Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Katedra softwarových technologií

Aplikace strojového učení pro mobilní zařízení

Autor diplomové práce: Dominik Šimáček

Vedoucí diplomové práce: Ing. Jan Panuš, Ph.D.

Rok obhajoby: 2023

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Dominik Šimáček**
Osobní číslo:	**I20219**
Studijní program:	**N0613A140007 Informační technologie**
Téma práce:	**Aplikace strojového učení pro mobilní zařízení**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem práce je vytvořit aplikaci pro mobilní zařízení s OS Android s využitím Neural Processing SDK od firmy Qualcomm, který je určen pro vývoj softwaru umožňující spouštět natrénované neuronové sítě na zařízeních, které potřebují spojení do cloudových úložišť. SDK umožňuje navrhnout aplikaci, která umožní spuštění i více neuronových sítí. V rámci diplomové práci dojde k nastudování dokumentace k daného SDK, dále bude vytvořena aplikace, ve které bude právě tento SDK využit a která bude umět zpracovat nahrané nebo vyfocené digitální obrázky pomocí neuronových sítí. Aplikace tak bude umět například detekci objektů na obrázku, popř. jejich klasifikaci. Předpokládá se znalost programovacího jazyka JAVA nebo Kotlin.

Rozsah pracovní zprávy: **50-60 stran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

Qualcomm Neural Processing SDK for AI. *Qualcomm.com* [online]. [cit. 2021-10-4]. Dostupné z: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk
GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. *Deep learning*. Cambridge: MIT Press, [2016]. Adaptive computation and machine learning (MIT Press). ISBN 978-026-2035-613.
TUČKOVÁ, Jana. *Úvod do teorie a aplikací umělých neuronových sítí*. Praha: Vydavatelství ČVUT, 2003. ISBN 80-010-2800-3.

Vedoucí diplomové práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **8. listopadu 2021**
Termín odevzdání diplomové práce: **20. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

prof. Ing. Antonín Kavička, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 30. listopadu 2021

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 11. 5. 2023

Dominik Šimáček v. r.

PODĚKOVÁNÍ

Poděkování patří všem, kdo mě podporovali a poskytli mi pomoc při zpracování této práce.

ANOTACE

Tato práce představí klíčové pojmy z oblasti strojového učení, počítačového vidění, konvolučních neuronových sítí, práce s Neural Processing SDK a technologiemi, které jsou pro tuto práci potřeba. Tyto pojmy budou popsány a vysvětleny. Dále v této práci bude pomocí ukázkové aplikace předvedena a popsána práce s Neural Processing SDK od firmy Qualcomm. Tato aplikace bude vyvinuta pro zařízení s OS Android a procesory Snapdragon. Aplikace bude pomocí předem natrénovaných neuronových sítí umět zpracovat nahrané digitální obrázky. Pomocí strojového učení bude v digitálním obrázku provedena detekce objektů a následně jejich klasifikace.

KLÍČOVÁ SLOVA

Mobilní telefon, neuronové sítě, Android, Snapdragon, Qualcomm, SNPE, strojové učení

TITTLE

Machine learning application for mobile devices

ANNOTATION

This thesis will introduce the key concepts of machine learning, computer vision, convolutional neural networks, Qualcomm's Neural Processing SDK and the technologies needed for this thesis. These terms will be described and explained. Further in this work, working with Qualcomm's Neural Processing SDK will be demonstrated on a sample application. This application will be developed for devices with Android OS and Snapdragon processors. This application will use pre-trained neural networks to process images uploaded to the device. Using machine learning, this application will classify a digital image.

KEYWORDS

Mobile device, neural network, Android, Snapdragon, Qualcomm, SNPE, machine learning

Obsah

Úvod

V posledních letech se umělá inteligence a strojové učení staly velmi slibnými a populárními technologiemi a našly využití v různých průmyslových odvětvích. Díky strojovému učení je možné, aby se programy učily a zlepšovaly se, a to přináší velkou výhodu a nové možnosti oproti klasickému přístupu, kde vše bylo explicitně naprogramováno. Díky tomu jsou tyto aplikace ideální pro použití při identifikaci a rozpoznávání obrázků a řeči, zpracování přirozeného jazyka a predikcích. V dnešní době poptávka právě po takových aplikacích, které umožnují tuto flexibilitu a nabízejí tyto možnosti roste a kvůli tomu, roste i poptávka po nástrojích, které by umožňovaly vytváření těchto aplikací a efektivní a výkonnou integraci neuronových sítí do těchto aplikací.

Qualcomm Snpadragon Neural Processing Engine (SNPE) SDK, umožňuje vývojářům používat neuronové sítě na zařízeních, které mají Snapdragon procesor. Díky SNPE lze tyto modely optimalizovat a zefektivnit jejich výkonnost, a to bez nutnosti použití cloudových nebo serverových technologií. Aplikace nepotřebuje žádné připojení do internetu a díky tomu se sníží latence a aplikace není závislá na internetovém připojení. Tento framework také umožňuje využít speciálního hardware od firmy Qualcomm a optimalizovat model neuronové sítě pro lepší a rychlejší běh na tomto hardware. Díky tomu může aplikace dosahovat přesvědčivého výkonu i na slabších procesorech a zároveň šetřit baterii.

V rámci této práce budou nejprve představeny a popsány principy strojového učení a počítačového vidění, které jsou důležité pro pochopení, jak se počítač může učit a jak fungují neuronové sítě. Dále budou představeny a popsány konvoluční neuronové sítě. Tyto sítě jsou využívány právě pro zpracování obrazových dat, proto jsou v rámci této práce blíže popsány. Následně budou popsány principy a funkčnost Qualcomm SNPE. Budou diskutovány výhody a nevýhody, funkčnost a použití tohoto frameworku a příprava modelů neuronových sítí, pro práci s tímto frameworkem.

