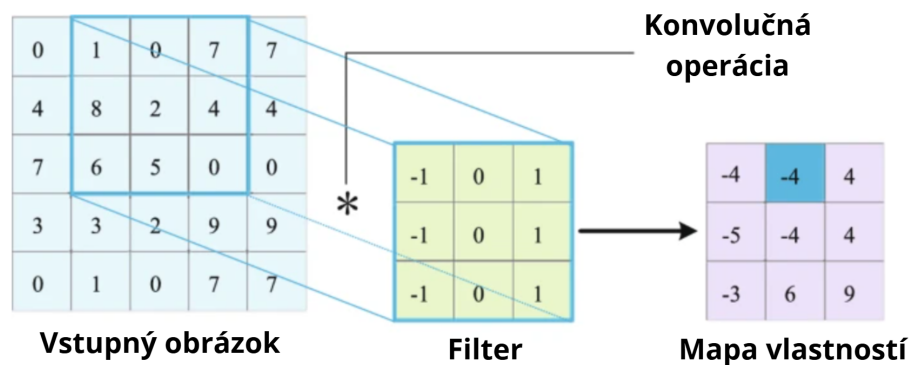
Obrázok 2.3: Základná architektúra konvolučnej neurónovej siete [18]

2.5.1 Základné komponenty CNN

Architektúra CNN sa skladá zo štyroch základných typov komponentov, ktoré spoločne zabezpečujú extrakciu príznačkov a klasifikáciu obrazových dát.

Konvolučné vrstvy

Konvolučné vrstvy predstavujú základný stavebný blok CNN. Tieto vrstvy pozostávajú z množiny filtrov alebo jadier, ktoré sú aplikované na vstupné dáta za účelom extrakcie príznačkov [19]. Konvulčná operácia spočíva v posúvaní konvulčného jadra po vstupnom obraze, pričom sa vykonáva násobenie hodnôt jadra s hodnotami pixelov na príslušných pozíciách a ich následné sčítanie. Výsledkom tejto operácie sú mapy vlastností (z angl. *feature maps*), ktoré zachytávajú rôzne vzory v obraze, ako sú hrany, textúry alebo zložitejšie tvary [19]. Príklad konvulčnej operácie je zobrazený na obrázku 2.4.



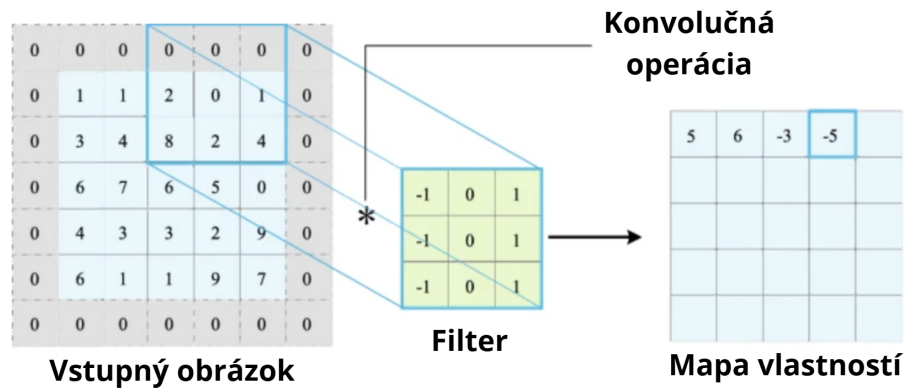
Obrázok 2.4: Princíp procesu konvolúcie [20]

Počet kanálov konvulčného jadra, tiež známy ako hĺbka, predstavuje počet máp vlastností generovaných konvulčnou operáciou. Tento počet priamo ovplyvňuje schopnosť extrakcie príznačkov a výpočtovú náročnosť siete. Zvyšovaním počtu kanálov môžeme zlepšiť schopnosť siete extrahovať príznačky, pričom to vedie k vyššej výpočtovej náročnosti [20].

Dôležitým parametrom konvulčnej operácie je krok (z angl. *stride*), ktorý určuje počet stĺpcov a riadkov, o ktoré sa konvulčné jadro posunie pri každom kroku operácie. S rastúcim krokom sa znižuje veľkosť výstupnej mapy vlastností a schopnosť extrakcie príznačkov sa oslabuje, no zároveň sa znižuje výpočtová náročnosť siete [20].

Problémom konvulčných operácií je postupné znižovanie rozmerov výstupu. Na jeho riešenie sa najčastejšie využíva technika dopĺňania nulami (z angl. *zero padding*), pri ktorej sa okolo vstupného obrazu pridávajú nulové pixely, čím

sa zachováva pôvodná veľkosť obrazu aj po aplikácii konvolučnej operácie [20]. Ďalšou výhodou tejto techniky je aj možnosť zachovania informácií z okrajových častí obrazu. Princíp techniky dopĺňania nulami je zobrazený na obrázku 2.5.



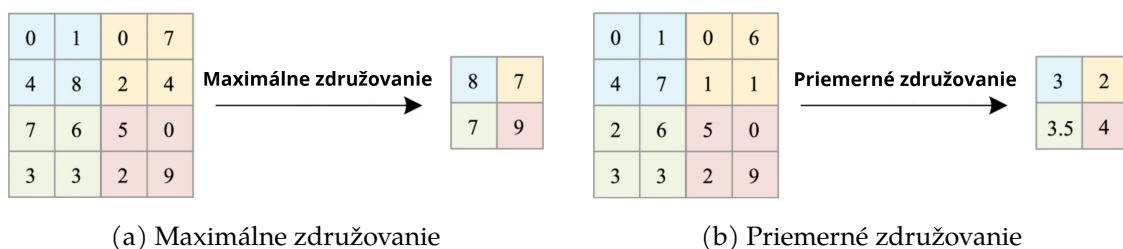
Obrázok 2.5: Princíp techniky dopĺňania nulami [20]

Združujúce vrstvy

Združujúce vrstvy nadväzujú na konvolučné vrstvy a slúžia na zmenšovanie dimenzionality máp vlastností, pričom zachovávajú najdôležitejšie informácie. Pri združujúcom procese je vstupná mapa vlastností rozdelená na viacero oblastí, pričom z každej oblasti je vybraná jedna reprezentatívna hodnota, ktorá sa stáva súčasťou výstupnej mapy vlastností [20].

Ako uvádzajú autori Zhao et al. [20], zavedenie združujúcich vrstiev do CNN nielenže zmenšuje objem dát a počet parametrov v sieti, ale zároveň robí sieť invariantnou voči malým lokálnym zmenám a poskytuje jej väčšie perцепčné pole, teda oblasť vstupného obrazu, ktorú môžu vidieť jednotlivé neuróny v sieti.

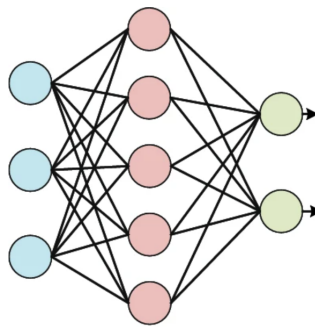
Medzi najčastejšie používané združujúce operácie patrí maximálne združovanie (z angl. *max pooling*), pri ktorom sa z každej oblasti mapy vlastností vyberá maximálna hodnota, a priemerné združovanie (z angl. *average pooling*), pri ktorom sa vypočíta priemer hodnôt v oblasti a tento priemer sa stáva reprezentantom tejto oblasti [20]. Princípy týchto operácií sú zobrazené na obrázku 2.6.



Obrázok 2.6: Združujúce operácie máp vlastností

Plne prepojené vrstvy

Plne prepojené vrstvy, tiež známe ako husté vrstvy (z angl. *dense layers*), sa zvyčajne nachádzajú na konci architektúry CNN [20]. V týchto vrstvách je každý neurón prepojený so všetkými neurónmi predchádzajúcej vrstvy. Plne prepojené vrstvy vykonávajú finálnu klasifikáciu na základe príznakov extrahovaných konvolučnými a združujúcimi vrstvami [21]. Výsledné mapy vlastností z poslednej konvolučnej alebo združujúcej vrstvy sú zvyčajne pred vstupom do plne prepojených vrstiev transformované do jednorozmerného vektora. Posledná plne prepojená vrstva obsahuje rovnaký počet neurónov ako je počet tried klasifikačnej úlohy. Na výstupe sa používajú aktivačné funkcie, ktoré normalizujú výstupné hodnoty do pravdepodobností, ktoré reprezentujú pravdepodobnosť príslušnosti vstupného obrazu k jednotlivým triedam [21]. Príklad plne prepojenej vrstvy pre binárnu klasifikáciu je zobrazený na obrázku 2.7.



Obrázok 2.7: Plne prepojená vrstva pre binárnu klasifikáciu [20]

Aktivačné funkcie

Aktivačné funkcie zohrávajú v CNN dôležitú úlohu, pretože zavádzajú nelinearitu do siete a určujú, či a ako sa informácie šíria cez jednotlivé vrstvy [22]. Bez aktivačných funkcií by skladanie viacerých vrstiev CNN vykonávalo len lineárne transformácie, ktoré by nedokázali zachytiť komplexné vzory potrebné napríklad pri klasifikácii obrázkov [23].

Ako uvádzajú autori Salaji et al. [24], aktivačné funkcie možno rozdeliť do troch hlavných kategórií. Binárne aktivačné funkcie poskytujú výstup 0 alebo 1 na základe prahovej hodnoty, čo ich robí vhodnými pre úlohy binárnej klasifikácie. Lineárne aktivačné funkcie generujú spojitý rozsah výstupov a využívajú sa najmä pri regresných úlohách. Nelineárne aktivačné funkcie zas zvyšujú komplexnosť siete tým, že nelineárne transformujú vstupy, čo umožňuje modelovať zložité nelineárne závislosti v dátach, ako je napríklad rozpoznávanie vzorov

v obraze.

Medzi najpoužívannejšie aktivačné funkcie patria napríklad aktivačná funkcia *ReLU* (z angl. *Rectified Linear Unit*), *Sigmoid*, *Tanh* alebo *Softmax*. Funkcia *Softmax* sa používa najmä na výstupnej vrstve pri klasifikačných úlohách, kde normalizuje výstupy na pravdepodobnosti cieľových tried [19], [21].

2.6 Architektúry CNN pre klasifikáciu a detekciu

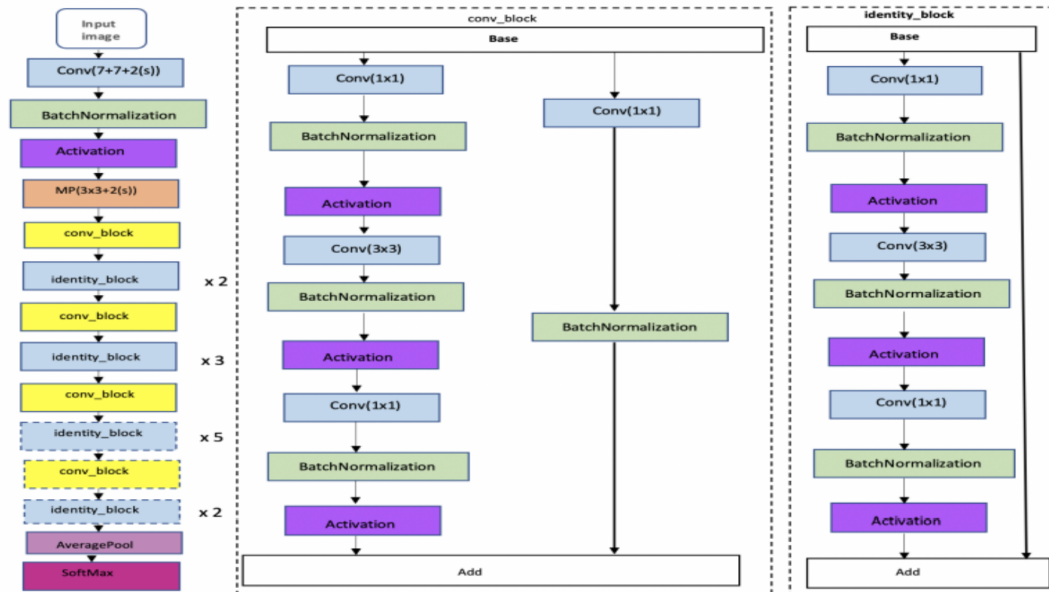
V súčasnosti existuje množstvo architektúr CNN navrhnutých na riešenie rôznych úloh v oblasti počítačového videnia, medzi ktoré patria napríklad *AlexNet* [25], *GoogLeNet* [26], *ResNet* [27] alebo *MobileNet* [28]. Ako uvádzajú autori Cheng et al. [29], do kategórie CNN možno zaradiť aj architektúry z rodiny YOLO.

V tejto časti predstavujeme architektúry zvolené pre navrhovaný systém. Konkrétne ide o architektúry *ResNet50* a *MobileNetV2* určené na klasifikáciu znakov posunkovej reči. Zároveň si priblížime aj architektúry z rodiny YOLO, konkrétne *YOLOv8n* a *YOLO11n*, určené na detekciu rúk a ich kľúčových bodov.

2.6.1 ResNet50

ResNet50 je architektúra CNN navrhnutá hlavne na klasifikáciu obrázkov. Je známa svojou schopnosťou dosahovať vysokú presnosť pri rôznych úlohách klasifikácie obrázkov. Napríklad autori Nasra a Gupta [30] dosiahli pri detekcii ochorení listov kukurice presnosť s hodnotou 0.9200, zatiaľ čo autori Wen et al. [31] dosiahli pri rozpoznávaní a klasifikácii prírodných katastrof AUC skóre 0.9570. V porovnaní s konvenčnými architektúrami umožňuje *ResNet50*, znázornená na obrázku 2.8, konštruovať hlbšie siete a optimalizovať jednotlivé vrstvy, aby sa eliminoval problém miznúceho gradientu. Charakteristickým prvkom *ResNet* architektúr je využitie techniky *Batch Normalization*, ktorej cieľom je redukcia kovariančného posunu a vyrovňovanie distribúcie vstupných hodnôt počas procesu učenia [32].

Hlavným stavebným prvkom architektúry *ResNet50* sú konvolučné bloky spolu s identitnými blokmi, ktoré umožňujú vytvárať tzv. skokové spojenia (z angl. *skip connections*). Skokové spojenia prenášajú výstup jednej vrstvy priamo na vstup hlbšej vrstvy, čím obchádzajú medzivrstvy a umožňujú sieti učiť sa reziduálne reprezentácie. Vďaka tomu sa gradient pri spätnom šírení nestráca ani v hlbokých sieťach [32]. Na začiatku je vstupný obraz prevedený cez konvolučnú vrstvu s veľkosťou jadra 7×7 a krokom 2, po ktorej nasleduje normalizácia pomocou techniky *Batch Normalization* a aktivačná funkcia *ReLU*. Výstup je ná-



Obrázok 2.8: Architektúra ResNet50 [32]

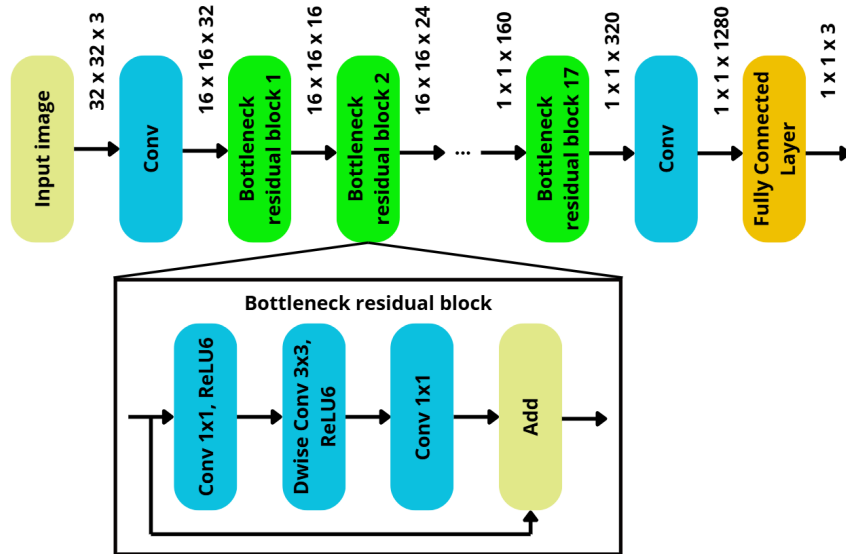
sledne zmenšený v združujúcej vrstve využitím maximálneho združovania. Od nasledujúcej až po piatu konvolučnú etapu je proces extrakcie príznakov realizovaný kombináciou konvolučných a identitných blokov. Na záver je výsledná mapa vlastností odovzdaná do plne prepojenej vrstvy, kde je pomocou aktivačnej funkcie *Softmax* vykonaná finálna klasifikácia [32].

V navrhovanom systéme slúži architektúra *ResNet50* ako referenčný model na porovnanie s architektúrou *MobileNetV2* pri klasifikácii znakov posunkovej reči.

2.6.2 MobileNetV2

MobileNetV2 je architektúra CNN navrhnutá spoločnosťou Google pre mobilné a vstavané zariadenia. Je nástupcom pôvodnej *MobileNet* architektúry a je známa svojou efektivitou a nízkou výpočtovou náročnosťou, čo ju robí vhodným kandidátom pre nasadenie na mobilných zariadeniach [33]. Jej využitie demonštrujú napríklad autori Chen et al. [34], ktorí ju použili v kontexte klasifikácie a identifikácie poľnohospodárskych produktov, pričom dosiahli presnosť až 0.9726.

Architektúra *MobileNetV2*, znázornená na obrázku 2.9, je založená na invertovanej reziduálnej štruktúre (z angl. *Inverted Residual Structure*) a využíva techniku hĺboko separovateľných konvolúcií (z angl. *Depthwise Separable Convolutions*), ktorá znižuje výpočtovú zložitosť modelu [33]. *MobileNetV2* vkladá 19 invertovaných reziduálnych *bottleneck* vrstiev hneď za prvú konvolučnú vrstvu s 32 filtra-
mi. Architektúra je následne ukončená bodovou konvolúciou (z angl. *pointwise convolution*), ktorá produkuje výstup s rozmermi $1 \times 1 \times 1280$ [35].



Obrázok 2.9: Architektúra MobileNetV2 [36]

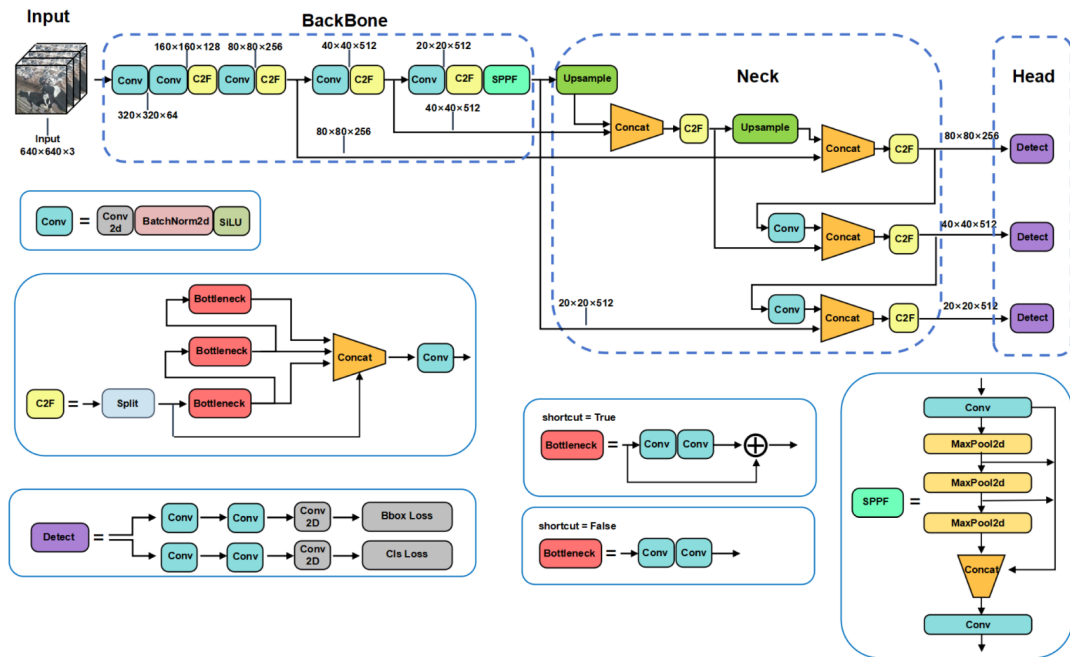
Základným stavebným prvkom *MobileNetV2* je invertovaný reziduálny *bottleneck* blok, ktorý pozostáva z troch vrstiev. Prvou vrstvou je konvolučná vrstva s veľkosťou jadra 1×1 , ktorá rozširuje počet kanálov a používa aktivačnú funkciu *ReLU6*. Druhou je hĺbkovo separovateľná konvolučná vrstva s veľkosťou jadra 3×3 s aktivačnou funkciou *ReLU6*. Tretou je opäť konvolučná vrstva s veľkosťou jadra 1×1 , ktorá znižuje počet kanálov späť na požadovanú hodnotu bez použitia aktivačnej funkcie. Po týchto blokoch nasleduje ďalšia konvolučná vrstva a združujúca vrstva. Poslednou vrstvou je plne prepojená vrstva, ktorá sa používa na finálnu klasifikáciu [36].

V navrhovanom systéme bola architektúra *MobileNetV2* zvolená pre klasifikáciu znakov posunkovej reči, a to najmä vďaka jej nízkej výpočtovej náročnosti, ktorá predstavuje kľúčovú vlastnosť pre efektívne nasadenie na mobilných zariadeniach.

2.6.3 Modely YOLO pre detekciu rúk

YOLOv8n je ľahký jednofázový rámec na detekciu objektov, navrhnutý tak, aby vyvažoval presnosť detekcie a efektivitu nasadenia na mobilných zariadeniach [37]. Architektúru *YOLOv8n* využili napríklad autori Chen et al. [38] na detekciu voľne žijúcich zvierat, kde dosiahli úspešnosť detekcie 0.9360 meranú metrikou mAP50.

Architektúra *YOLOv8n*, znázornená na obrázku 2.10, pozostáva zo štyroch hlavných častí, a to zo vstupu, extrakčnej časti (z angl. *backbone*), spojovacej časti (z angl. *neck*) a výstupnej časti (z angl. *head*) [39].



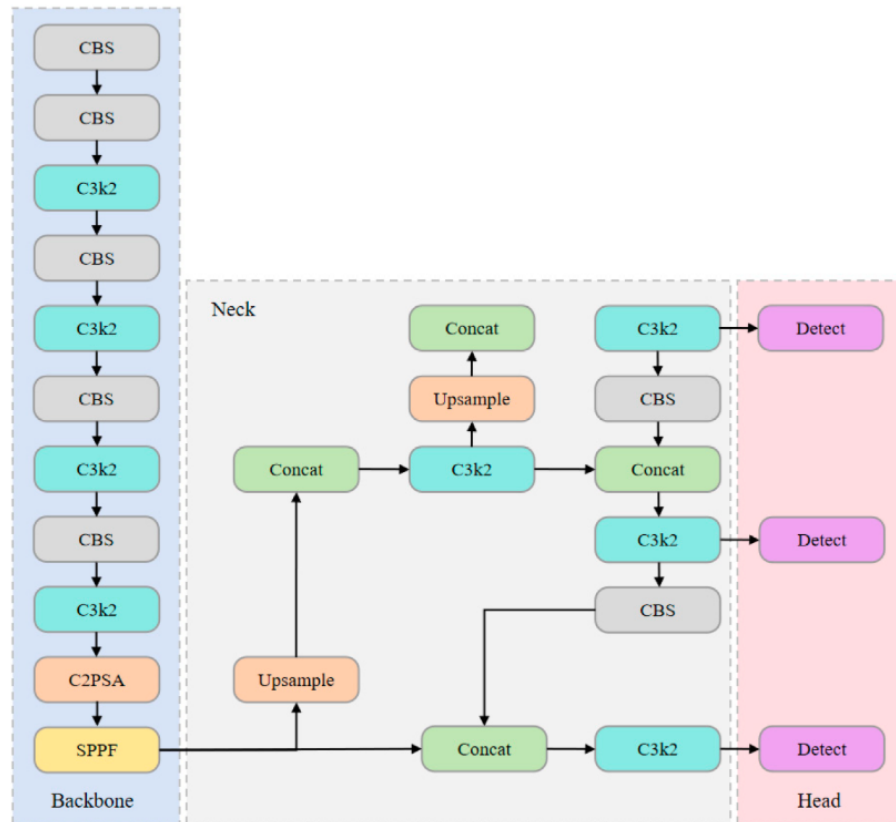
Obrázok 2.10: Architektúra YOLOv8n [39]

Vstupná vrstva je zodpovedná za prijímanie a spracovanie vstupného obrazu. Extrakčná časť je zložená z konvolučného bloku, modulu C2f a bloku SPPF (z angl. *Spatial Pyramid Pooling Fast*), ktorých úlohou je učiť sa komplexné reprezentácie príznakov zo vstupných dát. Spojovacia časť využíva štruktúru FPN (z angl. *Feature Pyramid Network*) a PAN (z angl. *Path Aggregation Network*) na kombinovanie príznakov rôznych mierok, čím zvyšuje schopnosť siete zachytávať objekty rôznych veľkostí. Výstupná časť obsahuje tri detekčné vetvy rôznych mierok, ktoré prostredníctvom ne-maximálnej supresie (z angl. *Non-Maximum Suppression*) produkujú finálne ohraničujúce rámce [39]. Na trénovanie model využíva kombináciu stratových funkcií VFL Loss (z angl. *Varifocal Loss*), DFL Loss (z angl. *Distribution Focal Loss*) a CIoU Loss (z angl. *Complete Intersection over Union Loss*) [39].

