VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

MOBILNÍ APLIKACE PRO NASKENOVÁNÍ HRY SUDOKU Z NOVIN A JEJÍ DOHRÁNÍ

MOBILE APPLICATION FOR SCANNING SUDOKU FROM NEWSPAPERS AND FINISHING IT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JURAJ LAZORÍK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ DYK**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Lazorík Juraj**
Program: Informační technologie
Název: **Mobilní aplikace pro naskenování hry Sudoku z novin a její dohrání**
Mobile Application for Scanning Sudoku from Newspapers and Finishing It
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy zpracování obrazu. Zaměřte se zejména na problematiku detekce a rozpoznání tištěných i ručně psaných číslic.
2. Prostudujte problematiku návrhu a tvorby mobilních aplikací pro android.
3. Navrhněte mobilní aplikaci, která nasnímá vytištěnou křížovku Sudoku a navrhne kroky pro její vyřešení.
4. Implementujte minimalistickou použitelnou verzi řešené aplikace.
5. Otestujte aplikaci na testovacím vzorku uživatelů.
6. Analyzujte zpětnou vazbu uživatelů a realizujte iterativní úpravy řešené aplikace.
7. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání,
- rozpracovaný bod 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Dyk Tomáš, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Táto bakalárska práca popisuje tvorbu mobilnej aplikácie, ktorá skenuje zadanie sudoku z novín a umožňuje jeho riešenie. Aplikácia je určená pre operačný systém Android a je implementovaná v jazyku Kotlin. Skenovanie funguje na princípe odfotenia zadania fotoaparátom. Na detekciu hracieho poľa zadania sú využité základné metódy spracovania obrazu ako prahovanie alebo Cannyho hranový detektor. Rozpoznávanie čísel zabezpečuje konvolučná neurónová sieť s presnosťou 99,08 %. Aplikácia umožňuje pri riešení hry nápovedu. Na hľadanie riešenia sú využité eliminačné metódy a backtracking. Každé naskenované zadanie je uložené do databázy a užívateľ sa tak vždy môže vrátiť ku každej hre. Finálna verzia aplikácie je zverejnená v obchode Google Play.

Abstract

This bachelor thesis describes the creation of a mobile application, that scans sudoku layout from the newspaper and allows its solving. The application is developed for the Android and is implemented in Kotlin. Scanning is done by taking picture of layout with camera. Basic image processing methods such as thresholding or Canny's edge detector are used to detect the sudoku layout. Number recognition is provided by convolution neural network with 99.08 % accuracy. The application allows solving sudoku with hints. Elimination methods and backtracking are used to find solution. Each scanned sudoku is stored in the database so the user can always return to any game. The final version of the application is published in the Google Play store.

Kľúčové slová

sudoku, mobilná aplikácia, spracovanie obrazu, neurónové siete, android, camerax, opencv, tensorflow, kotlin, backtracking, GUI

Keywords

sudoku, mobile application, image processing, neural network, android, camerax, opencv, tensorflow, kotlin, backtracking, GUI

Citácia

LAZORÍK, Juraj. *Mobilní aplikace pro naskenování hry Sudoku z novin a její dohrání.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Dyk

Mobilní aplikace pro naskenování hry Sudoku z novin a její dohrání

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Tomáša Dyka. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Juraj Lazorík
6. mája 2021

Poďakovanie

V prvom rade by som sa rád poďakoval môjmu vedúcemu bakalárskej práce Ing. Tomášovi Dykovi, za jeho odborné rady, priateľský prístup a čas, ktorý mi venoval na konzultáciách. Ďalej by som sa chcel poďakovať mojej partnerke, ktorá ma vždy podporovala a poskytla pomoc pri oprave gramatickej stránky práce. Chcem sa poďakovať aj mojim rodičom a rodine za veľkú podporu pri štúdiu. Taktiež ďakujem všetkým respondentom za čas, ktorý strávili pri vyplňovaní dotazníka a testovaní aplikácie. Nakoniec by som chcel poďakovať Laďovi z discordu za jeho cenné rady, ktoré vždy ochotne poskytol.

Obsah

Kapitola 1

Úvod

Mobilné aplikácie sa stávajú čoraz viac súčasťou našich dennodenných životov. O to viac sa stretávame s aplikáciami, ktoré využívajú spracovanie obrazu či už na zábavu napríklad v podobe zábavných filtrov alebo na serióznejšie a praktickejšie využitie, ako rozpoznávanie textu či iných údajov z obrazu. Jednou z hlavných výhod mobilných aplikácií je možnosť využitia širokého množstva vstavaných periférií na mobilnom zariadení. Tieto zariadenia majú taktiež malú a kompaktnú veľkosť, vďaka čomu je možné mať zariadenie so sebou kdekoľvek. Vzhľadom na túto možnosť mať zariadenie stále so sebou majú, podľa môjho názoru, mobilné aplikácie veľký potenciál.

Cieľom tejto práce je vytvoriť aplikáciu, ktorá umožní užívateľovi naskenovať všetky jeho rozpracované, či nové zadania sudoku a následne ich aj dokončiť. Pri riešení týchto zadaní aplikácia poskytne možnosť nápovedy. Vďaka tomu bude môcť užívateľ vyriešiť každé sudoku, s ktorým si nebude vedieť sám poradiť. Aplikácia taktiež uchová tieto zadania a užívateľ sa k nim tak bude môcť kedykoľvek vrátiť.

Práca je rozdelená do siedmych kapitol. Prvé štyri kapitoly sa zaoberajú teoretickými poznatkami potrebnými k implementácii aplikácie. Ďalšie dve kapitoly popisujú návrh aplikácie a jej následnú implementáciu. Posledná kapitola je zameraná na testovanie aplikácie. Kapitola 2 popisuje základy hry sudoku a možnosti jeho riešenia. Kapitola 3 sa zaoberá metódami na spracovanie obrazu. Kapitola 4 je zameraná na prácu s neurónovými sieťami. Popisuje taktiež datasety potrebné na prácu s nimi. V kapitole 5 sú informácie o operačnom systéme Android, ako aj informácie o využitých technológiách pri vývoji mobilnej aplikácie. V kapitole 6 sú rozobraté existujúce riešenia aplikácií, ktoré pracujú so sudoku. Následne popisuje návrh jednotlivých častí aplikácie. Kapitola 7 obsahuje informácie o implementácii jednotlivých častí, ktoré aplikácia využíva. Posledná kapitola 8 je zameraná na testovanie aplikácie z pohľadu užívateľa, z pohľadu presnosti a z pohľadu rýchlosti.

Kapitola 2

Sudoku

Sudoku je logická hra pre jedného hráča. Cieľom je doplniť čísla 1 až 9 do hracieho poľa tak, aby sa v každom stĺpci, riadku a štvorci (3x3) nenachádzali rovnaké čísla. Bežné hracie pole sa skladá z deviatich štvorcov o rozmere 3x3 (9 políčok). Vzniká tak pole o veľkosti 81 políčok s rozmermi 9x9. Na obrázku 2.1 je možné vidieť zadanie bežného sudoku a jeho riešenie.

	2	7				5		8
8	9			5				
5			8		2			9
9		5	2		8			
		4					9	7
				4	9		1	
	5	9	6	2	4	7		1
			3	8	5	9		4
	4	8		7	1	3	5	

4	2	7	1	9	6	5	3	8
8	9	3	4	5	7	1	6	2
5	1	6	8	3	2	4	7	9
9	7	5	2	1	8	6	4	3
1	8	4	5	6	3	2	9	7
6	3	2	7	4	9	8	1	5
3	5	9	6	2	4	7	8	1
7	6	1	3	8	5	9	2	4
2	4	8	9	7	1	3	5	6

Obr. 2.1: Ukážka zadania sudoku (vľavo) a jeho riešenia (vpravo). Šedou a bielou farbou sú rozlíšené jednotlivé štvorce s rozmerom 3x3

Okrem základnej verzie hry s hracím poľom o veľkosti 9x9 existujú aj iné verzie. Príkladom je verzia s hracím poľom o veľkosti 4x4 alebo 16x16. Je možné taktiež vytvoriť sudoku o veľkosti 6x6, ktoré sa už ale neskladá zo štvorcov, no obdĺžnikov o veľkosti 2x3. Správne zadané sudoku by malo mať práve jedno riešenie. V opačnom prípade je zadanie nesprávne.

Obtiažnosť sudoku nezávisí len na počte prázdnych políčok, ale hlavne na vzájomných väzbách medzi týmito políčkami. Preto aj zdanlivo ťažké zadanie s veľkým počtom nevyplnených políčok je možné niekedy vyriešiť rýchlejšie, ako to, kde je zadaných viacero čísel. Existujú rôzne metódy, ktoré môžu byť použité na riešenie sudoku. Podľa [4] vieme tieto metódy usporiadať na základe ich zložitosti od najjednoduchších po najzložitejšie a to takto:

- Naked Single,
- Hidden Single,

- Locked Candidates,
- Naked a Hidden Pair, Triplet, Quad, . . .,
- X-Wing,
- hádanie.

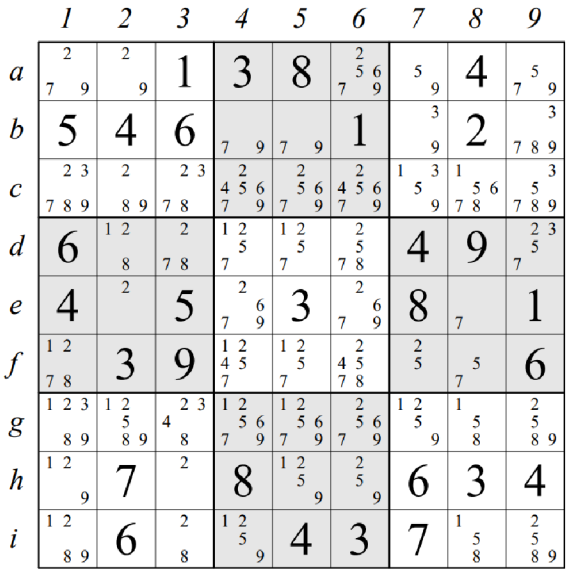
Existujú aj iné eliminačné metódy, no tieto sú tie najzákladnejšie a budú viac popísané v sekcii 2.2.

2.1 Spätné vyhľadávanie

Metóda spätného vyhľadávania alebo backtracking je algoritmus založený na metóde prehľadávania do hĺbky. Jeho základom je postupné dopĺňanie jednotlivých čísel tak, aby boli splnené všetky stanovené podmienky. V prípade, že doplnené číslo porušuje dané podmienky, je dané číslo zmazané a doplní sa iné. Ak už nie je možné doplniť iné číslo, je nutné zmazať aj číslo z predchádzajúceho kroku a pozmeniť ho. Vďaka tomuto vieme vylúčiť veľké množstvo nesprávnych riešení. Tento algoritmus je možné použiť v mnohých situáciách. Je úplný, čo znamená, že vždy nájde riešenie, no nie je optimálny, a teda riešenie nemusí byť to najlepšie. Pri správne zadanom sudoku počítame len s jedným možným riešením, a teda optimálnosť pre nás nie je dôležitý faktor. Dôležitá je časová zložitosť, ktorá je bohužiaľ veľká a nevýhodná. Naopak výhodou je relatívne jednoduchá implementácia.

2.2 Eliminačné metódy

Eliminačné metódy sa na rozdiel od backtrackingu snažia eliminovať možný počet kandidátov (čísel) pre jednotlivé políčka. K tomu je potrebné vytvoriť množinu možných kandidátov pre každé políčko (viď obrázok 2.2) a následne túto množinu redukovať. Vďaka tomuto vieme oveľa rýchlejšie určiť správne číslo, ktoré je potrebné doplniť. Problémom však je väčšia komplexita týchto metód, z čoho vyplýva náročnejšia implementácia. Eliminačné metódy tiež nemusia pokryť všetky riešenia, čo znamená, že nie sú úplné. Niektoré vybrané metódy budú v nasledujúcich sekciách vysvetlené podrobnejšie. Informácie o týchto metódach boli čerpané z [7].



Obr. 2.2: Ukážka sudoku s doplnenou množinou všetkých možných kandidátov[1]

2.2.1 Naked single

Naked Single znamená, že vo vybranom políčku ostal jediný možný kandidát. Množina možných kandidátov teda obsahuje len jedno prípustné číslo. Toto číslo musí patriť na dané pole. Túto situáciu je možné pozorovať napríklad v políčku *e2* na obrázku 2.2.

2.2.2 Hidden single

Hidden single znamená, že v celom riadku, stĺpci alebo štvorci sa dané číslo nachádza len v jednej množine možných kandidátov. Samotná množina ale môže obsahovať aj iné čísla a nemusí byť jednoprvková. Z tohto dôvodu sa táto metóda nazýva hidden, čiže skrytá. Dané číslo je schované medzi ostatnými kandidátmi. Na obrázku 2.2 je možné hidden single pozorovať v políčku *g2*, kde sa možný kandidát 5 nenachádza v žiadnej inej množine možných kandidátov daného štvorca, a preto ho môžeme doplniť.

2.2.3 Locked candidates

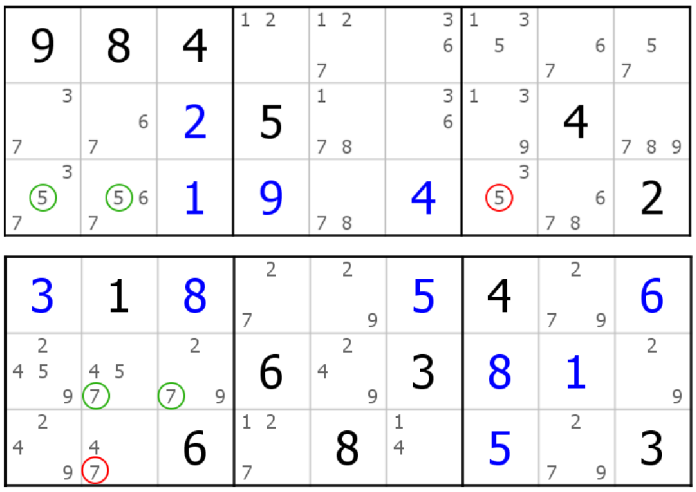
Existujú dva typy locked candidates:

- Pointing,
- Claiming.

Pointing sa využíva v prípade, keď sa v štvorci nachádza viac rovnakých možných kandidátov v rovnakom riadku alebo stĺpci. V takejto situácii sa daný kandidát nesmie nachádzať mimo štvorca v rámci riadku alebo stĺpca. Obrázok 2.3 zobrazuje túto situáciu a kandidáta číslo 5 v červenom krúžku je možné odstrániť. Vzniká tak jediný kandidát v danom políčku a to číslo 3.

[1]Obrázok bol prevzatý z [4]

Claiming funguje opačne ako pointing. Keď sa v riadku alebo stĺpci nachádza viacero rovnakých možných kandidátov a zároveň sa všetci títo kandidáti nachádzajú v rovnakom štvorci, potom je možné odstrániť daného kandidáta v ostatných políčkach štvorca. To opäť zobrazuje obrázok 2.3, kde sa v druhom riadku nachádza číslo 7 ako kandidát len v rámci prvého štvorca. To znamená, že kandidáta číslo 7 v červenom krúžku je možné odstrániť.



Obr. 2.3: Ukážka situácie locked candidate pointing (hore) a locked candidate claiming (dole)[2]

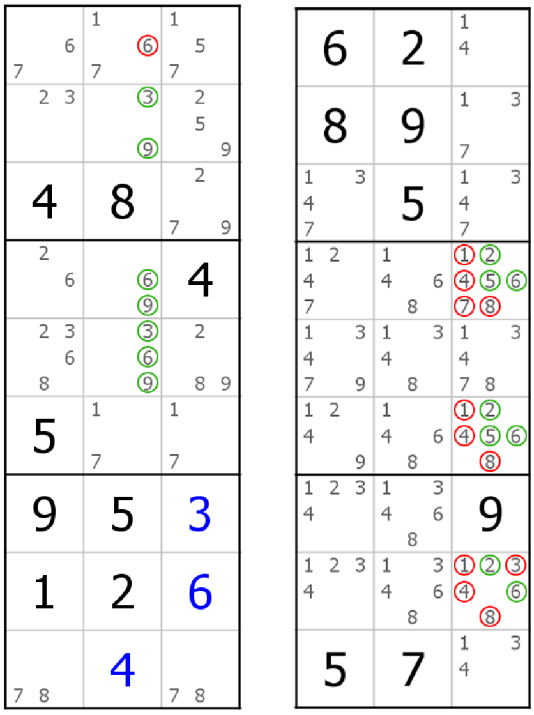
2.2.4 Naked pair, triplet, quad, . . .