V další části této práce bude tento framework použit v praxi a v rámci ukázkové aplikace bude implementováno jeho použití. Nejprve bude popsán postup přípravy prostředí, které je potřeba pro práci s tímto frameworkem a přípravu modelů tak, aby je uměl framework zpracovat. Následně bude připraven ukázkový model neuronové sítě a zprovozněna ukázková aplikace. V této práci bude také popsán kód ukázkové aplikace a jeho nejdůležitější části pro další použití nebo integraci do vlastní aplikace. Tato práce se také bude snažit popsat případné problémy, které během práce se SNPE frameworkem nastanou a popsat případné nevýhody, které jsou s tímto frameworkem spojeny.

1. Strojové učení

Pokud mluvíme o umělé inteligenci, existuje více definic. Někteří autoři definují umělou inteligenci jako systém, který přemýšlí jako lidé [1], jiní dokonce jako systém, který se chová podobně jako lidé [2]. Ať už je definice jakákoliv, jedná se například o systémy, které jsou schopné porazit mistra světa v šachách (Garry Kasparov) [3] nebo o systémy, které rozumí řeči, tomu, co člověk napíše, zpracovávají obrazová data pomocí kamer, řídí autonomní auta a pohání roboty. Strojové učení je poté podobor umělé inteligence [4].

Algoritmy strojového učení jsou procesy, které používají vstupní data pro dosažení výsledků. Na rozdíl od běžných algoritmů strojové učení dosahuje požadovaných výsledků, aniž by byly tyto algoritmy „hard coded." Tyto algoritmy se automaticky mění a přizpůsobují svoji architekturu pomocí učení, aby výsledky, které poskytují, byly lepší při každém dalším vyhodnocování vstupních dat. Proces, při kterém se tyto algoritmy vyvíjejí pro lepší výsledky, se nazývá trénování, kdy jsou poskytována vstupní data zároveň se správným výsledkem. Díky tomu se algoritmus může adaptovat právě tak, aby poskytnul správný výsledek pro konkrétní vstupní data, na kterých byl trénován, ale také poskytnul správný výsledek pro nová vstupní data, která předtím nebyla součástí trénovací množiny. Tento proces trénování nemusí být omezen pouze na začátek práce s algoritmy. Podobně jako u lidí, algoritmus se může trénovat po celou dobu svého fungování, tak jak zpracovává nová data, a může se poučit ze svých chyb. V ideálním případě strojové učení napodobí proces, jakým se učí lidé. V konkrétním případě může být cílem algoritmu zařazení jednotlivých objektů do různých kategorií, jako například rozeznat jablko a pomeranč. Každé jedno jablko a pomeranč jsou unikátní, a proto by „hard coded" program musel obsahovat definice mnoha konkrétních jablek a pomerančů. Lze však natrénovat algoritmus pomocí technik učení, aby rozpoznal jablko nebo pomeranč, který předtím nebyl definován. [5]

Algoritmy učení strojového učení, lze rozdělit do několika základních skupin. (i) učení s učitelem, (ii) učení bez učitele a (iii) učení částečně pod dohledem.

- Učení s učitelem: tyto algoritmy mapují vstupy na požadované výstupy. Standartně se tento algoritmus používá při řešení problému klasifikace, kdy se žák musí naučit, jak zařadit vstupní vektor do několika tříd. To se naučí tím způsobem, že je mu poskytnuto několik příkladů vstupu a správného výstupu. [6]
- Učení bez učitele: V tomto případě neexistuje soubor dat, podle kterých by šlo trénovat. Dalším rozdílem oproti učení s učitelem je to, že neexistuje

žádná záruka, že bylo nalezeno globálně optimální řešení. Toto učení probíhá na základě shlukování podle určitých kritérií. Vstupy s podobnou charakteristikou se poté sdružují k sobě. [7]

- Učení pod částečným dohledem: je kombinací obou přístupů.

Existují další přístupy učení, ale tyto se řadí mezi základní a nejčastější. [6]

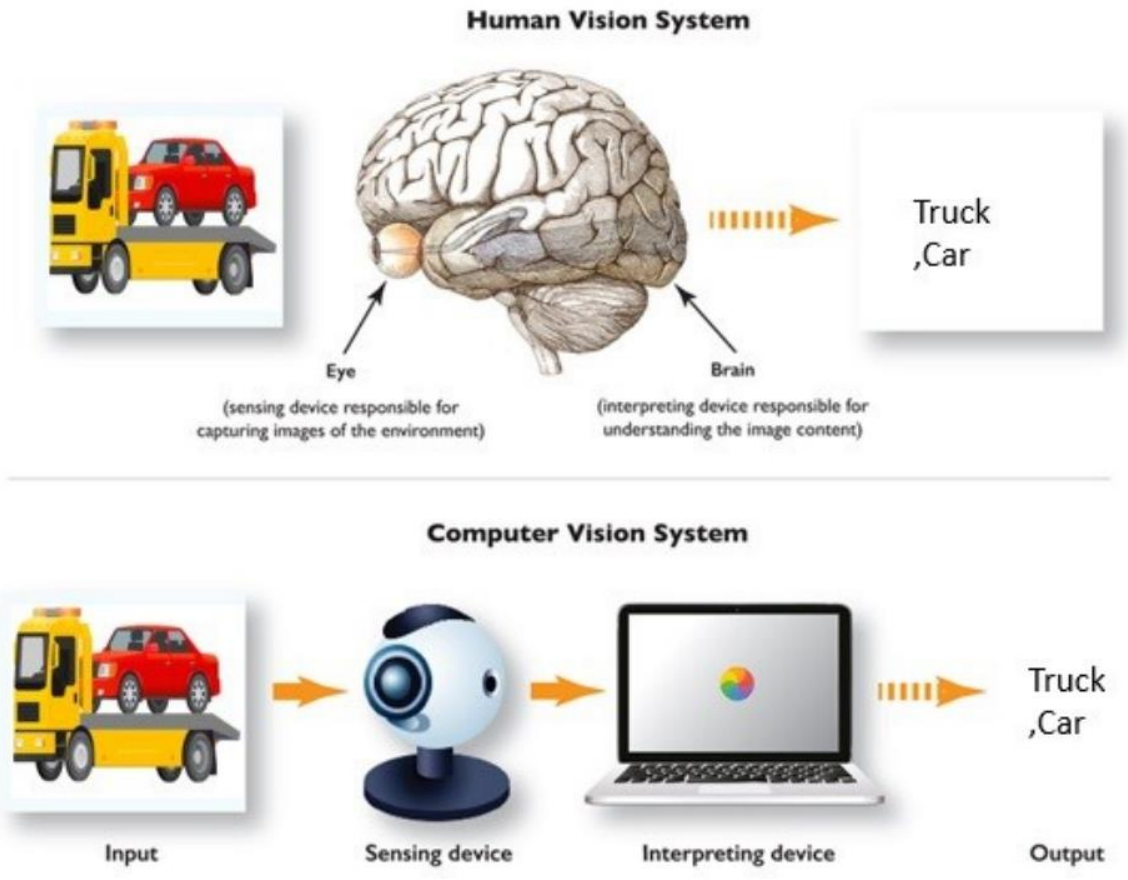
Strojové učení přináší dvě výhody. První výhodou je, že může nahradit pracné a opakující se práce. Druhou, významnější výhodou je to, že strojové učení může zpracovat mnohem více dat, a také mnohem komplexnější data, než je běžný pozorovatel schopný zpracovat a určit výsledek v poměrně krátkém čase. Konkrétní příklad této výhody můžeme vidět ve zdravotnictví, kde strojové učení může pomáhat v rozhodování nebo odhalování onemocnění. Dobře natrénovaný algoritmus může propojit zkušenosti mnoha zdravotníků a snížit tak nejistoty při rozhodování. [5]