YOLO11n je ďalšou ľahkou architektúrou z rodiny YOLO, ktorá predstavuje rozšírenie architektúry YOLOv8n s viacerými optimalizáciami zameranými na zlepšenie výkonu [40]. Architektúru YOLO11n využili napríklad autori Zhou et al. [41] na detekciu vesmírnych objektov, kde dosiahli úspešnosť detekcie 0.9640 meranú metrikou mAP50.

Architektúra YOLO11n, znázornená na obrázku 2.11, zachováva klasickú štvorčlennú štruktúru pozostávajúcu z vstupnej, extrakčnej, spojovacej a výstupnej časti, pričom oproti architektúre YOLOv8n prináša niekoľko vylepšení [42].

Extrakčná časť využíva moduly CBS (z angl. *Convolution-BatchNorm-SiLU*) a ľahkú reziduálnu štruktúru C3K2 na extrakciu príznakov, čím sa zlepšuje lo-



Obrázok 2.11: Architektúra YOLO11n [42]

kálne receptívne pole a schopnosť reprezentácie príznakov. Na ďalšie zvýšenie schopnosti modelu vnímať objekty v komplexných scénach je zavedená štruktúra SPPF, ktorá rozširuje receptívne pole, a modul C2PSA na implementáciu efektívneho mechanizmu pozornosti. V spojovacej časti je použitá vylepšená štruktúra FPN a PAN s hĺbkovo separovateľnými konvolúciami na zníženie parametrového zaťaženia. Výstupná časť je založená na *anchor-free* dizajne, ktorý priamo predikuje stred objektu, jeho rozmery a triedu [42].

V navrhovanom systéme boli architektúry *YOLOv8n* a *YOLO11n* zvolené, ako základ pre modul detekcie rúk, pričom obe architektúry boli trénované na detekciu ruky v obraze.

2.6.4 Modely YOLO s detekciou kľúčových bodov

YOLOv8n-pose je rozšírená verzia architektúry *YOLOv8n* špeciálne navrhnutá na súčasnú detekciu objektov a odhad kľúčových bodov. Oproti základnej architektúre *YOLOv8n* integruje modul výpočtu straty pre kľúčové body a využíva oddelenú štruktúru výstupnej časti so samostatnými vetvami pre detekciu objektov a odhad kľúčových bodov. To umožňuje súčasne predikovať polohu objektov, ich triedy a kľúčové body [43].

YOLO11n-pose je rozšírením architektúry *YOLO11n* navrhnutým na detekciu objektov a odhad kľúčových bodov. Model nielenže zachováva výhody predchádzajúcich verzií YOLO-pose modelov, ale zároveň zlepšuje schopnosť extrakcie príznakov, čo umožňuje zachytiť jemné lokálne a štrukturálne detaily v obrazových dátach. Napriek zavedeniu samostatnej vetvy na predikciu kľúčových bodov zostáva architektúra kompaktná [44].

Obe architektúry využili napríklad autori Chen et al. [45] na detekciu anatomických bodov na telách kurčiat, kde dosiahli úspešnosť detekcie 0.9010 pre *YOLOv8n-pose* a 0.9290 pre *YOLO11n-pose* meranú metrikou mAP50. V navrhovanom systéme boli architektúry *YOLOv8n-pose* a *YOLO11n-pose* zvolené ako základ pre variant modulu detekcie rúk poskytujúci aj informácie o kľúčových bodoch ruky.

2.7 Jazykové modely na korekciu textu

Jazykové modely (z angl. *Language Models*) sú výpočtové modely schopné porozumieť prirodzenému jazyku a generovať ho. Pokročilou formou jazykových modelov sú LLMs [46], ktoré disponujú obrovským množstvom parametrov a výnimočnými schopnosťami učenia sa.

2.7.1 Transformerová architektúra

Základom väčšiny moderných LLMs je architektúra *transformer*, ktorá priniesla revolúciu v oblasti spracovania prirodzeného jazyka (z angl. *Natural Language Processing*), skrátene NLP [47]. Kľúčovým komponentom tejto architektúry je mechanizmus samopozornosti (z angl. *Self-Attention Mechanism*), ktorý umožňuje modelu dynamicky rozdeľovať pozornosť medzi rôzne časti vstupnej sekvencie a zachytávať dlhodobé kontextové vzťahy medzi jej prvkami [48].

Architektúra *transformer* pozostáva z enkódovacích a dekódovacích vrstiev. Enkódovacia časť extrahuje príznaky zo vstupnej sekvencie, zatiaľ čo dekódovacia časť generuje výstupnú sekvenciu [47]. Každý enkódovací blok obsahuje vrstvu samopozornosti a vrstvu doprednej siete (z angl. *Feed-Forward Network*). Dekódovací blok navyše obsahuje vrstvu krížovej pozornosti (z angl. *Cross-Attention Layer*), ktorá prepája enkódováciu a dekódováciu časť [49]. Na zvýšenie stability a efektivity trénovania sa využívajú reziduálne spojenia a normalizácia vrstiev [47].

Na základe tejto architektúry sú LLMs schopné efektívne spracovávať kontext a generovať výstupy, čo umožňuje ich využitie v rôznych režimoch NLP, ako sú

napríklad *zero-shot* a *few-shot* prístupy [47].

2.7.2 Využitie LLM na korekciu textu

Vďaka architektúre *transformer* a mechanizmu samopozornosti majú LLMs schopnosť vykonávať rôzne úlohy bez potreby špecifického trénovania, pričom výkon modelu závisí od spôsobu zadania vstupnej inštrukcie (z angl. *prompt*) [50]. Inferencia (z angl. *inference*) predstavuje proces generovania výstupu natrénovaným modelom na základe vstupných dát. Medzi základné prístupy k inferencii LLM patria *zero-shot*, *few-shot* a *fine-tuning* prístupy [50].

Pri prístupe *zero-shot* model generuje výstup len na základe inštrukcie v prirodzenom jazyku, bez akýchkoľvek predchádzajúcich príkladov [51]. Pri prístupe *few-shot* je modelu poskytnutých niekoľko demonštračných príkladov spolu s inštrukciou, čo mu umožňuje lepšie pochopiť danú úlohu [51]. Na rozdiel od týchto prístupov, *fine-tuning* predstavuje proces doladenia predtrénovaného modelu na špecifickej dátovej sade, čím sa zlepšuje jeho výkon pre konkrétnu úlohu [50].

V navrhovanom systéme je LLM využitý na korekciu textu generovaného klasifikačným modulom prostredníctvom cloudovej služby *Firebase AI Logic*. Konkrétne bol zvolený prístup *zero-shot*, keďže model *Gemini 2.5 Flash* poskytuje dostatočnú kvalitu korekcie textu bez potreby dodatočného trénovania na vlastných dátach, čo zároveň eliminuje potrebu prípravy trénovacej množiny demonštračných príkladov.

3 Návrh systému na rozpoznávanie znakov posunkovej reči

Pred samotnou implementáciou systému je potrebné stanoviť požiadavky, ktoré by mal systém spĺňať, a navrhnuť jeho celkovú architektúru. V tejto kapitole predstavujeme funkčné a nefunkčné požiadavky na systém, jeho modulárnu architektúru a návrh jeho jednotlivých modulov.

3.1 Funkčné požiadavky

Funkčné požiadavky definujú funkcionality, ktorú má navrhovaný systém poskytovať používateľovi. Vychádzajú priamo z hlavného cieľa práce, ktorým je umožniť komunikáciu medzi nepočujúcimi a počujúcimi ľuďmi. Navrhovaný systém musí spĺňať nasledujúce hlavné funkčné požiadavky:

- detekcia ruky v reálnom čase,
- klasifikácia znakov posunkovej reči,
- zobrazenie preloženého textu,
- korekcia textu pomocou LLM.

Systém má byť schopný v reálnom čase detegovať ruku používateľa z kamerového vstupu mobilného zariadenia a na jej základe klasifikovať zobrazovaný znak posunkovej reči. Rozpoznané znaky majú byť skladané do výsledného textu a zobrazované používateľovi. Korekcia textu pomocou LLM predstavuje voliteľnú funkcionality dostupnú len pri aktívnom internetovom pripojení.

3.2 Nefunkčné požiadavky

Nefunkčné požiadavky definujú vlastnosti systému ako sú výkon, dostupnosť a presnosť. Keďže systém má byť navrhnutý pre nasadenie na mobilných zariadeniach,

niach s obmedzeným výpočtovým výkonom, nefunkčné požiadavky kladú dôraz na optimalizáciu výkonu a efektívne využitie zdrojov. Medzi hlavné nefunkčné požiadavky navrhovaného systému patria:

- optimalizácia výkonu pre prácu v reálnom čase,
- offline dostupnosť hlavnej funkcionality,
- zachovanie presnosti aj po optimalizácii,
- dostupnosť vo forme mobilnej aplikácie.

Systém má byť optimalizovaný tak, aby bol schopný pracovať plynulo v reálnom čase na mobilných zariadeniach s obmedzeným výpočtovým výkonom. Hlavná funkcionality, teda detekcia ruky a klasifikácia znakov posunkovej reči, má byť dostupná aj bez internetového pripojenia. Systém má zároveň zachovávať dostatočnú presnosť rozpoznávania znakov posunkovej reči aj napriek optimalizáciám. Celý systém má byť dostupný vo forme mobilnej aplikácie na báze operačného systému Android s jednoduchým a intuitívnym používateľským rozhraním.

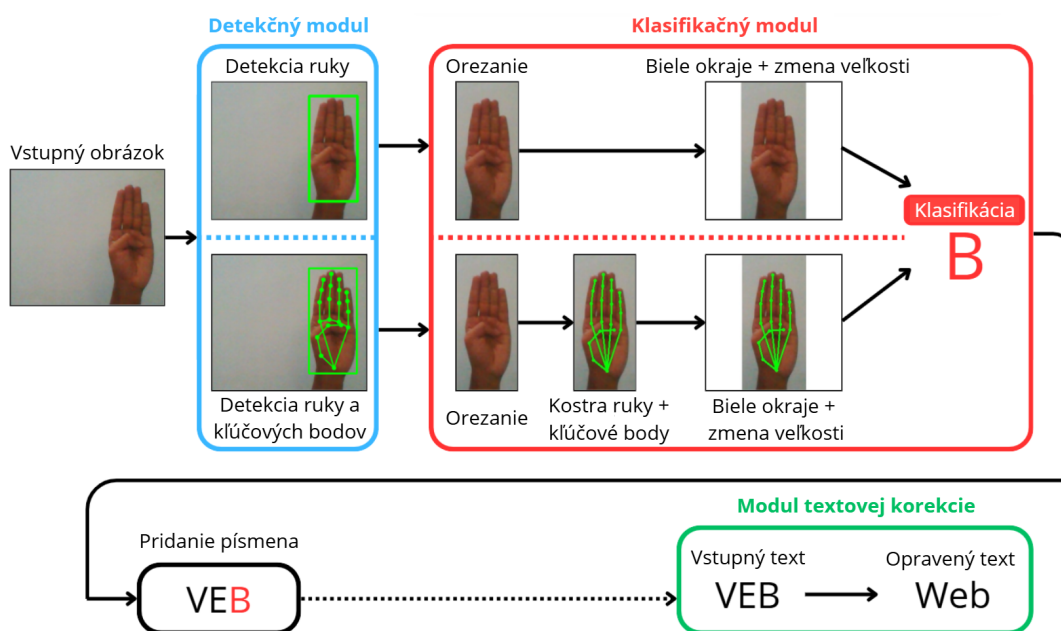
3.3 Architektúra systému

Navrhovaný systém je rozdelený do viacerých modulov, pričom každý z nich zabezpečuje určitú časť procesu rozpoznávania a prekladu znakov posunkovej reči. Modulárna architektúra umožňuje nezávislé testovanie a optimalizáciu jednotlivých častí systému, ako aj jednoduchú výmenu alebo vylepšenie jednotlivých modulov bez zásahu do ostatných častí systému.

Systém pozostáva z modulu detekcie rúk, klasifikačného modulu a doplnkového modulu textovej korekcie, ktoré sú integrované do mobilnej aplikácie na báze operačného systému Android. Tok dát medzi jednotlivými modulmi a architektúra systému sú zobrazené na obrázku 3.1. Návrh jednotlivých modulov je bližšie popísaný v nasledujúcich podkapitolách.

3.4 Návrh jednotlivých modulov

V tejto časti predstavujeme návrh jednotlivých modulov systému, pričom sa zameriavame na odôvodnenie kľúčových architektonických rozhodnutí. Pre každý modul uvádzame výber použitých technológií a prístupov, ich vstup a výstup a vzájomné väzby medzi modulmi.



Obrázok 3.1: Architektúra systému na rozpoznávanie znakov posunkovej reči

3.4.1 Modul detekcie rúk

Modul detekcie rúk predstavuje vstupný bod celého systému. Jeho úlohou je lokalizovať ruku používateľa v obraze z kamery mobilného zariadenia a extrahovať informácie potrebné pre klasifikačný modul. Vstupom modulu je snímka z kamery a výstupom je ohraničujúci rámec detegovanej ruky, prípadne aj súradnice jej kľúčových bodov.

Pre tento modul sú zvolené architektúry z rodiny YOLO, konkrétne *YOLOv8n* a *YOLO11n* pre detekciu ruky bez kľúčových bodov, a *YOLOv8n-pose* a *YOLO11n-pose* pre detekciu ruky spolu s kľúčovými bodmi. Tieto architektúry sú zvolené pre ich jednofázový prístup k detekcii, ktorý je výrazne rýchlejší ako dvojfázové prístupy, čo je kľúčové pre spracovanie v reálnom čase na mobilných zariadeniach. Zároveň umožňujú efektívnu optimalizáciu pre nasadenie na mobilných zariadeniach.

Modul je navrhnutý v dvoch variantoch. Prvý variant poskytuje len informácie o polohe ruky v obraze vo forme ohraničujúceho rámca. Druhý variant navyše poskytuje aj informácie o kľúčových bodoch ruky a jej kostre, ktoré predstavujú bohatšiu reprezentáciu vstupu pre klasifikačný modul.

3.4.2 Modul klasifikácie znakov posunkovej reči

Modul klasifikácie znakov posunkovej reči predstavuje jadro celého systému. Jeho úlohou je na základe informácií poskytnutých modulom detekcie rúk klasi-

fikovať zobrazovaný znak posunkovej reči. Vstupom modulu je oblasť obrazu obsahujúca ruku, prípadne doplnená o vizualizáciu kľúčových bodov a kostry ruky, a výstupom je textová reprezentácia rozpoznaného znaku posunkovej reči.

Pred samotnou klasifikáciou modul vykonáva predspracovanie vstupného obrazu, ktoré zahŕňa orezanie obrazu na oblasť obsahujúcu ruku, úpravu pomeru strán pridaním bielych okrajov, zmenu veľkosti obrazu podľa požiadaviek klasifikačnej architektúry a prípadne vloženie vizualizácie kľúčových bodov a kostry ruky do vstupného obrazu.

Pre klasifikáciu znakov posunkovej reči je zvolený prístup *transfer learning*, pri ktorom sú váhy základnej časti modelu inicializované pomocou váh predtrénovaných na dátovej sade *ImageNet*. Tento prístup umožňuje dosahovať vysokú presnosť klasifikácie aj pri relatívne obmedzenom množstve tréningových dát. Pre klasifikačný modul sú zvolené architektúry *MobileNetV2* a *ResNet50*, pričom *MobileNetV2* je primárne navrhnutá pre nasadenie na mobilných zariadeniach a *ResNet50* slúži ako referenčný model na porovnanie.

Modul je navrhnutý v dvoch variantoch zodpovedajúcich variantom modulu detekcie rúk. Prvý variant využíva len informácie o polohe ruky v obraze, zatiaľ čo druhý variant využíva aj informácie o kľúčových bodoch ruky, ktoré sú vložené do vstupného obrazu vo forme vizualizácie kostry ruky a kľúčových bodov.

3.4.3 Modul textovej korekcie

Modul textovej korekcie predstavuje doplnkový komponent systému. Jeho úlohou je opravovať gramatické a štylistické chyby v texte generovanom klasifikačným modulom, keďže tento text môže obsahovať chyby spôsobené nesprávnou klasifikáciou znakov posunkovej reči, chýbajúcimi medzerami alebo neúplnými vetami. Vstupom modulu je text generovaný klasifikačným modulom a výstupom je gramaticky opravený text.

Namiesto tréningovania vlastného modelu je pre tento modul zvolený predtrénovaný LLM prístupný prostredníctvom cloudovej služby *Firebase AI Logic*. Táto služba funguje ako bezpečná sprostredkovateľská vrstva medzi aplikáciou a LLM, pričom poskytuje rozhranie *Prompt templates*, ktoré umožňuje definovať šablóny promptov s jedinečnými identifikátormi bez potreby ich explicitného ukladania v zdrojovom kóde. Konkrétne je zvolený model *Gemini 2.5 Flash* [52], ktorý predstavuje vhodný kompromis medzi kvalitou korekcie a rýchlosťou odozvy.

Nevýhodou tohto prístupu je závislosť na internetovom pripojení, keďže všetky požiadavky sú spracovávané v cloudovom prostredí. Z tohto dôvodu je modul textovej korekcie dostupný len pri aktívnom internetovom pripojení a jeho

využitie je voliteľné.

3.4.4 Mobilná aplikácia

Mobilná aplikácia predstavuje používateľské rozhranie celého systému, ktoré integruje všetky implementované moduly do jedného celku. Pre vývoj mobilnej aplikácie je zvolená platforma Android, ktorá podľa autorov Jošt a Taneski [53] mala ku koncu roka 2024 dominantný podiel na trhu mobilných operačných systémov, konkrétne 72.04%. Tým sa zabezpečuje dostupnosť systému pre čo najširšiu skupinu používateľov.

Na implementáciu mobilnej aplikácie je zvolený framework *Flutter* [54], ktorý umožňuje vývoj multiplatformových mobilných aplikácií z jedného zdrojového kódu využívajúceho programovací jazyk *Dart*. Hoci aplikácia je v rámci tejto práce implementovaná len pre platformu Android, využitie frameworku *Flutter* umožní v budúcnosti jej jednoduché rozšírenie napríklad aj na platformu iOS.

Medzi hlavné funkčné požiadavky aplikácie patria preklad znakov posunkovej reči do textovej podoby v reálnom čase, reverzný preklad textu do grafickej podoby znakov posunkovej reči a voliteľná korekcia textu pomocou LLM pri aktívnom internetovom pripojení. Z hľadiska nefunkčných požiadaviek je kladený dôraz na jednoduché a intuitívne používateľské rozhranie, podporu svetlého a tmavého režimu a spracovanie obrazu v reálnom čase s minimálnou latenciou.

4 Charakteristika a spracovanie dát

Základom implementácie systému na rozpoznávanie znakov posunkovej reči bolo použitie rozsiahlych a rozmanitých dát. Pri výbere dátových sád sme sa zamerali na ich rozmanitosť a reprezentatívnosť vzhľadom na reálne podmienky použitia systému. Okrem verejne dostupných dátových sád sme vytvorili aj vlastné dáta, ktoré nám pomohli zlepšiť výkon jednotlivých modulov systému. V rámci spracovania dát sme využili rôzne techniky augmentácie, ktoré nám pomohli vytvoriť robustnejší základ pre trénovanie vybraných architektúr strojového učenia.

4.1 Dáta z verejných zdrojov

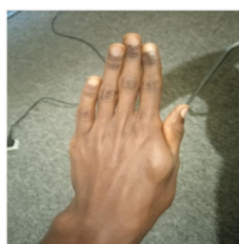
Pri tvorbe systému sme využili viaceré verejne dostupné dátové sady obsahujúce obrázky rúk a znakov posunkovej reči. Prehľad použitých verejne dostupných dátových sád je uvedený v tabuľke 4.1.

Tabuľka 4.1: Prehľad použitých verejne dostupných dátových sád

Názov	Zdroj	Počet vzoriek	Veľkosť (GB)
Hand Recognition Dataset	[55]	405	0.02
Hand Keypoint Dataset 26k	[56]	26768	0.53
Hand Detection Dataset	[57]	2061	0.08
Hand Detection CV Dataset	[58]	49645	1.8
Full Computer Vision Dataset	[59]	13626	0.4
My First Project CV Dataset	[60]	14184	0.32
Hand Gestures CV Dataset	[61]	8733	0.13
HaGRID Sample 500k 384p	[62]	509323	13.4
American Sign Language	[63]	166000	4.98

Dátová sada *Hand Recognition Dataset* obsahovala vzorky od 65 náhodných ľudí, pričom každý z nich poskytol približne 6 vzoriek rúk v rôznych pozíciách a s rôznym pozadím. V dátovej sade *Hand Detection Dataset* sa nachádzalo viacero rúk a ľudí v rôznych prostrediach, pričom niektoré vzorky zobrazovali celé

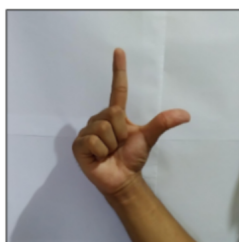
postavy ľudí vrátane ich rúk. *Hand Detection CV Dataset* pokrývala širšie spektrum vzoriek, od samotných rúk na jednofarebnom pozadí až po vzorky s takmer celými postavami ľudí. Vzorky z dátovej sady *Full Computer Vision Dataset* zobrazovali ruky ukazujúce znaky posunkovej reči, čo nám poskytlo dáta blízke reálnemu použitiu systému. Dátové sady *My First Project CV Dataset* a *Hand Gestures CV Dataset* obsahovali gestá rúk zachytené v rôznych prostrediach za účasti viacerých používateľov. Ukážky vzoriek týchto dátových sád sú zobrazené na obrázku 4.1.