Naked pair nastáva vtedy, ak dokážeme nájsť dve políčka, ktoré sa nachádzajú v jednom riadku, stĺpci alebo štvorci, a zároveň majú len dvoch rovnakých kandidátov. Vtedy je možné odstrániť týchto kandidátov z ostatných políčok v nájdenom riadku, stĺpci alebo štvorci. Trojice, štvorice a väčšie skupiny fungujú na rovnakom princípe len s jediným rozdielom, a to takým, že musíme nájsť tri políčka s troma rovnakými kandidátmi, štyri políčka so štyrmi rovnakými kandidátmi a podobne. Je dôležité poznamenať, že pri troch a viacerých políčkach nemusia všetky políčka obsahovať všetkých troch alebo viac kandidátov. Počet kandidátov však nesmie presiahnuť počet políčok. Takúto trojicu je možné pozorovať na obrázku 2.4. Kandidáti 3, 6 a 9 v zelených krúžkoch sa určite budú nachádzať na jednom z vybraných políčok (zatiaľ nie je jasné v akom poradí), čo znamená, že kandidáta 6 v červenom krúžku je možné odstrániť.

2.2.5 Hidden pair, triplet, quad, . . .

Táto metóda funguje na rovnakom princípe ako metóda naked pair, triplet, quad, ktorá bola popísaná vyššie. Hlavným rozdielom je, že v tomto prípade sa v daných políčkach môžu nachádzať aj iní kandidáti okrem tých spoločných. Tých je následne možné eliminovať, v čom je podstata metódy. Obrázok 2.4 zobrazuje príklad trojice 2, 5 a 6. Ostatných (červených) kandidátov v poliach s touto trojicou môžeme odstrániť.

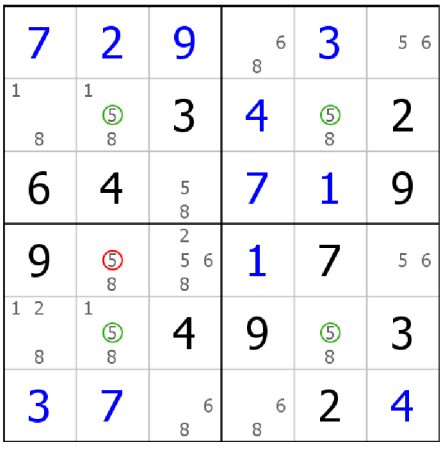
[2]Obrázok bol prevzatý a upravený z [7]



Obr. 2.4: Ukážka situácie naked triplet (vľavo) a hidden triplet (vpravo)[3]

2.2.6 X-Wing

Základom metódy X-Wing je nájdenie takého kandidáta, ktorý sa nachádza v štyroch rôznych štvorcoch. Takýto kandidát sa musí ďalej nachádzať na dvoch rovnakých riadkoch a stĺpcoch. Vytvára teda abstraktný obdĺžnik. Ak sa tento kandidát nachádza aj na iných políčkach ako v rohoch abstraktného obdĺžnika, tak je možné ho z týchto políčok odstrániť (viď obrázok 2.5). Táto metóda využíva fakt, že v jednom štvorci sa smie každé číslo nachádzať iba raz.



Obr. 2.5: Metóda X-wing, ktorá vytvára abstraktný štvorec[4]

[3]Obrázok bol prevzatý a upravený z [7]

2.2.7 Hádanie

Hádanie je metóda, pomocou ktorej je možné vyriešiť takmer každé sudoku. Ako už z názvu vyplýva, čísla do políčok dosadzujeme z množiny kandidátov náhodne. V prípade, že bude daný tip neúspešný, vrátime sa a výber kandidáta pozmeníme. Táto metóda by mala byť využívaná ako posledná možnosť pri riešení sudoku, keďže väčšina dobrých zadaní sudoku by nemala potrebovať využitie tejto metódy. Na základe takéhoto hádania je založený aj algoritmus backtrackingu. Problémom je, že pre človeka môže byť obtiažne sa rekurzívne vracať a meniť kandidátov, s čím narastá aj časová náročnosť.

[4]Obrázok bol prevzatý a upravený z [7]

Kapitola 3

Spracovanie obrazu

Spracovanie obrazu je akákoľvek forma spracovania signálu, kde vstupom je nejaký obrázok, ako napríklad fotografia či video a kde výstupom je buď obrázok alebo istá sada charakteristík či parametrov, ktoré majú nejaký vzťah so vstupným obrazom. Spracovanie obrazu zahŕňa modifikáciu existujúceho obrázka požadovaným spôsobom a pomáha z neho získať informáciu v čitateľnom formáte. Techniky, ktoré spracovávajú obraz, s ním väčšinou pracujú ako s dvojrozmerným poľom [22]. Táto kapitola popisuje niektoré základné techniky takéhoto spracovávania obrazu.

3.1 Adaptívne prahovanie

Vzhľadom na jednoduchosť, intuitívnosť a výpočetnú rýchlosť je prahovanie jedna z hlavných metód, ktorá sa využíva pri segmentácii obrazu. Na základe intenzity pixelu určuje výslednú farbu pixelu. Tá môže byť buď biela alebo čierna. Pre každý pixel obrazu platí, že:

$$G(x,y) = \begin{cases} 1 & \text{pre} \quad I(x,y) \geq T \\ 0 & \text{pre} \quad I(x,y) < T \end{cases} \tag{3.1}$$

kde x a y sú súradnice pixelu, *I(x, y)* je intenzita pixelu, T je hodnota prahu, *G(x, y)* je výsledná hodnota pixelu, *1* označuje čiernu farbu a *0* označuje bielu farbu.

Pri jednoduchom prahovaní je hodnota prahu stanovená ako pevná hodnota a nemení sa. To však nemusí byť vždy dostačujúce najmä pri nerovnomernej intenzite obrazu. V takejto situácii je vhodnejšie využiť adaptívne prahovanie, kde sa hodnota prahu mení. Metódy na výpočet týchto nových prahov fungujú zväčša na základe získavaní informácií o okolí jednotlivých pixelov. Existujú jednoduché metódy ako napríklad výpočet na základe strednej hodnoty alebo mediánu [26]. Zložitejšie metódy využívajú napríklad analýzu histogramu alebo určovanie váh jednotlivým pixelom s pomocou Gaussovej funkcie [15].

3.2 Erózia a dilatácia

Erózia a dilatácia (viď obrázok 3.1) sú základné morfologické operácie, ktoré sú zvyčajne využívané na binárnych obrázkoch. Ich hlavné využitie slúži na odstránenie šumu z obrazu, zjednodušenie tvarov alebo stenčenie či zhrubnutie objektov.

Základnou myšlienkou erózie $\ominus$ je narúšanie alebo odstraňovanie okrajov objektu. Erózia kombinuje 2 množiny pomocou vektorového rozdielu ich bodov. Majme vstupný obraz X a

štrukturálny element B. Eróziu $X \ominus B$ je možné vyjadriť nasledovne:

$$X \ominus B = \left\{ p \in \mathcal{E}^2 : p + b \in X \text{ pre každé } b \in B \right\} \tag{3.2}$$

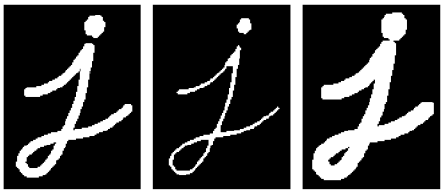
čo znamená, že každý bod p z X je testovaný a výsledok je daný takými bodmi p, ktorých vektorový súčet $p + b$ patrí do X [19].

Inými slovami erózia funguje na princípe posuvného okna prechádzajúceho cez obrázok. Pixel z pôvodného obrázku (čierny alebo biely) bude považovaný za biely na základe toho, či sa v okne nachádzajú všetky pixely bielej farby. Ak nie, pixel eroduje, a teda zostáva čierny. Efekt je možné upravovať na základe veľkosti okna [16].

Dilatácia $\oplus$ funguje opačne ako erózia a snaží sa o rozšírenie objektov. Dilatácia kombinuje dve množiny pomocou vektorového súčtu ich bodov. To je možné zapísať ako [19]:

$$X \oplus B = \left\{ p \in \mathcal{E}^2 : p = x + b, x \in X \text{ a } b \in B \right\} \tag{3.3}$$

V prípade, že sa v okne nachádza aspoň jeden pixel bielej farby, výsledný pixel bude taktiež biely. Takto sa zväčší plocha objektu v obrázku. To je vhodné najmä po použití erózie, ktorá síce zredukuje šum v obraze, ale taktiež zmenší náš objekt. Taktiež je táto operácia vhodná pri spájaní rozdelených častí v obraze [16].



Obr. 3.1: Ukážka pôvodného obrázku (vľavo), obrázku po erózii (v strede) a obrázku po dilatácii (vpravo)[1]

3.3 Gaussovo rozostrenie

Gaussovo rozostrenie využíva Gaussovu funkciu pomocou ktorej zbavuje obraz šumu a taktiež spôsobuje rozmazanie obrazu. Jedná sa o dolnopriepustný filter, ktorý zachováva nízku priestorovú frekvenciu, redukuje šum a zanedbateľné detaily v obraze. To je dosiahnuté konvolúciou obrazu s Gaussovým jadrom. Toto jadro je v dvojrozmernom priestore definované ako:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.4}$$

kde σ je smerodajná odchýlka rozloženia a x a y sú indexy pixelu. Hodnota σ udáva rozptyl okolo strednej hodnoty Gaussovho rozloženia. To udáva ako veľmi bude okolie pixelov rozmazané [11].

3.4 Cannyho hranový detektor

Detekcia hrán je jedna z najviac používaných operácií pri spracovaní obrazu a pravdepodobne by sme nenašli žiaden iný algoritmus, ktorý by bol v literatúre spomenutý častejšie.

[1] Obrázky boli prevzaté a upravené z [16]

Dôvodom toho je, že hrany formujú obrys objektu a tieto objekty sú základom pri práci s obrazom. Pomocou hrán a obrysov vieme určiť hranicu medzi objektom a pozadím, a taktiež hranicu medzi dvoma prekrývajúcimi sa objektami. V prípade, že sú tieto hranice identifikované správne, vieme nájsť nielen pozíciu objektov v obraze, ale aj ich základné vlastnosti ako ich plochu, obvod a tvar. Pri spracovaní obrazu je teda detekcia hrán esenciálnym nástrojom [17]. K tomu sa využívajú rôzne hranové detektory, ktorých existuje veľké množstvo. Jedným z nich je Cannyho hranový detektor, ktorý bol zároveň využitý pri tvorbe tejto práce.

V roku 1986 definoval John Canny v článku *A Computational Approach to Edge Detection* [2] sadu cieľov, ktoré by mal spĺňať hranový detektor a popísal optimálne metódy na ich dosiahnutie. Táto sada obsahuje 3 problémy, ktoré musí detektor riešiť. Podľa zdroja [17] medzi tieto problémy patria:

- **Chybovosť** – detektor by mal rozoznávať len hrany, mal by ich nájsť všetky a žiadne by nemali byť vynechané.
- **Lokalizácia** – vzdialenosť medzi skutočnými hranami v obraze a nájdenými hranami by mala byť čo najmenšia.
- **Odozva** – detektor by nemal detekovať viacero hrán na mieste, kde sa nachádza jedna hrana, má detekovať len jednu odozvu na každú hranu.

Na to, aby boli splnené všetky tieto podmienky je potrebné realizovať Cannyho detektor vo viacerých krokoch. Tieto kroky, podľa zdroja [14] sú:

1. **Odstránenie šumu:**

   Vzhľadom na to, že je hranový detektor náchylný na šum, prvým krokom by malo byť jeho odstránenie. Táto operácia je väčšinou vykonávaná pomocou Gaussoveho rozostrenia, ktoré bolo popísané v sekcii 3.3. To je ešte možné rozšíriť o použitie erózie a dilatácie, ktoré tiež odstraňujú šum, čo bolo popísane v časti 3.2.

2. **Určenie intenzity gradientu:**

   Následne je potrebné vypočítať gradient obrazu. Na to je možné využiť rôzne hranové filtre. Medzi takéto filtre patrí napríklad: Prewittov filter, Sobelov filter a Laplaceov filter. Tento hranový filter je následne použitý v horizontálnom aj vertikálnom smere a je z neho vypočítaný výsledný gradient.

   Knižnica OpenCV, ktorá bola využitá pri tvorbe tejto práce, implementuje Cannyho detektor s pomocou Sobelovho filtra. Jeho konvolučná maska v horizontálnom smere je definovaná v tvare [25]:

$$\mathbf{G}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{3.5}$$

   Konvolučná maska vo vertikálnom smere je definovaná v tvare:

$$\mathbf{G}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.6}$$

Na získanie výsledného hranového obrazu je nutné využiť následovný vzťah na výpočet intenzity gradientu:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \tag{3.7}$$

Pomocou tejto intenzity je taktiež možné vypočítať smer gradientu a to takto:

$$\Theta = \arctan \frac{\mathbf{G}_y}{\mathbf{G}_x} \tag{3.8}$$

3. **Nájdenie lokálnych maxím:**

   Po získaní gradientu a jeho smeru je potrebné potlačiť všetky hodnoty, ktoré nie sú lokálne maximá. Pre každý pixel v obraze sa overí, či je daný pixel lokálne maximum pre jeho okolie, ktoré je dané smerom gradientu. Ak áno, je zachovaný na ďalšie spracovanie. V opačnom prípade je potlačený. Tento krok spôsobí stenčenie hrán, vďaka čomu je získaná jednoznačná odozva na hranu.

4. **Prahovanie hysteréziou:**

   Posledná fáza rozhoduje, ktoré hrany sú naozaj reálne hrany a ktoré nie. Na to je potrebné určiť 2 hraničné hodnoty a to: minimálny a maximálny prah. Hranu, ktorej intenzita gradientu je väčšia ako maximálny prah, môžeme s istotou označiť za hranu. Tie hrany, ktoré majú intenzitu gradientu menšiu ako minimálny prah, môžeme naopak s istotou odstrániť, pretože sa nejedná o hrany. Ostatné hrany, ktoré majú hodnotu gradientu medzi týmito hraničnými hodnotami, musíme rozdeliť na základe ich náväznosti k iným hranám. V prípade, že nadväzujú k pixelom, ktoré sú určite hranou, sú sami označené za hrany. Naopak, ak nenadväzujú, tak môžeme takéto hrany odstrániť. Je veľmi dôležité vybrať hodnoty prahov správne pre dosiahnutie dobrých výsledkov. Tento krok taktiež odstráni drobný šum, avšak za predpokladu, že hrany sú dlhšie čiary.

Medzi všetkými existujúcimi detektormi je Cannyho hranový detektor jeden z najstriktnejších algoritmov, vďaka čomu poskytuje kvalitnú a spoľahlivú detekciu hrán. Vzhľadom na jeho optimálnosť a jednoduchosť implementácie patrí medzi jeden z najpoužívanejších algoritmov na detekciu hrán.

3.5 Houghova transformácia

Houghova transformácia je metóda, ktorá slúži na detekciu určitých štruktúr v obraze. Vzhľadom na to, že využíva analytický popis tvaru hľadaných objektov, je často používaná na detekciu pravidelných kriviek ako napríklad: priamky, kružnice, elipsy a podobne. Využíva sa teda hlavne pri segmentácii objektov v obraze, ktorých hranice je možné popísať jednoduchými krivkami. Hlavnou výhodou tejto transformácie je, že aj napriek nedokonalostiam v ohraničeniach objektov alebo napriek šumu v obraze, dokáže fungovať spoľahlivo. Pri tvorbe tejto práce bola potrebná detekcia priamok, a práve preto je v práci rozobratá podrobnejšie.

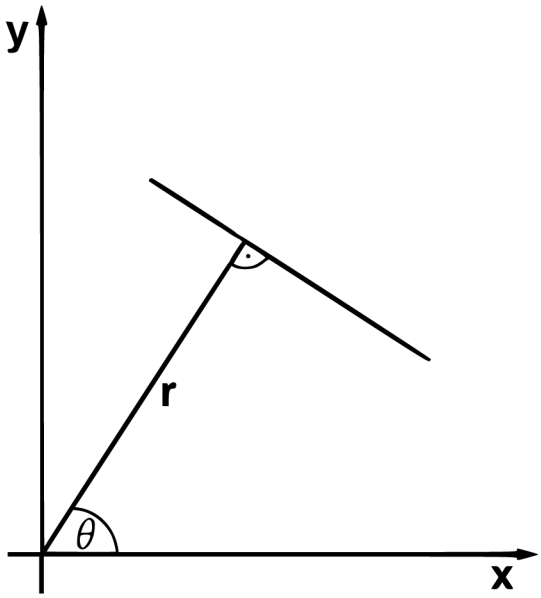
Priamku je možné analyticky popísať viacerými formami. Napríklad v karteziánskej sústave ju môžeme popísať ako vzťah $y = ax + b$. Problém však nastáva v prípade, ak je priamka rovnobežná s osou y a parameter a sa stáva nekonečným. Vzhľadom na to nie je

možné využiť tento tvar pri Houghovej transformácii. Ak však využijeme polárnu sústavu, dostaneme oveľa vhodnejší tvar priamky, a to:

$$x \cos \theta + y \sin \theta = r \tag{3.9}$$

kde r je dĺžka normály od začiatku sústavy k priamke a θ je uhol, ktorý zviera r vzhľadom na os x. Toto znázorňuje obrázok 3.2.