Jak bylo zmíněno v této práci, existují tři základní přístupy ke strojovému učení a počítačovému vidění: s učitelem, bez učitele a s částečným dohledem. Učení s učitelem je velmi drahé a časově náročné, protože je potřeba označit trénovací množinu dat. Na druhou stranu učení pod částečným dohledem má pouze část trénovací množiny označenou a zbytek dat je bez značení. Avšak většina problémů reálného světa spadá do učení bez učitele, protože se vzory vyvíjejí na základě klastrování neboli hierarchického shlukování. [8]

2. Počítačové vidění

Počítačové vidění umožňuje počítači vidět a pochopit obsah obrázku a díky tomu je schopný zareagovat podobně jako člověk. Základním cílem počítačového vidění je poskytnout počítači stejné hodnoty jako poskytuje oko člověku a díky dalším technologiím replikovat lidské schopnosti vidění. [9]

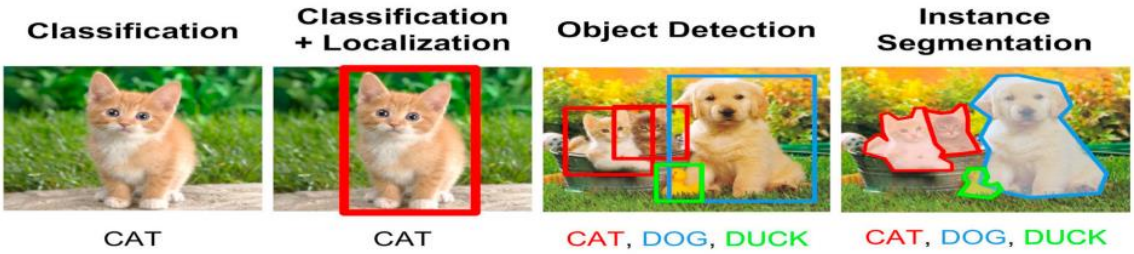
Počítačové vidění můžeme chápat jako automatickou analýzu a pochopení informací z jednoho nebo více obrázků. Také můžeme počítačové vidění chápat jako techniku zpracování obrázků, která bere obrázek jako vstup a na výstupu vrací interpretaci obrázku. Cílem počítačového vidění je tedy poskytnout podobnou, nebo lepší vizualizaci světa, jako dokáže lidský mozek. [10]



Obrázek 1 – Porovnání lidského a počítačového vidění, zdroj: [11]

Na obrázku 1 lze vidět porovnání a podobnosti lidského vidění a počítačového vidění. Z tohoto obrázku je patrné, že si jsou oba systémy dost podobné.

Počítačové vidění používá obrázky a mapování vzorů pro nalezení řešení. Díky kombinaci počítačového vidění a strojového učení je možné provádět automatickou analýzu nebo anotaci videa nebo obrazových dat. [8]



Obrázek 2 – klasifikace, detekce a segmentace, zdroj: [12]

Na obrázku 2 je zobrazeno několik možných přístupů klasifikace obrázku, popřípadě objektů a také lokalizace těchto objektů. Klasifikace obrázku do jedné kategorie většinou probíhá podle nejvýraznějšího objektu na tomto obrázku. Avšak obrázky mohou být komplexnější a samotná klasifikace obrázku jako celku by mohla být velice nepřesná. Zde pomůže detekce objektů, díky které můžeme označit a klasifikovat jednotlivé objekty na jednom obrázku. Další výhodou, kterou tato klasifikace jednotlivých objektů přináší je to, že tyto objekty lze v obrázku lokalizovat. Dále se jednotlivé modely mohou lišit podle přesnosti lokalizace objektu v obrázku. [12]