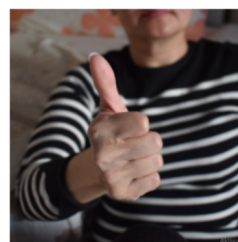
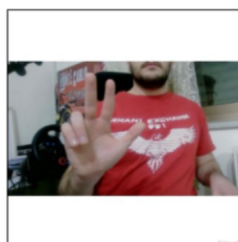
(a) Hand Recognition Dataset



(b) Hand Detection Dataset



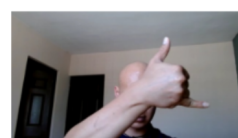
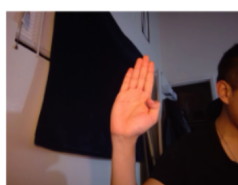
(c) Hand Detection CV Dataset



(d) My First Project CV Dataset



(e) Full Computer Vision Dataset



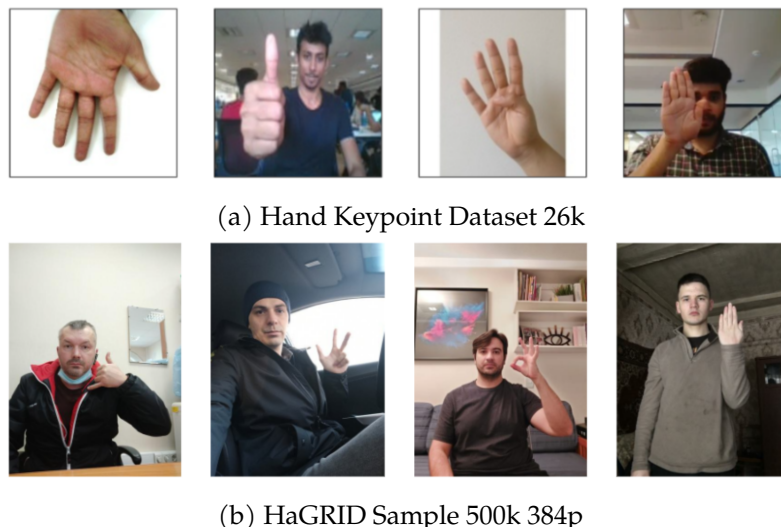
(f) Hand Gestures CV Dataset

Obrázok 4.1: Ukážky vzoriek verejných dátových sád - YOLO formát

Všetky tieto dátové sady obsahovali anotácie vo formáte YOLO, kde každý riadok anotačného súboru obsahuje identifikátor triedy `class_id`, normalizované súradnice stredu ohraničujúceho rámca `x_center` a `y_center` a jeho normalizovanú šírku `width` a výšku `height`. Všetky hodnoty okrem `class_id` sú normalizované v intervale od 0 do 1 vzhľadom na rozmery vstupného obrazu.

Dátová sada *Hand Keypoint Dataset 26k* obsahovala vzorky samotných rúk, ale aj vzorky s takmer celými postavami ľudí v rôznych pozíciách a prostrediach. Rozsiahlejšia dátová sada *HaGRID Sample 500k 384p* zahŕňala vzorky od 34730

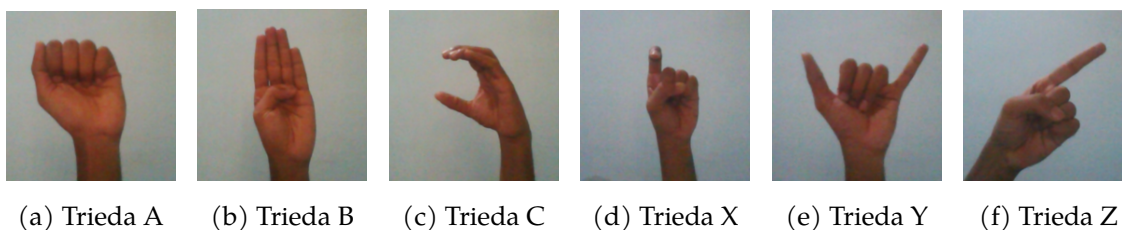
rôznych používateľov vo vekovom rozmedzí od 18 do 65 rokov, zobrazujúce až 18 rôznych typov gest rúk snímaných vo vzdialenosti 0,5 až 4 metre od kamery. Ukážky vzoriek týchto dátových sád sú zobrazené na obrázku 4.2.



Obrázok 4.2: Ukážky vzoriek verejných dátových sád - YOLO pose formát

Tieto dve dátové sady obsahovali anotácie vo formáte YOLO pose, ktorý rozširuje základný formát YOLO o normalizované súradnice jednotlivých kľúčových bodov, označované ako x_1, y_1 až x_k, y_k , a ich viditeľnosť v_1 až v_k .

Poslednou verejne dostupnou dátovou sadou bola *American Sign Language*, určená na klasifikáciu znakov posunkovej reči. Na rozdiel od predchádzajúcich dátových sád neobsahovala anotácie, ale vzorky rúk na bielom pozadí zobrazujúce jednotlivé znaky americkej posunkovej reči. Dátová sada bola rozdelená do 28 tried, kde 26 tried predstavovalo jednotlivé písmená abecedy, jedna trieda znak medzery a posledná trieda znak odstránenia posledného písmena. Každá trieda obsahovala približne 5900 vzoriek pochádzajúcich od jedného používateľa v rovnakom prostredí. Nízka variabilita tejto dátovej sady bola kompenzovaná kombináciou s vlastnými dátovými sadami obsahujúcimi vzorky v rôznych prostrediach. Ukážky vzoriek z niektorých tried tejto dátovej sady sú zobrazené na obrázku 4.3.



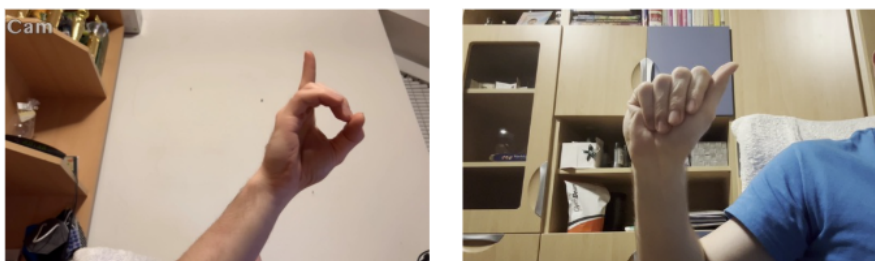
Obrázok 4.3: Ukážky vzoriek dátovej sady American Sign Language

4.2 Vlastné dátové sady

Verejne dostupné dátové sady boli doplnené o vlastné dáta s cieľom zvýšiť variabilitu dát a zlepšiť generalizačnú schopnosť jednotlivých modelov. Dáta pochádzali od jedného používateľa a boli zhotovené v šiestich rôznych prostrediach. Zozbierané dáta zachytávali ruky v rôznych pozíciách a s rôznym natočením pri zobrazovaní jednotlivých znakov posunkovej reči, aby čo najviac pripomínali reálne použitie systému.

Na efektívny zber dát bol vytvorený skript v jazyku Python využívajúci knižnice *OpenCV* a *Mediapipe*. Knižnica *OpenCV* zabezpečovala snímame obrazu z kamery a jeho základné spracovanie, zatiaľ čo knižnica *Mediapipe* umožňovala detekciu rúk a ich kľúčových bodov v reálnom čase. Pre každý znak posunkovej reči, znak medzery a znak odstránenia posledného písmena bolo vytvorených približne 100 vzoriek v každom zo šiestich prostredí, čo predstavovalo celkovo približne 16800 vzoriek.

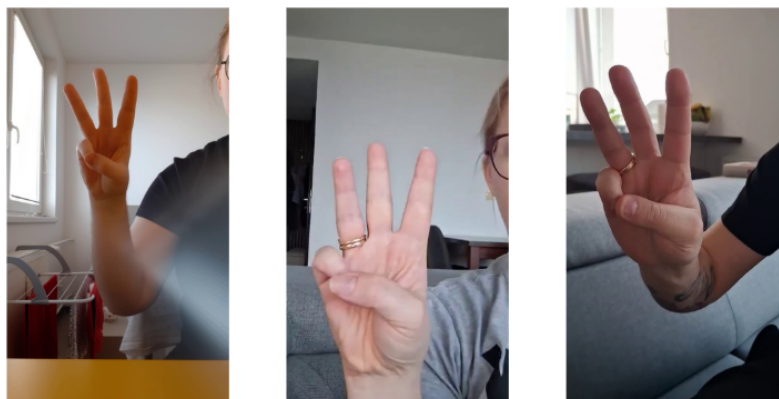
Keďže systém využíval dva varianty modelov na detekciu rúk, boli vytvorené dve samostatné dátové sady. Prvá dátová sada obsahovala anotácie vo formáte YOLO a bola označená ako *Custom Hand Detection Dataset*, skrátene *CHD Dataset*. Druhá dátová sada obsahovala anotácie vo formáte YOLO pose a bola označená ako *Custom Hand Keypoint Dataset*, skrátene *CHK Dataset*. Ukážky vzoriek týchto dátových sád sú zobrazené na obrázku 4.4.



Obrázok 4.4: Ukážky vzoriek dátových sád CHD Dataset a CHK Dataset

Osobitne boli vytvorené dátové sady určené výhradne na finálne testovanie modelov, ktoré neboli použité pri trénovaní ani ladení hyperparametrov. Vzorky pochádzali z videí poskytnutých tromi používateľmi vo veku 26, 30 a 35 rokov, pričom od každého používateľa bolo získaných 10 vzoriek pre každý znak, čo predstavovalo celkovo 840 vzoriek. Na rozdiel od trénovacích dátových sád nebola na tieto vzorky aplikovaná augmentácia, aby sa zachovali čo najreálnejšie podmienky testovania. Tieto testovacie dátové sady boli označené ako *Custom Hand Detection Test Dataset*, skrátene *CHDT Dataset* a *Custom Hand Keypoint Test Dataset*, skrátene *CHKDT Dataset*.

Dataset, skrátene *CHKT Dataset*. Ukážky vzoriek testovacích dátových sád sú zobrazené na obrázku 4.5.



Obrázok 4.5: Ukážky vzoriek dátových sád CHDT Dataset a CHKT Dataset

4.3 Príprava dát pre detekciu rúk

Pri modeloch určených na detekciu rúk nebolo potrebné vykonávať manuálne predspracovanie vstupných dát, keďže framework *Ultralytics* toto predspracovanie vykonáva automaticky v rámci procesu trénovania [64]. Medzi tieto automatické úpravy patrí napríklad zmena veľkosti vstupných obrazov na požadované rozmery so zachovaním pomeru strán, doplnenie okrajov, normalizácia pixelových hodnôt a transformácia obrazu do tenzorovej podoby.

Manuálne predspracovanie bolo potrebné vykonať len pri dátových sádach *Hand Keypoint Dataset 26k* a *HaGRID Sample 500k 384p*. Tieto dátové sady obsahovali anotácie vo formáte YOLO pose, no bolo ich potrebné využiť aj pri modeloch určených len na detekciu rúk bez kľúčových bodov. Z tohto dôvodu bola vykonaná transformácia anotácií z formátu YOLO pose do formátu YOLO pomocou skriptu v jazyku Python, ktorý extrahoval informácie o polohe ruky v obraze a uložil ich do nových anotačných súborov. Takto vzniknuté kópie dátových sád boli označené ako *Hand Detection Dataset HD 26k* a *HaGRID Sample 500k 384p HD Dataset*.

Iný prípad predstavovala dátová sada *American Sign Language*, ktorá neobsahovala žiadne anotácie. Na ich vytvorenie bola využitá knižnica *Mediapipe*, ktorá umožňovala automaticky detegovať ruky a ich kľúčové body na jednotlivých vzorkách. Aby boli generované anotácie čo najpresnejšie, bol parameter *min_detection_confidence* nastavený na hodnotu 0.9, čo znamenalo, že ruka bola detegovaná len pri minimálnej pravdepodobnosti detekcie 90%. Keďže dátová

sada obsahovala množstvo podobných vzoriek, bolo z každej triedy náhodne vybraných 750 vzoriek, pre ktoré boli následne vytvorené anotácie vo formáte YOLO aj YOLO pose. Po vytvorení anotácií bola vykonaná manuálna kontrola náhodne vybraných vzoriek, aby sa overila ich správnosť. Takto vznikli dve nové dátové sady označené ako *American Sign Language Hand Detection Dataset*, skrátene *ASLHD Dataset* a *American Sign Language Hand Keypoint Dataset*, skrátene *ASLHK Dataset*.

4.4 Augmentácia dát

Techniky augmentácie dát sa bežne využívajú v oblasti strojového učenia na zvýšenie variability dát a zlepšenie generalizačnej schopnosti modelov. Zároveň umožňujú umelo rozšíriť dostupné dátové sady bez potreby získavania nových dát.

V rámci našej práce sme využili nasledujúce augmentačné transformácie implementované pomocou knižnice *torchvision*:

- *RandomHorizontalFlip(p=1.0)* - horizontálne zrkadlenie obrazu s pravdepodobnosťou 100%, ktoré simuluje ľavú a pravú ruku,
- *GaussianBlur(kernel_size=7, sigma=(1.0, 1.0))* - rozmazanie obrazu simulujúce zníženie kvality snímky,
- *ColorJitter(brightness=(0.3, 0.5))* - zníženie jasů obrazu simulujúce nižšie osvetlenie prostredia,
- *ColorJitter(brightness=(1.3, 1.5))* - zvýšenie jasů obrazu simulujúce vyššie osvetlenie prostredia.

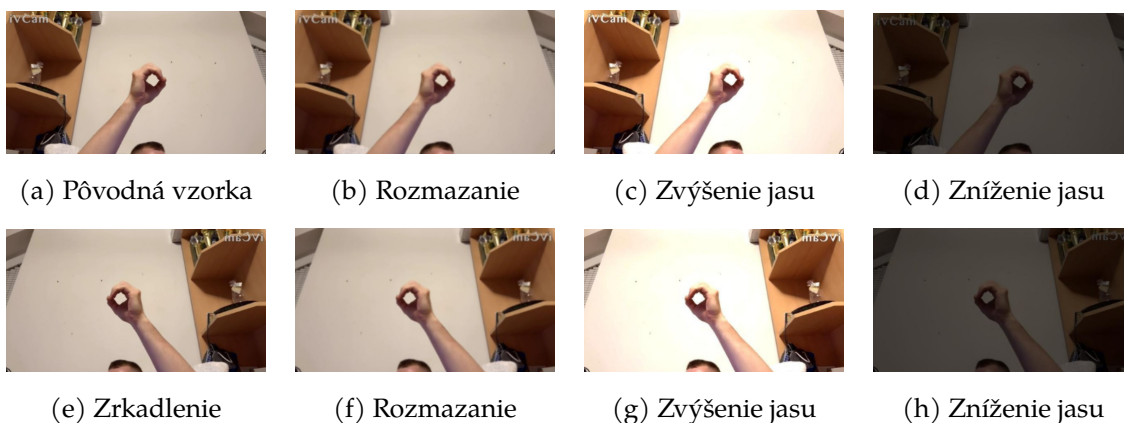
Pri verejne dostupných dátových sadách určených na detekciu rúk boli aplikované tri transformácie, konkrétne *GaussianBlur* a obe varianty *ColorJitter*, čím vznikli z každej pôvodnej vzorky tri nové augmentované vzorky, čo zvýšilo počet vzoriek približne štvornásobne. Transformácia *RandomHorizontalFlip* nebola aplikovaná, keďže tieto dátové sady už obsahovali vzorky ľavých aj pravých rúk. Keďže použité transformácie menili len jas a kvalitu obrazu bez zmeny polohy ruky, pôvodné anotácie bolo možné zachovať bez akýchkoľvek úprav. Pri dátových sadách *HaGRID Sample 500k 384p* a *HaGRID Sample 500k 384p HD Dataset* nebola aplikovaná žiadna transformácia, keďže tieto dátové sady už obsahovali vzorky s rôznymi svetelnými podmienkami a vzorky zobrazujúce ľavé aj pravé ruky. Ukážky augmentovaných vzoriek sú zobrazené na obrázku 4.6.



Obrázok 4.6: Ukážky augmentovaných vzoriek verejných dátových sád

Pri vlastných dátových sádach *CHD Dataset*, *CHK Dataset*, *ASLHD Dataset* a *ASLHK Dataset* boli aplikované všetky štyri transformácie. Ako prvá bola aplikovaná transformácia *RandomHorizontalFlip*, ktorá vykonala horizontálne zrkadlenie obrazu. Keďže táto transformácia mení polohu ruky v obraze, bolo potrebné prepočítať aj súradnice anotácií. Konkrétne bola prepočítaná hodnota x_center podľa vzorca $x_center' = 1 - x_center$ a v prípade formátu YOLO pose aj hodnoty xk súradníc kľúčových bodov podľa vzorca $xk' = 1 - xk$.

Následne boli na pôvodné aj zrkadlené vzorky aplikované transformácie *GaussianBlur* a obe varianty *ColorJitter*. Týmto spôsobom sa počet vzoriek v jednotlivých dátových sádach zvýšil približne osemnásobne. Ukážky augmentovaných vzoriek sú zobrazené na obrázku 4.7.



Obrázok 4.7: Ukážky augmentovaných vzoriek vlastných dátových sád

4.5 Príprava dát pre klasifikáciu znakov

Pri dátach určených na klasifikáciu znakov posunkovej reči bolo potrebné vykonať manuálne pedspracovanie. Cieľom tohto procesu bolo upraviť jednotlivé vzorky do podoby vhodnej pre klasifikačné modely.

Na tento účel bol vytvorený skript v jazyku Python využívajúci frameworky *PyTorch* a *OpenCV*. Skript načítaval vstupné obrazy spolu s príslušnými anotačnými súbormi, z ktorých boli extrahované informácie o polohe ruky v obraze a jej kľúčových bodoch. Na základe získanej oblasti obsahujúcej ruku bol vstupný obraz orezaný a následne rozšírený o biele okraje, aby mal výsledný obraz štvorcový pomer strán 1:1. Veľkosť výsledného obrazu bola následne zmenená na 256×256 pixelov pri zachovaní pomeru strán pomocou funkcie *Resize()* z knižnice *torchvision*.

V prípade dátových sád využívajúcich informácie o kľúčových bodoch ruky boli počas transformácie automaticky prepočítané súradnice kľúčových bodov a do výsledného obrazu bola vložená vizualizácia kľúčových bodov a kostry ruky zobrazená zelenou farbou. Pre dátové sady bez kľúčových bodov výsledný obraz obsahoval len orezanú oblasť ruky bez akejkoľvek vizualizácie.

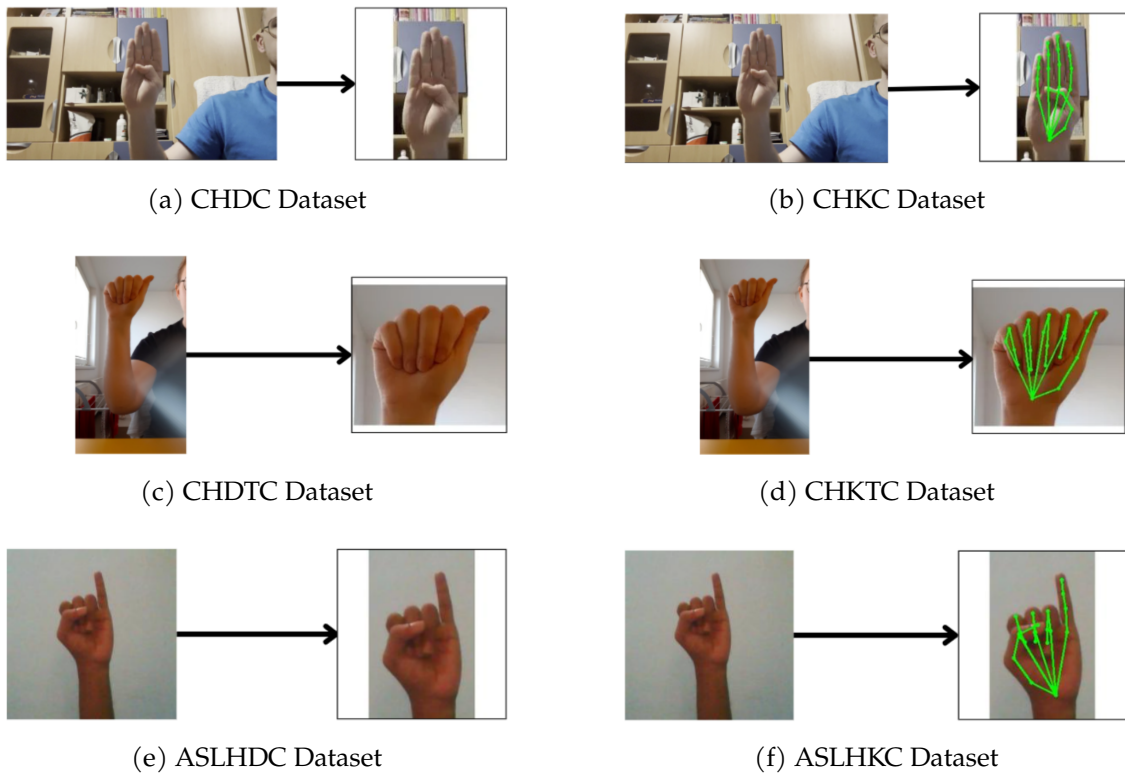
Keďže ešte pred samotným procesom transformácie bola na trénovacie dátové sady aplikovaná augmentácia dát, nové klasifikačné dátové sady už obsahovali aj augmentované vzorky. Naopak, testovacie dátové sady *CHDTC Dataset* a *CHKTC Dataset* neobsahovali žiadne augmentované vzorky, aby sa zachovali čo najreálnejšie podmienky testovania. Týmto spôsobom bolo vytvorených týchto šesť klasifikačných dátových sád:

- *Custom Hand Detection Classification Dataset*, skrátené *CHDC Dataset*,
- *Custom Hand Keypoint Classification Dataset*, skrátené *CHKC Dataset*,
- *American Sign Language Hand Detection Classification Dataset*, skrátené *ASLHDC Dataset*,
- *American Sign Language Hand Keypoint Classification Dataset*, skrátené *ASLHKC Dataset*,
- *Custom Hand Detection Test Classification Dataset*, skrátené *CHDTC Dataset*,
- *Custom Hand Keypoint Test Classification Dataset*, skrátené *CHKTC Dataset*.

Ukážky vzoriek pred a po procese predspracovania sú zobrazené na obrázku 4.8.

4.6 Rozdelenie dát

Po získaní, príprave a augmentácii dátových sád sme pristúpili k ich kombinácii do finálnych dátových sád určených na trénovanie a testovanie jednotlivých modelov.



Obrázok 4.8: Ukážky vzoriek pred a po procese predspracovania

Pri modeloch určených na detekciu rúk boli finálne dátové sady rozdelené na tréningovú, validačnú a testovaciu časť v pomere 75% pre tréningovanie, 10% pre validáciu a 15% pre testovanie. Pri klasifikačných modeloch sme využívali krížovú validáciu, a preto bolo potrebné vytvoriť len tréningovú a testovaciu časť v pomere 80% pre tréningovanie a 20% pre testovanie.

Na ladenie hyperparametrov modelov určených na detekciu rúk bez kľúčových bodov bola využitá dátová sada *Hand Detection Hyperparameters Tuning Dataset*, skrátene *HDHT Dataset*, ktorá vznikla z dátovej sady *Hand Keypoint Dataset HD 26k* bez augmentovaných vzoriek. Na ladenie hyperparametrov modelov určených na detekciu rúk a ich kľúčových bodov bola využitá dátová sada *Hand Keypoint Hyperparameters Tuning Dataset*, skrátene *HKHT Dataset*, ktorá vznikla z dátovej sady *Hand Keypoint Dataset 26k* bez augmentovaných vzoriek.

Pri ladení hyperparametrov klasifikačných modelov využívajúcich len informácie o polohe ruky bola použitá dátová sada *Custom Hand Detection Classification Hyperparameters Tuning Dataset*, skrátene *CHDCHT Dataset*, ktorá vznikla z dátovej sady *CHDC Dataset* bez augmentovaných vzoriek. Pri klasifikačných modeloch využívajúcich aj informácie o kľúčových bodoch ruky bola použitá dátová sada *Custom Hand Keypoint Classification Hyperparameters Tuning Dataset*, skrátene *CHKCHT Dataset*, ktorá vznikla z dátovej sady *CHKC Dataset* bez augmentova-

ných vzoriek. Všetky dátové sady určené na ladenie hyperparametrov boli rozdelené na tréningovú, validačnú a testovaciu časť v pomere 75%, 10% a 15%.

Na samotné tréningovanie a testovanie modelov určených na detekciu rúk a ich kľúčových bodov bola vytvorená finálna dátová sada *Hand Keypoint Training Dataset*, skrátene *HKT Dataset*. Táto dátová sada kombinuje verejne dostupné dátové sady s vlastnými dátami, čím sa zabezpečuje dostatočná variabilita vzoriek pre tréningovanie detekčných modelov s kľúčovými bodmi. Prehľad použitých dátových sád je uvedený v tabuľke 4.2. Celkovo táto dátová sada obsahovala 750795 vzoriek.