Pri hľadaní priamky sú využité súradnice pixelov (x, y) ako vstup a parametre r a θ sú neznáme. Ak do rovnice priamky dosadíme súradnice nejakého bodu, tak množina všetkých riešení tejto rovnice (r, θ) vytvorí v Houghovom polárnom priestore spojitú krivku. Pri vykreslení všetkých bodov, ktoré ležia na rovnakej priamke do Houghovho polárneho priestoru, je možné pozorovať ako sa tieto krivky pretnú v jednom bode so súradnicami r a θ. Táto výsledná dvojica obsahuje parametre, ktoré definujú hľadanú priamku [5]. Vo všeobecnosti je teda možné detekovať priamku nájdením počtu kriviek, ktoré sa medzi sebou pretínajú. Čím viac je takýchto kriviek, tým viac pixelov patrí danej priamke. V obraze môžu tieto priamky značiť hrany, ktoré hľadáme. Pri takejto detekcii je vhodné definovať minimálny počet pretínajúcich sa kriviek, a teda minimálny počet pixelov, kedy bude hrana detekovaná.



Obr. 3.2: Definícia priamky v polárnej sústave

Kapitola 4

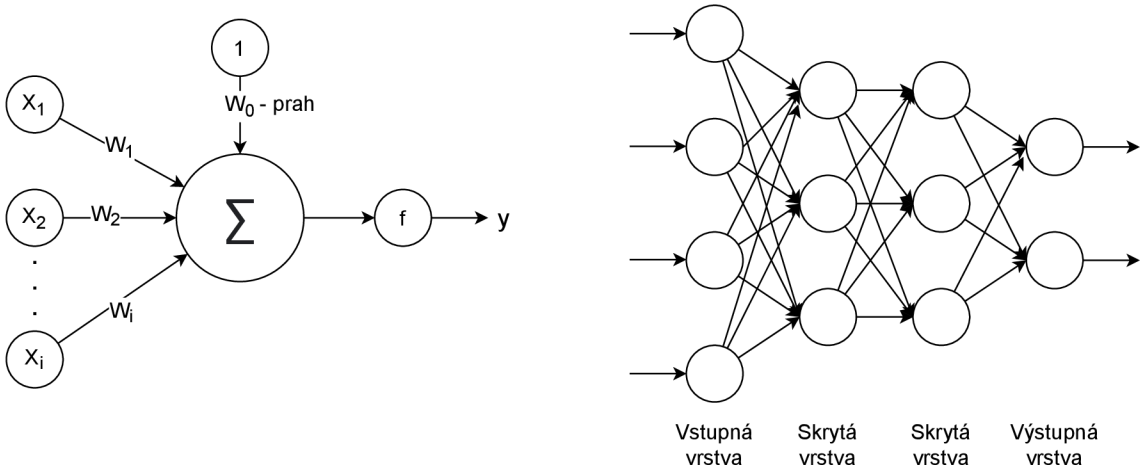
Neurónové siete

Neurónové siete sú výpočetné modely inšpirované biologickými neurónovými sústavami. Ich základnou jednotkou je takzvaný umelý neurón. Sieť sa následne skladá z veľkého počtu takýchto neurónov, ktoré sú medzi sebou vzájomne prepojené tak, že výstup jedného neurónu je vstupom pre ďalšie neuróny. To je možné využiť v rôznych oblastiach ako napríklad predikcia, klasifikácia, aproximácia, rozpoznávanie obrazu a v mnohých ďalších [8].

Neurón môže mať ľubovoľný počet vstupov x_i a práve jeden výstup y. Každý vstup je ohodnotený váhou w_i, ktorá určuje priepustnosť. Táto priepustnosť môže nadobúdať aj záporné hodnoty. Následne sa hodnoty vstupov vynásobia s príslušnými váhami, vďaka čomu je možné upravovať silu prepojenia neurónov. Tento model umelého neurónu zobrazuje obrázok 4.1. Po získaní vstupných hodnôt sa urobí zo všetkých vstupov súčet. K nemu je ešte pripočítaná hodnota prahu (*bias*). Výsledok sa ďalej použije ako argument aktivačnej funkcie. To je viac popísané v sekcii 4.1. Vrátená hodnota je výstupom neurónu. To je definované následujúcim matematickým vzťahom:

$$y = f\left(\sum_{i=1}^{N} w_i x_i + w_0\right) \tag{4.1}$$

Topológia siete určuje vzájomné usporiadanie neurónov. Na základe topológie vieme neurónové siete rozdeliť na cyklické (rekurentné) a acyklické (dopredné). Väčšina sietí má neuróny usporiadané vo vrstvách. Rozlišujeme 3 typy vrstiev: vstupná, skrytá a výstupná vrstva. Vstupná vrstva prijíma externé vstupné dáta a posiela ich ďalšej (skrytej) vrstve. Vstupom skrytej vrstvy môže byť výstup vstupnej vrstvy, ale taktiež aj výstup inej skrytej vrstvy. V prípade viacerých skrytých vrstiev hovoríme o hlbokej neurónovej sieti. Výstupná vrstva je posledná a vracia predikciu výsledku [24]. Ukážku jednoduchej neurónovej siete je možné vidieť na obrázku 4.1.



Obr. 4.1: Model umelého neurónu (vľavo) a neurónová sieť s dvomi skrytými vrstvami (vpravo)

4.1 Aktivačné funkcie

Aktivačné funkcie sú funkcie, ktoré na základe váženého súčtu zo vstupov neurónu a prahu rozhodujú, či je neurón aktívny. Aktivačné funkcie môžu byť lineárne alebo nelineárne na základe toho, akú funkciu reprezentujú. Správny výber aktivačnej funkcie zlepšuje výsledky neurónovej siete a urýchľuje učenie. Medzi najpoužívanejšie funkcie patrí: sigmoida, hyperbolický tangens, softmax a rectified linear unit [13].

4.1.1 Sigmoida

Sigmoida alebo logistická funkcia je nelineárna aktivačná funkcia, ktorá sa využíva hlavne v dopredných neurónových sieťach. Jej definičný obor sú všetky reálne čísla a nadobúda hodnoty na intervale (0, 1). Sigmoida je daná nasledujúcim vzťahom:

$$f(x) = \left(\frac{1}{(1 + \exp^{-x})} \right) \tag{4.2}$$

Zvyčajne sa objaví vo výstupných vrstvách sietí a používa sa na určenie pravdepodobnosti javov. Vzhľadom na jej jednoduchosť sa hlavne využíva v plytkých neurónových sieťach. Má však veľa nedostatkov ako napríklad prudký gradient pri spätnom prehľadávaní (viď sekcia 4.3) z hlbších skrytých vrstiev do vstupných vrstiev, pomalá konvergencia alebo nenulový stred, čo spôsobuje nesprávnu spätnú úpravu váh.

4.1.2 Hyperbolický tangens

Hyperbolický tangens (*tanh*) je aktivačná funkcia veľmi podobná sigmoide, avšak odstraňuje niektoré jej nedostatky. Má plynulejší priebeh, nulový stred a nadobúda hodnoty z intervalu (-1, 1). Môžeme ju definovať ako:

$$f(x) = \left(\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \right) \tag{4.3}$$

Funkcia tanh je v porovnaní so sigmoidou viac preferovaná, pretože podáva lepšie výsledky pri trénovaní viacvrstvových neurónových sietí. Hlavnou výhodou je, že jej výstup je centrovaný okolo nuly, čím negatívne neovplyvňuje spätnú úpravu váh. Hyperbolický tangens sa najčastejšie využíva v rekurentných neurónových sieťach na spracovanie jazyka a rozpoznávanie reči.

4.1.3 Softmax

Softmax je ďalší typ aktivačnej funkcie a využíva sa hlavne na výpočet rozdelenia pravdepodobnosti z vektorov s reálnymi číslami. Softmax je možné popísať nasledovne:

$$f\left(x_i\right) = \frac{\exp\left(x_i\right)}{\sum_j \exp\left(x_j\right)} \tag{4.4}$$

Jej výstupom sú hodnoty z intervalu $(0, 1)$ a suma pravdepodobností sa rovná 1. Práve preto sa najčastejšie vyskytuje v posledných vrstvách neurónových sietí, ktoré majú viacero klasifikačných tried. Výsledná hľadaná trieda tak má najväčšiu pravdepodobnostnú hodnotu. Hlavný rozdiel medzi sigmoidou a softmaxom je ten, že sigmoida sa používa pri binárnej klasifikácii a softmax pri klasifikáciách s viacerými triedami.

4.1.4 Rectified Linear Unit

Aktivačná funkcia Rectified Linear Unit (ReLU) bola navrhnutá dvojicou Nair a Hinton v roku 2010 a odvtedy je jednou z najpoužívanejších aktivačných funkcií s perfektnými výsledkami. Je veľmi rýchla, pretože nevyužíva exponenciály a delenie, čo je jej veľkou výhodou. Jednou z horších vlastností ReLU je, že sa pri jeho použití model jednoduchšie pretrénuje (viď sekcia 4.3) v porovnaní so sigmoidovou funkciou. Najčastejšie sa objavuje v neurónových sieťach, ktoré klasifikujú objekty alebo rozpoznávajú reč. ReLU môžeme matematicky zapísať takto:

$$f(x) = \max(0, x) = \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases} \tag{4.5}$$

4.2 Konvolučná neurónová sieť

Konvolučná neurónová sieť je špeciálny typ neurónovej siete, ktorý sa najčastejšie využíva pri spracovaní obrazových dát. Príkladom takýchto dát môžu byť napríklad obrazové dáta, ktoré je možné usporiadať ako dvojrozmernú mriežku pixelov. Ako už z názvu vyplýva, táto neurónová sieť využíva matematickú operáciu s názvom konvolúcia. Konvolučné neurónové siete sú teda také neurónové siete, ktoré využívajú konvolúciu aspoň v jednej z ich vrstiev. Takáto vrstva sa nazýva konvolučná vrstva.

Konvolúcia je lineárna matematická funkcia definovaná nasledujúcim vzťahom [6]:

$$s(t) = (x * w)(t) \tag{4.6}$$

kde x označuje vstup neurónovej siete, w označuje jadro konvolúcie a $*$ symbolizuje samotnú konvolúciu. Ako už bolo spomenuté vyššie, vstupom je zvyčajne mriežka alebo viacrozmerné pole, ktoré sa nazýva tenzor. Jadro konvolúcie má zvyčajne rovnaký počet rozmerov ako

má vstup. Pri spracovaní obrazových dát sú to typicky 2 alebo 3 rozmery podľa farebnej hĺbky obrazu. Konvolúcia je v kontexte neurónových sietí pre dvojrozmerný obrázok s dvojrozmerným jadrom definovaná ako [6]:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{4.7}$$

kde I je vstupný obrázok, K je jadro konvolúcie, i a j reprezentujú súradnice bodu obrázku a m a n sú rozmery jadra.

4.3 Trénovanie

Trénovanie alebo učenie siete zabezpečuje algoritmus, ktorý iteratívne počas učenia mení váhy jednotlivých prepojení neurónov s cieľom minimalizovať hodnotu chybovej funkcie. Tá vyjadruje, ako dobre sieť predikuje výsledok. Teória, na základe ktorej je založené učenie sietí, vychádza z Hebbovho zákona. Ten je inšpirovaný učením mozgu. V prípade, ak sú 2 neuróny v jednom okamihu aktívne (jeden vybudil druhý), je potrebné väzbu medzi nimi zosilniť (upraviť príslušné váhy). V opačnom prípade treba väzbu zoslabiť. Tento zákon je v rôznych obmenách implementovaný vo väčšine učiacich algoritmov. To, ako je sieť vytrénovaná, je uložené vo váhach spojov medzi neurónmi a samotná topológia sa pri tom nemení. Ak by to bolo potrebné, je možné nastaviť váhu spojenia na 0 a tým realizovať prerušenie spojenia medzi dvoma neurónmi [8].

K trénovaniu neurónových sietí sa dá pristupovať dvoma spôsobmi: učenie s učiteľom a bez učiteľa. V prvom prípade sú dopredu známe výstupné hodnoty pre vstupné dáta. Tie spolu tvoria trénovaciu množinu (trénovací dataset). Pomocou tejto množiny sa sieť snaží naučiť takú funkciu, ktorá by najlepšie aproximovala vzťah medzi vstupnými a výstupnými dátami. To dosiahne tým, že porovnáva požadovaný výstup s reálnym výstupom siete a upraví váhy prepojení tak, aby sa výstup čo najviac priblížil požadovanej hodnote. Toto je vykonané s každým prvkom z trénovacej množiny. Pri učení bez učiteľa sa nevyužívajú žiadne dopredu známe výstupné hodnoty. Sieť využíva iba vstupné dáta a pomocou zhlukovania sa sama snaží nájsť vzťahy medzi dátami [8].

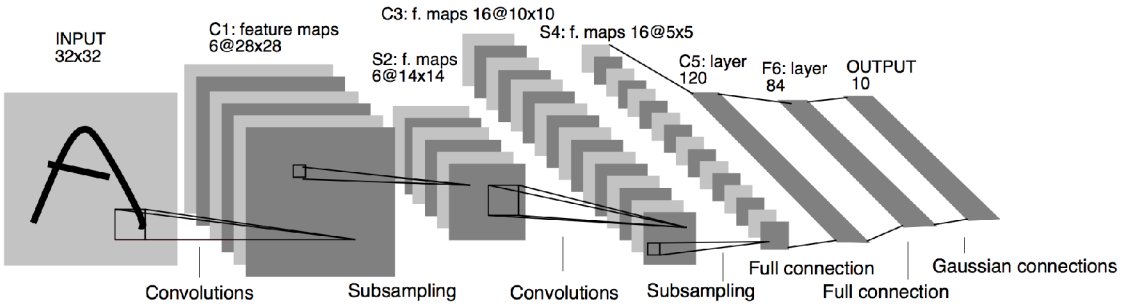
Algoritmus spätného šírenia chyby (backpropagation) sa používa v približne 80 % [24] zo všetkých aplikácií neurónových sietí. Skladá sa z troch etáp: dopredné (feedforward) šírenie vstupného signálu tréningového vzoru, spätné šírenie chyby a úprava váh prepojení neurónov. V prvej etape prejdú vstupné dáta cez všetky vrstvy siete, kde si každý neurón vypočíta svoju aktivačnú hodnotu a v poslednej vrstve sa vygenerujú odhadované výstupy. Druhá etapa, a teda samotné spätné šírenie chyby, spočíva v šírení informácie smerom od výstupných vrstiev k vstupným. Počas adaptácie neurónovej siete sú pri metóde backpropagation porovnávané vypočítané aktivačné hodnoty výstupných neurónov s výstupnou hodnotou z trénovacej množiny. Na základe toho sa definuje chyba siete, z ktorej sa vypočíta gradient pre jednotlivé neuróny. Ten sa následne spätne šíri z každého neurónu ku všetkým neurónom z predchádzajúcej vrstvy. Tento proces sa opakuje vo všetkých vrstvách až k vstupnej vrstve. Pri tomto šírení taktiež dochádza k úprave váh prepojení, a to na základe hodnoty gradientu [24].

S trénovaním sa spája aj problém pretrénovania siete. Tento problém nastáva, ak sieť dokáže perfektne určiť správne výstupy pre vstupné dáta z trénovacej množiny, ale nie je schopná generalizovať a odpovede na skutočné dáta nie sú príliš presné [8]. Jednoduchšie povedané, si sieť zapamätala všetky hodnoty z trénovania, ale nenaučila sa vzťah na základe ktorého by mala tieto hodnoty detekovať. K tomuto dochádza v prípade, keď trénovacia

množina obsahuje málo dát alebo trénovanie prebieha príliš dlho. Na určenie, kedy nastáva pretrénovanie, je vhodné využiť validačnú množinu. Tá obsahuje také dáta, na ktorých sieť nebola trénovaná a je teda možné pozorovať, ako sa sieť správa za reálnych podmienok. Pri snahe eliminovať pretrénovanie je vhodné využiť takzvaný dropout (výpadok). Dropout zabezpečí, že niektoré neuróny a ich hodnoty budú ignorované. Tieto neuróny sú vyberané náhodne na základe zadanej pravdepodobnosti. Tým sa zamedzí vytvoreniu závislosti pri trénovaní medzi neurónmi vo vrstvách, a teda aj samotnému pretrénovaniu. Tento postup sa v spojení s neurónovými sieťami nazýva regularizácia [1].