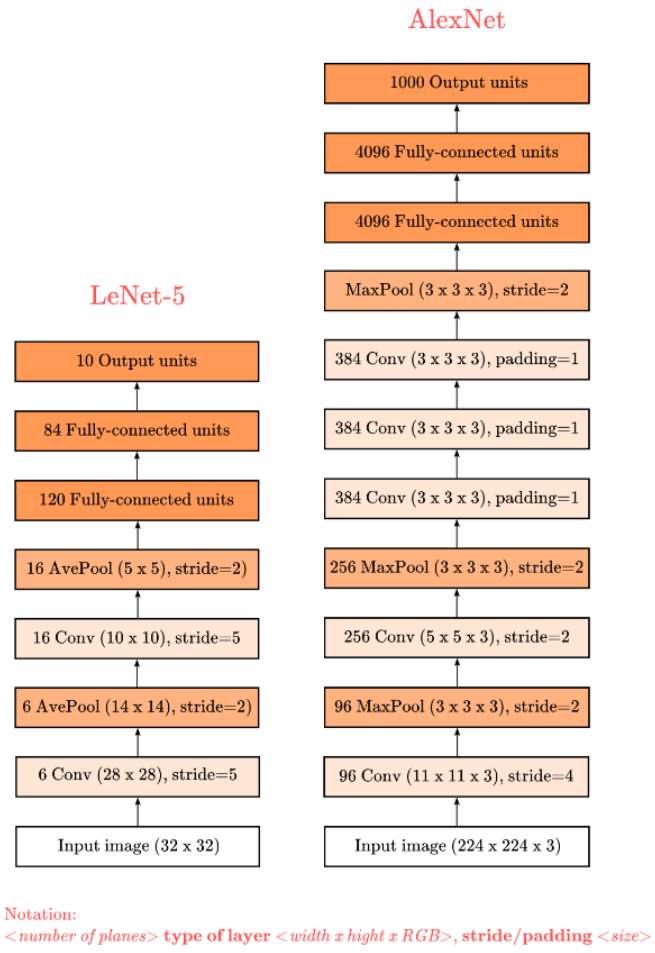
Klasifikace je využívána pro určování, které objekty jsou na obrázku. Hlavním záměrem je zjistit, které třídy (objekty) jsou na obrázku přítomny. Může být využito více přístupů, a to klasifikace podle nejdominantnějšího objektu nebo přiřazení všech tříd podle všech objektů, které se na obrázku vyskytují. Klasifikace je nejčastěji využívána v situacích při rozhodování s odpovědí ano nebo ne, zda obrázek obsahuje objekt nebo anomálii. Detekce objektů mimo klasifikaci objektu ještě určuje, kde objekt na obrázku nebo videu je. K samotné klasifikaci tedy přidává ještě něco navíc. Takto nalezené objekty označí nejčastěji do rámečku a k tomuto rámečku připíše třídu objektu (provede jeho klasifikaci). Segmentace obrázku rozdělí jednotlivé objekty do regionů, každý se specifickým tvarem a hranicí. Segmentace neoznačuje veškeré objekty na obrázku, ale pouze ty, které budou předmětem dalšího zpracování, jako například klasifikace. Důvodem pro segmentaci může být například to, aby byly vybrány objekty v popředí obrázku a pouze ty následně zpracovávat. Segmentace poskytuje detaily pixel po pixelu o objektu, a tím se liší od klasifikace a detekce objektů. [13]

Zde přicházejí konvoluční neuronové sítě, které se používají při rozpoznávání a klasifikaci obrázků. Obsahují neurony, které mají různé dimenze: šířku, výšku a hloubku. Konvoluční neuronové sítě nabraly na popularitě díky rozsáhlým a dostupným datům. [14]

3. Konvoluční neuronové sítě

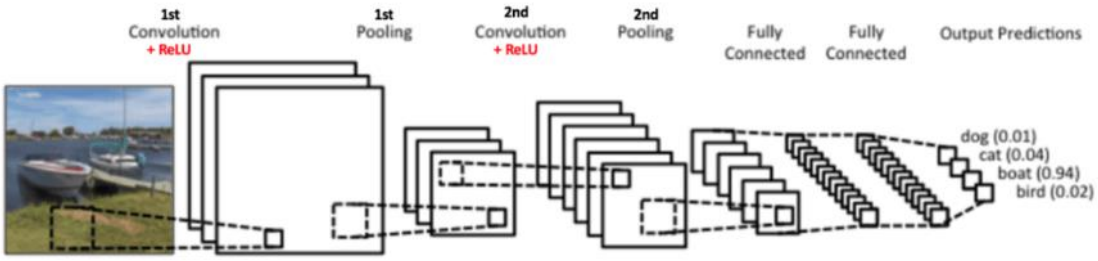
Pro lepší pochopení konvolučních neuronových sítí, budou použity zjednodušené koncepty. Matematické vyjádření konceptů a operací bude zjednodušeno nebo přeskočeno za účelem snazšího pochopení architektury a fungování neuronových sítí.

Jednou z prvních konvolučních sítí byla síť s názvem LeNet. Ačkoliv od jejího vytvoření uplynula již dlouhá doba a novější sítě ji předčily, základní koncept zůstává stejný, jak je patrné z obrázku 3.



Obrázek 3 – Porovnání architektury LeNet-5 a AlexNet, zdroj: [15]

Na obrázku 3 je zobrazeno porovnání architektur LeNet-5 (1998) a AlexNet (2012). Z tohoto obrázku je patrné, že je architektura složitější a jsou zde vrstvy navíc, ale princip je podobný. Pro vysvětlení fungování neuronových sítí lze tedy využít starší síť, protože princip se doposud nijak zásadně nezměnil.



Obrázek 4 – Jednoduchá architektura konvoluční neuronové sítě, zdroj: [16]

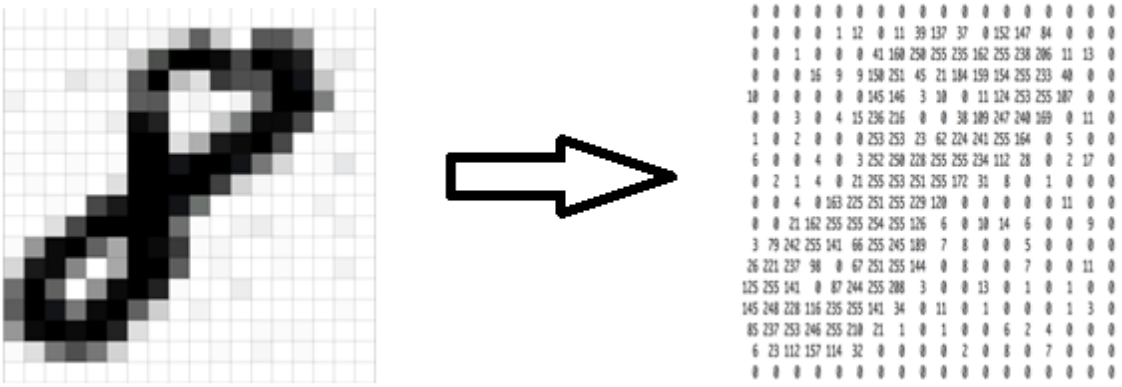
Obrázek 4 zobrazuje jednoduchou architekturu konvoluční neuronové sítě. Tato síť je schopna klasifikovat vstupní obrázek do jedné ze čtyř kategorií (pes, kočka, loď a pták) společně s pravděpodobností.