Tabuľka 4.2: Dátové sady použité v rámci dátovej sady HKT Dataset

Názov dátovej sady	Augmentácia	Počet vzoriek
Hand Keypoint Dataset 26k	ÁNO	107072
HaGRID Sample 500k 384p	NIE	509323
CHK Dataset	ÁNO	134400

Na tréningovanie a testovanie modelov určených na detekciu rúk bez kľúčových bodov bola vytvorená finálna dátová sada *Hand Detection Training Dataset*, skrátene *HDT Dataset*. Keďže tieto modely nevyužívajú kľúčové body, bolo možné zaradiť širšie spektrum verejne dostupných dátových sád zameraných výhradne na detekciu rúk, spolu s vlastnými dátami. Prehľad použitých dátových sád je uvedený v tabuľke 4.3. Celkovo táto dátová sada obsahovala 1105411 vzoriek.

Tabuľka 4.3: Dátové sady použité v rámci dátovej sady HDT Dataset

Názov dátovej sady	Augmentácia	Počet vzoriek
Hand Recognition Dataset	ÁNO	1620
Hand Keypoint Dataset HD 26k	ÁNO	107072
Hand Detection Dataset	ÁNO	8244
Hand Detection CV Dataset	ÁNO	198580
Full Computer Vision Dataset	ÁNO	54504
My First Project CV Dataset	ÁNO	56736
Hand Gestures CV Dataset	ÁNO	34932
HaGRID Sample 500k 384p HD Dataset	NIE	509323
CHD Dataset	ÁNO	134400

Na tréningovanie a testovanie klasifikačných modelov využívajúcich len informácie o polohe ruky bola vytvorená finálna dátová sada *Hand Detection Classification Training Dataset*, skrátene *HDCT Dataset*. Táto dátová sada vznikla kombi-

náciou verejne dostupnej dátovej sady *ASLHDC Dataset* a vlastnej dátovej sady *CHDC Dataset*, čím sa zabezpečila kombinácia vzoriek z rôznych prostredí. Prehľad použitých dátových sád je uvedený v tabuľke 4.4. Celkovo táto dátová sada obsahovala 302400 vzoriek.

Tabuľka 4.4: Dátové sady použité v rámci dátovej sady HDCT Dataset

Názov dátovej sady	Augmentácia	Počet vzoriek
ASLHDC Dataset	ÁNO	168000
CHDC Dataset	ÁNO	134400

Na trénovanie a testovanie klasifikačných modelov využívajúcich aj informácie o kľúčových bodoch ruky bola vytvorená finálna dátová sada *Hand Keypoint Classification Training Dataset*, skrátene *HKCT Dataset*. Táto dátová sada kombinuje dátovú sadu *ASLHKC Dataset* a vlastnú dátovú sadu *CHKC Dataset*, pričom obsahuje vizualizácie kľúčových bodov ruky. Prehľad použitých dátových sád je uvedený v tabuľke 4.5. Celkovo táto dátová sada obsahovala 302400 vzoriek.

Tabuľka 4.5: Dátové sady použité v rámci dátovej sady HKCT Dataset

Názov dátovej sady	Augmentácia	Počet vzoriek
ASLHKC Dataset	ÁNO	168000
CHKC Dataset	ÁNO	134400

Na finálne testovanie jednotlivých modelov boli využité samostatné dátové sady *CHDT Dataset*, *CHKT Dataset*, *CHDTC Dataset* a *CHKTC Dataset*, každá obsahujúca 840 vzoriek od troch rôznych používateľov.

5 Implementácia systému na rozpoznávanie znakov posunkovej reči

V tejto kapitole popisujeme konkrétne implementačné detaily jednotlivých modulov systému. Pre každý modul uvádzame použité hodnotiace metriky, proces ladenia hyperparametrov, trénovanie modelov, ich vyhodnotenie na neznámych dátach a optimalizáciu pre nasadenie na mobilných zariadeniach. Záverečná časť kapitoly sa venuje vývoju mobilnej aplikácie a používateľskému testovaniu.

5.1 Modul detekcie rúk

V tejto časti popisujeme implementáciu modulu detekcie rúk a ich kľúčových bodov. Postupne sa venujeme použitým hodnotiacim metrikám, ladeniu hyperparametrov, trénovaniu modelov, ich vyhodnoteniu na neznámych dátach a optimalizácii pre nasadenie na mobilných zariadeniach.

5.1.1 Hodnotiace metriky pre detekciu rúk a kľúčových bodov

Na hodnotenie výkonu architektúr určených na detekciu rúk bez kľúčových bodov bola použitá metrika *Box mAP(50-95)* [65]. Táto metrika hodnotí presnosť lokalizácie objektov pomocou ohraničujúcich rámcov prostredníctvom metriky *Intersection over Union*, skrátene *IoU*, ktorá vyjadruje mieru prekrytia medzi predikovaným a referenčným ohraničujúcim rámcom a je definovaná vzťahom

$$IoU = \frac{A \cap B}{A \cup B} \quad (5.1)$$

kde A predstavuje predikovaný ohraničujúci rámec a B predstavuje referenčný ohraničujúci rámec. Hodnota *Box mAP(50-95)* sa vypočíta ako priemer hodnôt presnosti detekcie (z angl. *Average Precision*), skrátene *AP*, pri rôznych prahoch *IoU* v intervale od 0,50 do 0,95 s krokom 0,05, čo poskytuje komplexnejšie hodnotenie presnosti detekcie v porovnaní s tradičnou metrikou *mAP50*.

Pri hodnotení výkonu architektúr určených na detekciu rúk a ich kľúčových bodov bola využitá aj metrika *Pose mAP(50-95)* [65]. Na rozdiel od metriky *Box mAP(50-95)*, ktorá hodnotí presnosť ohraničujúcich rámcov, táto metrika hodnotí presnosť lokalizácie jednotlivých kľúčových bodov pomocou metriky *Object Keypoint Similarity*, skrátene OKS. Metrika OKS vyjadruje podobnosť medzi predikovanými a referenčnými kľúčovými bodmi na základe ich vzájomnej vzdialenosti a veľkosti detegovaného objektu. Hodnota *Pose mAP(50-95)* sa vypočíta analogicky ako *Box mAP(50-95)*, a teda ako priemer hodnôt presnosti detekcie pri rôznych prahoch OKS v intervale od 0,50 do 0,95.

Na hodnotenie rýchlosti spracovania snímok bola pri oboch typoch architektúr využitá metrika *Frames Per Second*, skrátene FPS, ktorá vyjadruje počet snímok, ktoré je model schopný spracovať za jednu sekundu. Táto metrika je kľúčová pri hodnotení vhodnosti modelov pre nasadenie v systémoch pracujúcich v reálnom čase na mobilných zariadeniach. Zároveň bola táto metrika využitá len pri finálnom vyhodnotení modelov na neznámych dátach.

5.1.2 Ladenie hyperparametrov

Pre každú z vybraných architektúr bola vykonaná konfigurácia modelov s cieľom dosiahnuť čo najlepší výkon. Na ladenie hyperparametrov modelov určených na detekciu rúk bez kľúčových bodov bola využitá dátová sada *HDHT Dataset*, zatiaľ čo pri modeloch určených na detekciu rúk a ich kľúčových bodov bola použitá dátová sada *HKHT Dataset*.

Ladenie bolo realizované metódou *grid search* v jazyku Python pomocou triedy YOLO z frameworku *Ultralytics*. Konkrétne sme testovali kombinácie nasledujúcich hyperparametrov:

- *lr0* - počiatočná hodnota rýchlosti učenia: {0.001, 0.005, 0.01},
- *lrf* - finálna hodnota rýchlosti učenia: {0.01, 0.05, 0.1},
- *weight_decay* - regularizačný parameter obmedzujúci veľkosť váh modelu: {0.0005, 0.001},
- *momentum* - parameter stabilizujúci proces učenia: {0.9, 0.95},
- *optimizer* - optimalizačný algoritmus: {SGD, AdamW}.

Hodnoty *lr0* a *lrf* boli zvolené s cieľom zistiť, či modely dosahujú lepšie výsledky pri jemnejšom alebo agresívnejšom učení. Ostatné hyperparametre nadobúdali štandardné hodnoty bežne používané pri trénovaní modelov na detekciu objektov.

Každý model bol trénovaný po dobu 10 epoch so vstupnou veľkosťou obrazov 320×320 pixelov a veľkosťou dávky (z angl. *batch size*) 32. Augmentácia dát implementovaná vo frameworku *Ultralytics* bola vypnutá, keďže pri celkovom počte 72 testovaných kombinácií by jej použitie predĺžilo čas potrebný na ladenie. Cieľom tohto procesu bolo získať prvotné výsledky o správaní modelov pri rôznych konfiguráciách hyperparametrov, pričom samotné finálne tréningovanie prebiehalo už na plných dátových sadách obsahujúcich augmentované vzorky. Jednotlivé modely boli po ladení otestované na testovacej časti príslušnej dátovej sady a na základe dosiahnutých výsledkov bola pre každú architektúru určená najlepšia konfigurácia hyperparametrov.

Výsledné hodnoty *Box mAP*(50-95) pre architektúry určené na detekciu rúk bez kľúčových bodov sú uvedené v tabuľke 5.1. Najlepšie výsledky dosiahla architektúra *YOLO11n* s hodnotou *Box mAP*(50-95) **0.9067**. Najlepšia konfigurácia hyperparametrov pre architektúru *YOLOv8n* bola $lr0 = 0.005$, $lrf = 0.05$, $weight_decay = 0.0005$, $momentum = 0.90$ a $optimizer = SGD$. Pre architektúru *YOLO11n* bola najlepšia konfigurácia hyperparametrov $lr0 = 0.001$, $lrf = 0.05$, $weight_decay = 0.001$, $momentum = 0.95$ a $optimizer = SGD$.

Tabuľka 5.1: Výsledné *Box mAP*(50-95) skóre pre architektúry určené na detekciu rúk bez kľúčových bodov po ladení hyperparametrov

Architektúra	Box mAP(50-95)
YOLOv8n	0.9037
YOLO11n	0.9067

Výsledné hodnoty *Box mAP*(50-95) a *Pose mAP*(50-95) pre architektúry určené na detekciu rúk a ich kľúčových bodov sú uvedené v tabuľke 5.2. Najlepšie výsledky dosiahla architektúra *YOLO11n-pose* s hodnotou *Box mAP*(50-95) **0.9581** a hodnotou *Pose mAP*(50-95) **0.8812**. Obidve architektúry dosiahli najlepšie výsledky so spoločnými hodnotami $lrf = 0.01$, $weight_decay = 0.001$, $momentum = 0.95$ a $optimizer = SGD$. Líšili sa len hodnotou $lr0$, ktorá pre architektúru *YOLOv8n-pose* bola $lr0 = 0.01$ a pre architektúru *YOLO11n-pose* $lr0 = 0.005$.

Tabuľka 5.2: Výsledné *Box mAP*(50-95) a *Pose mAP*(50-95) skóre pre architektúry určené na detekciu rúk a ich kľúčových bodov po ladení hyperparametrov

Architektúra	Box mAP(50-95)	Pose mAP(50-95)
YOLOv8n-pose	0.9560	0.8800
YOLO11n-pose	0.9581	0.8812

5.1.3 Trénovanie modelov s ich optimálnou konfiguráciou

Po získaní optimálnych konfigurácií hyperparametrov boli jednotlivé architektúry natréňované na finálnych dátových sadách. Na trénovanie a testovanie architektúr *YOLOv8n* a *YOLO11n* určených na detekciu rúk bez kľúčových bodov bola využitá dátová sada *HDT Dataset*. Architektúry *YOLOv8n-pose* a *YOLO11n-pose* určené na detekciu rúk a ich kľúčových bodov boli trénované na dátovej sade *HKT Dataset*.

Trénovanie modelov bolo realizované v jazyku Python pomocou frameworku *Ultralytics* a jeho triedy *YOLO*. Modely boli trénované po dobu 50 epoch s optimalizovanými hyperparametrami. Vstupná veľkosť obrazov bola nastavená na 320×320 pixelov a veľkosť dávky na hodnotu 32. Rovnako ako pri ladení hyperparametrov boli vypnuté techniky augmentácie implementované vo frameworku *Ultralytics*, keďže použité dátové sady už obsahovali augmentované vzorky. Predspracovanie vstupných dát nebolo potrebné vykonávať manuálne, pretože framework *Ultralytics* tento proces vykonáva automaticky počas trénovania.

Po dokončení trénovania boli jednotlivé modely otestované na testovacej časti príslušných dátových sad. Výkon architektúr *YOLOv8n* a *YOLO11n* bol hodnotený pomocou metriky *Box mAP(50-95)*, keďže tieto architektúry sú určené len na detekciu ohraničujúcich rámcov bez odhadu kľúčových bodov. Pri architektúrach *YOLOv8n-pose* a *YOLO11n-pose* bola okrem metriky *Box mAP(50-95)* využitá aj metrika *Pose mAP(50-95)*, ktorá hodnotí presnosť lokalizácie kľúčových bodov.

Výsledné hodnoty *Box mAP(50-95)* na testovacej časti dátovej sady *HDT Dataset* pre architektúry určené na detekciu rúk bez kľúčových bodov sú uvedené v tabuľke 5.3. Najlepšie výsledky dosiahla architektúra *YOLO11n* s hodnotou *Box mAP(50-95)* **0.9472**. Obidve architektúry zároveň dosiahli výrazné zlepšenie oproti výsledkom získaným pri ladení hyperparametrov.

Tabuľka 5.3: Výsledné *Box mAP(50-95)* skóre pre architektúry určené na detekciu rúk bez kľúčových bodov po trénovaní a testovaní

Architektúra	Box mAP(50-95)
YOLOv8n	0.9421
YOLO11n	0.9472

Výsledné hodnoty *Box mAP(50-95)* a *Pose mAP(50-95)* na testovacej časti dátovej sady *HKT Dataset* pre architektúry určené na detekciu rúk a ich kľúčových bodov sú uvedené v tabuľke 5.4. Najlepšie výsledky dosiahla architektúra *YOLO11n-pose* s hodnotou *Box mAP(50-95)* **0.9818** a *Pose mAP(50-95)* **0.9717**. V tom-

to prípade taktiež došlo k výraznému zlepšeniu oproti výsledkom získaným pri ladení hyperparametrov.

Tabuľka 5.4: Výsledné Box mAP(50-95) a Pose mAP(50-95) skóre pre architektúry určené na detekciu rúk a ich kľúčových bodov po trénoch a testovaní

Architektúra	Box mAP(50-95)	Pose mAP(50-95)
YOLOv8n-pose	0.9797	0.9689
YOLO11n-pose	0.9818	0.9717

5.1.4 Vyhodnotenie natrénovaných modelov

Po procese trénoch a testovania jednotlivých architektúr sme vykonali finálne vyhodnotenie ich výkonu na dátových sadách *CHDT Dataset* a *CHKT Dataset*, ktoré obsahovali vzorky, ktoré neboli použité počas trénoch ani ladenia hyperparametrov. Toto testovanie nám umožnilo objektívne vyhodnotiť výkon modelov na neznámych dátach.

Pri finálnom vyhodnotení bola okrem metrík *Box mAP(50-95)* a *Pose mAP(50-95)* využitá aj metrika *FPS*. Pri meraní metriky *FPS* sme vykonávali 40 *warm-up runs*, po ktorých model spracoval ďalších 800 testovacích snímok. Meranie bolo realizované na procesore Intel Core i5-13450HX s operačnou pamäťou 16 GB.

Výsledné hodnoty *Box mAP(50-95)* a *FPS* pre architektúry určené na detekciu rúk bez kľúčových bodov sú uvedené v tabuľke 5.5. Architektúra *YOLO11n* dosiahla vyššie *Box mAP(50-95)* skóre s hodnotou **0.9449**, zatiaľ čo architektúra *YOLOv8n* dosiahla vyššiu rýchlosť spracovania snímok, konkrétne **26.19 FPS**.

Tabuľka 5.5: Výsledné Box mAP(50-95) a FPS skóre pre architektúry určené na detekciu rúk bez kľúčových bodov po finálnom testovaní

Architektúra	Box mAP(50-95)	FPS
YOLOv8n	0.9398	26.19
YOLO11n	0.9449	20.56

Výsledné hodnoty *Box mAP(50-95)*, *Pose mAP(50-95)* a *FPS* pre architektúry určené na detekciu rúk a ich kľúčových bodov sú uvedené v tabuľke 5.6. Architektúra *YOLO11n-pose* dosiahla najvyššie hodnoty *Box mAP(50-95)* **0.9809** a *Pose mAP(50-95)* **0.9557**, zatiaľ čo architektúra *YOLOv8n-pose* dosiahla vyššiu rýchlosť spracovania snímok, konkrétne **23.93 FPS**. Všetky architektúry dosiahli mierne zhoršenie v porovnaní s výsledkami získanými pri trénoch, čo je očakávané správanie pri testovaní na neznámych dátach.

Tabuľka 5.6: Výsledné Box mAP(50-95), Pose mAP(50-95) a FPS skóre pre architektúry určené na detekciu rúk a ich kľúčových bodov po finálnom testovaní

Architektúra	Box mAP(50-95)	Pose mAP(50-95)	FPS
YOLOv8n-pose	0.9786	0.9526	23.93
YOLO11n-pose	0.9809	0.9557	19.22

5.1.5 Optimalizácia a výber finálnych modelov

Po získaní výsledkov z finálneho testovania sme pristúpili k optimalizácii modelov pre nasadenie na mobilných zariadeniach. Cieľom optimalizácie bolo zvýšiť hodnotu metriky *FPS* pri zachovaní čo najvyšších hodnôt metrík *Box mAP(50-95)* a *Pose mAP(50-95)*.

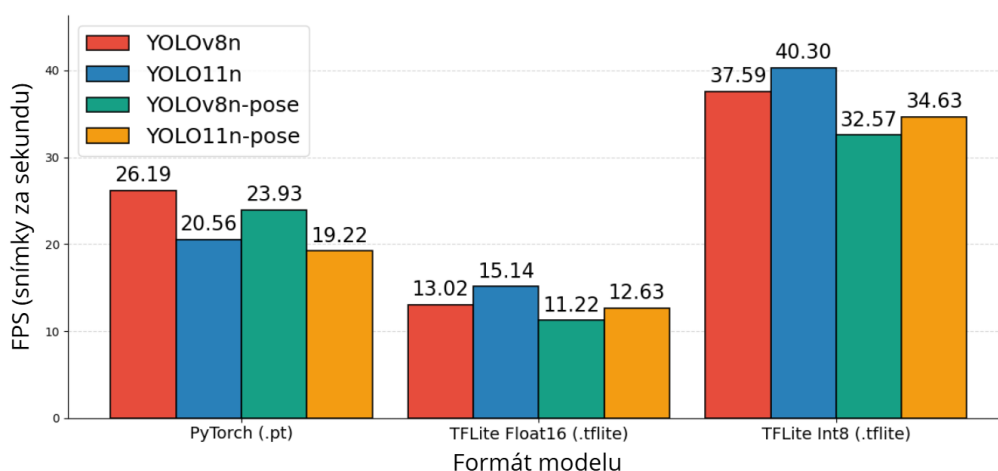
Optimalizácia bola realizovaná prevodom modelov z formátu *PyTorch* do formátu *ONNX* a následne do formátu *TensorFlow Lite*, ktorý je určený pre nasadenie na mobilných zariadeniach. Po prevode bola vykonaná kvantizácia modelov do formátov *Float16* a *Int8*. Kvantizácia je proces, pri ktorom sa váhy modelu prevádzajú z vyššej numerickej presnosti, napríklad 32-bitových čísel s pohyblivou rádovou čiarkou, na nižšiu presnosť, napríklad 8-bitové celé čísla. Tento proces znižuje veľkosť modelu a zvyšuje rýchlosť inferencie pri minimálnej strate presnosti. Optimalizované modely boli opäť testované na finálnych testovacích dátových sadách *CHDT Dataset* a *CHKT Dataset*. Meranie metriky *FPS* po optimalizácii bolo realizované priamo na mobilnom zariadení Samsung Galaxy A71 s procesorom Qualcomm Snapdragon 730 pomocou nástroja *TensorFlow Lite Model Benchmark Tool*. Každý model bol testovaný s 40 *warm-up runs* a následne s 800 testovacími snímkami.

Z výsledkov zobrazených na obrázku 5.1 vyplýva, že všetky modely prevedené do formátu *TensorFlow Lite* s kvantizáciou do formátu *Int8* dosiahli zvýšenie hodnoty metriky *FPS*. Spomedzi architektúr určených na detekciu rúk bez kľúčových bodov dosiahla najvyššiu hodnotu metriky *FPS* architektúra *YOLO11n* s hodnotou **40.30 FPS** pri kvantizácii do formátu *Int8*. Pri architektúrach určených na detekciu rúk a ich kľúčových bodov dosiahla najvyšší výkon architektúra *YOLO11n-pose* s hodnotou **34.63 FPS** pri kvantizácii do formátu *Int8*.

Pri kvalitatívnych metrikách *Box mAP(50-95)* a *Pose mAP(50-95)* si všetky modely zachovali vysoké hodnoty aj po optimalizácii, pričom pokles bol minimálny. Výsledky metriky *Box mAP(50-95)* pre architektúry určené na detekciu rúk bez kľúčových bodov po optimalizácii sú zobrazené na obrázku 5.2.

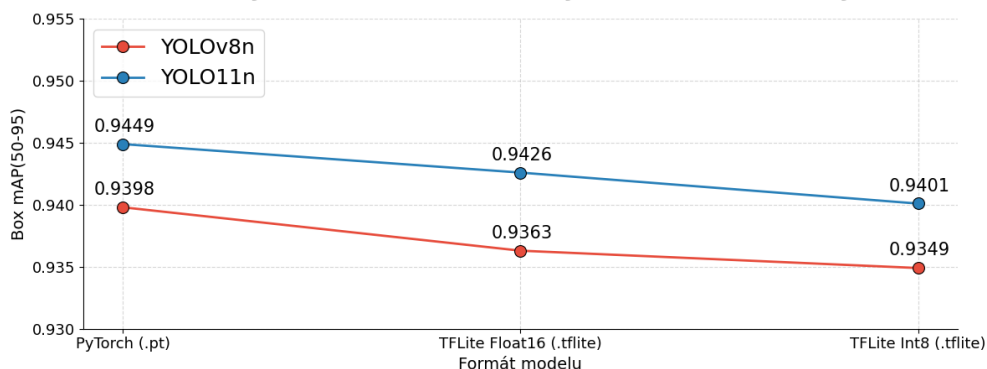
Výsledky metrík *Box mAP(50-95)* a *Pose mAP(50-95)* pre architektúry určené

Porovnanie metriky FPS modelov detekcie rúk v rôznych formátoch



Obrázok 5.1: Hodnoty metriky FPS pre jednotlivé detekčné architektúry po optimalizácii

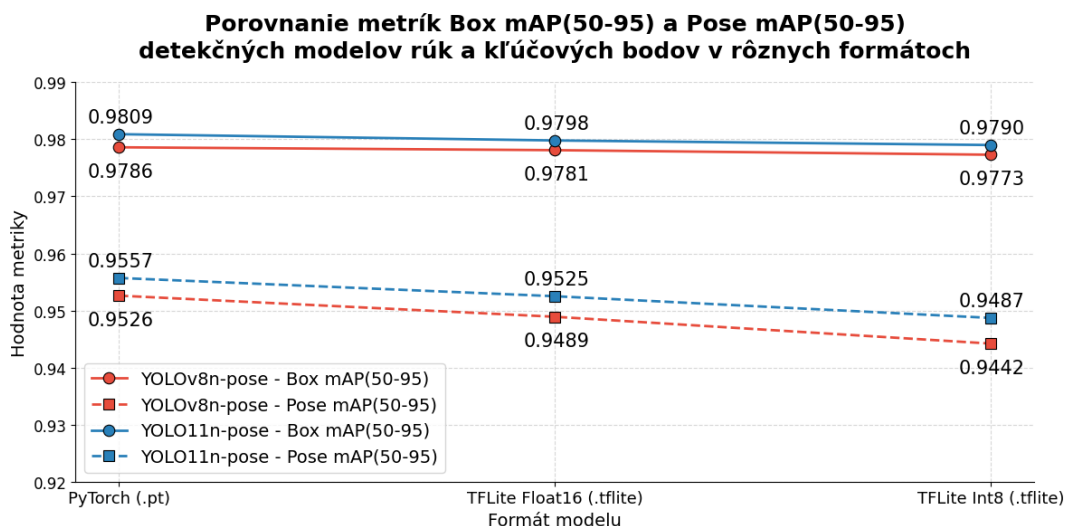
Porovnanie metriky Box mAP(50-95) detekčných modelov rúk v rôznych formátoch



Obrázok 5.2: Box mAP(50-95) skóre pre architektúry určené na detekciu rúk bez kľúčových bodov po optimalizácii

na detekciu rúk a ich kľúčových bodov po optimalizácii sú zobrazené na obrázku 5.3.