4.4 Architektúra LeNet-5

LeNet-5 je konvolučná neurónová sieť, ktorú v roku 1998 prvýkrát popísal Yann LeCun v publikácii *GradientBased Learning Applied to Document Recognition* [9]. Sieť sa skladá zo siedmich vrstiev a je zobrazená na obrázku 4.2. Zameriava sa na rozpoznávanie ručne písaných a strojovo tlačených znakov. Vstupom tejto siete je šedý obrázok s rozmermi 32x32 pixelov.



Obr. 4.2: Zobrazenie jednotlivých vrstiev architektúry LeNet-5[1]

4.5 Dataset

Dataset je v kontexte neurónových sietí súbor dát, ktoré majú určité spoločné charakteristiky. Pomocou datasetu sa následne trénuje neurónová sieť. Zvyčajne sa skladá zo vstupných dát a ich popisu.

4.5.1 MNIST

Dataset MNIST je zložený z ručne písaných čísel. Obsahuje 60 000 trénovacích vzoriek a 10 000 testovacích vzoriek. MNIST je podmnožinou väčšieho datasetu NIST, ktorý obsahuje okrem ručne písaných čísel aj písmená. Čísla v datasete MNIST sú šedé, normalizované, centrované na stred a majú fixnú veľkosť, a to 28x28 pixelov. Dataset je uložený v štyroch súboroch, kde prvé dva súbory obsahujú trénovacie vzorky a ich popis, a druhé dva súbory obsahujú testovacie vzorky a ich popis [12]. Existuje taktiež dataset EMNIST (Extended MNIST), ktorý rozširuje MNIST. EMNIST zachováva pôvodný formát a vlastnosti aké má MNIST, no obsahuje až 240 000 trénovacích vzoriek a 40 000 testovacích vzoriek [3].

[1]Obrázok bol prevzatý z [9]

4.5.2 Vlastný dataset

Pri tvorbe tejto práce bol vytvorený vlastný dataset ručne písaných čísel, a to z dôvodu, že dataset MNIST obsahuje obrázky čísel vo veľmi dobrej kvalite, so zarovnaním na stred a bez deformácií, čo neodzrkadľuje reálnu formu orezaných a spracovaných obrázkov čísel zo sudoku. Hlavne v prípade, ak sú čísla vpísané ručne a nemusia sa nachádzať v strede políčka alebo nemusia byť dostatočne výrazne vpísané. Tieto drobné rozdiely znižujú šancu na správnu detekciu čísla pomocou neurónovej siete.

Vzorky boli získavané od respondentov pomocou jednoduchej úlohy. Každý respondent obdržal 9 prázdnych zadaní sudoku (štvorcová mriežka o rozmeroch 9x9). Následne bolo potrebné každé zadanie vyplniť jednou a tou istou číslicou od 1 do 9. To znamená, že prvé vyplnené zadanie bolo vyplnené iba jednotkami, druhé dvojkami a tak ďalej. Výsledkom bolo 729 čísel. Na jednu číslicu pripadalo 81 vzoriek. Túto úlohu vykonalo 44 respondentov. Dataset bol taktiež doplnený o 7 tlačených zadaní. Tie boli vyplnené rovnakým spôsobom ako predošlé zadanie, avšak s tým rozdielom, že čísla boli strojovo tlačené. Výsledný dataset obsahuje 37 179 vzoriek čísel od 1 do 9. Na jednu číslicu teda pripadá 4 131 vzoriek. Jednotlivé obrázky sú šedé a majú veľkosť 28x28 pixelov. Dataset sa skladá z dvoch súborov, kde jeden obsahuje samotné vzorky a druhý ich popis.

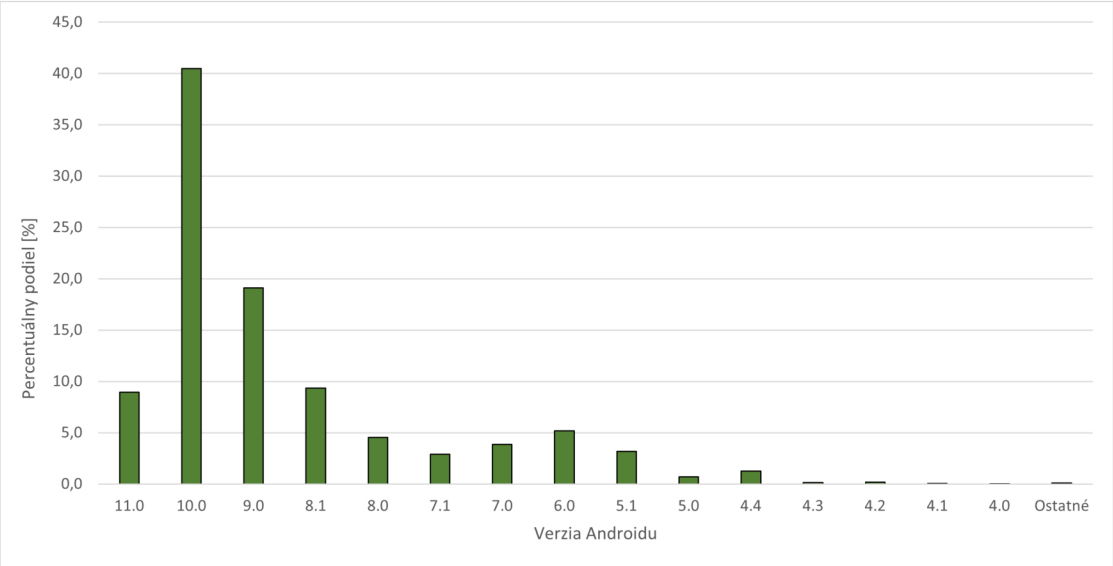
Kapitola 5

Vývoj mobilnej aplikácie pre Android

Táto kapitola popisuje operačný systém Android a technológie, pomocou ktorých je možné vytvoriť výslednú mobilnú aplikáciu. Kapitola zahŕňa taktiež spôsob, ako sa dá takáto aplikácia zverejniť pre užívateľov.

5.1 Android

Android je *open source* operačný systém založený na Linuxovom jadre a je určený pre mobilné zariadenia ako smartfóny, tablety či inteligentné hodinky. Tento systém bol vyvinutý spoločnosťou *Open Handset Alliance* pod vedením spoločnosti *Google*. Jednou z hlavných výhod Androidu je, že ponúka jednotný prístup pri vývoji aplikácií, vďaka čomu môžu tieto aplikácie fungovať na rôznych zariadeniach, pokiaľ tieto zariadenia využívajú Android [23]. Na základe najnovšej štatistiky z marca 2021 vyplýva, že spomedzi všetkých operačných systémov, určených pre mobilné zariadenia, sa Android umiestnil na prvom mieste s podielom 71,81 % [20]. Zastúpenie jednotlivých verzií Androidu zobrazuje obrázok 5.1.

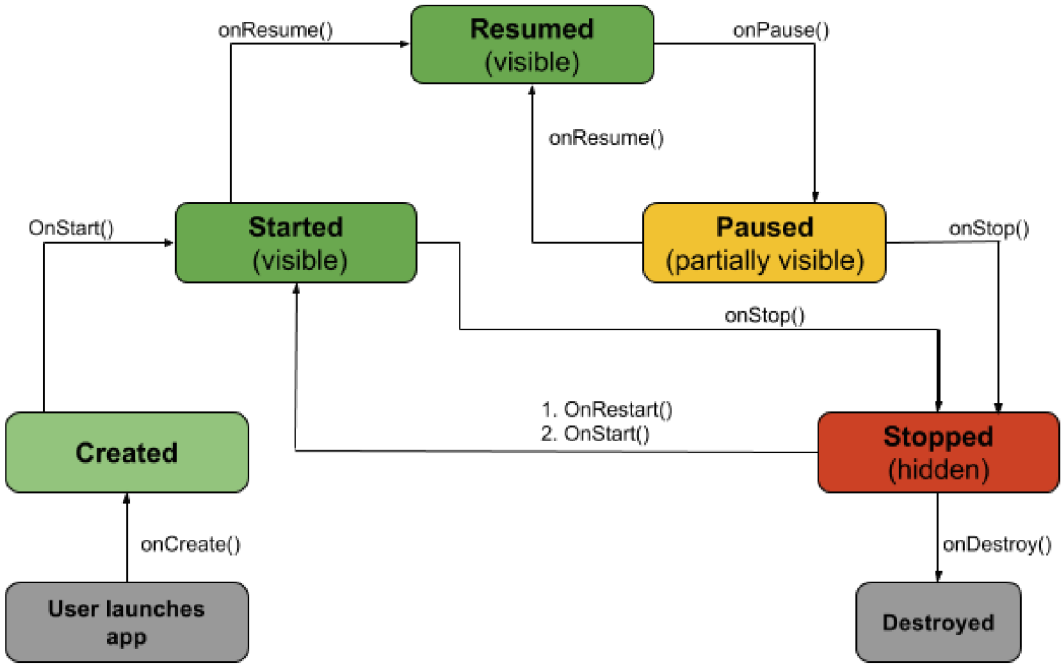


Obr. 5.1: Celosvetové percentuálne zastúpenie verzií Androidu v marci 2021[1]

5.2 Kotlin

Kotlin[2] je staticky typovaný programovací jazyk vytvorený spoločnosťou *JetBrains*[3]. V máji 2019 spoločnosť *Google* oznámila, že Kotlin je preferovaný jazyk na vývoj aplikácii pre Android. Pred týmto vyhlásením bol preferovaným programovacím jazykom jazyk Java[4]. Kotlin nie je syntakticky kompatibilný s jazykom Java, no aj napriek tomu je navrhnutý pre úplnú interoperabilitu s jej knižnicami. Kotlin sa navyše snaží odstrániť niektoré nedostatky, ktoré vznikajú pri práci s jazykom Java. Medzi hlavné výhody patrí vyššia bezpečnosť a rýchlosť kompilácie programu. Jazyk taktiež značne uľahčuje a skracuje písanie kódu, vďaka jeho kompaktnosti a intuitívnosti. Kotlin tiež podporuje multiplatformový vývoj a môže byť použitý pri tvorbe aplikácií nielen na Android ale aj iOS, či inú platformu [18].

Základnými prvkami pri tvorbe aplikácie pomocou jazyka Kotlin sú takzvané aktivity. Tie reprezentujú jednotlivé obrazovky aplikácie a uchovávajú v sebe ďalšie prvky. Vďaka ním môže užívateľ ovládať aplikáciu. Každá aktivita má svoj životný cyklus (viď obrázok 5.2). Ten je definovaný metódami a tie sa spúšťajú v definovaných momentoch. Tieto metódy sú: `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, `onRestart()` a `onDestroy()`.



Obr. 5.2: Životný cyklus aktivity v aplikácii[5]

Metóda `onCreate()` sa volá pri prvom vytvorení aktivity, po čom nasleduje metóda `onStart()`, ktorá sprístupní aktivitu užívateľovi. Po tomto sprístupnení je vždy volaná

[1]Dáta boli prevzaté z https://gs.statcounter.com/os-version-market-share/android/mobile-tablet/worldwide

[2]https://kotlinlang.org/

[3]https://www.jetbrains.com/

[4]https://www.java.com/

[5]Obrázok bol prevzatý z https://www.raywenderlich.com/2705552-introduction-to-android-activities-with-kotlin

metóda `onResume()` a aktivita sa presúva do popredia a môže byť využívaná. Na schovanie aktivity do pozadia slúži `onPause()`. V stave, kedy už užívateľ nemá prístup k aktivite, je volaná metóda `onStop()`. Ďalej je možné aktivitu znova prebrať a obnoviť s metódou `onRestart()` alebo ju úplne odstrániť pomocou metódy `onDestroy()`.

5.3 Google Play

Google Play je distribučná služba, prostredníctvom ktorej je možné publikovať a sťahovať aplikácie určené pre Android. Táto služba umožňuje okrem sťahovania aplikácii aj online nákup filmov alebo kníh. Google Play je dostupný pomocou prehliadača alebo v predinštalovanej aplikácií na mobilných zariadeniach.

Na to, aby aplikácia mohla byť zverejnená v službe Google Play, je potrebné vytvorenie účtu pre vývojára. Pri tvorbe tohto účtu treba zaplatiť jednorázový poplatok 25 dolárov. Ďalej je potrebné vytvoriť profil pre samotnú aplikáciu. Na to slúži Google Play Console. Pri vytváraní profilu je vyžadovaný názov aplikácie, jej popis, logo, snímky z aplikácie a taktiež vyplnenie formuláru o obsahu aplikácie. Aplikáciu je možné okrem finálneho zverejnenia vydať aj ako verziu určenú na rôzne druhy testovania. Zverejnené aplikácie ďalej môžu byť dostupné buď zdarma alebo za poplatok. Z týchto poplatkov si služba Google Play nárokuje 30 %[6]. Po zverejnení aplikácie si vývojár môže v Google Play Console sprístupniť veľké množstvo štatistík o aplikácii ako napríklad: koľko ľudí si aplikáciu stiahlo a v akých krajinách, koľkokrát sa aplikácia objavila vo vyhľadávaní alebo na akých zariadeniach je aplikácia nainštalovaná. Navyše sa dajú zobraziť chyby, kvôli ktorým aplikácia užívateľom nefungovala, a tak získať lepšie informácie o prípadných problémoch.

[6] https://support.google.com/googleplay/android-developer/answer/112622?hl=en

Kapitola 6

Návrh aplikácie

Táto kapitola sa zameriava na návrh aplikácie a popisuje, aké postupy boli použité pri implementácii samotnej aplikácie. Ďalej rozoberá existujúce riešenia a ich jednotlivé pozitívne a negatívne stránky. Vďaka týmto zisteniam je možné navrhnúť takú aplikáciu, ktorá sa vyhne návrhovým chybám, ktoré sa vyskytli v týchto riešeniach. Hlavnými zásadami pri tvorbe aplikácie boli:

- **Jednoduchosť** – každý bežný riešiteľ sudoku zvládne ovládať túto aplikáciu.
- **Spoľahlivosť** – aplikácia dokáže spoľahlivo detekovať pozíciu sudoku, rozpoznávať čísla a správne riešiť zadané sudoku.
- **Praktickosť** – aplikácia bude rýchla, nebude obsahovať nadbytočné funkcie alebo kroky a bude vyžadovať čo najmenej zásahov od užívateľa.

6.1 Existujúce riešenia

Existuje veľké množstvo aplikácií prostredníctvom ktorých je možné riešiť sudoku. Tie je možné vyhľadať napríklad v obchode Google Play[1], odkiaľ boli vybrané existujúce riešenia. Zvolené aplikácie museli spĺňať tieto kritériá: možnosť naskenovať sudoku pomocou kamery a schopnosť vyriešiť dané sudoku. Ich pozitívne a negatívne stránky sú popísané v tejto sekcii.

1. **Sudoku Solver (Camera) - Robinson Industries:**[2]

   Táto aplikácia v reálnom čase spracováva obraz z kamery a v prípade, keď zdetekuje zadanie sudoku, tak sa ho pokúsi riešiť a vypísať. Tu vzniká hneď niekoľko problémov. Keďže je spracovanie sudoku a jeho riešenie relatívne výkonne náročné, tak obraz z kamery začne sekať pri snahe riešiť sudoku. To spôsobí akoby zamrznutie aplikácie. Ďalším problémom je, že aplikácia detekuje sudoku ešte počas pohybu kamery a tak sa môže stať, že je obraz nekvalitný, a preto sa nerozpoznajú všetky čísla v sudoku správne. Často sa stalo, že sa na obrazovke zobrazila len časť zadania a nebolo možné vidieť všetky vyriešené hodnoty (viď obrázok 6.1). Chýba tu taktiež možnosť vypĺňať každé políčko individuálne používateľom a možnosť kontroly a úpravy naskenovaného zadania. Aplikácia taktiež nezvládne rozpoznávať ručne písané čísla, takže nie je možné skenovať čiastočne riešené sudoku.