V této architektuře jsou čtyři hlavní operace:

1. Konvoluce
2. Nelinearita (ReLU)
3. Pooling
4. Klasifikace (plně propojená vrstva)

Tyto operace jsou základními stavebními kameny každé konvoluční neuronové sítě. Z obrázku 3 je patrné, že jsou použity tyto základní operace pro obě sítě, jen se liší jejich posloupnost, kombinace a počet. Také se jednotlivé operace odlišují svými parametry a nastavením.

Vstupní obrázek je třeba předat do neuronové sítě tak, aby s ním bylo možné tyto operace provádět, a to nejlépe v podobě čísel. Obrázek lze reprezentovat jako matici hodnot jednotlivých pixelů. Rozlišení obrázku je potom velikost této matice. Standartní obrázek bude mít pixely rozděleny do tří základních barev (červená, zelená a modrá). Pro každou barvu je zde matice a tyto matice jsou naskládány přes sebe. Jednotlivé hodnoty v maticích (pixely) mají hodnoty od 0 do 255. Pokud je však obrázek černobílý je zde matice pouze jedna a její hodnoty reprezentují, jak moc černý daný pixel je (0 pro černou a 255 pro bílou barvu). Pro lepší představu je zde obrázek 5. [17]



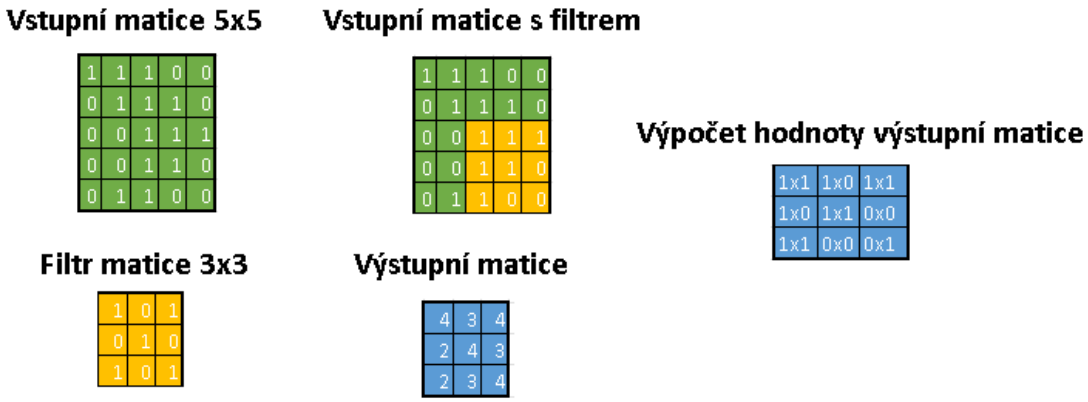
Obrázek 5 – Převod černobílého obrázku na maticovou reprezentaci, zdroj: [17], vlastní zpracování

Nyní, když máme obrázek reprezentován pomocí matice pixelů, můžeme začít provádět jednotlivé operace.

1. Konvoluce

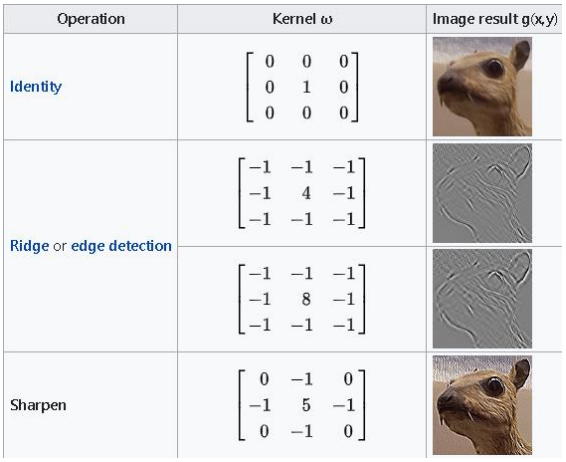
Konvoluční neuronové sítě mají své jméno podle této operace. Je to také hlavní operace v rámci celého procesu. Tato operace má za cíl extrahovat dominantní rysy ze vstupního obrázku. Tento proces zachovává prostorový vztah mezi pixely tím, že se učí jednotlivé rysy pomocí malých matic se vstupními daty. Jak bylo předvedeno výše v této práci, obrázek můžeme vidět jako matici hodnot jednotlivých pixelů. Pokud si představíme matici o velikosti 5x5 s hodnotami pixelů 0 nebo 1 (v matici v obrázku 5 můžeme vidět hodnoty od 0 do 255, tato matice by byla speciálním případem, kde jsou hodnoty pouze 0 a 1) a menší matici 3x3 s hodnotami 0 a 1 (nastavení hodnot této matice a její funkce bude vysvětlena později v této práci), vypadal by proces konvoluce nějak takto:

Matice 3x3 se umístí do levého horního rohu a je vypočítána hodnota pomocí násobení jednotlivých prvků. Tento výsledek je poté zaznamenán do výstupní matice. Následně je menší matice posunuta o 1 pixel (krok) a opět je vypočítána hodnota, která se zaznamená do výstupní matice. Takto se proces opakuje pro celou vstupní matici (5x5). Na konci máme vypočítanou výstupní matici. V tomto procesu se vždy analyzuje pouze část vstupní matice a pro každý krok je tato část jiná.[18] Tento proces je zachycen na obrázku 6, kde je vidět vstupní matice, filtr (matice 3x3), výsledná matice a výpočet hodnoty výstupní matice v posledním kroku.