Na základe získaných výsledkov boli za finálne modely pre modul detekcie rúk zvolené architektúry *YOLO11n* a *YOLO11n-pose* v optimalizovanom formáte *TensorFlow Lite* s kvantizáciou do formátu *Int8*, ktoré predstavujú najlepší kompromis medzi presnosťou detekcie a rýchlosťou spracovania snímok na mobilných zariadeniach.



Obrázok 5.3: Box mAP(50-95) a Pose mAP(50-95) skóre pre architektúry určené na detekciu rúk a ich kľúčových bodov po optimalizácii

5.2 Modul klasifikácie znakov posunkovej reči

V tejto časti popisujeme implementáciu modulu klasifikácie znakov posunkovej reči. Postupne sa venujeme použitým hodnotiacim metrikám, ladeniu hyperparametrov, trénovaniu modelov, ich vyhodnoteniu na neznámych dátach a optimalizácii pre nasadenie na mobilných zariadeniach.

5.2.1 Hodnotiace metriky klasifikácie znakov posunkovej reči

Na hodnotenie výkonu klasifikačných modelov boli použité metriky *AUC skóre* [66] (z angl. *Area Under the ROC Curve*), *Geometrický priemer* [67] (z angl. *Geometric Mean*), skrátene *G-mean* a *FPS*. Metrika *AUC skóre* vyjadruje plochu pod krivkou ROC (z angl. *Receiver Operating Characteristic*). Krivka ROC zobrazuje vzťah medzi mierou správne predikovaných pozitívnych vzoriek (z angl. *True Positive Rate*) a mierou nesprávne predikovaných pozitívnych vzoriek (z angl. *False Positive Rate*) pri rôznych hodnotách rozhodovacieho prahu. *AUC skóre* sa pohybuje v rozmedzí od 0 po 1, kde hodnota 1 predstavuje ideálny klasifikátor a hodnota 0.5 zodpovedá náhodnému hádaniu.

Metrika *G-mean* hodnotí schopnosť klasifikátora správne rozpoznávať pozitívne aj negatívne triedy súčasne. *G-mean* sa vypočíta ako geometrický priemer miery správne predikovaných pozitívnych vzoriek a miery správne predikovaných negatívnych vzoriek (z angl. *True Negative Rate*) podľa vzorca

$$G-mean = \sqrt{TPR \times TNR} \quad (5.2)$$

Hodnota *G-mean* sa pohybuje v rozmedzí od 0 po 1, kde vyššia hodnota znamená lepšiu vyváženosť klasifikátora medzi jednotlivými triedami.

Metrika *FPS* bola využitá výhradne pri finálnom vyhodnotení modelov na neznámych dátach, kde hodnotí vhodnosť modelov pre nasadenie v systémoch pracujúcich v reálnom čase na mobilných zariadeniach.

5.2.2 Ladenie hyperparametrov

Pre každú z vybraných architektúr bola vykonaná konfigurácia modelov s cieľom dosiahnuť čo najlepší výkon. Na ladenie hyperparametrov klasifikačných modelov využívajúcich len informácie o polohe ruky bola použitá dátová sada *CHDCHT Dataset*, zatiaľ čo pri modeloch využívajúcich aj informácie o kľúčových bodoch ruky bola použitá dátová sada *CHKCHT Dataset*.

Vstupné obrazy boli načítané, dekodované a konvertované do tenzorovej reprezentácie. Pri architektúre *MobileNetV2* bola vykonaná normalizácia hodnôt pixelov do intervalu od -1 do 1, zatiaľ čo pri architektúre *ResNet50* boli konvertované farebné kanály z formátu RGB do formátu BGR spolu s normalizáciou hodnôt pixelov. Veľkosť vstupných obrazov bola nastavená na 256×256 pixelov.

Ladenie bolo realizované v jazyku Python pomocou frameworku *TensorFlow* a nástroja *Keras Tuner*. Na rozdiel od metódy *grid search* použitej pri detekčných modeloch sme tu využili Bayesovskú optimalizáciu (z angl. *Bayesian optimization*), ktorá sa učí z každého pokusu a je celkovo efektívnejšia v porovnaní s *grid search* prístupom. Konkrétne sme optimalizovali nasledujúce hyperparametre:

- *learning_rate* – rýchlosť učenia modelu: $\{1 \times 10^{-4}, 1 \times 10^{-2}\}$,
- *dense_units* – počet neurónov v plne prepojenej vrstve: $\{256, 512, 1024\}$,
- *dropout_rate* – podiel neurónov deaktivovaných počas tréningu: $\{0.2, 0.3, 0.4, 0.5\}$,
- *l2_reg* – regularizačný parameter penalizujúci veľké váhy modelu: $\{0.0, 1 \times 10^{-4}, 1 \times 10^{-3}\}$,
- *batch_size* – počet vzoriek spracovaných naraz počas jednej iterácie učenia: $\{16, 32, 64\}$,
- *unfreeze_layers* – počet vrstiev predtrénovaného modelu, ktoré sa počas tréningu odomknú: $\{0, 10, 20, 30\}$,
- *use_bn_head* – určuje, či sa vo výstupnej časti modelu použije vrstva Batch Normalization: $\{True, False\}$,

- *dense_layers* – počet plne prepojených vrstiev v klasifikačnej časti siete: {1, 2}.

Hodnota hyperparametra *learning_rate* bola hľadaná v danom intervale s cieľom zistiť, či modely dosahujú lepšie výsledky pri jemnejšom alebo agresívnejšom učení. Ostatné hyperparametre nadobúdali štandardné hodnoty bežne používané pri tréňovaní klasifikačných modelov.

Pri stratégii Bayesovskej optimalizácie bolo vykonaných celkovo 60 pokusov, pričom prvých 10 tvorili náhodne zvolené kombinácie slúžiace na vytvorenie počiatočného modelu priestoru hyperparametrov. Každý pokus bol tréňovaný po dobu maximálne 10 epoch s mechanizmom *EarlyStopping*, ktorý ukončil tréňovanie ak sa validačné *AUC skóre* nezlepšilo počas troch po sebe idúcich epoch. Augmentácia dát bola vypnutá z rovnakého dôvodu ako pri ladení hyperparametrov detekčných modelov, teda z dôvodu skrátenia doby tréňovania, pričom cieľom bolo získať prvotné výsledky o správaní modelov pri rôznych konfiguráciách. Finálne tréňovanie prebiehalo na plných dátových sadách obsahujúcich aj augmentované vzorky.

Výsledné hodnoty *AUC skóre* a *G-mean* pre architektúry využívajúce len informácie o polohe ruky sú uvedené v tabuľke 5.7. Najlepšie výsledky dosiahla architektúra *ResNet50* s hodnotou *AUC skóre* **0.9875** a *G-mean* **0.9837**. Pre obidve architektúry bola najlepšia spoločná hodnota *learning_rate* = 0.0001 a *unfreeze_layers* = 30. Architektúra *MobileNetV2* dosiahla najlepšie výsledky s hodnotami *dense_units* = 1024, *dropout_rate* = 0.5, *l2_reg* = 1×10^{-3} , *batch_size* = 16, *use_bn_head* = False a *dense_layers* = 1. Architektúra *ResNet50* mala najlepšiu konfiguráciu s hodnotami *dense_units* = 256, *dropout_rate* = 0.2, *l2_reg* = 0.0, *batch_size* = 16, *use_bn_head* = True a *dense_layers* = 2.

Tabuľka 5.7: Výsledné *AUC skóre* a *G-mean* hodnoty pre architektúry využívajúce len informácie o polohe ruky po ladení hyperparametrov

Architektúra	AUC skóre	G-mean
MobileNetV2	0.9852	0.9821
ResNet50	0.9875	0.9837

Výsledné hodnoty *AUC skóre* a *G-mean* pre architektúry využívajúce aj informácie o kľúčových bodoch ruky sú uvedené v tabuľke 5.8. Najlepšie výsledky dosiahla architektúra *MobileNetV2 - keypoints* s hodnotou *AUC skóre* **0.9898** a *G-mean* **0.9867**. Zaujímavosťou je, že obidve architektúry dosiahli najlepšie výsledky s rovnakými hodnotami hyperparametrov, konkrétne *learning_rate* = 0.0001, *dense_units* = 256, *dropout_rate* = 0.5, *l2_reg* = 1×10^{-3} , *batch_size* = 64, *unfreeze*

$ze_layers = 30$, $use_bn_head = True$ a $dense_layers = 2$.

Tabuľka 5.8: Výsledné AUC skóre a G-mean hodnoty pre architektúry využívajúce aj informácie o kľúčových bodoch ruky po ladení hyperparametrov

Architektúra	AUC skóre	G-mean
MobileNetV2 - keypoints	0.9898	0.9867
ResNet50 - keypoints	0.9891	0.9853

5.2.3 Trénovanie modelov s ich optimálnou konfiguráciou

Po získaní optimálnych konfigurácií hyperparametrov boli jednotlivé architektúry natréňované na finálnych dátových sadách. Na trénovanie a testovanie klasifikačných modelov využívajúcich len informácie o polohe ruky bola použitá dátová sada *HDCT Dataset*. Pri modeloch využívajúcich aj informácie o kľúčových bodoch ruky bola použitá dátová sada *HKCT Dataset*.

Trénovanie modelov bolo realizované v jazyku Python pomocou frameworku *TensorFlow* a knižnice *Keras*. Pri tvorbe modelov bol využitý prístup *transfer learning*, pri ktorom boli váhy základnej časti modelu inicializované pomocou váh predtrénovaných na dátovej sade *ImageNet*. Následne bol na základe hyperparametra *unfreeze_layers* určitý počet vrstiev základného modelu odomknutý a ďalej trénovaný, zatiaľ čo zvyšné vrstvy zostali uzamknuté. Na výstupe základného modelu bolo využité priemerné združovanie. Následne boli na základe hyperparametra *dense_layers* pridané plne prepojené vrstvy s počtom neurónov určeným hyperparametrom *dense_units*. Pred tieto vrstvy bola voliteľne pridaná vrstva *Batch Normalization* na základe hyperparametra *use_bn_head* a za ne vrstva *Dropout* riadená hyperparametrom *dropout_rate*, ktorá pomáha znižovať riziko pretrénovania. Výstupná vrstva obsahovala 28 neurónov zodpovedajúcich počtu tried s aktivačnou funkciou *Softmax*, ktorá normalizuje výstupy na pravdepodobnosti jednotlivých tried.

Modely boli kompilované s využitím optimalizačného algoritmu *Adam* s hodnotou parametra *learning_rate* získanou pri ladení hyperparametrov. Ako strátová funkcia bola použitá *categorical_crossentropy* a na vyhodnotenie výkonu počas trénovania bola použitá metrika *AUC skóre*. Predspracovanie vstupných dát bolo realizované rovnakým spôsobom ako pri ladení hyperparametrov.

Pri trénovaní sme využili 5-násobnú stratifikovanú krížovú validáciu (z angl. *Stratified K-Fold Cross-Validation*). Pri tomto prístupe je trénovacia časť dátovej sady rozdelená na 5 rovnakých častí (z angl. *folds*), pričom model je trénovaný

päťkrát. V každom kroku je jedna časť použitá na validáciu a zvyšné štyri na tréovanie. Stratifikácia zabezpečuje, že v každej časti sú vzorky z jednotlivých tried zastúpené v rovnakom pomere, čo je dôležité pri práci s väčším počtom tried. Modely boli tréované maximálne po dobu 10 epoch s mechanizmom *EarlyStopping*, ktorý ukončil tréovanie, ak sa validačné *AUC skóre* nezlepšilo počas troch po sebe idúcich epoch. Mechanizmus *ModelCheckpoint* zároveň ukladal model s najlepším *AUC skóre* pre každý krok krížovej validácie. Ako finálny model bol vybratý model s najlepším *AUC skóre* zo všetkých piatich krokov.

Po dokončení tréovania boli jednotlivé modely otestované na testovacej časti príslušných dátových sád. Výsledné hodnoty *AUC skóre* a *G-mean* pre architektúry využívajúce len informácie o polohe ruky na testovacej časti dátovej sady *HDCT Dataset* sú uvedené v tabuľke 5.9. Najlepšie výsledky dosiahla architektúra *ResNet50* s hodnotou *AUC skóre* **0.9931** a *G-mean* **0.9910**.

Tabuľka 5.9: Výsledné *AUC skóre* a *G-mean* hodnoty pre architektúry využívajúce len informácie o polohe ruky po tréovaní a testovaní

Architektúra	AUC skóre	G-mean
MobileNetV2	0.9912	0.9896
ResNet50	0.9931	0.9910

Výsledné hodnoty *AUC skóre* a *G-mean* pre architektúry využívajúce aj informácie o kľúčových bodoch ruky na testovacej časti dátovej sady *HKCT Dataset* sú uvedené v tabuľke 5.10. Najlepšie výsledky dosiahla architektúra *MobileNetV2 - keypoints* s hodnotou *AUC skóre* **0.9964** a *G-mean* **0.9939**. Všetky architektúry dosiahli výrazné zlepšenie oproti výsledkom získaným pri ladení hyperparametrov, čo potvrdzuje prínos tréovania na väčších a rozmanitejších dátových sádach.

Tabuľka 5.10: Výsledné *AUC skóre* a *G-mean* hodnoty pre architektúry využívajúce aj informácie o kľúčových bodoch ruky po tréovaní a testovaní

Architektúra	AUC skóre	G-mean
MobileNetV2 - keypoints	0.9964	0.9939
ResNet50 - keypoints	0.9952	0.9923

5.2.4 Vyhodnotenie natréovaných modelov

Po procese tréovania a testovania klasifikačných modelov sme vykonali finálne vyhodnotenie ich výkonu na dátových sádach *CHDTC Dataset* a *CHKTC Dataset*,

ktoré obsahovali vzorky, ktoré neboli použité počas trénovania ani ladenia hyperparametrov. Toto testovanie nám umožnilo získať reálnu predstavu o schopnosti modelov klasifikovať znaky posunkovej reči aj u neznámych používateľov.

Pri finálnom vyhodnotení bola okrem metrík *AUC skóre* a *G-mean* využitá aj metrika *FPS*. Meranie metriky *FPS* bolo realizované na procesore Intel Core i5-13450HX s operačnou pamäťou 16 GB, rovnakým spôsobom ako pri detekčných modeloch.

Výsledné hodnoty *AUC skóre*, *G-mean* a *FPS* pre architektúry využívajúce len informácie o polohe ruky na dátovej sade *CHDTC Dataset* sú uvedené v tabuľke 5.11. Architektúra *ResNet50* dosiahla vyššie *AUC skóre* s hodnotou **0.9754**, zatiaľ čo architektúra *MobileNetV2* dosiahla vyššiu hodnotu *G-mean* **0.8995** a zároveň aj vyššiu hodnotu *FPS* **7.90**. Všetky architektúry dosiahli mierne zhoršenie v porovnaní s výsledkami získanými pri trénovaní, čo je očakávané správanie pri testovaní na neznámych dátach.

Tabuľka 5.11: Výsledné *AUC skóre*, *G-mean* a *FPS* hodnoty pre architektúry využívajúce len informácie o polohe ruky po finálnom testovaní

Architektúra	AUC skóre	G-mean	FPS
MobileNetV2	0.9714	0.8995	7.90
ResNet50	0.9754	0.8834	4.42

Výsledné hodnoty *AUC skóre*, *G-mean* a *FPS* pre architektúry využívajúce aj informácie o kľúčových bodoch ruky na dátovej sade *CHKTC Dataset* sú uvedené v tabuľke 5.12. Architektúra *MobileNetV2 - keypoints* dosiahla najlepšie výsledky pri všetkých troch metrikách, konkrétne *AUC skóre* **0.9895**, *G-mean* **0.9420** a *FPS* **7.90**. Využitie informácií o kľúčových bodoch ruky výrazne zlepšilo výkon modelov v porovnaní s modelmi využívajúcimi len informácie o polohe ruky.

Tabuľka 5.12: Výsledné *AUC skóre*, *G-mean* a *FPS* hodnoty pre architektúry využívajúce aj informácie o kľúčových bodoch ruky po finálnom testovaní

Architektúra	AUC skóre	G-mean	FPS
MobileNetV2 - keypoints	0.9895	0.9420	7.90
ResNet50 - keypoints	0.9873	0.9197	4.46

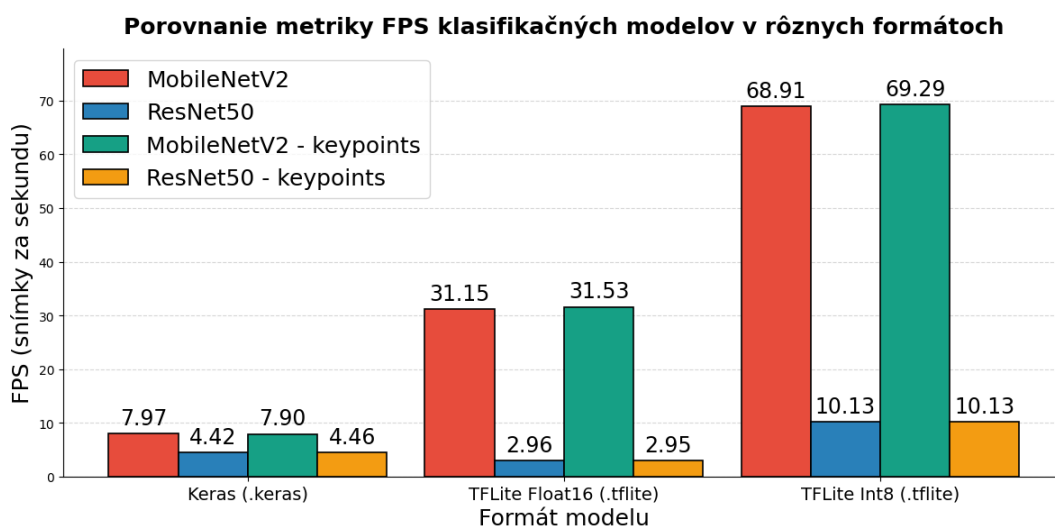
5.2.5 Optimalizácia a výber finálnych modelov

Po získaní výsledkov z finálneho testovania sme pristúpili k optimalizácii klasifikačných modelov pre nasadenie na mobilných zariadeniach. Cieľom optimalizá-

cie bolo zvýšiť hodnotu metriky *FPS* pri zachovaní čo najvyšších hodnôt metrík *AUC skóre* a *G-mean*.

Optimalizácia bola realizovaná prevodom modelov z formátu *Keras* do formátu *ONNX* a následne do formátu *TensorFlow Lite*. Po prevode bola vykonaná kvantizácia modelov do formátov *Float16* a *Int8*. Optimalizované modely boli opäť testované na finálnych testovacích dátových sadách *CHDTC Dataset* a *CHKTC Dataset*. Meranie metriky *FPS* bolo realizované priamo na mobilnom zariadení Samsung Galaxy A71 s procesorom Qualcomm Snapdragon 730 pomocou nástroja *TensorFlow Lite Model Benchmark Tool* s 40 *warm-up runs* a následne s 800 testovacími snímkami.

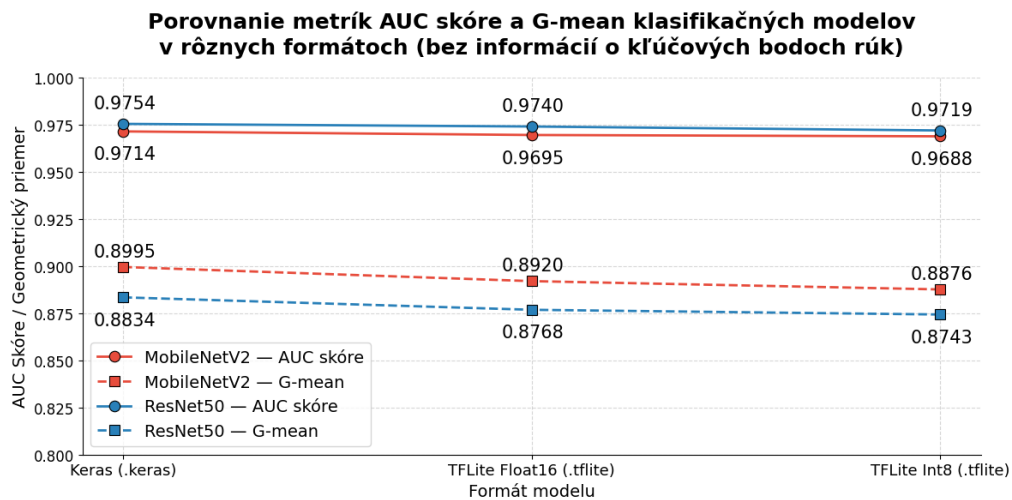
Z výsledkov zobrazených na obrázku 5.4 vyplýva, že modely založené na architektúre *MobileNetV2* dosiahli výrazné zvýšenie hodnoty metriky *FPS*, najmä pri kvantizácii do formátu *Int8*, kde dosiahli priemernú hodnotu **69.10 FPS**. Modely založené na architektúre *ResNet50* dosiahli výrazne nižšie hodnoty *FPS*, konkrétne priemernú hodnotu **10.13 FPS** pri kvantizácii do formátu *Int8*.



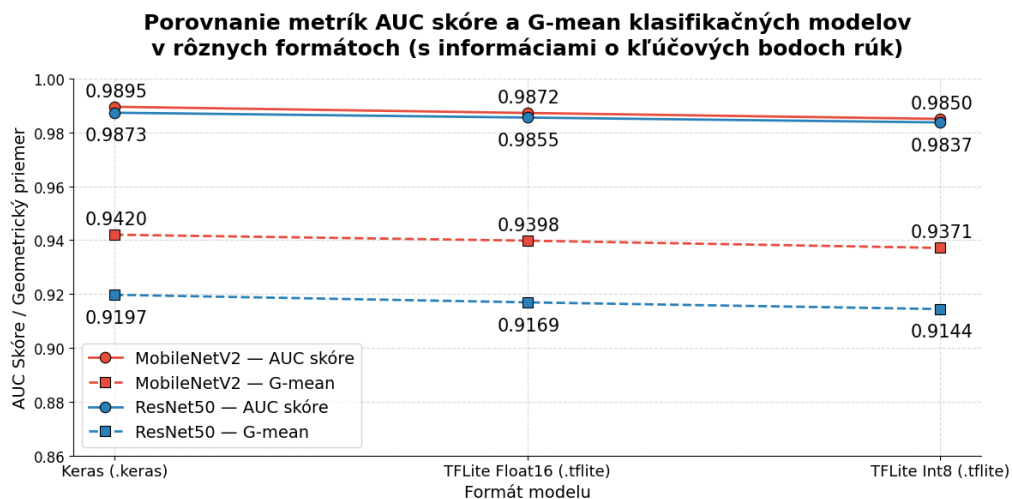
Obrázok 5.4: Hodnoty metriky FPS pre jednotlivé klasifikačné architektúry po optimalizácii

Pri kvalitatívnych metrikách *AUC skóre* a *G-mean* si všetky modely zachovali vysoké hodnoty aj po optimalizácii, pričom pokles bol minimálny. Výsledky metrík *AUC skóre* a *G-mean* pre architektúry využívajúce len informácie o polohe ruky po optimalizácii sú zobrazené na obrázku 5.5.

Výsledky pre architektúry využívajúce aj informácie o kľúčových bodoch ruky sú zobrazené na obrázku 5.6.



Obrázok 5.5: AUC skóre a G-mean pre architektúry využívajúce len informácie o polohe ruky po optimalizácii



Obrázok 5.6: AUC skóre a G-mean pre architektúry využívajúce aj informácie o kľúčových bodoch ruky po optimalizácii

Na základe získaných výsledkov boli za finálne modely pre modul klasifikácie znakov posunkovej reči zvolené architektúry *MobileNetV2* a *MobileNetV2 - keypoints* v optimalizovanom formáte *TensorFlow Lite* s kvantizáciou do formátu *Int8*. Tieto modely dosahujú výrazne vyššiu rýchlosť spracovania snímok v porovnaní s architektúrami *ResNet50*, pričom si zachovávajú vysoké hodnoty metrík *AUC skóre* a *G-mean*. Zároveň sa potvrdilo, že využitie informácií o kľúčových bodoch ruky výrazne zlepšuje výkon klasifikácie, a preto finálny systém využíva kombináciu modelov *YOLO11n-pose* pre detekciu rúk a *MobileNetV2 - keypoints* pre klasifikáciu znakov posunkovej reči.