[1] https://play.google.com/store

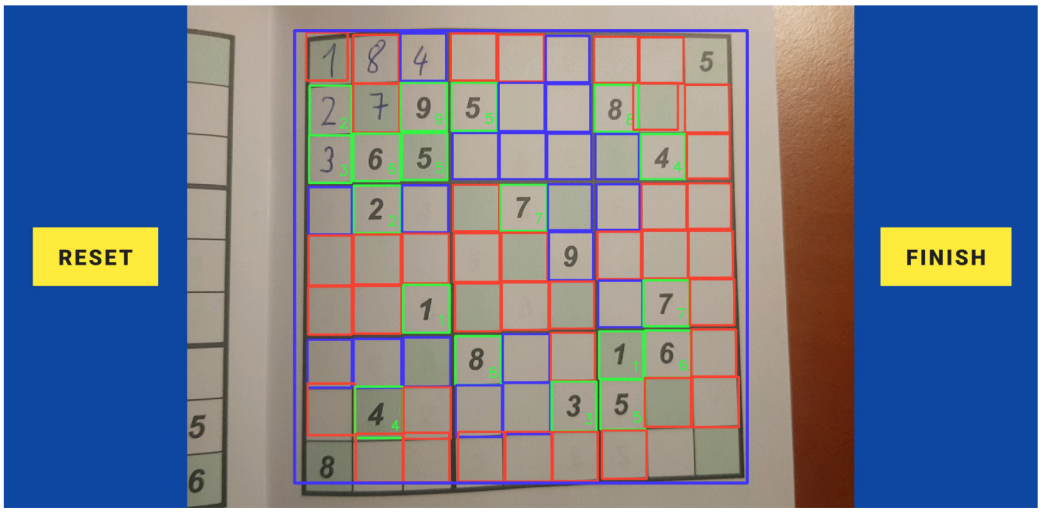
[2] https://play.google.com/store/apps/details?id=com.RobinsonIndustries.SudokuSolver

Obr. 6.1: Ukážka aplikácie Sudoku Solver (Camera)

2. **Sudoku Solver - Scan and Solve Sudoku - Divya Mamgai:**[3]

   Táto aplikácia upravuje niektoré funkcie, ktoré ponúkala predošlá aplikácia. Opäť sa sudoku spracováva v reálnom čase, avšak len v prípade, ak užívateľ drží prst na displeji. To vytvára priestor na to, aby bolo možné sudoku zarovnať s rámčekom, ktorý je zobrazený na obrazovke. Pri spustení detekcie sa sudoku rozdelí na jednotlivé štvorce a začnú sa detekovať čísla. To všetko stále v reálnom čase. V tomto prípade už obrazovka nezamŕza, ale užívateľ je nútený držať telefón v stále rovnakej pozícií. To je celkom obtiažne, málo stabilné a treba byť pri tom veľmi precízny. Doba rozpoznávania čísel je dlhšia čo viac komplikuje detekciu. Aj pri veľkej snahe bolo často nereálne naskenovať sudoku správne. Rozpoznávanie ručne písaných čísel opäť nefungovalo a hľadanie riešenia trvalo aplikácii dlho. Znova nie je umožnené riešiť sudoku postupne, ale je vypísané iba celé výsledné riešenie. Postup skenovania sudoku zachytáva obrázok 6.2.

[3] https://play.google.com/store/apps/details?id=com.divyamamgai.sudokusolver

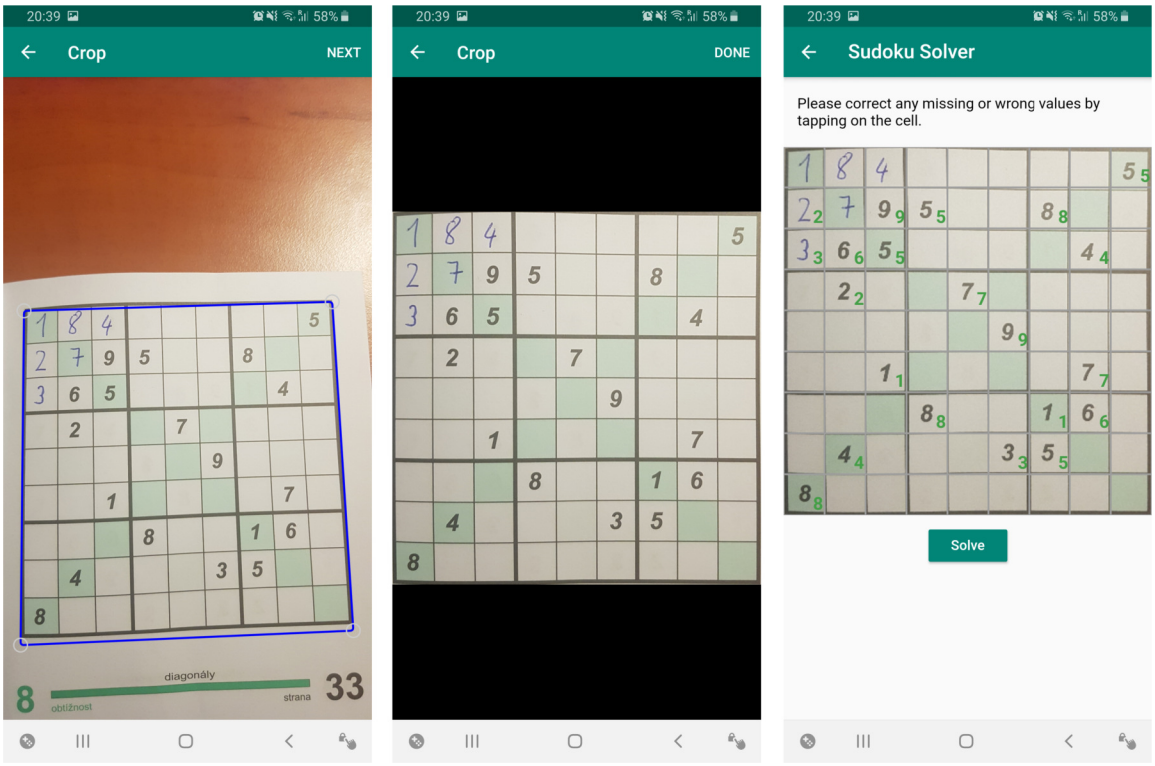


Obr. 6.2: Ukážka skenovania sudoku v aplikácii Sudoku Solver - Scan and Solve Sudoku

3. **Sudoku Solver - Scanner app using camera - Publicida Media:**[4]

   Nasledujúca aplikácia už nespracováva sudoku v reálnom čase, ale využíva fotografiu, z ktorej následne získava potrebné informácie. Výhodou tohto prístupu je, že je veľmi jednoduchý pre užívateľa a prakticky sa od neho vyžaduje len stlačenie jedného tlačidla. V tomto prípade je ešte užívateľ požiadaný o upravenie miesta, kde bude fotografia orezaná pre lepšie určenie pozície sudoku. Aplikácia navyše oproti predošlým riešeniam umožňuje upraviť naskenované zadanie, čo je veľkou výhodou v prípade chyby. Naopak stále chýba schopnosť detekovať ručne vpísané čísla a možnosť riešiť sudoku postupne. Rozpoznávanie čísel zaberá stále veľmi dlhú dobu. Aplikácia tiež neoveruje správnosť zadania, čo značne uberá na jej dôveryhodnosti. Jednotlivé kroky aplikácie sú zobrazené na obrázku 6.3.

[4] https://play.google.com/store/apps/details?id=publicida.media.sudokusolver

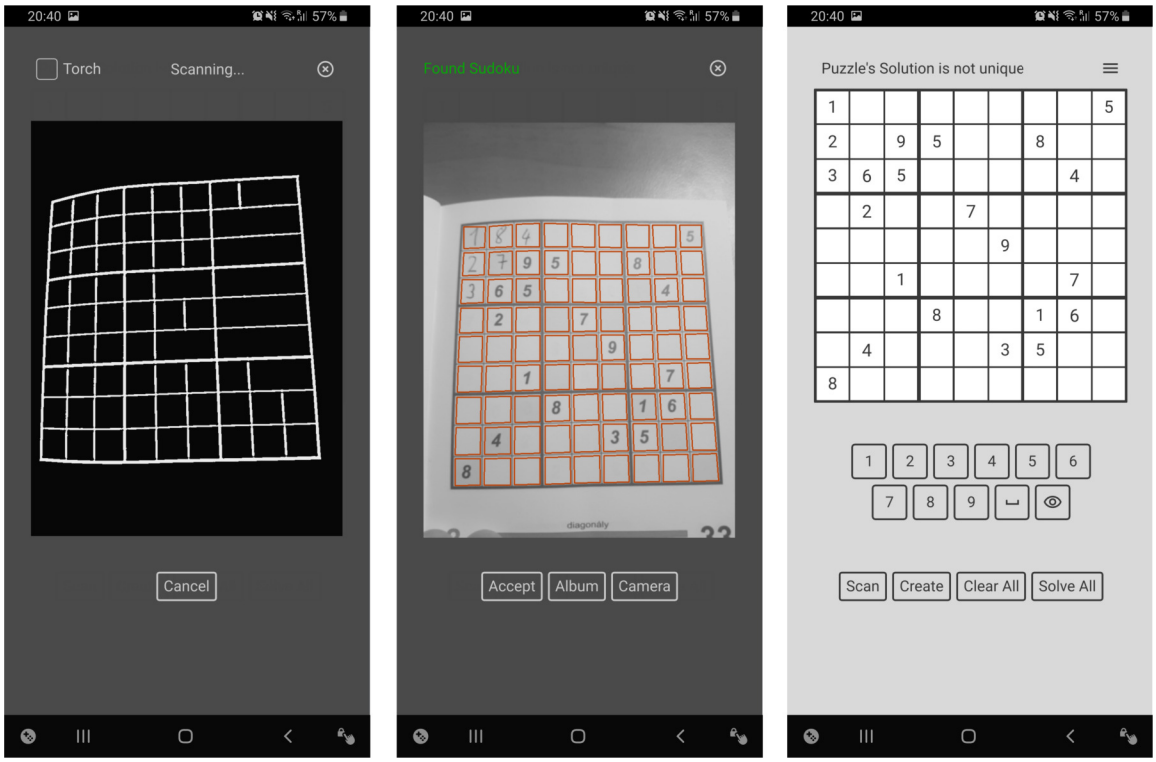


Obr. 6.3: Ukážka aplikácie Sudoku Solver - Scanner app using camera, kde obrázok vľavo zachytáva možnú úpravu bodov orezania, stredný obrázok ukazuje výsledok orezania a na obrázku vpravo je sudoku s rozpoznanými číslami

4. **Sudoku Scan&Solve - Stephan Widor:**[5]

   Posledná aplikácia síce využíva spracovanie v reálnom čase, ale detekuje len pozíciu sudoku. To spôsobí pokles snímkov pri zobrazovaní na displeji, ale nespôsobí to zamrznutie aplikácie. Až po nájdení správnej pozície a potvrdení od užívateľa sa rozpoznajú čísla v zadaní. Toto riešenie funguje veľmi dobre. Aplikácia je schopná rozpoznať sudoku aj z nahranej fotografie z galérie a ako prvá umožňuje riešiť sudoku postupne. Umožňuje aj nápovedu a nie len celkové riešenie zadania, čo je veľkou výhodou. Ďalšou novinkou oproti predchádzajúcim riešeniam je generovanie nového zadania sudoku. Ako aj pri ostatných aplikáciach aj táto má niekoľko nedostatkov. Nedokáže správne rozpoznávať ručne písané čísla a pri riešení sudoku nie je rozlíšené zadanie od nami doplnených čísel. Vzniká tak šanca, že si užívateľ prepíše samotné zadanie a bude riešiť iné sudoku ako pôvodne chcel. Obrázok 6.4 zachytáva fungovanie tejto aplikácie.

[5] https://play.google.com/store/apps/details?id=com.stephanwidor.sudokuscanandsolve



Obr. 6.4: Ukážka aplikácie Sudoku Scan&Solve, kde snímka vľavo zobrazuje detekciu hrán zadania sudoku v reálnom čase, obrázok v strede zobrazuje zachytený obrazok s detekovaným zadaním a posledný obrázok vpravo zachytáva aktivitu, kde sa rieši sudoku

Na základe týchto existujúcich riešení je možné určiť, čomu je potrebné sa pri návrhu vyhnúť alebo naopak, čo je dôležité zahrnúť a prípadne akým spôsobom implementovať. Aby teda aplikácia poskytovala čo najlepší užívateľský zážitok je potrebné:

- využiť fotografiu na rozpoznávanie sudoku a nie spracovanie v reálnom čase,
- mať schopnosť detekovať a rozpoznávať aj ručne vpísané čísla v zadaní,
- umožniť užívateľovi upraviť skenované zadanie v prípade chyby,
- overiť správnosť zadania,
- umožniť užívateľovi riešiť sudoku postupne (s nápovedou) a nie len zobraziť kompletné riešenie,
- detekovať a riešiť sudoku v relatívne krátkej dobe,
- poskytnúť intuitívne a jasné rozhranie na ovládanie.

6.2 Detekcia hracieho poľa

Prvým krokom pri spracovávaní sudoku je nájdenie hracieho poľa. Vzhľadom na rýchlosť a jednoduchosť ovládania som sa rozhodol pre detekciu z nasnímanej fotografie. Tú je potrebné vo väčšine prípadov zmenšiť. To z dôvodu, že veľké množstvo telefónov využíva

kameru s takým rozlíšením, ktoré je zbytočne veľké. Vďaka zmenšeniu sa značne urýchli doba spracovania fotky a stále sa zachová taká kvalita, aká je potrebná na rozpoznanie sudoku. Ďalej je potrebné rozpoznať hrany zadania. Aby bolo toto rozpoznávanie čo najúspešnejšie, je potrebné obraz predspracovať tak, aby sa odstránil šum. To som sa rozhodol doplniť ešte o prahovanie a obrázok tak obsahuje len najnutnejšie informácie.

Hranový detektor následne zdetekuje všetky hrany. Z nich sa vyberú také, ktoré spolu vytvárajú objekt s najväčšou plochou. Ten by mal obsahovať hľadané hracie pole. Rohy tohto objektu by teda mali označovať rohy hracieho poľa. Pomocou nich je možné z celej fotografie získať polohu zadania a odstrániť z fotografie iné nepotrebné časti. Tieto body tiež umožňujú pomocou transformácie upraviť obraz tak, aby sa odstránili rôzne deformácie. Tie môžu vzniknúť napríklad pri skenovaní zadania z rôznych uhlov alebo pri pokrivenom zadaní.

6.3 Rozpoznávanie čísel

Na rozpoznávanie čísel bola navrhnutá konvolučná neurónová sieť, ktorá vychádza z architektúry LeNet-5 (viď sekcia 4.4). Sieť bola upravená tak, že obsahuje 6 konvolučných vrstiev a 2 plne prepojené vrstvy. Všetky vrstvy okrem poslednej výstupnej využívajú aktivačnú funkciu ReLU. Výstupná vrstva využíva funkciu softmax. Upravená sieť taktiež používa dropout, čím redukuje pretrénovanie siete. Vstupom tejto siete sú obrázky s rozmerom 28x28 pixelov. Tieto obrázky sa získavajú z detekovaného hracieho poľa. To sa rozdelí podľa rozmerov na 9x9 políčok. Vzhľadom na to, že takto získané políčko môže okrem čísla obsahovať aj samotné ohraničenie políčka, je nutné ešte orezať okraje obrázku. Na určovanie či sa na políčku nachádza nejaké číslo som sa rozhodol využiť počet tmavých pixelov. Obrázok je vložený na rozpoznávanie do neurónovej siete vtedy, ak množstvo čiernych pixelov prekoná nastavenú prahovú hodnotu.

6.4 Riešenie sudoku

Algoritmus riešenia sudoku bolo potrebné navrhnúť tak, aby zvládol správne vyriešiť každé zadanie. Dôležitým faktorom bola taktiež doba, za ktorú zvládne algoritmus nájsť riešenie. V prvom návrhu aplikácie slúžil na nájdenie riešenia algoritmus backtracking, ktorý je popísaný v sekcii 2.1.

Ako sa ale ukázalo po testovaní, ktoré podrobnejšie rozoberá kapitola 8, v niektorých prípadoch algoritmus nezvládol nájsť riešenie v dostatočne krátkej dobe. To spôsobilo efekt zamrznutia a celkovú nefunkčnosť aplikácie. Z toho dôvodu bola do aplikácie doplnená eliminačná metóda naked single, ale aj analýza rozloženia hracieho poľa a možných kandidátov. Vďaka eliminačnej metóde sa zmenšil počet možných kandidátov. Analýza rozloženia hracieho poľa a možných kandidátov zase pomáha určiť lepšiu štartovaciu pozíciu pre backtracking. Analýza spočíva v tom, že na základe počtu kandidátov a na základe počtu vyplnených polí vo vybraných oblastiach určí takú pozíciu, kde bude mať algoritmus menší počet možností. To spôsobí menšie vetvenie, a tým sa urýchli vyhľadávanie riešenia.

6.5 Databáza hier

Pri návrhu aplikácie bola taktiež vytvorená databáza hier. Táto databáza uchováva každé naskenované zadanie sudoku, ktoré užívateľ hral. Vďaka tomu sa užívateľ môže kedykoľ-

vek vrátiť k rozohranej hre sudoku. Aplikácia tiež umožňuje odstrániť hru, ktorú vyberie užívateľ. Samotná databáza sa skladá z:

- unikátny identifikátor hry,
- časový údaj o tom, ako dlho užívateľ riešil sudoku,
- naskenované zadanie sudoku,
- aktuálny stav riešenia sudoku,
- celé riešenie pre naskenované zadanie,
- cesta k obrázku s naskenovaným zadaním,
- dátum a čas kedy bolo sudoku naposledy uložené.