Obrázek 6 – Proces výpočtu konvoluce na zjednodušených maticích, zdroj: vlastní zpracování

Právě menší matice, která je posunována o určitý počet pixelů (krok) po vstupní matici (vstupní obrázek přetvořený na pixelovou matici), je nazývána filtr. Hlavní funkcí tohoto filtru je to, že ve vstupních datech hledá rysy, které jsou poté reprezentovány ve výstupní matici. Jak je zřejmé z obrázku 6, tak odlišné hodnoty filtru budou mít zásadní vliv na výsledné hodnoty výstupní matice. Pomocí nastavování filtru lze tedy v obrázku hledat různé rysy. Filtry lze specificky nastavit například na detekci hran, zaostření nebo rozmazání, pomocí úpravy hodnot v této matici. Příklad aplikace rozdílných filtrů je zachycen na obrázku 7. [19]

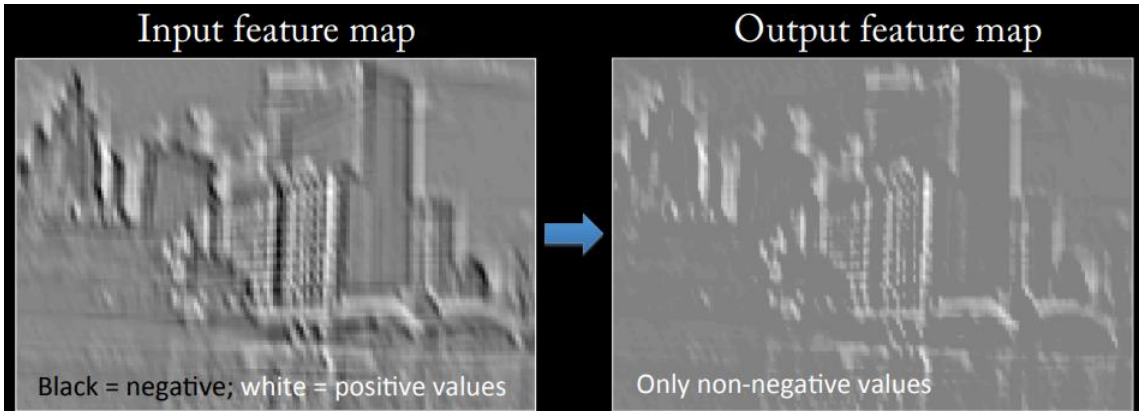


Obrázek 7 – Aplikace rozdílných filtrů v procesu konvoluce, zdroj: [20]

V konvolučním procesu může být použito více filtrů, přičemž každý se specifikuje na detekci jiných rysů ve vstupních datech. Čím více filtrů je použito, tím více rysů je zachyceno a dostáváme přesnější rozpoznávání vzorů ve vstupních datech, které konvoluční neuronová síť předtím „neviděla“. Konvoluční síť si poté v procesu trénování upravuje hodnoty jednotlivých filtrů sama. Je však potřeba nastavit jisté parametry jako například počet filtrů, velikost filtru (matice), krok (počet pixelů, o který se posunuje filtr) a další, a to ještě před trénováním sítě. [18]

2. Nelinearita (ReLU)

Další operace, která je doplňkovou k operaci konvoluce, je nelineární funkce ReLU (rectified linear unit). Tato funkce je v poslední době jednou z nejčastěji používaných. Ostatní funkce jsou například tanh nebo sigmoid. ReLU funguje tak, že přemění všechny negativní hodnoty ve výstupní matici na 0 a ostatní hodnoty ponechá nezměněné. Hlavním účelem této funkce je zavést nelinearitu do konvoluční neuronové sítě.[21] Hlavní výhodou nelineárních dat je to, že většina reálných dat je nelineární a protože proces konvoluce vytvoří lineární data, tato operace výsledek upraví do nelineární podoby a výsledky při rozpoznávání nových obrázků se díky tomu zlepší. [18]

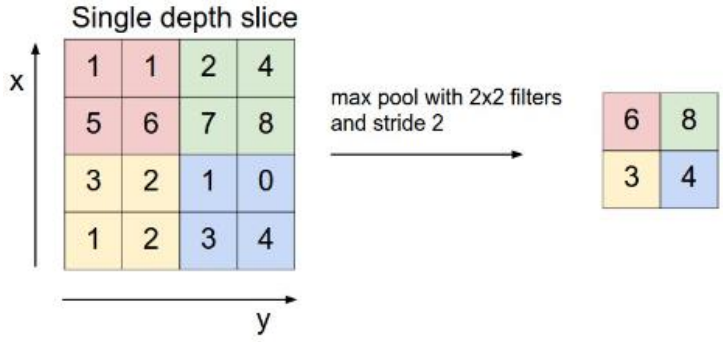


Obrázek 8 – ReLU operace a její výsledek, zdroj: [22]

3. Pooling

Pooling proces často následuje po konvolučním procesu. Hlavní funkcí tohoto procesu je zmenšení velikosti výstupní matice. To má za následek méně parametrů, které se musí v dalších vrstvách zpracovávat. Tato funkce však i při zmenšení matice zachovává nejvýznamnější informace, které jsou ve výstupní matici obsaženy. Díky tomu se zdokonaluje výkon celé sítě a zabrání se tomu, aby bylo pro rozhodování

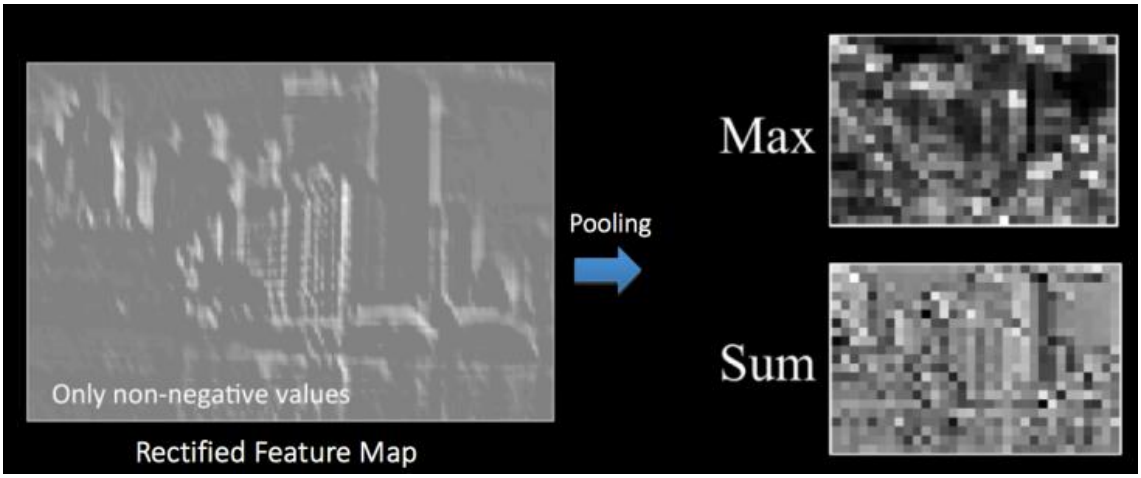
zpracováváno příliš mnoho informací, které nejsou tak podstatné pro určení výsledku (overfitting). Nad každou jednotlivou výstupní matici se provádí tato funkce, která zmenší rozměry pomocí některé z funkcí maximum, průměr, součet nebo jiných funkcí. Samotný proces poté probíhá například s filtrem 2x2 a krokem 2, který je aplikován na výstupní matici z procesu konvoluce. V každém kroku je pak podle zvolené funkce vypočítána nová hodnota do výstupní matice pooling procesu. Tento proces a jeho výsledek je vidět na obrázku 9. [18, 23]