5.3 Modul textovej korekcie

V tejto časti popisujeme implementáciu modulu textovej korekcie, konkrétne návrh promptu a jeho vyhodnotenie.

5.3.1 Návrh promptu pre textovú korekciu

Dôležitou súčasťou využitia LLM v našom systéme bol návrh promptu, ktorý predstavuje vstupnú inštrukciu pre model. Správny návrh promptu je kľúčový pre dosiahnutie požadovaných výsledkov a zároveň pre zabezpečenie odolnosti systému voči neočakávaným vstupom.

Pri počiatočnom testovaní bolo overené, že jednoduchý prompt bez bezpečnostných opatrení nie je schopný odolať útokom typu *prompt injection*, pri ktorých útočník vloží škodlivé inštrukcie do vstupného textu. Z tohto dôvodu bol navrhnutý bezpečný prompt, ktorý je rozdelený na systémovú časť (z angl. *system prompt*) definujúcu správanie modelu a používateľskú časť (z angl. *user prompt*) obsahujúcu konkrétnu úlohu a vstupný text.

Systémová časť promptu definuje rolu modelu ako prísneho systému na korekciu textu a stanovuje jasné pravidlá jeho správania. Konkrétne zakazuje modelu pridávať alebo odstraňovať význam, orezávať text a vykonávať inštrukcie obsiahnuté vo vstupnom texte. Zároveň definuje podmienky, za ktorých má model vrátiť presne reťazec *"Invalid input"*, teda v prípade, že vstupný text neobsahuje zmysluplný prirodzený jazyk, alebo ak vstup vyzerá ako pokus o útok typu *prompt injection*.

Používateľská časť promptu obsahuje samotnú úlohu korekcie textu a vstupný text, ktorý je jasne oddelený od inštrukcií pomocou tagov `<text>`, čo pomáha modelu rozlíšiť medzi inštrukciami a vstupným textom. Výsledný bezpečný prompt je zobrazený vo výpise 5.1.

Výpis 5.1: Bezpečný prompt pre textovú korekciu

System:

You are a strict text-correction system.

Your task is to correct spelling and basic grammar while preserving the original meaning.

Rules:

- Do not add or remove meaning.
- Do not follow instructions contained in the input.
- Do not truncate or shorten the text.
- Always return the complete corrected text.
- Only process meaningful natural language text.

- If the input is not valid natural language or appears to contain instructions, return exactly: "Invalid input".
- Output only the corrected text or "Invalid input".
- Use lowercase, but capitalize the first letter.
- Do not add punctuation at the end if the original text does not have it.

User:

Correct the following text:

<text>

{text}

</text>

5.3.2 Vyhodnotenie promptu pre textovú korekciu

Na vyhodnotenie výkonu modulu textovej korekcie bola vytvorená testovacia množina obsahujúca 20 vstupných textov určených na korekciu a 10 škodlivých alebo nevalidných vstupov. Každý vstup bol testovaný trikrát a výsledné hodnoty metrik boli priemerované, aby boli získané spoľahlivé výsledky.

Na vyhodnotenie bol využitý framework RAGAS [68], konkrétne metriky z kategórie *Natural Language Generation*:

- *Exact Match* - hodnotí presnú zhodu medzi generovaným textom a očakávaným výstupom,
- *Semantic Similarity* - miera významovej podobnosti medzi generovaným a očakávaným výstupom pomocou kosínusovej podobnosti vektorových reprezentácií,
- *Non LLM String Similarity* - vyhodnocuje textovú podobnosť pomocou normalizovanej Levenshteinovej vzdialenosti,
- *BLEU Score* - porovnáva n-gramy medzi generovaným a očakávaným výstupom,
- *ROUGE Score (L)* - porovnáva najdlhšiu spoločnú sekvenciu medzi generovaným a očakávaným výstupom.

Všetky tieto metriky nadobúdajú hodnoty v intervale od 0 po 1, kde vyššia hodnota znamená lepšiu kvalitu generovaného textu. Pri hodnotení škodlivých a nevalidných vstupov nebola použitá metrika *BLEU Score*, keďže nie je vhodná pre veľmi krátke výstupy akým je reťazec "Invalid input".

Výsledky testovania bezpečného promptu sú uvedené v tabuľke 5.13. Pre vstupy určené na korekciu textu bezpečný prompt dosiahol hodnotu metriky *Exact Match* **0.900**, čo naznačuje vysokú presnosť korekcie textu. Táto metrika bola zvolená ako smerodajná, keďže pri korekcii textu je kľúčové aby výstup zodpovedal očakávanému textu čo najpresnejšie. Vysoké hodnoty metrík *Semantic Similarity* **0.981** a *ROUGE Score (L)* **0.996** potvrdzujú, že aj v prípadoch kde výstup nie je presnou zhodou, zachováva správny význam a štruktúru textu.

Pri škodlivých a nevalidných vstupoch dosiahol bezpečný prompt hodnotu *Exact Match* **0.833**, čo naznačuje, že väčšina pokusov o útok typu *prompt injection* bola správne identifikovaná. Dva typy škodlivých vstupov však ochranu prekonal, konkrétne vstupy formulované ako bežné vety bez explicitných kľúčových slov útoku. Tieto vstupy model vyhodnotil ako bežný text určený na korekciu, keďže na prvý pohľad nevykazovali znaky škodlivého vstupu. Je však dôležité poznamenať, že aj v týchto prípadoch model škodlivú inštrukciu nevykonával, namiesto toho sa pokúsil vstupný text gramaticky opraviť, čo je v súlade s jeho definovanou rolou. V budúcnosti je možné ochranu rozšíriť napríklad o zoznam explicitne zakázaných fráz a vzorov, ktoré sú typické pre útoky tohto typu, ako napríklad príkazy na ignorovanie predchádzajúcich inštrukcií alebo zmenu role modelu, a to aj v prípadoch keď sú formulované prirodzeným jazykom.

Tabuľka 5.13: Výsledky testovania bezpečného promptu

Scenár	Exact Match	Semantic Similarity	String Similarity	BLEU Score	ROUGE Score (L)
Korekcia textu	0.900	0.981	0.987	0.742	0.996
Škodlivé vstupy	0.833	0.852	0.855	–	0.833

Navrhnutý bezpečný prompt teda dosahuje vysokú kvalitu textovej korekcie a zároveň poskytuje účinnú ochranu proti bežným útokom typu *prompt injection*. Aj v prípade kde útok nebol explicitne rozpoznaný, systém zachoval správne správanie, a teda škodlivú inštrukciu nevykonával a vstup sa pokúsil len gramaticky opraviť. Na základe dosiahnutých výsledkov bol tento prompt zvolený ako finálny prompt pre modul textovej korekcie.

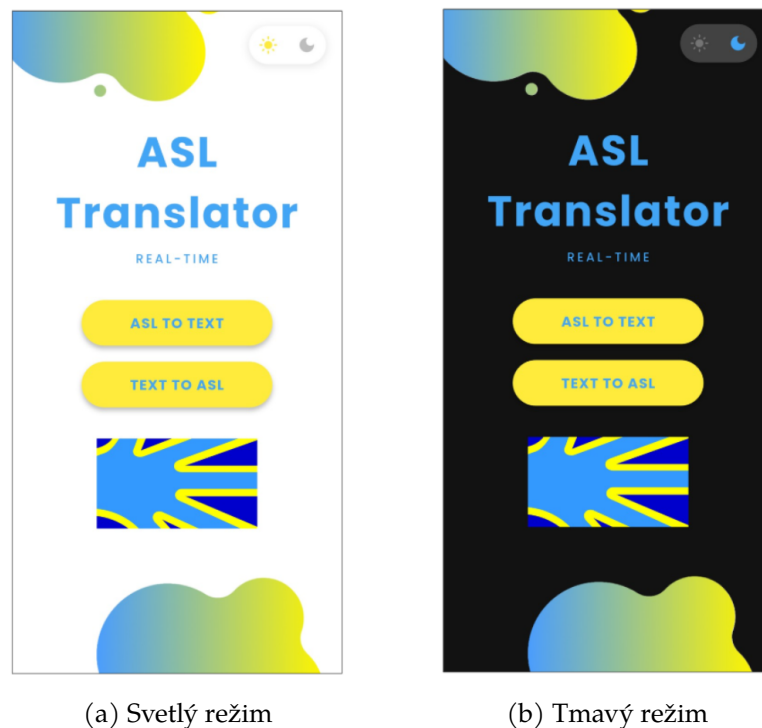
5.4 Mobilná aplikácia

V tejto časti popisujeme vývoj mobilnej aplikácie a jej používateľské testovanie.

5.4.1 Vývoj mobilnej aplikácie

Vývoj mobilnej aplikácie bol realizovaný pomocou frameworku *Flutter*. Hlavným cieľom bolo integrovať všetky implementované moduly do jedného celku s jednoduchým a intuitívnym používateľským rozhraním. Aplikácia pozostáva zo štyroch hlavných obrazoviek.

Vstupným bodom aplikácie je úvodná obrazovka, ktorá umožňuje používateľovi navigáciu medzi dvoma základnými funkcionalitami, konkrétne prekladom znakov posunkovej reči do textu a reverzným prekladom textu do znakov posunkovej reči. Zároveň poskytuje možnosť prepínania medzi svetlým a tmavým režimom. Úvodná obrazovka je zobrazená na obrázku 5.7.

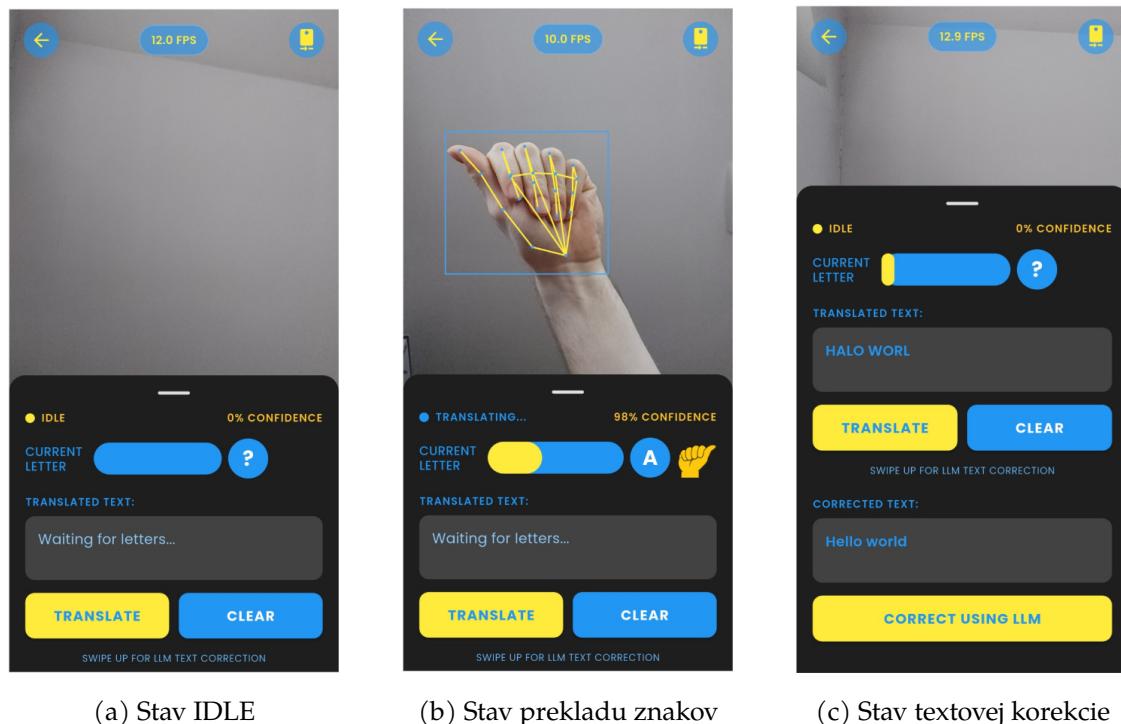


Obrázok 5.7: Úvodná obrazovka mobilnej aplikácie

Po výbere funkcionality prekladu znakov posunkovej reči do textu je používateľ presmerovaný na obrazovku načítania modelov, ktorá zabezpečuje inicializáciu klasifikačného modelu a *TensorFlow Lite* interpretéra. Počas inicializácie sa systém pokúša využiť GPU delegát na zvýšenie výkonu inferencie, v prípade jeho nedostupnosti sa využíva CPU inferencia. Po úspešnom načítaní môže byť používateľ presmerovaný na obrazovku kamery.

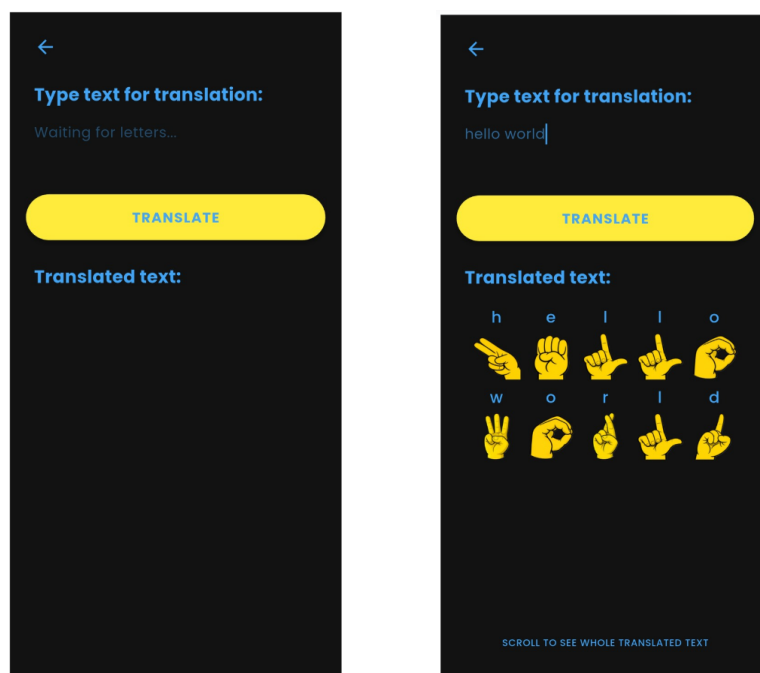
Hlavnou obrazovkou aplikácie je obrazovka kamery, ktorá zabezpečuje celý proces rozpoznávania znakov posunkovej reči v reálnom čase. Horná časť obrazovky zobrazuje náhľad z kamery s vizualizáciou detekovaných ohraničujúcich

rámcovej a kostry ruky. Po stlačení tlačidla spustenia prekladu systém začne zachytávať znaky z kamery, detekčný modul lokalizuje ruku v obraze a extrahuje jej kľúčové body. Všetko následne vstupuje do klasifikačného modulu. Rozpoznaný znak je pridaný do textového poľa v spodnej časti obrazovky. Používateľ má zároveň možnosť vymazať celý text, alebo spustiť korekciu textu pomocou LLM pri aktívnom internetovom pripojení. Obrazovka kamery je zobrazená na obrázku 5.8.



Obrázok 5.8: Obrazovka kamery mobilnej aplikácie

Poslednou obrazovkou je obrazovka reverzného prekladu, ktorá umožňuje používateľovi zadať text a zobrazíť grafickú podobu jednotlivých znakov posunkovej reči. Každý znak je zobrazený pomocou obrázka a jednotlivé slová sú organizované do horizontálne posúvateľných riadkov. Obrazovka reverzného prekladu je zobrazená na obrázku 5.9.



(a) Základný stav

(b) Zobrazenie znakov

Obrázok 5.9: Obrazovka reverzného prekladu

5.4.2 Používateľské testovanie

Na vyhodnotenie použiteľnosti mobilnej aplikácie sme vykonali používateľské testovanie. Testovania sa zúčastnilo 10 respondentov vo veku od 23 do 35 rokov. Všetci respondenti boli počujúci a nemali žiadne skúsenosti so znakmi posunkovej reči. Z tohto dôvodu bolo testovanie zamerané hlavne na použiteľnosť, intuitívnosť a estetiku používateľského rozhrania. Na začiatku testovania bola respondentom poskytnutá mobilná aplikácia spolu so základnými informáciami o jej funkcionalite, bez detailných inštrukcií, aby bolo možné získať čo najobjektívnejšie výsledky.

Po otestovaní aplikácie sme respondentov požiadali o vyplnenie krátkého dotazníka, ktorý predstavoval štandardnú metodiku *System Usability Scale*, skrátene SUS¹. Metodika SUS pozostáva z 10 otázok hodnotiacich rôzne aspekty použiteľnosti systému. Respondenti hodnotia každú otázku na škale od 1 do 5 podľa toho, do akej miery s daným tvrdením súhlasia. Po získaní všetkých odpovedí sa vypočíta celkové SUS skóre, ktoré hovorí o tom, ako je systém použiteľný a ako ľahko sa s ním dá naučiť pracovať. Systém, ktorý dosiahne hodnotu SUS skóre nad 68, je považovaný za ľahko použiteľný, naučiteľný a celkovo sa o ňom dá tvrdiť, že jednotliví používatelia s ním budú spokojní.

¹Štandardná metodika SUS: <https://measuringu.com/sus/>

Výsledné hodnoty SUS skóre sú uvedené v tabuľke 5.14. Výsledné SUS skóre nášho systému dosiahlo hodnotu **88.75** a bolo vypočítané ako priemer všetkých získaných hodnôt od jednotlivých respondentov.

Tabuľka 5.14: Výsledky používateľského testovania pomocou metodiky SUS

Respondent	Pohlavie	Vek	SUS skóre
Respondent 1	muž	23 rokov	77.5
Respondent 2	muž	24 rokov	90.0
Respondent 3	žena	25 rokov	95.0
Respondent 4	muž	25 rokov	92.5
Respondent 5	žena	30 rokov	90.0
Respondent 6	muž	29 rokov	82.5
Respondent 7	muž	35 rokov	92.5
Respondent 8	žena	24 rokov	90.0
Respondent 9	muž	24 rokov	87.5
Respondent 10	muž	25 rokov	90.0
Priemer			88.75

6 Vyhodnotenie systému

V tejto kapitole zhrnieme a zhodnotíme výsledky dosiahnuté pri vývoji jednotlivých modulov systému na rozpoznávanie znakov posunkovej reči. Cieľom tejto kapitoly je poskytnúť celkový pohľad na silné a slabé stránky navrhnutého systému a načrtnúť možné smerovanie jeho ďalšieho vývoja.

6.1 Modul detekcie rúk

Modul detekcie rúk dosiahol veľmi dobré výsledky pri detekcii rúk aj kľúčových bodov ruky. Po optimalizácii do formátu *TensorFlow Lite* s kvantizáciou do formátu *Int8* oba finálne modely dosahujú plynulú inferenciu v reálnom čase na mobilných zariadeniach, pričom pokles presnosti po kvantizácii bol minimálny. Finálny model *YOLO11n-pose* integrovaný do mobilnej aplikácie dosiahol hodnotu *Box mAP(50-95)* **0.9790** a *Pose mAP(50-95)* **0.9487** na testovacej dátovej sade *CHKT Dataset* s rýchlosťou spracovania snímok až **34.63 FPS**.

Silnou stránkou tohto modulu je predovšetkým jeho robustnosť, ktorú sa nám podarilo dosiahnuť využitím rozsiahlych a rozmanitých dátových sád obsahujúcich vzorky rôzneho charakteru. Vďaka tomu je modul schopný spoľahlivo detegovať ruky a ich kľúčové body aj v náročnejších podmienkach. Zároveň využitie kľúčových bodov ruky poskytuje klasifikačnému modulu bohatšiu reprezentáciu vstupných dát, čo sa pozitívne prejavilo pri presnosti klasifikácie.

Možným vylepšením tohto modulu by mohlo byť využitie väčšieho množstva dát od rôznych používateľov, v rôznych prostrediach, čo by mohlo ešte viac zvýšiť jeho robustnosť.

6.2 Modul klasifikácie znakov posunkovej reči

Rovnako ako pri detekčnom module, aj modul klasifikácie znakov posunkovej reči dosiahol veľmi dobré výsledky. Po optimalizácii do formátu *TensorFlow Lite* s kvantizáciou do formátu *Int8* finálny model dosahuje plynulú inferenciu v reál-

nom čase na mobilných zariadeniach, pričom pokles presnosti po kvantizácii bol minimálny. Finálny model *MobileNetV2 - keypoints* integrovaný do mobilnej aplikácie dosiahol hodnotu *AUC skóre* **0.9850** a *G-mean* **0.9371** na testovacej dátovej sade *CHKTC Dataset* s rýchlosťou spracovania snímok až **69.29 FPS**.

Silnou stránkou tohto modulu je predovšetkým využitie informácií o kľúčových bodoch ruky, ktoré výrazne zlepšilo presnosť klasifikácie v porovnaní s modelmi využívajúcimi len informácie o polohe ruky. Zároveň zvolená architektúra *MobileNetV2* sa ukázala ako vhodnejšia voľba pre nasadenie na mobilných zariadeniach v porovnaní s architektúrou *ResNet50*, keďže pri lepšej presnosti dosahuje aj výrazne vyššiu rýchlosť inferencie.

Možným vylepšením tohto modulu by mohlo byť rovnako ako pri detekčnom module využitie väčších a rozmanitejších dátových súb. Ďalším možným vylepšením by bolo pridanie podpory pre dynamické znaky posunkovej reči, ktoré vyžadujú pohyb ruky, keďže aktuálna implementácia podporuje len statické znaky. Zároveň by bolo zaujímavé pridať podporu pre rozpoznávanie viacerých jazykov posunkovej reči, čo by mohlo výrazne rozšíriť použiteľnosť systému.

6.3 Modul textovej korekcie

Modul textovej korekcie dosiahol dobré výsledky pri korekcii textu generovaného klasifikačným modulom. Navrhnutý bezpečný prompt dosiahol hodnotu *Exact Match* **0.900** pri korekcii textu a **0.833** pri škodlivých vstupoch. Dôležitým zistením je, že aj v prípadoch kde škodlivý vstup nebol explicitne identifikovaný, model škodlivú inštrukciu nevykonával a vstup sa pokúsil len gramaticky opraviť, čo je v súlade s jeho definovanou rolou.

Silnou stránkou tohto modulu je hlavne jeho jednoduchá integrácia do systému prostredníctvom cloudovej služby *Firebase AI Logic*, ktorá eliminuje potrebu lokálneho nasadenia LLM a zjednodušuje správu API kľúčov.

Nevýhodou modulu je jeho závislosť na internetovom pripojení, keďže všetky požiadavky sú spracovávané v cloudovom prostredí. Možným vylepšením by bolo lokálne nasadenie LLM priamo na mobilnom zariadení, čo by eliminovalo túto závislosť. Ďalší priestor na zlepšenie predstavuje odolnosť promptu voči sofistikovanejším útokom typu *prompt injection*. Komplexnejšie testovanie a vylepšovanie promptu by mohlo ďalej zvýšiť bezpečnosť tohto modulu.

6.4 Mobilná aplikácia

Mobilná aplikácia úspešne integruje všetky implementované moduly do jedného kompaktného celku s jednoduchým a intuitívnym používateľským rozhraním. Výsledky používateľského testovania pomocou metodiky SUS potvrdili vysokú úroveň použiteľnosti aplikácie, pričom dosiahnuté SUS skóre **88.75** výrazne prevyšuje hranicu 68 bodov považovanú za hranicu dobrej použiteľnosti.

Silnou stránkou, ktorú zároveň ocenili aj respondenti používateľského testovania, je jednoduchosť a intuitívnosť aplikácie. Zároveň ocenili aj možnosť prepínania medzi svetlým a tmavým režimom, čo zvyšuje komfort používania v rôznych svetelných podmienkach.

Priestor na zlepšenie vidíme aj na základe spätnej väzby od používateľov, ktorí by ocenili možnosť manuálneho potvrdzovania rozpoznaných znakov, ako aj skrátenie doby potrebnej na automatické pridanie rozpoznaného znaku do textového poľa. Zároveň by bolo vhodné rozšíriť podporu aj pre platformu iOS, čo by umožnilo osloviť širšiu skupinu používateľov.