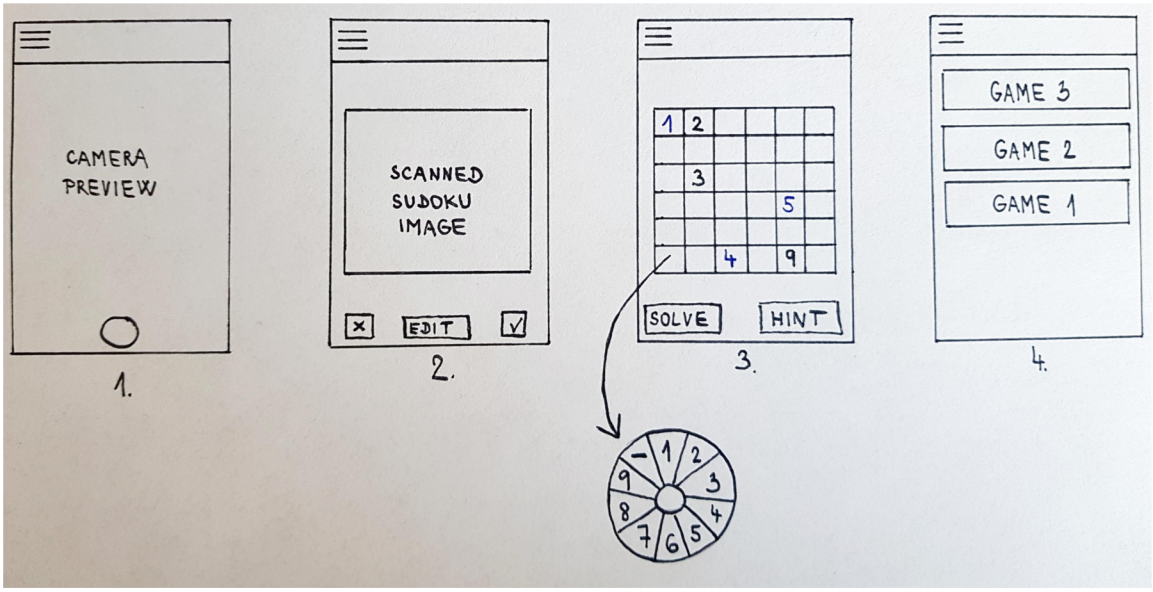
Databáza neuchováva veľké množstvo dát, a preto je uložená v internej pamäti zariadenia.

6.6 Grafické rozhranie

Pri navrhovaní grafického rozhrania je potrebné sa zamerať hlavne na jednoduchosť a intuitívnosť, no netreba taktiež zabúdať na estetickosť. Zlé ovládanie, neschopnosť porozumieť aplikácii, ale aj nemoderný dizajn môže odplašiť užívateľa. Cieľom je teda navrhnúť takú aplikáciu, ktorá bude nie len plne funkčná, ale aj vzhľadovo lákavá.

6.6.1 Prvotný návrh

V prvotnom návrhu, ktorý zobrazuje obrázok 6.5 sa pri spustení aplikácie spustil fotoaparát a užívateľ mohol hneď skenovať zadanie. Po naskenovaní zadania sa zobrazilo okno, kde mohol užívateľ spustiť úpravu rozpoznaného zadania v prípade chyby, zatvoriť okno a skenovať znovu alebo potvrdiť rozpoznané zadanie. Po potvrdení sa zobrazila aktivita na riešenie sudoku. Tá obsahovala samotné hracie pole, tlačidlo na vyriešenie celého sudoku a tlačidlo na nápovedu, ktorá vyplní zvolené pole. Vkladanie číslic do sudoku malo fungovať tak, že pri dlhšom stlačení prázdneho políčka sa zobrazí kruh s možnými číslami a následným potiahnutím k číslu sa potvrdí výber. Na navigáciu v aplikácii malo slúžiť menu. Pomocou neho by sa bolo možné dostať na aktivitu s uloženými hrami.



Obr. 6.5: Prvotný návrh aplikácie

Tento návrh zachytáva základnú funkcionalitu aplikácie, avšak stále obsahuje isté nedokonalosti. Vďaka iteratívnym úpravám boli tieto nedokonalosti nájdené a opravené tak, aby sa výsledná aplikácia čo najviac približovala stanoveným cieľom.

6.6.2 Iteratívne úpravy

Iteratívne úpravy boli robené v dvoch fázach. V prvej fáze som sa sám vžil do role náročného kritického užívateľa a snažil som sa v každej časti aplikácie nájsť všetko, čo by mohlo byť náročné, ťažko pochopiteľné, nejasné alebo dokonca nepotrebné. Keďže je však takáto kritika veľmi subjektívna, je nutná aj druhá fáza, kde spätnú väzbu poskytnú viacerí testujúci užívatelia. O tom, ako prebiehalo samotné testovanie, hovorí kapitola 8.

Po prvej fáze nastalo v aplikácii niekoľko významných zmien. Pri spustení aplikácie sa otvorí aktivita s uloženými hrami. V prípade, ak chce užívateľ spustiť už uloženú hru, jednoducho klikne na položku v zozname a otvorí sa aktivita riešenia. V prípade potreby zmazania uloženej hry stačí potiahnuť položku smerom doľava. Položka obsahuje základné informácie o hre a to tieto: počet správne vyplnených políčok, doba hrania a obrázok skenu zadania. Na spustenie aktivity, ktorá skenuje nové sudoku vzniklo nové tlačidlo v pravom dolnom rohu. Užívateľ môže naďalej upraviť prípadné chyby, no teraz to môže urobiť hneď a nemusí otvárať ďalšie okno. Po naskenovaní a prípadnej úprave zadania nasleduje aktivita riešenia sudoku. V nej sa zmenil spôsob vypĺňania čísel. Tlačidlá s číslami sa presunuli pod hraciu plochu a umožňujú tak rýchlejšie zadávať hodnoty. Taktiež pribudli nové tlačidlá na overenie správnosti vyplneného sudoku a na priebežné ukladanie. Tlačidlo na vyriešenie celého zadania bolo odstránené. Hra sa automaticky uloží aj pri zatvorení aktivity. Taktiež bol pridaný časovač, ktorý ukazuje ako dlho už užívateľ rieši sudoku. Prípadná kolízia v hracom poli sa zobrazí až po kliknutí na jedno z čísel, ktoré sú v kolízii.

Na základe druhej fázy už nepribudlo také veľké množstvo zmien a jednalo sa skôr o menšie úpravy. Pribudol úvodný text, ktorý napovedá ako začať skenovať sudoku. Text sa zobrazuje iba v prípade, ak aplikácia nemá pri spustení uloženú žiadnu hru. V zozname uložených hier ďalej pribudla informácia, kedy bola hra naposledy uložená. Podľa toho sú aj

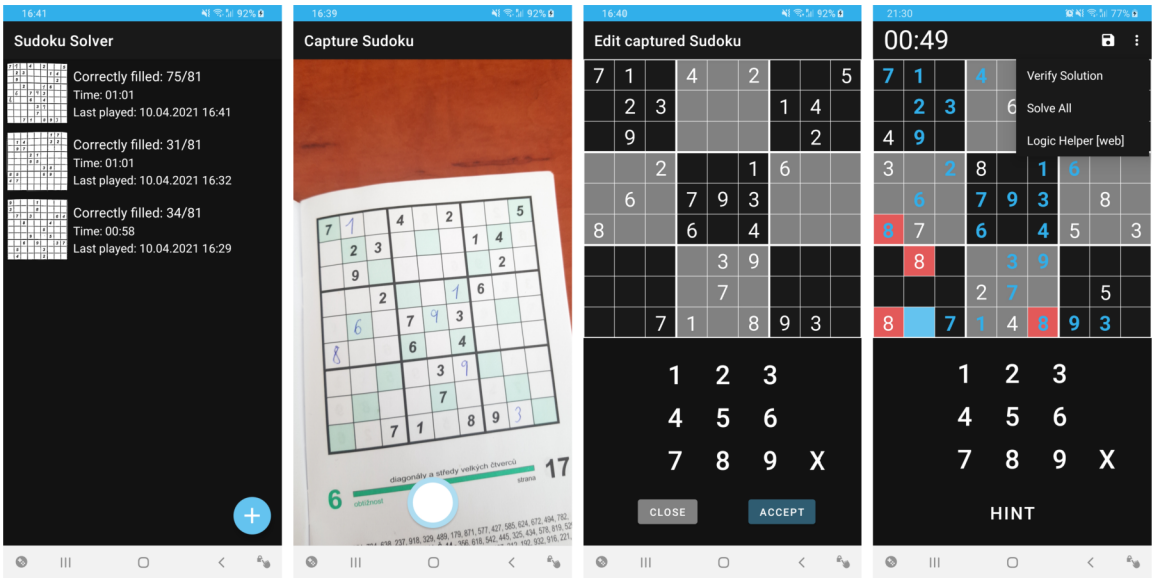
hry v zozname zoradené. Najviac zmien pribudlo v aktivite riešenia sudoku. Opäť pribudlo tlačidlo na vyriešenie celého sudoku, no tentokrát sa nachádza hore v rozbaľovacom menu. Do tohto menu sa presunulo aj tlačidlo na overovanie správnosti vyplneného sudoku. Posledná zmena sa týkala toho, kedy sa zobrazujú kolízie. Tie sa už nezobrazujú len po kliknutí na číslo v kolízii. Po úprave sa kolízia zobrazuje vždy, až do jej odstránenia.

6.6.3 Finálny vzhľad aplikácie

Výsledná aplikácia sa teda skladá z troch aktivít, ktoré zachytáva obrázok 6.6. Prvá zobrazuje uložené hry, druhá umožňuje skenovanie a úpravu sudoku a posledná obsahuje samotné riešenie sudoku. Farby aplikácie sa automaticky prispôsobujú na základe toho aký režim má užívateľ nastavený v zariadení. Pri dennom režime je aplikácia ladená do bielej farby, a naopak pri nočnom režime do čierna. Táto funkcionalita je v moderných aplikáciách čoraz viac využívaná a podporuje moderný dizajn.

Aktivita, ktorá zabezpečuje skenovanie a úpravu skenovaného zadania nie je rozdelená do dvoch aktivít z dôvodu rýchlosti. Spustenie fotoaparátu, ktorý táto aktivita využíva, je relatívne pomalý proces. V prípade rozdelenia tejto aktivity do dvoch aktivít by sa fotoaparát musel otvárať vždy znova, ak by užívateľ nebol spokojný so skenovaním. To by zbytočne zabralo veľa času. Práve preto okno určené na úpravu len prekryje výstup z fotoaparátu a až keď je užívateľ spokojný so skenom, tak sa aktivita ukončí a fotoaparát zavrie.

Rozbaľovacie menu v aktivite riešenia sudoku bolo doplnené o tlačidlo, pomocou ktorého sa odošle práve riešené sudoku na webovú stránku Sudoku Helper[6] [21], ktorá obsahuje program na hľadanie riešenia logickou cestou.



Obr. 6.6: Výsledný vzhľad aplikácie v nočnom režime. Druhý a tretí obrázok sú súčasťou rovnakej aktivity.

[6] https://hojkas-sudoku-helper.herokuapp.com/

Kapitola 7

Implementácia

Mobilná aplikácia bola implementovaná vo vývojovom prostredí Android Studio[1] a v programovacom jazyku Kotlin. Výsledná aplikácia funguje na operačnom systéme Android od verzie 5.0 a vyššie. To znamená, že funguje na 94,1 % zariadení s operačným systémom Android (viď obrázok 7.1).

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,8%
4.2 Jelly Bean	17	99,2%
4.3 Jelly Bean	18	98,4%
4.4 KitKat	19	98,1%
5.0 Lollipop	21	94,1%
5.1 Lollipop	22	92,3%
6.0 Marshmallow	23	84,9%
7.0 Nougat	24	73,7%
7.1 Nougat	25	66,2%
8.0 Oreo	26	60,8%
8.1 Oreo	27	53,5%
9.0 Pie	28	39,5%
10. Android 10	29	8,2%

Obr. 7.1: Percento zariadení, na ktorých bude aplikácia fungovať pre vybranú verziu Androidu[2]

[1] https://developer.android.com/studio

[2] Obrázok bol prevzatý z aplikácie Android Studio

Skripty potrebné na tvorbu datasetu a trénovanie neurónovej siete boli implementované v programovacom jazyku Python[3]. Na prácu s obrazom bola využitá knižnica OpenCV[4], ktorá obsahuje všetky potrebné funkcie k úprave obrazu. Práca s neurónovou sieťou bola realizovaná pomocou knižnice TensorFlow[5]. Na tvorbu databázy bola použitá knižnica Room[6].

7.1 Fotoaparát

Základným prvkom aplikácie je fotoaparát. Ten je implementovaný pomocou knižnice CameraX[7]. Tá je aktuálne ešte len v beta verzii, no už teraz je stabilná a zvláda základnú funkcionalitu. Knižnica značne zjednodušuje implementáciu fotoaparátu a je kompatibilná s verziou Androidu 5.0 a vyššou. Pri implementácii bola využitá ukážka z dokumentácie knižnice[8]. Na to, aby bolo možné využívať v aplikácii fotoaparát, je potrebné získať povolenie od užívateľa. Pri prvom spustení aplikácie a otvorení aktivity s fotoaparátom je užívateľ o toto povolenie požiadaný. Snímok získaný z fotoaparátu je uložený do pamäte zariadenia a po jeho spracovaní je odstránený.

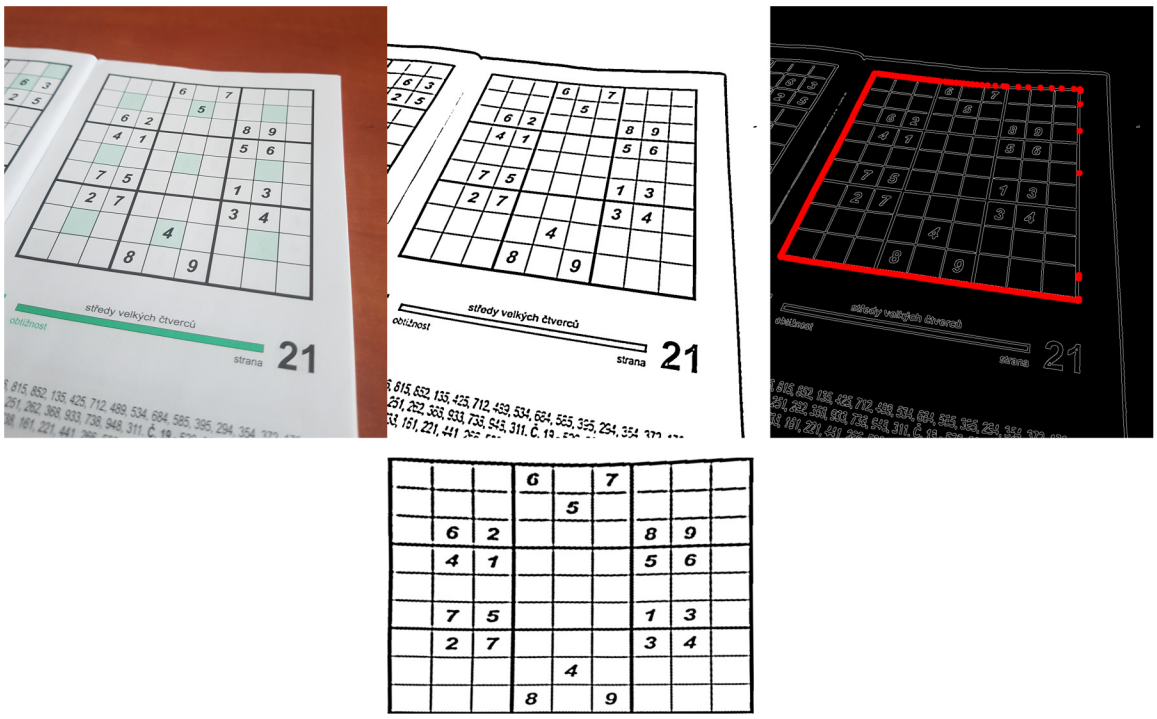
7.2 Detekcia hracieho poľa

Zachytená snímka je upravená na obrázok so šírkou 1 000 pixelov a takou výškou, ktorá sa dopočíta podľa pomeru strán pôvodného obrázku. Následne sa aplikuje Gaussovo rozostrenie s jadrom o veľkosti 9x9 pixelov a nulovou smerodajnou odchýlkou. Ďalším krokom je adaptívne prahovanie, ktoré využíva ako prahovú hodnotu priemer okolia pixelu (parameter `ADAPTIVE_THRESH_MEAN_C`). Toto okolie má rozmery 11x11 pixelov. Ďalej je potrebné určiť konštantu, ktorá sa využíva pri výpočte prahovej hodnoty. Tá je nastavená na hodnotu 4. Tieto hodnoty boli zvolené tak, aby výsledný obraz obsahoval čo najmenej šumu, no zároveň zachoval čo najviac potrebných informácií. Ďalej boli tieto aj ďalšie hodnoty parametrov funkcii určené na základe experimentov s rôznymi konštantami. Z týchto experimentov boli vybrané tie, ktoré poskytovali najideálnejšie výsledky.