Obrázek 9 – Pooling proces a jeho výsledek, zdroj: [23]

Obrázek 9 zachycuje pooling proces s filtrem 2x2 a krokem 2. Funkce v tomto případě je maximum, kdy ze zvolené oblasti (vyznačena barvou) je vždy vybráno maximum. Takto zůstanou zachovány nejvýznamnější rysy dané oblasti a zároveň je zredukován počet parametrů pro další vrstvu procesu.

Výsledek tohoto procesu je zachycen na obrázku 10, kde je vidět výsledek po konvolučním procesu, na který byla aplikována ReLU funkce. Výsledky jsou zde dva. Jeden pro funkci maximum a druhý pro funkci průměr.



Obrázek 10 – Výsledek pooling procesu po aplikaci max a sum funkce, zdroj: [22]

4. Klasifikace

Tyto první tři operace se mohou opakovat vícekrát za sebou v různých pořadích s různými parametry. Výsledek řetězce operací má za účel shrnout hlavní rysy a vlastnosti ve vstupním obrázku, aplikovat nelinearitu a snížit počet parametrů, které se dále analyzují pro finální klasifikaci. Výsledkem těchto operací je sada matic, které obsahují informace o vstupním obrázku. Jejich počet závisí na počtu aplikovaných filtrů v konvolučním procesu a dalších parametrech. Tyto matice jsou základem pro vstup do plně propojené vrstvy (neuronová síť), kde probíhá samotná klasifikace. [24]

Plně propojená vrstva je již klasickou vícevrstvou neuronovou sítí s jednotlivými neurony, které jsou plně propojeny. To znamená, že každý neuron má propojení ke všem neuronům v předchozí vrstvě. Její architektura, počet vrstev a neuronů v jednotlivých vrstvách, závisí na konkrétní implementaci. Ve většině případů se ve výstupní vrstvě používá softmax funkce, která pouze převede čísla na pravděpodobnosti (tak, aby součet vycházel 1).[12]

Plně propojená vrstva má za úkol za pomoci informací, které jsou zpracovány předešlými procesy, zařadit daný obrázek do určité kategorie. V příkladu, který byl uveden v úvodu této kapitoly, se jedná o čtyři kategorie (pes, kočka, loď a pták) a do jedné z těchto kategorií, je obrázek s určitou pravděpodobností zařazen.

4. Strojové učení na mobilních zařízeních

Mobilní zařízení se stala nedílnou součástí moderního života. Poskytují celou řadu komunikačních, zábavních a produktivních nástrojů na dosah ruky. Díky pokročilé technologii vývoje, jsou tato zařízení velice výkonná a zároveň se vejdou do jedné ruky. Mobilní zařízení, jako smartphony nebo tablety, zásadně změnily to, jak komunikujeme, pracujeme, bavíme se ve volných chvílích a vnímáme svět kolem nás. Charakteristické pro tato zařízení je právě jejich přenositelnost a konektivita a díky tomu je může uživatel mít vždy u sebe, ať už je kdekoliv a dělá cokoliv. Možnosti, které mobilní zařízení poskytují, se během posledních pár let zdokonalily a v dnešní době je mobilní zařízení vybaveno nejpokročilejšími hardwarovými a softwarovými technologiemi. Moderní smartphony a tablety jsou vybaveny displeji s vysokým rozlišením, výkonnými procesory a grafickými čipy, kvalitními kamerami a nejrůznějšími senzory. Pozadu není ani operační systém pro tato zařízení a různé aplikace, které jsou na těchto zařízeních nabízeny a poskytují nejrůznější funkce pro zábavu nebo usnadnění každodenního života nebo pracovních povinností.

1. Výhody

Díky tomu, že se mobilní zařízení stala velice výkonnými, je možné je využívat pro běh aplikací, které využívají strojové učení pro svůj běh. Tyto aplikace využívají algoritmy na analýzu a učení z rozsáhlých datasetů a díky tomu poskytují uživateli pokročilé funkce a výhody, které strojové učení přináší. Takové aplikace jsou například hlasový asistenti, nástroje pro práci s obrázky, personalizované nástroje pro doporučování a mnoho dalších.

Strojové učení si získalo velkou oblibu, a v poslední době se stalo velmi populárním a diskutovaným tématem díky své schopnosti extrahovat poznatky a vzory z velkých množství dat. Jak je zmíněno výše v této práci, u mobilních zařízení nastal také značný pokrok a díky tomu mohou využívat strojové učení, které je náročné na hardware. Díky pokroku v technologiích jsou mobilní zařízení dostatečně výkonná, aby umožnila provádět výpočty potřebné pro algoritmy strojového učení.