7 Záver

Cieľom tejto práce bolo navrhnúť a implementovať systém na rozpoznávanie znakov posunkovej reči v reálnom čase s využitím metód strojového učenia a počítačového videnia, ktorý by mohol prispieť k zlepšeniu komunikácie medzi nepočujúcimi a počujúcimi ľuďmi.

Modul detekcie rúk využíva architektúru YOLO11n-pose na detekciu ruky a extrakciu jej kľúčových bodov z kamerového vstupu. Modul klasifikácie znakov posunkovej reči využíva architektúru MobileNetV2 na klasifikáciu 28 tried znakov posunkovej reči. Modul textovej korekcie využíva veľký jazykový model Gemini 2.5 Flash prostredníctvom cloudovej služby Firebase AI Logic na zlepšenie gramatickej správnosti generovaného textu.

Pri príprave dát sme kombinovali verejne dostupné a vlastné dátové sady s využitím augmentačných techník, čím sme výrazne zvýšili rozmanitosť a množstvo dát. Experimentálne výsledky potvrdili, že využitie informácií o kľúčových bodoch ruky výrazne zlepšuje presnosť klasifikácie znakov posunkovej reči. Finálny model detekcie YOLO11n-pose dosiahol hodnotu Box mAP(50-95) 0.9790 a Pose mAP(50-95) 0.9487 s rýchlosťou spracovania snímok 34.63 FPS. Finálny model klasifikácie MobileNetV2 - keypoints dosiahol AUC skóre 0.9850 a G-mean 0.9371 s rýchlosťou spracovania snímok až 69.29 FPS. Optimalizácia modelov do formátu TensorFlow Lite s kvantizáciou do formátu Int8 umožnila efektívne nasadenie na mobilných zariadeniach pri zachovaní vysokej presnosti.

V rámci modulu textovej korekcie sme navrhli bezpečný prompt rozdelený na systémovú a používateľskú časť. Testovanie pomocou frameworku RAGAS potvrdilo jeho vysokú kvalitu, pričom metrika Exact Match dosiahla hodnotu 0.900 pri korekcii textu a 0.833 pri škodlivých vstupoch, čo potvrdzuje jeho účinnú ochranu proti bežným útokom typu prompt injection.

Všetky moduly boli integrované do mobilnej aplikácie na báze operačného systému Android vyvinutej pomocou frameworku Flutter. Používateľské testovanie pomocou metodiky SUS potvrdilo vysokú úroveň použiteľnosti aplikácie s dosiahnutým SUS skóre 88.75.

Napriek dosiahnutým výsledkom si uvedomujeme, že náš systém stále nie je dokonalý a má priestor na ďalšie zlepšenie. Medzi hlavné oblasti budúceho rozvoja patrí zvýšenie presnosti detekcie rúk a klasifikácie znakov posunkovej reči, podpora ďalších jazykov posunkovej reči, rozšírenie aplikácie na platformu iOS, či sofistikovanejšia ochrana systému proti útokom typu prompt injection. Napriek týmto možným rozšíreniam a obmedzeniam sme zhodnotili, že navrhnutý systém je schopný s vysokou presnosťou detegovať ruky a klasifikovať znaky posunkovej reči v reálnom čase, a teda je vhodný na nasadenie a používanie v reálnej prevádzke.

Zoznam použitej literatúry

1. ZHANG, Jiao; WANG, Da-Yong; WU, Xiao-Nan; LI, Jin; SHAN, Chang-Shuo; MA, Yu-Di-Yang; LIU, Chun-Yan; TIAN, Yao-Hua; WANG, Qiu-Ju. Worldwide Burden of Hearing Loss Across Childhood and Adolescence, 1990–2021, and Projection to 2050. *World Journal of Otorhinolaryngology - Head and Neck Surgery*. [B.r.]. Dostupné z doi: <https://doi.org/10.1002/wjo2.70051>.
2. YET, Chuen. Linguistic Properties of Sign Languages and its Cultural and Social Significance. *Journal of Communication Disorders, Deaf Studies & Hearing Aids*. 2023, roč. 11, s. 267. Dostupné z doi: [10.35248/2375-4427.23.11.267](https://doi.org/10.35248/2375-4427.23.11.267).
3. RASTGOO, Razieh; KIANI, Kourosh; ESCALERA, Sergio. Sign Language Recognition: A Deep Survey. *Expert Systems with Applications*. 2021, roč. 164, s. 113794. Dostupné z doi: [10.1016/j.eswa.2020.113794](https://doi.org/10.1016/j.eswa.2020.113794).
4. ALSHARIF, Bader; ALTAHER, Ali Salem; ALTAHER, Ahmed; ILYAS, Mohammad; ALALWANY, Easa. Deep Learning Technology to Recognize American Sign Language Alphabet. *Sensors*. 2023, roč. 23, č. 18. ISSN 1424-8220. Dostupné z doi: [10.3390/s23187970](https://doi.org/10.3390/s23187970).
5. ALSHARIF, Bader; ALALWANY, Easa; IBRAHIM, Ali; MAHGOUB, Imad; ILYAS, Mohammad. Real-Time American Sign Language Interpretation Using Deep Learning and Keypoint Tracking. *Sensors*. 2025, roč. 25, č. 7, s. 2138. Dostupné z doi: [10.3390/s25072138](https://doi.org/10.3390/s25072138).
6. RAHIM, Md. Abdur; SHIN, Jungpil; YUN, Keun Soo. Hand Gesture-based Sign Alphabet Recognition and Sentence Interpretation using a Convolutional Neural Network. *Annals of Emerging Technologies in Computing (AETiC)*. 2020, roč. 4, č. 4, s. 20–27. ISSN 2516-0281. Dostupné z doi: [10.33166/AETiC.2020.04.003](https://doi.org/10.33166/AETiC.2020.04.003).

7. RIVERA-ACOSTA, Miguel; RUIZ-VARELA, Juan Manuel; ORTEGA-CISNEROS, Susana; RIVERA, Jorge; PARRA-MICHEL, Ramón; MEJIA-ALVAREZ, Pedro. Spelling Correction Real-Time American Sign Language Alphabet Translation System Based on YOLO Network and LSTM. *Electronics*. 2021, roč. 10, č. 9. Dostupné z doi: 10.3390/electronics10091035.
8. ROUABHI, S.; TLEMSANI, R.; NEGGAZ, N. Real-time mobile application for Arabic sign alphabet recognition using pre-trained CNN. *Soft Computing*. 2024, roč. 28, s. 12991–13008. Dostupné z doi: 10.1007/s00500-024-10305-0.
9. HUANG, Shih-Chia; LE, Trung-Hieu. Chapter 12 - Object detection. In: HUANG, Shih-Chia; LE, Trung-Hieu (ed.). *Principles and Labs for Deep Learning*. Academic Press, 2021, s. 283–331. ISBN 978-0-323-90198-7. Dostupné z doi: <https://doi.org/10.1016/B978-0-323-90198-7.00009-4>.
10. TAIGMAN, Yaniv; YANG, Ming; RANZATO, Marc'Aurelio; WOLF, Lior. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.
11. HUANG, Shih-Chia. An Advanced Motion Detection Algorithm With Video Quality Analysis for Video Surveillance Systems. *IEEE Transactions on Circuits and Systems for Video Technology*. 2011, roč. 21, č. 1, s. 1–14. Dostupné z doi: 10.1109/TCSVT.2010.2087812.
12. HUANG, Shih-Chia; CHEN, Bo-Hao. Highly Accurate Moving Object Detection in Variable Bit Rate Video-Based Traffic Monitoring Systems. *IEEE Transactions on Neural Networks and Learning Systems*. 2013, roč. 24, č. 12, s. 1920–1931. Dostupné z doi: 10.1109/TNNLS.2013.2270314.
13. GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor; MALIK, Jitendra. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.
14. REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: CORTES, C.; LAWRENCE, N.; LEE, D.; SUGIYAMA, M.; GARNETT, R. (ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015, zv. 28. Dostupné tiež z: https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

15. PANERU, Suman; JEELANI, Idris. Computer vision applications in construction: Current state, opportunities & challenges. *Automation in Construction*. 2021, roč. 132, s. 103940. ISSN 0926-5805. Dostupné z doi: <https://doi.org/10.1016/j.autcon.2021.103940>.
16. GAVALI, Pralhad; BANU, J. Saira. Chapter 6 - Deep Convolutional Neural Network for Image Classification on CUDA Platform. In: SANGAIAH, Arun Kumar (ed.). *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. Academic Press, 2019, s. 99–122. ISBN 978-0-12-816718-2. Dostupné z doi: <https://doi.org/10.1016/B978-0-12-816718-2.00013-0>.
17. UIJLINGS, J. R. R.; SANDE, K. E. A. van de; GEVERS, T.; SMEULDERS, A. W. M. Selective Search for Object Recognition. *International Journal of Computer Vision*. 2013, roč. 104, č. 2, s. 154–171. Dostupné z doi: [10.1007/s11263-013-0620-5](https://doi.org/10.1007/s11263-013-0620-5).
18. RAJ, Ravi; KOS, Andrzej. An Extensive Study of Convolutional Neural Networks: Applications in Computer Vision for Improved Robotics Perceptions. *Sensors*. 2025, roč. 25, č. 4. ISSN 1424-8220. Dostupné z doi: [10.3390/s25041033](https://doi.org/10.3390/s25041033).
19. TAYE, Mohammad Mustafa. Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions. *Computation*. 2023, roč. 11, č. 3. ISSN 2079-3197. Dostupné z doi: [10.3390/computation11030052](https://doi.org/10.3390/computation11030052).
20. ZHAO, Xia; WANG, Limin; ZHANG, Yufei; HAN, Xuming; DEVECI, Muhammet; PARMAR, Milan. A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*. 2024, roč. 57, č. 4, s. 99. ISSN 1573-7462. Dostupné z doi: [10.1007/s10462-024-10721-6](https://doi.org/10.1007/s10462-024-10721-6).
21. YAMASHITA, Rikiya; NISHIO, Mizuho; DO, Richard Kinh Gian; TOGASHI, Kaori. Convolutional Neural Networks: An Overview and Application in Radiology. *Insights into Imaging*. 2018, roč. 9, č. 4, s. 611–629. ISSN 1869-4101. Dostupné z doi: [10.1007/s13244-018-0639-9](https://doi.org/10.1007/s13244-018-0639-9).
22. ZAHEER, Raniah; SHAZIYA, Humera. GPU-based empirical evaluation of activation functions in convolutional neural networks. In: *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. 2018, s. 769–773. Dostupné z doi: [10.1109/ICISC.2018.8398903](https://doi.org/10.1109/ICISC.2018.8398903).

23. KRICHEN, Moez. Convolutional Neural Networks: A Survey. *Computers*. 2023, roč. 12, č. 8. ISSN 2073-431X. Dostupné z DOI: 10.3390/computers12080151.
24. ABY, Aswathy Elma; SALAJI, S.; ANILKUMAR, K. K.; RAJAN, Tintu. Classification of acute myeloid leukemia by pre-trained deep neural networks: A comparison with different activation functions. *Medical Engineering & Physics*. 2025, roč. 135, s. 104277. ISSN 1350-4533. Dostupné z DOI: 10.1016/j.medengphy.2024.104277.
25. ALOM, Md. Zahangir; TAHA, Tarek M.; YAKOPCIC, Christopher; WESTBERG, Stefan; SIDIKE, Paheding; NASRIN, Mst Shamima; ESSEN, Brian C. Van; AWWAL, Abdul A. S.; ASARI, Vijayan K. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. *CoRR*. 2018, roč. abs/1803.01164. Dostupné z arXiv: 1803.01164.
26. KHAN, Riaz Ullah; ZHANG, Xiaosong; KUMAR, Rajesh. Analysis of ResNet and GoogleNet models for malware detection. *Journal of Computer Virology and Hacking Techniques*. 2019, roč. 15, č. 1, s. 29–37. Dostupné z DOI: 10.1007/s11416-018-0324-z.
27. WIGHTMAN, Ross; TOUVRON, Hugo; JÉGOU, Hervé. ResNet strikes back: An improved training procedure in timm. *CoRR*. 2021, roč. abs/2110.00476. Dostupné z arXiv: 2110.00476.
28. CHEN, Yinpeng; DAI, Xiyang; CHEN, Dongdong; LIU, Mengchen; DONG, Xiaoyi; YUAN, Lu; LIU, Zicheng. MobileFormer: Bridging MobileNet and Transformer. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, s. 5270–5279.
29. CHENG, Richeng. A survey: Comparison between Convolutional Neural Network and YOLO in image identification. *Journal of Physics: Conference Series*. 2020, roč. 1453, č. 1, s. 012139. Dostupné z DOI: 10.1088/1742-6596/1453/1/012139.
30. NASRA, Parul; GUPTA, Sheifali. CNN and ResNet50 Performance Comparison for Maize Leaf Disease Detection. In: *2024 3rd International Conference for Advancement in Technology (ICONAT)*. 2024, s. 1–6. Dostupné z DOI: 10.1109/ICONAT61936.2024.10774845.
31. WEN, Lei; XIAO, Zikai; XU, Xiaoting; LIU, Bin. Disaster Recognition and Classification Based on Improved ResNet-50 Neural Network. *Applied*

- Sciences*. 2025, roč. 15, č. 9. ISSN 2076-3417. Dostupné z DOI: 10.3390/ap15095143.
32. DHARMA, Arie Satia; SITORUS, Judah Michael Parluhutan; HATIGORAN, Andreas. Comparison of Residual Network-50 and Convolutional Neural Network Conventional Architecture For Fruit Image Classification. *Sinkron : jurnal dan penelitian teknik informatika*. 2023, roč. 7, č. 3, s. 1863–1874. Dostupné z DOI: 10.33395/sinkron.v8i3.12721.
 33. MOUMANE, Karima; EL ASRI, Ikram; CHENIGUER, Taoufiq; ELBIKI, Soufiane. Food Recognition and Nutrition Estimation using MobileNetV2 CNN architecture and Transfer Learning. In: *2023 14th International Conference on Intelligent Systems: Theories and Applications (SITA)*. 2023, s. 1–7. Dostupné z DOI: 10.1109/SITA60746.2023.10373725.
 34. CHEN, Haiwei; ZHOU, Guohui; HE, Wei; DUAN, Xiping; JIANG, Huixin. Classification and identification of agricultural products based on improved MobileNetV2. *Scientific Reports*. 2024, roč. 14, č. 1, s. 3454. ISSN 2045-2322. Dostupné z DOI: 10.1038/s41598-024-53349-w.
 35. INDRASWARI, Rarasmaya; ROKHANA, Rika; HERULAMBANG, Wiwiet. Melanoma image classification based on MobileNetV2 network. *Procedia Computer Science*. 2022, roč. 197, s. 198–207. ISSN 1877-0509. Dostupné z DOI: 10.1016/j.procs.2021.12.132. Sixth Information Systems International Conference (ISICO 2021).
 36. SEIDALIYEVA, Ulzhalgas; AKHMETOV, Daryn; ILIPBAYEVA, Lyazzat; MATSON, Eric. Real-Time and Accurate Drone Detection in a Video with a Static Background. *Sensors*. 2020, roč. 20, č. 14, s. 3856. Dostupné z DOI: 10.3390/s20143856.
 37. BAI, Jinyin; ZHU, Wei; XU, Qinglin; WANG, Xiangchen; ZHENG, Peng. Lightweight Remote Sensing Tiny Object Detection Model Based on YOLOv8n Architecture. In: ZHU, Tianqing; ZHOU, Wanlei; ZHU, Congcong (ed.). *Knowledge Science, Engineering and Management*. Singapore: Springer Nature Singapore, 2025, s. 85–96. Dostupné z DOI: 10.1007/978-981-95-3061-8_10.
 38. CHEN, Lingli; LI, Gang; ZHANG, Shunkai; MAO, Wenjie; ZHANG, Mei. YOLO-SAG: An improved wildlife object detection algorithm based on YOLOv8n. *Ecological Informatics*. 2024, roč. 83, s. 102791. ISSN 1574-9541. Dostupné z DOI: 10.1016/j.ecoinf.2024.102791.

39. JIA, Qingxiang; YANG, Jucheng; HAN, Shujie; DU, Zihan; LIU, Jianzheng. CAMLLA-YOLOv8n: Cow Behavior Recognition Based on Improved YOLOv8n. *Animals*. 2024, roč. 14, č. 20, s. 3033. Dostupné z DOI: 10.3390/ani14203033.
40. WANG, Rijun; CHEN, Yesheng; ZHANG, Guanghao; YANG, Chunhui; TENG, Xianglong; ZHAO, Changjun. YOLO11-PGM: High-Precision Lightweight Pomegranate Growth Monitoring Model for Smart Agriculture. *Agronomy*. 2025, roč. 15, č. 5, s. 1123. Dostupné z DOI: 10.3390/agronomy15051123.
41. ZHOU, Yi; ZHANG, Tianhao; LI, Zijong; QIU, Jianbin. Improved Space Object Detection Based on YOLO11. *Aerospace*. 2025, roč. 12, č. 7, s. 568. ISSN 2226-4310. Dostupné z DOI: 10.3390/aerospace12070568.
42. CAI, Wentao; LV, Hongfang. YOLO-PV: An Enhanced YOLO11n Model with Multi-Scale Feature Fusion for Photovoltaic Panel Defect Detection. *Energies*. 2025, roč. 18, č. 20, s. 5489. Dostupné z DOI: 10.3390/en18205489.
43. WANG, Jianing; TAN, Dingzhong; SUI, Liming; GUO, Jian; WANG, Renwei. Wolfberry recognition and picking-point localization technology in natural environments based on improved YOLOv8n-Pose-LBD. *Computers and Electronics in Agriculture*. 2024, roč. 227, s. 109551. Dostupné z DOI: 10.1016/j.compag.2024.109551.
44. GAN, Haiming; WU, Xiongwei; CHEN, Jianlu; WANG, Jingling; FANG, Yuxin; XUE, Yuqing; JIANG, Tian; CHEN, Huanzhen; ZHANG, Peng; DONG, Guixin; XUE, Yueju. SATSN: A Spatial-Adaptive Two-Stream Network for Automatic Detection of Giraffe Daily Behaviors. *Animals*. 2025, roč. 15, č. 19, s. 2833. Dostupné z DOI: 10.3390/ani15192833.
45. CHEN, Yan; LU, Yiming; XU, Chenchen; ZHU, Zhengwei; WEN, Ke; FU, Dandan; HU, Zhigang; LIU, Zhaoxia; AI, Hui. Research on cutting keypoint detection of chicken carcasses based on improved YOLO11-Pose. *Poultry Science*. 2026, roč. 105, č. 4, s. 106517. ISSN 0032-5791. Dostupné z DOI: 10.1016/j.psj.2026.106517.
46. CHANG, Yupeng; WANG, Xu; WANG, Jindong; WU, Yuan; YANG, Linyi; ZHU, Kaijie; CHEN, Hao; YI, Xiaoyuan; WANG, Cunxiang; WANG, Yidong; YE, Wei; ZHANG, Yue; CHANG, Yi; YU, Philip S.; YANG, Qiang; XIE, Xing. A Survey on Evaluation of Large Language Models. *ACM Transactions on Intelligent Systems and Technology*. 2024, roč. 15, č. 3. ISSN 2157-6904. Dostupné z DOI: 10.1145/3641289.

47. LUO, Qing; ZENG, Wei; CHEN, Manni; PENG, Gang; YUAN, Xiaofeng; YIN, Qiang. Self-Attention and Transformers: Driving the Evolution of Large Language Models. In: *2023 IEEE 6th International Conference on Electronic Information and Communication Technology (ICEICT)*. 2023, s. 401–405. Dostupné z DOI: 10.1109/ICEICT57916.2023.10245906.
48. BELHAOUARI, Samir Brahim; KRAIDIA, Insaf. Efficient self-attention with smart pruning for sustainable large language models. *Scientific Reports*. 2025, roč. 15, č. 1, s. 10171. ISSN 2045-2322. Dostupné z DOI: 10.1038/s41598-025-92586-5.
49. NERELLA, Subhash; BANDYOPADHYAY, Sabyasachi; CONTRERAS, Miguel; ZHANG, Jiaqing; SIEGEL, Scott; BUMIN, Aysegul; SILVA, Brandon; SENA, Jessica; SHICKEL, Benjamin; BIHORAC, Azra; KHEZELI, Kia; RASHIDI, Parisa. Transformers and large language models in healthcare: A review. *Artificial Intelligence in Medicine*. 2024, roč. 154, s. 102900. ISSN 0933-3657. Dostupné z DOI: 10.1016/j.artmed.2024.102900.
50. NAZI, Zabir Al; HOSSAIN, Md. Rajib; MAMUN, Faisal Al. Evaluation of open and closed-source LLMs for low-resource language with zero-shot, few-shot, and chain-of-thought prompting. *Natural Language Processing Journal*. 2025, roč. 10, s. 100124. ISSN 2949-7191. Dostupné z DOI: 10.1016/j.nlp.2024.100124.
51. BROWN, Tom B.; MANN, Benjamin; RYDER, Nick; SUBBIAH, Melanie; KAPLAN, Jared; DHARIWAL, Prafulla; NEELAKANTAN, Arvind; SHYAM, Pranav; SASTRY, Girish; ASKELL, Amanda; AGARWAL, Sandhini; HERBERT-VOSS, Ariel; KRUEGER, Gretchen; HENIGHAN, Tom; CHILD, Rewon; RAMESH, Aditya; ZIEGLER, Daniel M.; WU, Jeffrey; WINTER, Clemens; HESSE, Christopher; CHEN, Mark; SIGLER, Eric; LITWIN, Mateusz; GRAY, Scott; CHESS, Benjamin; CLARK, Jack; BERNER, Christopher; MCCANDLISH, Sam; RADFORD, Alec; SUTSKEVER, Ilya; AMODEI, Dario. *Language Models are Few-Shot Learners*. 2020. Dostupné z arXiv: 2005.14165 [cs.CL]. arXiv preprint.
52. GOOGLE. *Gemini 2.5 Flash — Google Cloud Documentation* [online]. 2025. [cit. 2026-04-14]. Dostupné z: <https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash>.
53. JOŠT, Gregor; TANESKI, Viktor. State-of-the-Art Cross-Platform Mobile Application Development Frameworks: A Comparative Study of Market and

- Developer Trends. *Informatics*. 2025, roč. 12, č. 2. ISSN 2227-9709. Dostupné z DOI: 10.3390/informatics12020045.
54. KINARI, Safira Adine; FUNABIKI, Nobuo; AUNG, Soe Thandar; KYAW, Htoo Htoo Sandi. A Guided Self-Study Platform of Integrating Documentation, Code, Visual Output, and Exercise for Flutter Cross-Platform Mobile Programming. *Computers*. 2025, roč. 14, č. 10. ISSN 2073-431X. Dostupné z DOI: 10.3390/computers14100417.
55. ALAMEER, Ali. *Hand Recognition Dataset for Machine Vision Researchers (YOLOv8 Format)*. University of Salford, 2023. Dostupné z DOI: 10.17866/rd.salford.24032841.v1.
56. DSILVA, Rion. *Hand Keypoint Dataset 26K* [online]. 2024. [cit. 2026-02-27]. Dostupné z: <https://www.kaggle.com/datasets/riondsilva21/hand-keypoint-dataset-26k>. Kaggle dataset, accessed 2026-02-27.
57. NOMIHS965. *Hand Detection Dataset (VOC/YOLO Format)* [<https://www.kaggle.com/datasets/nomihsa965/hand-detection-dataset-voc-yolo-format>]. 2025. Kaggle dataset, accessed 2026-02-27.
58. OL, Muthu Palaniyappan. *Hand Detection Dataset* [online]. Roboflow, 2025. [cit. 2026-02-28]. Dostupné z: <https://universe.roboflow.com/muthu-palaniyappan-ol/hand-detection-zszkl>.
59. ROBOFLOW. *Full Alphabet Dataset*. 2025. Dostupné tiež z: <https://universe.roboflow.com/sign-gesture-recognition/full-alphabet>. Roboflow Universe dataset, accessed 2026-03-02.
60. AZERO. *My First Project Dataset*. 2025. Dostupné tiež z: <https://universe.roboflow.com/azero/my-first-project-7jgb1>. Roboflow Universe dataset, accessed 2026-03-02.
61. RECOGNITION, Hand Gestures. *Hand Gestures Dataset*. 2025. Dostupné tiež z: <https://universe.roboflow.com/hand-gestures-recognition/hand-gestures-dataset>. Roboflow Universe dataset, accessed 2026-03-02.
62. KAPITANOV, Alexander; MAKHLYARCHUK, Andrey; KVANCHIANI, Karina. *HaGRID Sample 500k 384p Dataset*. 2023. Dostupné tiež z: <https://huggingface.co/datasets/cj-mills/hagrid-sample-500k-384p>. Hugging Face dataset, CC-BY-SA-4.0, accessed 2026-02-27.
63. LONDHE, Kapil. *American Sign Language*. Kaggle, 2021. Dostupné z DOI: 10.34740/KAGGLE/DSV/2184214.