Detekciu hrán zabezpečuje Cannyho hranový detektor. Tu je potrebné nastaviť hodnoty hysterézie. Dolná prahová hodnota je nastavená na 100 a horná na 300. Veľkosť Sobelovho filtra je ponechaná na prednastavenej hodnote 3x3. Z výsledných hrán sú pomocou funkcie `findContours` získané vonkajšie obrysy (parameter `RETR_EXTERNAL`) nájdených kriviek. Ďalší parameter (`CHAIN_APPROX_SIMPLE`) zaistí, že sa uložia len tie body krivky, ktoré nie sú nadbytočné. Z nich je vybraná tá s najväčšou plochou. Za predpokladu, že táto krivka ohraničuje hľadané hracie pole, by mala mať obdĺžnikový tvar. Aproximáciou tejto krivky je teda možné získať body, ktoré definujú rohy obdĺžnika a teda hracieho poľa. Na aproximáciu je použitá funkcia `approxPolyDP`, kde parameter epsilon je hodnota obvodu získanej krivky vynásobená hodnotou 0,03. V prípade, ak ani po aproximácii nie je počet získaných bodov redukovaný na 4, je užívateľ požiadaný o opätovné skenovanie zadania. Po získaní presne štyroch bodov sú tieto body zoradené na základe vzdialenosti od pixelu s nulovými súradnicami. Vzhľadom na to, že hracie pole môže byť snímané pod rôznymi uhlami, je potrebné snímku vhodne transformovať. Výpočet matice, na základe ktorej je vykonaná

[3] https://www.python.org/
[4] https://opencv.org/
[5] https://www.tensorflow.org/
[6] https://developer.android.com/training/data-storage/room
[7] https://developer.android.com/training/camerax
[8] https://codelabs.developers.google.com/codelabs/camerax-getting-started

transformácia, zabezpečuje funkcia `getPerspectiveTransform`. Táto matica sa ďalej použije vo funkcii `warpPerspective`, ktorá vykoná samotnú transformáciu obrazu. Výsledkom je transformovaný obrázok, ktorý obsahuje len samotné hracie pole. Detekciu hracieho poľa zachytáva obrázok 7.2. Nakoniec sa tento modifikovaný obrázok uloží do vnútornej pamäte a slúži ako náhľad pri uloženej hre. Taktiež sa odstráni pôvodne zachytená snímka.



Obr. 7.2: Zobrazenie jednotlivých krokov spracovania obrazu, ktoré sú potrebné pri detekcii hracieho poľa sudoku

7.3 Rozpoznávanie čísel

Po získaní hracieho poľa je pole rozdelené na 81 políčok. Rozmery každého políčka sú upravené na pevnú veľkosť a to 28x28 pixelov. Ďalšia funkcia zafarbí na bielo také pixely, ktoré sú od okraja vzdialené menej ako 5 pixelov. Tým sa odstráni väčšina pozostatkov po ohraničení políčka. Na určenie, či sa na políčku nachádza nejaké číslo, bola implementovaná funkcia `fieldContainNumber`. Tá spočíta, koľko pixelov je tmavších ako prahová hodnota 127. Ak je takýchto pixelov viac ako 18, tak je políčko poslané do neurónovej siete na rozpoznanie. Tieto rozpoznané čísla sú vložené do výsledného dvojrozmerného poľa, pomocou ktorého je inicializované hracie pole.

7.3.1 Trénovanie neurónovej siete

Konvolučná neurónová sieť bola trénovaná na datasete, ktorý je popísaný v sekcii 4.5.2. Tento dataset bol rozdelený a posledných 4 000 vzoriek bolo využitých ako testovací set. Počet iterácií (epoch) trénovania bol nastavený na 1 000. Trénovanie prebiehalo na grafickej karte Nvidia GeForce GTX 1660 a trvalo približne 2 hodiny a 20 minút. Výsledná pres-

nosť neurónovej siete dosahuje 99,08 % na testovacom datasete. Sieť bola následne uložená a konvertovaná na verziu TensorFlow Lite[8], ktorá je zameraná na využitie v mobilných zariadeniach.

7.4 Riešenie sudoku

Hľadanie riešenia zadaného sudoku sa spustí v momente, keď užívateľ potvrdí rozpoznané alebo upravené sudoku. V prvom kroku sa vygeneruje množina všetkých možných kandidátov pre každé nevyplnené pole pomocou funkcie `generateCandidates`. Generovanie tejto množiny prebieha dovtedy, dokiaľ sa novo vygenerovaná množina nebude l od tej predchádzajúcej. V prípade, ak sa množina nelíši, tak to znamená, že už nebolo možné doplniť žiadne ďalšie číslo pomocou eliminačnej metódy. Vznikla tak rovnaká množina kandidátov ako tá pôvodná. Číslo sa doplňuje vtedy, keď sú splnené všetky podmienky eliminačnej metódy naked single. To znamená, že sa nájde pre nejaké políčko taká množina, ktorá obsahuje len jedného kandidáta. Potom môže byť tento kandidát doplnený do políčka. Funkcia `eliminateAndAnalyzeCandidates`, ktorá zabezpečuje elimináciu kandidátov taktiež analyzuje hracie pole a kandidátov. Analýza spočíva v dvoch krokoch. V prvom kroku sa z prvého (horného) a posledného (spodného) riadku hracieho poľa určí také políčko, ktoré má najmenší počet možných kandidátov. Ak je toto políčko v hornom riadku, tak je algoritmus backtracking spustený v smere zhora dole. V opačnom prípade smeruje zdola hore. V situácii, kedy je minimálny počet kandidátov zhodný, rozhoduje počet vyplnených políčok. Množstvo vyplnených políčok sa počíta v horných a dolných troch riadkoch. Tentokrát hľadáme čo najväčšie množstvo vyplnených polí. Štartovacia pozícia pre backtracking je vybraná na základe toho, v ktorej časti hracieho poľa je vyplnených najviac políčok. Ak je počet vyplnených políčok rovnaký, tak je algoritmus backtracking spustený v smere zdola. V ostatných prípadoch je štartovacia pozícia v hornom riadku. To, ako je toto riešenie spoľahlivé a rýchle, popisuje sekcia 8.4.

7.5 Databáza hier

Knižnica Room umožňuje veľmi jednoducho implementovať databázu v mobilnej aplikácii. Pri tvorbe databázy je potrebné vytvoriť niekoľko základných tried, z ktorých sa samotná databáza skladá. Tieto triedy je potrebné označiť anotáciou. Prvá trieda, ktorú je potrebné vytvoriť definuje samotnú databázu a je potrebné ju označiť anotáciou `@Database`. Pomocou tejto triedy je možné nastaviť parametre databázy a pristupovať k uloženým dátam. Trieda označená ako `@Entity` reprezentuje tabuľku v databáze. V tejto triede možno definovať aké hodnoty budú uložené v databáze. V tejto aplikácii bola implementovaná len jedna tabuľka a obsahuje údaje, ktoré boli popísané v časti 6.5. Keďže knižnica Room umožňuje uchovávať len jednoduché dátové typy, bolo nutné implementovať konvertor. Konvertor konvertuje z dátového typu `ArrayList<Int>` na typ `String`. Využitie konvertorov sa značí anotáciou `@TypeConverters` pri triede, ktorá reprezentuje tabuľku. Nakoniec je potrebné implementovať rozhranie s anotáciou `@Dao`. Toto rozhranie slúži na definovanie operácii, ktoré je možné vykonávať nad databázou. Každá operácia je definovaná ako funkcia so špecifickou anotáciou `@Query()`, do ktorej je možné vpísať príkaz v jazyku SQL. Niektoré najzákladnejšie operácie implementuje samotná knižnica Room a majú svoju vlastnú anotáciu. Tieto ope-

[8] https://www.tensorflow.org/lite

rácie sú napríklad vkladanie (`@Insert`), aktualizácia (`@Update`) alebo vymazanie (`@Delete`) záznamu.

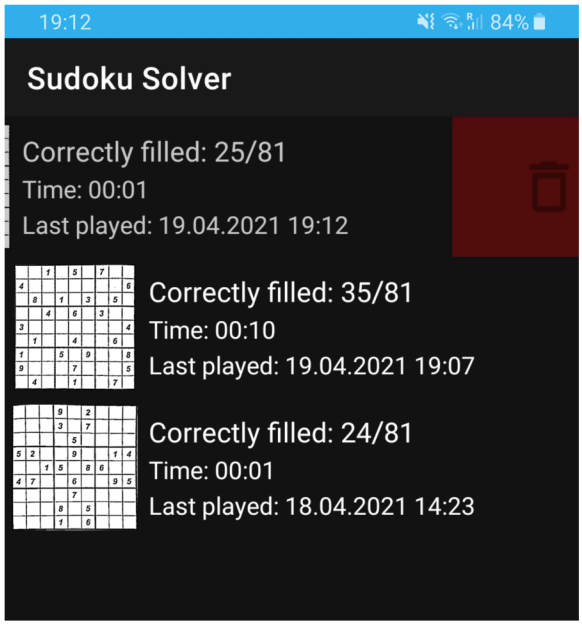
7.6 Grafické rozhranie

Grafické rozhranie bolo implementované na základe finálneho návrhu z časti 6.6.3. Pri implementácii boli využité hlavne základné komponenty, ktoré poskytuje Android, avšak pri tvorbe hracieho poľa bol vytvorený aj vlastný prvok užívateľského rozhrania. V nasledujúcich podkapitolách sú popísané tie časti grafického rozhrania, ktoré boli zložitejšie na implementáciu.

7.6.1 Zoznam uložených hier

Zoznam uložených hier je implementovaný pomocou komponentu RecyclerView, ktorý umožňuje zobrazovať dynamicky generovaný zoznam. Predchodcom tohoto komponentu je ListView.

Pri využívaní komponentu RecyclerView je nutné vykonať viacero krokov. Ako prvé si treba definovať vzhľad položiek v zozname v jazyku XML (súbor `row.xml`). Ďalej je potrebné implementovať adaptér a ViewHolder (obe položky sú implementované v triede `ListAdapter`). Tieto položky zabezpečujú správne zobrazenie dát. ViewHolder je zodpovedný za správne zobrazovanie individuálnych položiek a adaptér zasa za dáta, ktoré zobrazujú tieto položky. V tejto práci adaptér načítava údaje, ktoré sú uložené v databáze. V prípade, ak chce užívateľ zmazať uloženú hru, stačí potiahnuť položku doľava. Táto funkcionalita je implementovaná pomocou triedy `SwipeToDelete`, ktorá dedí vlastnosti z triedy `ItemTouchHelper`. RecyclerView a funkcionalitu mazania je vidieť na obrázku 7.3.



Obr. 7.3: RecyclerView s uloženými hrami sudoku, kde prvá uložená hra je vo fáze mazania (ťahanie položky doľava)

7.6.2 Hracie pole

Hracie pole bolo implementované so snahou využiť návrhový vzor Model-View-Viewmodel (MVVM). Vďaka tomu sa oddelila grafická a logická časť, čo značne sprehľadnilo implementáciu. Grafická stránka (View) je implementovaná v triede `SudokuBoardView`. Hracie pole je vykresľované pomocou triedy `Canvas`. Hracie pole sa skladá len z čiar, štvorcov a čísel (textu), čo je ideálne na využitie tejto triedy. Trieda `SudokuBoardView` taktiež získava súradnice z obrazovky a z nich následne prepočítava, na ktoré pole užívateľ klikol. Tieto získané súradnice políčka ďalej posiela triede `SudokuGame` (Model), ktorá už zabezpečí potrebnú logiku dopĺňania čísel. Táto implementácia hracieho poľa sa využíva okrem samotného riešenia sudoku aj pri editácii naskenovaného zadania.

Kapitola 8

Testovanie aplikácie

Testovanie prebiehalo v dvoch etapách. V prvej etape bola aplikácia testovaná užívateľmi. Cieľom bolo zistiť, ako rôzni užívatelia hodnotia aplikáciu, či už po grafickej alebo funkčnej stránke.

V druhej etape som aplikáciu testoval ja sám na väčšej vzorke zadaní. Snahou bolo overiť celkovú správnu funkcionalitu aplikácie, ako aj zistiť rôzne štatistické údaje o jednotlivých krokoch spracovávania sudoku. Toto testovanie prebiehalo na zariadení Samsung Galaxy S8+ s procesorom Exynos 8895 Octa a operačným systémom Android s verziou 9.0. Aplikácia bola testovaná na 150 rôznych zadaniach sudoku z viacerých časopisov. Tieto zadania boli doplnené o 7 ďalších zadaní z dokumentu *The most difficult Sudoku puzzles are quickly solved by a straightforward depth-first search algorithm* [10], ktorý analyzuje hľadanie riešení sudoku pomocou prehľadávania do hĺbky.

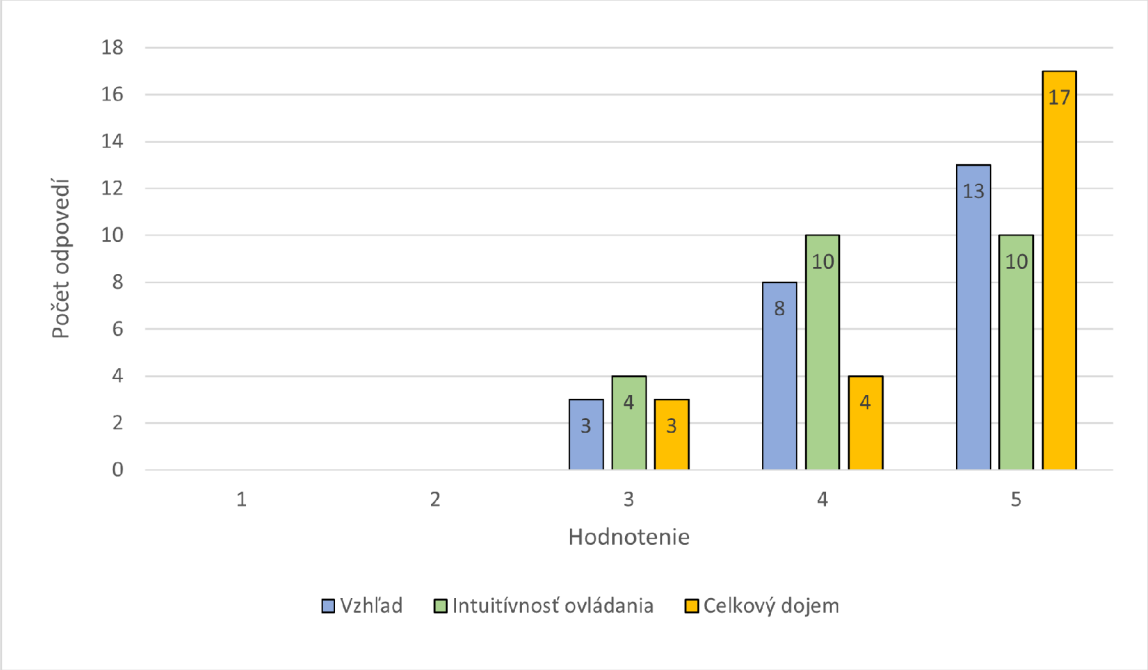
8.1 Užívateľské testovanie

Pri užívateľskom testovaní bol užívateľom buď poskytnutý mobilný telefón s predinštalovanou aplikáciou (bez predošlého spustenia) alebo si mohli aplikáciu stiahnuť z obchodu Google Play. Pri testovaní nemali zadané žiadne špecifické pravidlá a postupy, ako by malo testovanie prebiehať. Užívateľom bolo povedané len to, na čo aplikácia slúži. Po otestovaní mali užívatelia vyplniť dotazník z prílohy A. V dotazníku sa nachádzali otázky na ktoré bolo možné odpovedať pomocou:

- krátkej stupnice od 1 do 5, kde číslo 1 reprezentuje najhoršie hodnotenie a číslo 5 najlepšie,
- dlhej stupnice od 0 do 10, kde 0 reprezentuje 0 % a 10 reprezentuje 100 %,
- výberu zo zadaných možností,
- otvorenej odpovede.

Otázky boli zamerané na hodnotenie vzhľadu aplikácie, intuitívnosti ovládania či presnosti detekcie a rozpoznávania sudoku. Dotazník vyplnilo celkovo 24 respondentov.

Ako znázorňuje graf na obrázku 8.1 bol vzhľad aplikácie ohodnotený užívateľmi celkom pozitívne. Hodnotenie intuitívnosti dopadlo o trochu horšie, no celkový výsledok bol stále uspokojivý. Keďže bol ale v oboch prípadoch priestor na zlepšenie, bola aplikácia pozmenená tak, aby poskytla lepší užívateľský zážitok. Všetky tieto zmeny popisuje sekcia o iteratívnych úpravách 6.6.2.



Obr. 8.1: Odpovede respondentov na otázky s krátkou stupnicou

Detekcia hracieho poľa bola pri užívateľskom testovaní úspešná na 92,5 %. Detekcia podľa užívateľov zlyhávala hlavne v prípadoch, keď skenovali zadanie z príliš veľkej diaľky. Rozpoznávanie čísel bolo podľa hodnotení úspešné na 90 %. Užívatelia tiež popísali, že rozpoznávanie bolo problémové väčšinou vtedy, keď skúšali skenovať zadanie z rôznych veľmi šikmých uhlov.

Celkový dojem aplikácie však užívatelia ohodnotili veľmi pozitívne a boli spokojní s testovanou aplikáciou. Presný počet odpovedí ohľadom celkového dojmu je vidieť na obrázku 8.1. Aplikáciu by odporúčalo svojím známym 95,8 % respondentov, a teda len jeden respondent by aplikáciu neodporúčal. Tento respondent neodporúčal aplikáciu kvôli tomu, že aplikácia nepodporuje slovenský jazyk (podporovaným jazykom je angličtina) a on sám angličtinu neovláda. Z výsledkov dotazníka bolo zistené, že užívatelia nepokladajú za dôležité mať možnosť pomenovať uložené sudoku. Podľa získaných informácii by uložené zadania mali byť zoradené podľa času posledného hrania danej hry.

Posledná otázka s otvorenou odpoveďou, kde mohli respondenti napísať svoje pripomienky či nápady, priniesla okrem dobrých nápadov na zlepšenie aplikácie aj jednu zaujímavú poznámku. Jednému z užívateľov sa aplikácia zasekla v štádiu, keď hľadala riešenie pre naskenované sudoku. Na základe tejto odozvy boli implementované zmeny v algoritme, ktorý hľadá riešenie. Tieto zmeny popisuje sekcia 6.4.

8.2 Testovanie detekcie hracieho poľa

Pri testovaní, ako aplikácia zvláda detektovať hracie pole, neboli zistené žiadne väčšie problémy. Aplikácia správne detekovala hracie pole pod rôznymi uhlami a pri rôznom osvetlení. Aplikácia môže nesprávne rozpoznať sudoku v prípade, ak sa na obraze rozpozná iný väčší štvorec, no to nastáva pri skenovaní s bežnou vzdialenosťou medzi fotoaparátom a zadaním veľmi zriedkavo. Ako udáva tabuľka 8.1, detekcia trvala v priemere 334,57 ms.

8.3 Testovanie rozpoznávania čísel

Rozpoznávanie čísel dopadlo taktiež veľmi pozitívne. Zo 150 naskenovaných zadaní rozpoznala aplikácia chybne len 8 čísel. Na rozpoznávanie najviac vplývali 2 faktory a to: uhol skenovania zadania a typ písma (font), akým sú zadané číslice v zadaní. Sudoku skenované z príliš veľkého uhla začne strácať práve tie drobné detaily, ktoré definujú samotnú číslicu pri rozpoznávaní. Rôzne fonty a najmä hrubý alebo tučný font môžu podporiť práve toto vytrácanie detailu. Ako sa však pri testovaní ukázalo, aplikácia si aj s týmto dokázala vo väčšine prípadoch poradiť. Čo sa týka doby rozpoznávania, tak je táto časť časovo najviac náročná a ako uvádza tabuľka 8.1, priemerná doba rozpoznávania čísel bola 412,99 ms.

8.4 Testovanie schopnosti nájsť riešenie

Nájsť správne riešenie pre každé sudoku je jednou z najdôležitejších častí aplikácie. Algoritmy, ktoré riešia tento problém fungujú veľmi spoľahlivo a je dokázané, že dokážu nájsť riešenie. To potvrdzuje aj fakt, že všetkých 150 skenovaných zadaní bolo vyriešených správne. Práve preto sa táto časť zamerala skôr na testovanie toho, v akom čase je aplikácia schopná nájsť riešenie. Hľadanie riešenia pozostáva z eliminačnej metódy a backtrackingu. Pri skenovaní zadaní z rôznych novín a časopisov trvalo eliminovanie kandidátov priemerne 5,40 ms. Nasledujúci backtracking s analýzou hracieho poľa našiel po eliminácii riešenie priemerne za 23,11 ms. Backtracking našiel v najhoršom prípade riešenie za 618 ms. Presnejšie štatistické údaje sú uvedené v tabuľke 8.1.

Pri overovaní, ako si aplikácia poradí s tými najťažšími zadaniami, boli využité zadania sudoku z už spomenutého dokumentu [10]. Ako je možné vidieť v tabuľke 8.2, väčšina riešení bola získaná v krátkej alebo akceptovateľnej dobe. V najhoršom prípade bolo riešenie nájdené až po približne 11 sekundách, čo je síce dlhá doba, no nespôsobí to úplné zamrznutie aplikácie a tiež je potrebné myslieť na to, že sa jedná o zadania, ktoré majú čo najviac komplikovať nájdenie riešenia.

Tabuľka 8.1: Získané štatistické údaje o dobe trvania jednotlivých krokov pri práci so sudoku [ms]

	Detekcia hracieho poľa	**Rozpoznávanie čísel**	**Eliminačná metóda**	**Backtracking**
Priemer	334,57	412,99	5,40	23,11
Minimum	307,00	344,00	1,00	1,00
Maximum	410,00	687,00	35,00	618,00
Smerodajná odchýlka	11,21	43,23	4,87	71,59

Tabuľka 8.2: Doba [ms], za akú zvládnu jednotlivé metódy riešiť zadania sudoku z dokumentu *The most difficult Sudoku puzzles are quickly solved by a straightforward depth-first search algorithm* [10]

Označenie	**Eliminačná metóda**	**Backtracking**
A	7	11
B	3	2 565
B reversed	3	1 130
C	22	11 072
D	1	6 921
E	2	240
The most difficult	1	271

Kapitola 9

Záver

Cieľom tejto práce bolo vytvoriť mobilnú aplikáciu, ktorá zvládne naskenovať zadanie hry sudoku z novín a následne ho s pomocou nápovedy umožní dohrať.

Implementovaná aplikácia určená pre operačný systém Android umožňuje užívateľovi vytvoriť fotografiu zadania sudoku. Z tejto fotografie je pomocou základných techník spracovania obrazu rozpoznané hracie pole. V tomto poli sú následne rozpoznané čísla a to s využitím konvolučnej neurónovej siete. Tá je vytrénovaná s presnosťou 99,08 %. Po rozpoznaní čísel môže užívateľ ešte sudoku dodatočne upraviť a následne riešiť. Pri hľadaní riešenia môže v prípade potreby využiť nápovedu. Každé hrané sudoku je uložené a užívateľ sa k nemu môže kedykoľvek vrátiť a pokračovať v hraní.

Aplikácia prešla niekoľkými fázami testovania a v jednej z nich bola aplikácia testovaná aj užívateľmi. Na základe ich spätnej väzby bola aplikácia upravená, no celkovo aplikáciu hodnotili kladne. Finálna verzia bola zverejnená pod názvom *Sudoku Solver - Scanner*[1] v obchode Google Play, kde je zdarma k dispozícii na stiahnutie. Na základe týchto informácii je možné považovať zadaný cieľ práce za splnený.

Aplikáciu by bolo ešte možné v rôznych smeroch vylepšiť či doplniť o novú funkcionalitu. Príkladom takejto novej funkcionality môže byť generovanie rôznych nových zadaní sudoku. Užívateľ by si tak mohol zahrať sudoku aj v prípade, ak by nemal po ruke žiadne zadanie z novín alebo žiadnu uloženú hru. Ďalším rozšírením by mohla byť možnosť detekovať sudoku nielen z práve získanej snímky, ale aj z interného úložiska zariadenia či albumu. Užívateľ by si pri pridávaní nového zadania sám zvolil, či chce zadanie skenovať alebo len nahrať z pamäte. Do aplikácie by tiež mohla byť pridaná podpora pre rôzne jazyky.

[1] https://play.google.com/store/apps/details?id=com.androidjl.sudokusolver

Literatúra

[1] Budhiraja, A. *Dropout in (Deep) Machine learning* [online]. 2016 [cit. 2021-4-18]. Dostupné z: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5.

[2] Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1986, PAMI-8, č. 6, s. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[3] Cohen, G., Afshar, S., Tapson, J. a Schaik, A. EMNIST: an extension of MNIST to handwritten letters. *CoRR* [online]. 2017, abs/1702.05373. Dostupné z: http://arxiv.org/abs/1702.05373.

[4] Davis, T. *The Mathematics of Sudoku* [online]. September 2012 [cit. 2020-12-31]. Dostupné z: http://www.geometer.org/mathcircles/sudoku.pdf.

[5] Fisher, R., Perkins, S., Walker, A. a Wolfart, E. *Hough Transform* [online]. 2003 [cit. 2021-2-28]. Dostupné z: http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm.

[6] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Hobiger, B. *Hodoku* [online]. 2012 [cit. 2021-1-17]. Dostupné z: http://hodoku.sourceforge.net/en/techniques.php.

[8] Horký, L. a Břinda, K. *Neuronové sítě* [online]. [cit. 2021-3-13]. Dostupné z: http://buon.fjfi.cvut.cz/raws/FS/FyzSemBest/prezentace/neurony.pdf.

[9] Lecun, Y., Bottou, L., Bengio, Y. a Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, zv. 86, č. 11, s. 2278–2324. DOI: 10.1109/5.726791.

[10] Matos, A. *The most difficult Sudoku puzzles are quickly solved by a straightforward depth-first search algorithm* [online]. 2016 [cit. 2021-4-23]. Dostupné z: https://www.dcc.fc.up.pt/~acm/sudoku.pdf.

[11] Misra, S., Li, H. a He, J. *Machine Learning for Subsurface Characterization*. 1. vyd. Gulf Professional Publishing, 2019. ISBN 9780128177365.

[12] *THE MNIST DATABASE of handwritten digits* [online]. [cit. 2021-3-19]. Dostupné z: http://yann.lecun.com/exdb/mnist/.

[13] Nwankpa, C., Ijomah, W., Gachagan, A. a Marshall, S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *CoRR* [online]. 2018, abs/1811.03378. Dostupné z: https://arxiv.org/abs/1811.03378.

[14] OpenCV. *Canny Edge Detection* [online]. Február 2021 [cit. 2021-2-27]. Dostupné z: https://docs.opencv.org/master/da/d22/tutorial_py_canny.html.

[15] OpenCV. *Image Thresholding* [online]. Február 2021 [cit. 2021-2-6]. Dostupné z: https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html.

[16] OpenCV. *Morphological Transformations* [online]. Február 2021 [cit. 2021-2-7]. Dostupné z: https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html.

[17] Parker, J. R. *Algorithms for Image Processing and Computer Vision*. 2. vyd. Wiley, 2010. ISBN 9780470643853.

[18] ShareIT. *What is Kotlin and Why Do Mobile App Developers Love It?* [online]. 2021 [cit. 2021-4-19]. Dostupné z: https://www.shareitsolutions.com/blog/what-is-kotlin.

[19] Sonka, M., Hlavac, V. a Boyle, R. *Image Processing, Analysis, and Machine Vision*. 4. vyd. Cengage Learning, 2014. ISBN 9781133593607.

[20] StatCounter. *Mobile Operating System Market Share Worldwide - March 2021* [online]. 2021 [cit. 2021-4-19]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[21] Strnadová, I. *Výukový program pro hraní netradičních forem hry SUDOKU*. Brno, 2021. Bakalárska práca. Vysoké učení technické v Brně, Fakulta informačních technologií.

[22] Sumithra, K., Buvana, S. a Somasundaram, R. A Survey on Various Types of Image Processing Technique. *International Journal of Engineering Research & Technology (IJERT)* [online]. Marec 2015, zv. 4, č. 3. ISSN 2278-0181. Dostupné z: https://www.ijert.org/research/a-survey-on-various-types-of-image-processing-technique-IJERTV4IS030552.pdf.

[23] Tutorialspoint. *Android - Overview* [online]. 2021 [cit. 2021-4-19]. Dostupné z: https://www.tutorialspoint.com/android/android_overview.htm.

[24] Volná, E. *Neuronové sítě 1* [online]. 2002 [cit. 2021-3-13]. Dostupné z: https://files.klaska.net/sites/files.klaska.net/files/manual_files/cvut/Neuronove%20site%201/volna.pdf.

[25] Šikudová, E. *Počítačové videnie: detekcia a rozpoznávanie objektov*. 1. vyd. Wikina, 2014. ISBN 9788087925065.

[26] Španěl, M. a Milet, T. *Redukce barevného prostoru* [online]. 2019 [cit. 2021-2-6]. Dostupné z: https://www.fit.vutbr.cz/study/courses/IZG/private/lecture/izg_slide_omezeni_barev_print_rev2019.pdf.

Príloha A

Dotazník na získanie spätnej väzby

Sudoku Solver - Spätná väzba

Tento dotazník slúži na získanie spätnej väzby k aplikácii Sudoku Solver (https://play.google.com/store/apps/details?id=com.androidjl.sudokusolver), ktorá bola vytvorená na základe zadania mojej bakalárskej práce. Výsledky tohto dotazníku budú použité na prípadnú úpravu aplikácie a budú taktiež spomenuté v samotnej práci. Za každú jednu odpoveď a za strávený čas vopred ďakujem. V prípade akýchkoľvek otázok či nejasností ma prosím kontaktujte pomocou emailu: lazorik.juraj@gmail.com.

* Povinné

1. Váš vek: *

Označte iba jednu elipsu.

- 13 a menej
- 14 - 17
- 18 - 25
- 26 - 35
- 36 - 55
- viac ako 55

2. Ako hodnotíte vzhľad aplikácie? *

Označte iba jednu elipsu.

	1	2	3	4	5	
Nepáči sa mi						Páči sa mi

3. Ako hodnotíte intuitívnosť ovládania aplikácie? *

Označte iba jednu elipsu.

	1	2	3	4	5	
Málo intuitívne	◯	◯	◯	◯	◯	Veľmi intuitívne

4. Na koľko percent bola detekcia hracieho poľa úspešná? *

Označte iba jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
0%	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	100%

5. Na koľko percent bolo rozpoznávanie čísel zo zadania úspešne? *

Označte iba jednu elipsu.

	0	1	2	3	4	5	6	7	8	9	10	
0%	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	100%

6. Je pre vás potrebné mať možnosť pomenovať uložené sudoku? *

Označte iba jednu elipsu.

- ◯ Áno
- ◯ Nie

7. Na základe čoho by mali byť zoradené uložené zadania sudoku? *

Označte iba jednu elipsu.

- ◯ Názov
- ◯ Čas posledného hrania
- ◯ Počet vyplnených políčok
- ◯ Dĺžka riešenia
- ◯ Iné: _____

8. Ako na vás pôsobí aplikácia ako celok? *

Označte iba jednu elipsu.

	1	2	3	4	5	
Veľmi zle	◯	◯	◯	◯	◯	Veľmi dobre

9. Budete aplikáciu aj naďalej používať? *

Označte iba jednu elipsu.

- ◯ Áno
- ◯ Nie

10. Odporúčili by ste aplikáciu svojim známym? *

Označte iba jednu elipsu.

- ◯ Áno
- ◯ Nie

11. Sem môžete napísať vaše postrehy, pripomienky, požiadavky a iné...

Ďakujem za vyplnenie dotazníka a váš čas.

Príloha B

Obsah CD

Priložené CD obsahuje:

```
/
├── DatasetImages/ - získané obrázky z ktorých sa skladá vytvorený dataset
├── Documentation-src/ - zdrojové súbory textovej časti tejto práce
├── NeuralNetwork-src/ - zdrojové súbory pracujúce s neurónovou sieťou
├── ProcessDataset-src/ - zdrojové súbory na spracovanie datasetu
├── app-release.apk - inštalačný súbor aplikácie
├── NeuralNetworkFinal.tflite - model vytrénovanej neurónovej siete
├── readme.md - užívateľský manuál na spustenie aplikácie
├── SudokuSolver-src.zip - zdrojové súbory mobilnej aplikácie
└── xlazor02-BP.pdf - textová časť tejto bakalárskej práce
```