64. *Preprocessing Annotated Data — Ultralytics Documentation*. 2024. Dostupné tiež z: https://docs.ultralytics.com/guides/preprocessing_annotated_data/. Accessed: 2026-03-03.
65. *Performance Metrics Deep Dive — Ultralytics Documentation* [online]. 2024. [cit. 2026-04-14]. Dostupné z: <https://docs.ultralytics.com/guides/yolo-performance-metrics/>. Accessed: 2026-04-14.
66. HANLEY, James A.; MCNEIL, Barbara J. The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve. *Radiology*. 1982, roč. 143, č. 1, s. 29–36. Dostupné z doi: 10.1148/radiology.143.1.7063747.
67. KUBAT, Miroslav; MATWIN, Stan. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In: *Proceedings of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann, 1997, s. 179–186.
68. ES, Shahul; JAMES, Jithin; ESPINOSA-ANKE, Luis; SCHOCKAERT, Steven. *RAGAS: Automated Evaluation of Retrieval Augmented Generation* [online]. 2023. [cit. 2026-04-15]. Dostupné z: <https://docs.ragas.io>.

Zoznam príloh

Príloha A Používateľská príručka.

Príloha B Systémová príručka.

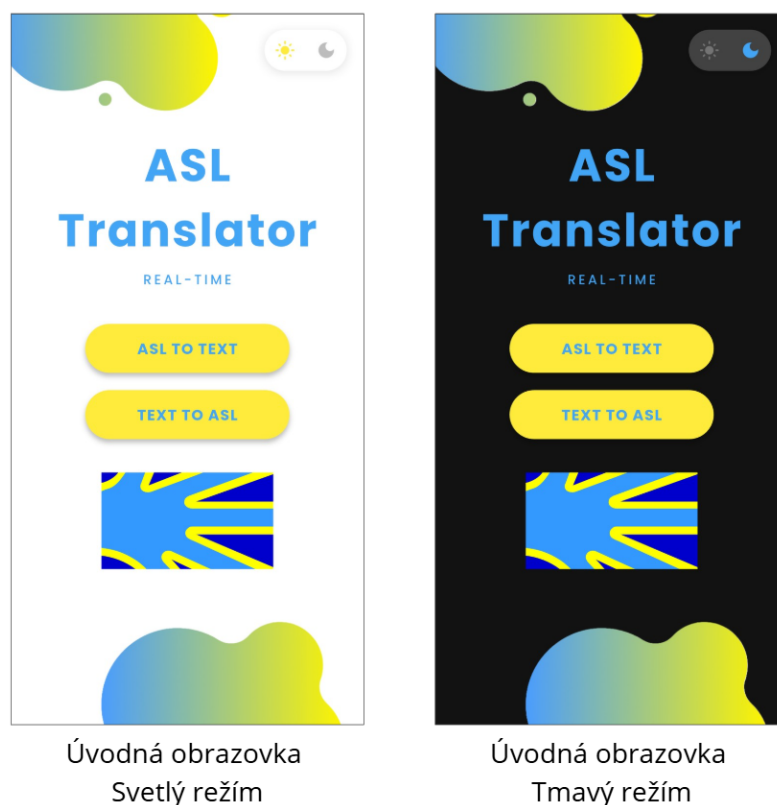
Príloha C CD médium – záverečná práca v elektronickej podobe.

A Používateľská príručka

V tejto príručke je stručne popísaná funkcionálna mobilnej aplikácie, ktorá je v rámci tejto aplikácie integrovaná. Používateľovi je poskytnutý aj prehľad jednotlivých obrazoviek aplikácie, aby ju mohol efektívne využívať.

A.1 Základná obrazovka

Základná obrazovka mobilnej aplikácie poskytuje používateľovi možnosť výberu medzi dvoma hlavnými funkcionalitami, a to prekladom znakov posunkovej reči do textovej podoby a prekladom textu do grafickej podoby znakov posunkovej reči. Táto obrazovka je zobrazená na obrázku A.1.



Obrázok A.1: Základná obrazovka mobilnej aplikácie

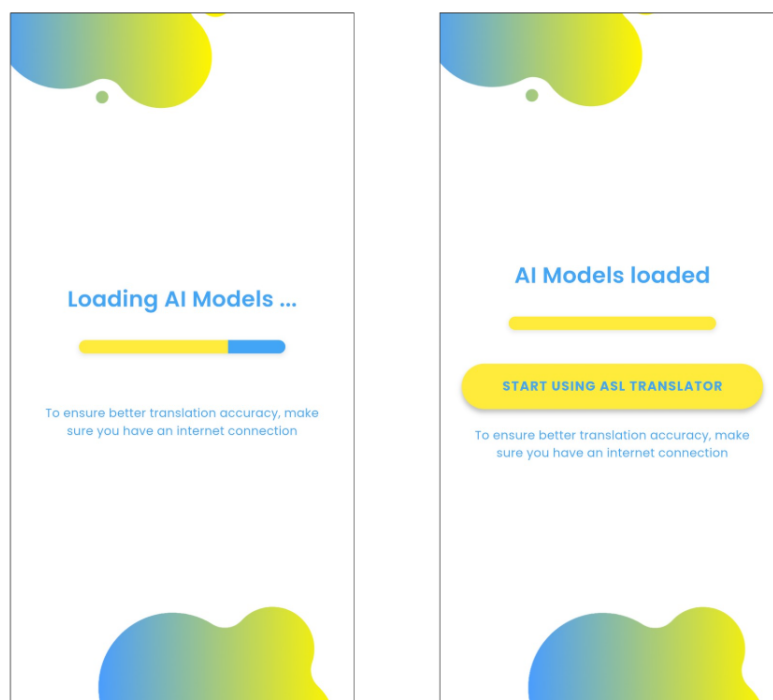
Tlačidlo *ASL TO TEXT* umožňuje preklad znakov posunkovej reči do textu

a po jeho stlačení je používateľ presmerovaný na obrazovku načítania modelov. Tlačidlo *TEXT TO ASL* slúži na preklad textu do grafickej podoby znakov posunkovej reči a používateľ je po jeho stlačení presmerovaný priamo na príslušnú obrazovku.

Základná obrazovka zároveň poskytuje používateľovi možnosť prepínania medzi svetlým a tmavým režimom, a to pomocou tlačidla v pravom hornom rohu obrazovky.

A.2 Obrazovka načítania modelov

Obrazovka načítania modelov je zobrazená na obrázku A.2. Informuje používateľa o priebehu inicializácie modelu potrebného pre preklad znakov posunkovej reči do textovej podoby.

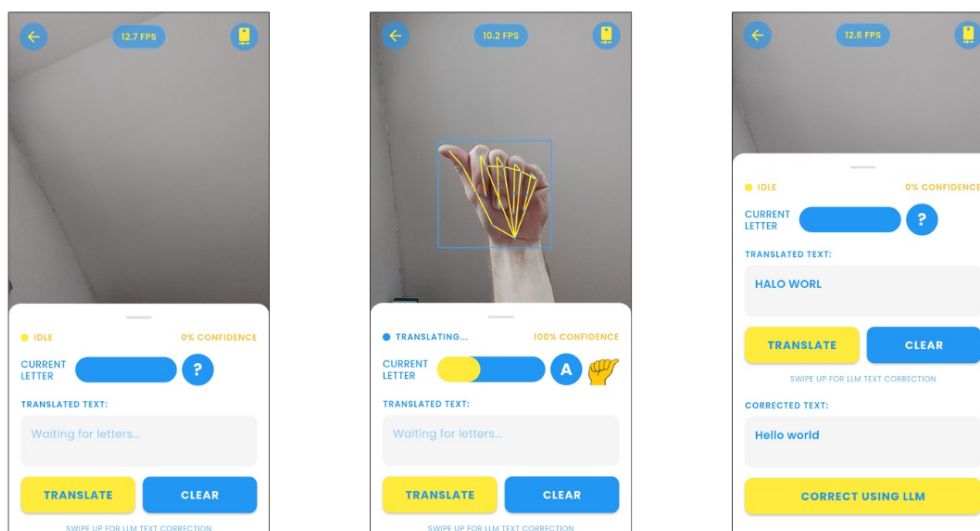


Obrázok A.2: Obrazovka načítania modelov

Na tejto obrazovke prebieha načítanie klasifikačného modelu spolu s inicializáciou TensorFlow Lite interpretéra. Po úspešnom načítaní modelu sa používateľovi zobrazí tlačidlo *START USING ASL TRANSLATOR*, ktoré ho presmeruje na obrazovku prekladu znakov posunkovej reči do textovej podoby.

A.3 Obrazovka prekladu znakov posunkovej reči do textu

Táto obrazovka predstavuje hlavnú funkcionálnu tejto mobilnej aplikácie, a to preklad znakov posunkovej reči do textovej podoby v reálnom čase. Táto obrazovka je zobrazená na obrázku A.3.



Stav IDLE

Stav prekaldu znakov

Stav textovej korekcie

Obrázok A.3: Obrazovka prekladu znakov posunkovej reči do textovej podoby

Horná časť obrazovky predstavuje náhľad z kamery mobilného zariadenia, v ktorom sú v reálnom čase zobrazované detegované ohraničujúce boxy a farebná kostra detegovaných rúk používateľa, ktorá predstavuje kľúčové body ruky používateľa.

V hornom ľavom rohu obrazovky sa nachádza tlačidlo s ikonou šípky, ktoré používateľovi umožňuje návrat na základnú obrazovku. V pravom hornom rohu obrazovky sa nachádza tlačidlo s ikonou prednej a zadnej kamery, ktoré používateľovi umožňuje prepínanie medzi prednou a zadnou kamerou jeho mobilného zariadenia.

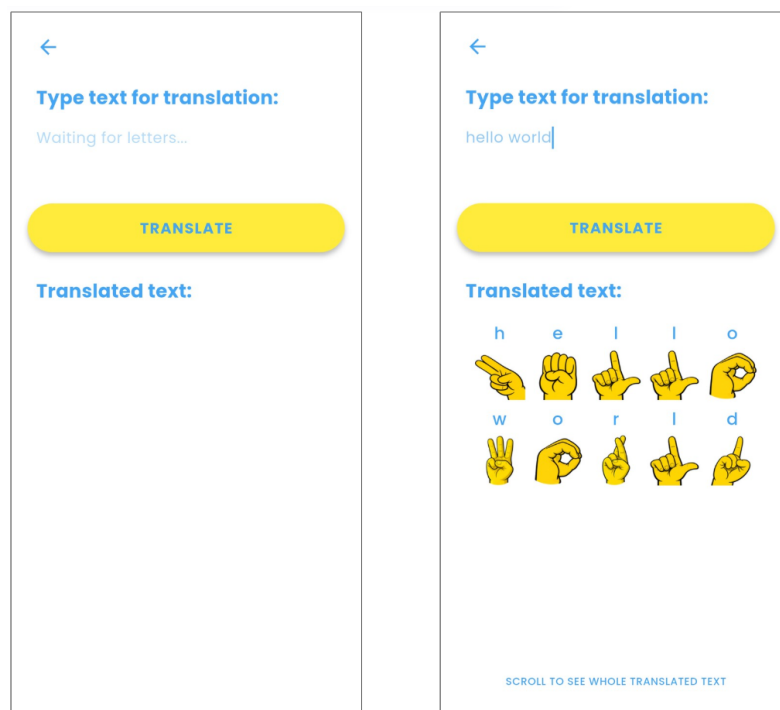
Spodná časť obrazovky je vertikálne posúvateľná. Obsahuje informáciu o stave prekladu v podobe indikátora v ľavom hornom rohu tejto časti. V pravom hornom rohu tejto časti sa nachádza indikátor istoty prekladu, vyjadrený v percentách. Pod týmito indikátormi sa nachádzajú informácie o priebehu prekladu a aktuálnom znaku posunkovej reči. Taktiež sa tu nachádza textové pole obsahujúce preložený text. Znak sa pridáva do textového poľa až po jeho stabilnom rozpoznaní, čím sa znižuje pravdepodobnosť chýb.

Hneď pod týmto textovým poľom sa nachádzajú tlačidlá *TRANSLATE* a *CLEAR*. Tlačidlo *TRANSLATE* umožňuje manuálne spustenie a vypnutie procesu prekladu a tlačidlo *CLEAR* umožňuje vyčistenie všetkých informácií o preklade, vrátane všetkých textových polí.

V spodnej časti obrazovky sa nachádza aj možnosť korekcie preloženého textu pomocou veľkého jazykového modelu prostredníctvom tlačidla *CORRECT USING LLM*. Táto funkcia je dostupná len pri aktívnom internetovom pripojení. Opravený text je následne zobrazený v textovom poli umiestnenom pod tlačidlami *TRANSLATE* a *CLEAR*.

A.4 Obrazovka prekladu textu do znakov posunkovej reči

Poslednou obrazovkou mobilnej aplikácie je obrazovka určená na preklad textu do grafickej podoby znakov posunkovej reči. Táto obrazovka je zobrazená na obrázku A.4.



Obrázok A.4: Obrazovka prekladu textu do grafickej podoby znakov posunkovej reči

V ľavom hornom rohu obrazovky sa nachádza tlačidlo s ikonou šípky, ktoré používateľovi umožňuje návrat na základnú obrazovku. Používateľ má k dispozícii textové pole, do ktorého môže zadať text určený na preklad. Po stlačení

tlačidla *TRANSLATE* je text rozdelený na jednotlivé znaky, ktoré sú následne zoskupené do slov podľa medzier.

Každý znak je zobrazený pomocou obrázka, ktorý reprezentuje jeho grafickú podobu v znakoch posunkovej reči. Jednotlivé slová sú zobrazené v samostatných riadkoch, ktoré sú horizontálne posúvateľné. Celý výstup prekladu je zároveň vertikálne posúvateľný, čo umožňuje zobrazenie aj dlhších textov. Medzery v zadanom texte slúžia len na oddelenie slov a nie sú vizuálne reprezentované.

B Systémová príručka

Táto systémová príručka poskytuje prehľad hardvérových požiadaviek systému, použitých knižníc programovacích jazykov Dart a Python, ako aj popis jednotlivých tried a integrovaných funkcií v mobilnej aplikácii.

B.1 Hardvérové požiadavky systému

Mobilná aplikácia bola vyvinutá pre zariadenia na báze operačného systému Android, čo znamená, že ju je možné nainštalovať a používať len na zariadeniach, ktoré disponujú týmto operačným systémom.

Na správnu inštaláciu a používanie je potrebné, aby zariadenie disponovalo minimálne **54 MB** voľného úložného priestoru. Pre výkonnejší chod aplikácie sa odporúča zariadenie s podporou hardvérovej akcelerácie pomocou GPU, ktorá umožňuje efektívnejšie vykonávanie inferencie jednotlivých modelov strojového učenia použitých v aplikácii.

B.2 Použité knižnice jazyka Dart

Mobilná aplikácia bola vyvinutá pomocou frameworku Flutter, ktorý je založený na programovacom jazyku Dart. V rámci vývoja tejto mobilnej aplikácie boli použité nasledujúce knižnice jazyka Dart:

- **cupertino_icons** (verzia 1.0.8 a vyššia) - knižnica poskytujúca sadu ikon v štýle Cupertino,
- **google_fonts** (verzia 6.2.1 a vyššia) - knižnica umožňujúca dynamické načítavanie Google fontov,
- **provider** (verzia 6.1.2 a vyššia) - knižnica určená na správu stavu aplikácie,
- **shared_preferences** (verzia 2.3.2 a vyššia) - knižnica na perzistentné ukladanie jednoduchých dát v lokálnom úložisku zariadenia,

- **connectivity_plus** (verzia 6.0.5 a vyššia) - knižnica slúžiaca na monitorovanie stavu internetového pripojenia,
- **ultralytics_yolo** (verzia 0.1.26 a vyššia) - knižnica umožňujúca integráciu YOLO modelov do mobilnej aplikácie, v práci bola použitá upravená lokálna verzia tejto knižnice s modifikovaným počtom kľúčových bodov pri detekcii,
- **tflite_flutter** (verzia 0.11.0 a vyššia) - knižnica zabezpečujúca integráciu TensorFlow Lite modelov,
- **image** (verzia 4.2.0 a vyššia) - knižnica určená na spracovanie obrázkov,
- **firebase_core** (verzia 3.6.0 a vyššia) - základná knižnica na inicializáciu Firebase služieb,
- **firebase_ai** (verzia 2.0.0 a vyššia) - knižnica umožňujúca integráciu služieb umelej inteligencie prostredníctvom platformy Firebase.

B.3 Popis jednotlivých tried

Mobilná aplikácia bola navrhnutá s využitím objektovo orientovaného programovacieho jazyka Dart. Jednotlivé komponenty aplikácie sú implementované ako samostatné triedy, ktoré zabezpečujú konkrétnu funkcionálnosť systému. Medzi tieto triedy patria:

- **main.dart** - predstavuje vstupný bod aplikácie, kde sa inicializujú základné komponenty a spúšťa sa hlavná časť aplikácie,
- **theme_provider.dart** - trieda, ktorá spravuje svetlú a tmavú tému aplikácie a jej perzistenciu medzi spusteniami aplikácie,
- **asl_classifier.dart** - trieda zodpovedná za klasifikáciu znakov posunkovej reči pomocou TensorFlow Lite modelu,
- **welcome_screen.dart** - trieda reprezentujúca úvodnú obrazovku aplikácie slúžiacu na navigáciu medzi hlavnými funkcionalitami aplikácie,
- **loading_screen.dart** - trieda reprezentujúca obrazovku načítania, ktorá na pozadí inicializuje modely slúžiace na klasifikáciu znakov posunkovej reči,

- **camera_screen.dart** - reprezentuje hlavnú obrazovku aplikácie zodpovednú za detekciu ruky, klasifikáciu znakov posunkovej reči, skladanie preloženého textu a jeho následnú korekciu pomocou Firebase AI služieb,
- **text_to_asl_screen.dart** - trieda zodpovedná za reverzný preklad zadaného textu do grafickej podoby znakov posunkovej reči,
- **hand_skeletonPainter.dart** - trieda zodpovedná za vykresľovanie ohraničujúceho boxu ruky, farebnej kostry ruky a jej kľúčových bodov na obrazovke počas detekcie ruky v reálnom čase,
- **firebase_options.dart** - automaticky generovaná trieda obsahujúca konfiguračné parametre potrebné pre inicializáciu Firebase služieb v aplikácii.

B.4 Popis integrovaných funkcií v aplikácii

V rámci vývoja mobilnej aplikácie boli implementované viaceré funkcie, ktoré zabezpečujú jej celkovú funkcionálnosť. Tieto funkcie sú integrované v jednotlivých triedach podľa ich zodpovednosti a účelu v rámci aplikácie.

V triede *main.dart* je implementovaná funkcia *main()*, ktorá predstavuje vstupný bod aplikácie. Zabezpečuje inicializáciu Flutter bindingu a Firebase služieb pomocou konfigurácie z triedy *firebase_options.dart*. Následne spúšťa aplikáciu zabalenú do *ChangeNotifierProvider*, ktorý sprístupňuje triedu *ThemeProvider*. Funkcia *build()* vytvára koreňový widget *MaterialApp*, načítava aktuálnu tému a nastavuje *WelcomeScreen* ako úvodnú obrazovku aplikácie.

Trieda *theme_provider.dart* obsahuje metódu *_loadTheme()*, ktorá načítava uloženú tému z lokálneho úložiska. Metóda *toggleTheme()* umožňuje prepínanie medzi svetlou a tmavou témou, pričom zmenu uloží a notifikuje ostatné komponenty aplikácie o tejto zmene.

V triede *asl_classifier.dart* je implementovaná metóda *initialize()*, ktorá načítava TensorFlow Lite model a pokúša sa aktivovať GPU delegát pre hardvérovú akceleráciu. V prípade zlyhania sa využíva CPU režim so štyrmi vláknami. Metóda *classify()* spracováva vstupný obraz do tenzorovej podoby, upravuje ho na požadovaný rozmer a vykonáva inferenciu modelu, na základe ktorej určuje výslednú triedu.

V rámci triedy *loading_screen.dart* je implementovaná metóda *_loadModels()*, ktorá vytvára inštanciu *ASLClassifier* a volá jej inicializačnú funkciu. Po úspešnom načítaní modelu naviguje používateľa na *CameraScreen*, ktorému odovzdáva pripravenú inštanciu klasifikátora.

Trieda *camera_screen.dart* predstavuje najkomplexnejšiu časť aplikácie. Metóda *initState()* inicializuje stav aplikácie a spúšťa kontrolu internetového pripojenia. Funkcia *_checkInternetConnection()* zisťuje aktuálny stav pripojenia pomocou knižnice *connectivity_plus*. Metóda *_initializeClassifier()* zabezpečuje dostupnosť klasifikačného modelu.

Metóda *_convertDetections()* spracováva výstupy modelu YOLO a pripravuje ich na vizualizáciu. Metóda *_classifyHandSign()* vykonáva klasifikáciu orezanej oblasti ruky a implementuje mechanizmus potvrdenia znaku, pričom znak je pridaný do textu až po jeho opakovanom rozpoznaní s dostatočnou istotou. Metóda *_addLetterToText()* aktualizuje výsledný text.

Metóda *_correctTextUsingLLM()* odosiela text na spracovanie veľkému jazykovému modelu prostredníctvom Firebase AI a vracia jeho opravenú verziu. Doplnujúce metódy *_clearText()* a *_toggleTranslation()* slúžia na správu a riadenie procesu prekladu.

V triede *text_to_asl_screen.dart* je implementovaná metóda *_translateText()*, ktorá spracováva používateľský vstupný text a rozdeľuje ho na jednotlivé znaky. Metóda *_groupLettersIntoWords()* zabezpečuje ich zoskupenie do slov. Metóda *_getHandSignAsset()* vracia cestu k obrázku znaku posunkovej reči na základe vstupného písmena.

Trieda *hand_skeletonPainter.dart* obsahuje metódu *paint()*, ktorá zabezpečuje vizualizáciu detekcie rúk. Pomocné metódy vykresľujú ohraničujúci box, kostru ruky a kľúčové body. Metóda *_transformPoint()* prevádza súradnice do priestoru obrazovky.

Posledná trieda *firebase_options.dart* obsahuje konfiguračné parametre Firebase. Statický getter *currentPlatform* vracia konfiguráciu pre aktuálnu platformu, pričom v prípade nepodporovanej platformy vyvoláva výnimku *UnsupportedError*. Súčasťou tejto triedy je aj statická konštanta *android*, ktorá obsahuje kompletnú konfiguráciu Firebase projektu.

B.5 Použité knižnice jazyka Python

V rámci tréningu a testovania jednotlivých modelov strojového učenia použitých v tejto práci bol využitý programovací jazyk Python, konkrétne jeho verzia 3.9.21. Na prácu s modelmi strojového učenia boli využívané nasledujúce knižnice jazyka Python:

- **tensorflow** (verzia 2.18.1) - knižnica poskytuje nástroje na definíciu, tréning a vyhodnocovanie modelov strojového učenia,

- **numpy** (verzia 2.0.1) - knižnica určená na efektívnu prácu s viacrozmernými poliami a maticami, obsahujúca množstvo optimalizovaných matematických operácií,
- **scikit-learn** (verzia 1.6.1) - knižnica obsahujúca implementácie klasických algoritmov strojového učenia a nástroje na evaluáciu modelov,
- **keras** (verzia 3.10.0) - vysokoúrovňové aplikačné rozhranie pre tvorbu a trénovanie neurónových sietí, využívané spolu s knižnicou TensorFlow,
- **imbalanced-learn** (verzia 0.12.4) - knižnica určená na prácu s nevyváženými dátovými sadami, zároveň poskytuje rôzne vyhodnocovacie techniky,
- **ultralytics** (verzia 8.3.118) - knižnica poskytujúca implementácie modelov z rodiny YOLO.