Jednou z hlavních výhod strojového učení na mobilním zařízení je schopnost poskytovat uživatelům personifikované služby, díky přístupu k jejich datům, která jsou získána z mobilního zařízení. Mobilní aplikace mohou analyzovat chování, preference a zvyky uživatelů a na základě těchto informací mohou poskytnout uživateli personalizované doporučení a upravit své služby na míru uživateli. Dobrým příkladem takové aplikace, která využívá výhod strojového učení, je hudební

aplikace, která analyzuje chování uživatelů při poslechu hudby a na základě tohoto chování navrhuje nové skladby, které odpovídají vzorům předchozího chování uživatele. Další výhodou strojového učení na mobilním zařízení je schopnost zpracovávat data v reálném čase přímo na zařízení. Není nutné data posílat přes síť a čekat na odpověď. To značně sníží celkový čas potřebný pro zpracování požadavku a zlepší práci s aplikací. [25, 26]

Strojové učení nemusí přinášet výhody pouze pro aplikace, ale také pro zabezpečení mobilního zařízení. Na mobilním zařízení má uživatel velké množství citlivých dat a jejich bezpečnost je nanejvýš důležitá. Algoritmy strojového učení lze použít k detekci a prevenci narušení bezpečnosti, ať už škodlivým softwarem nebo neoprávněným přístupem k zařízení. Kromě bezpečnosti lze také těžit z výhod strojového učení z hlediska optimalizace a energetické účinnosti. Algoritmy strojového učení mohou optimalizovat využití zdrojů jako je CPU nebo baterie podle chování konkrétního uživatele, což může mít za následek delší životnost baterie a lepší výkon zařízení. Mimo tyto výhody může strojové učení také přinášet nové možnosti v různých odvětvích jako například zdravotnictví, fitness, finance a další. Tyto aplikace díky strojovému učení mohou poskytovat upozornění na neobvyklé události, různá doporučení a spoustu dalších výhod, které budou personifikované pro konkrétního uživatele a jeho potřeby. [26, 27]

2. Limitace

Díky rozvoji v oblasti mobilních zařízení a zvyšování jejich výkonu a kvality hardwaru je možné tato zařízení používat pro provádění složitých úloh strojového učení. Stále však existují určitá omezení, která limitují strojové učení na mobilním zařízení.

Přestože jsou mobilní zařízení čím dál výkonnější, stále se nemohou rovnat špičkovým stolním počítačům nebo serverům. Mobilní zařízení jsou díky svému výkonu schopna provádět jednodušší neuronové sítě a různé algoritmy, avšak prostředky na mobilním zařízení jsou omezené, ať už jde o výpočetní výkon, paměť nebo baterii. Tato omezení lze obejít pomocí připojení mobilu pomocí internetové sítě na vzdálený server, kde je dostatečný výkon, ale tím mizí některé výhody, které strojové učení na mobilním zařízení má a vše je závislé na kvalitě a stabilitě internetového připojení.

S tím souvisí i další omezení. Pro trénování a zlepšování neuronových sítí je potřeba velké množství dat, avšak paměť mobilního zařízení je značně omezená. Ukládání velkých datasetů tedy nelze provádět přímo na zařízení. Řešením je do jisté míry některý typ cloudového úložiště nebo komprese dat. Další z možností

je předem natrénovat a optimalizovat síť během jejího vývoje za použití výkonného hardwaru a následně přesunutí předtrénované sítě na mobilní zařízení.[28]

Dalším omezením, která mají mobilní zařízení, je absence specializovaného hardwaru. Algoritmy strojového učení lze optimalizovat pro konkrétní hardwarové architektury jako například GPU. Mobilní zařízení však většinou postrádá specializovaný hardware potřebný pro efektivní spuštění těchto algoritmů. Algoritmy mohou běžet na CPU, ale běh nemusí být optimální a kvůli tomu jsou výpočty náročnější jak časově, tak z hlediska spotřeby energie a ostatních prostředků. Toto omezení lze do jisté míry vyřešit pomocí optimalizace softwaru a předělání algoritmů pro běh na hardwaru, který je dostupný na mobilním zařízení. Další optimalizací by mohla být komprese modelu nebo zmenšení sítě. To však přináší jiné nevýhody, a proto je toto řešení vhodné pouze v určitých případech.[28, 29]

V neposlední řadě k limitacím řadíme určitým způsobem také bezpečnost a soukromí uživatele. Algoritmy často vyžadují přístup k osobním a citlivým datům uživatele, což může vést k jejich ohrožení, pokud s nimi není zacházeno správně. Komplexně je otázka soukromí a bezpečí s použitím strojového učení složité téma a je třeba dbát zvýšené opatrnosti. Na vyřešení této limitace existují různé techniky na ochranu soukromí jako například federované učení (technologie na minimalizaci dat, která využívá anonymizované informace z více zařízení a na jejich základě vytváří modely strojového učení) nebo techniky diferenčního soukromí. [29, 30]

3. Nástroje

Spolu s rozvojem výkonu mobilních zařízení a s tím spojené poptávky po aplikacích, které využívají této technologie, vzrostla i poptávka po nástrojích, které by umožňovaly, ulehčovaly a zrychlily vývoj těchto aplikací, jež by běžely nativně na zařízeních. Vývoj těchto aplikací je náročný z důvodů omezených zdrojů a dalších limitací mobilních zařízení, které jsou popsány výše v této práci. Z tohoto důvodu vznikly různé frameworky, které umožňují vývojářům překonat tyto limitace a vytvářet výkonné modely a použitelné aplikace, které fungují efektivně a jsou optimalizované pro běh na mobilních zařízeních.

Jedním z velmi populárních frameworků, který umožňuje vytváření efektivních modelů neuronových sítí a tím vytváření aplikací pro mobilní zařízení, je framework TensotFlow Lite. Jedná se o odlehčenou verzi populárního TensorFlow frameworku, která je optimalizována pro zařízení s omezenými zdroji. Výhoda TensorFlow Lite frameworku je jeho multiplatformní podpora. Podporuje jak Android, tak iOS a poskytuje řadu funkcí, které usnadňují vývoj a nasazení modelů neuronových sítí na mobilní zařízení. Funkcemi, které stojí za zmínění, jsou například podpora

různých formátů modelů, včetně modelů ve formátu TensorFlow, podpora trénování sítě přímo na zařízení, podpora hardwarové akcelerace modelů prostřednictvím nástrojů jako například Android Neural Networks API. [31]

Dalším nástrojem pro efektivní využívání strojového učení na mobilních zařízeních je Core ML. Jedná se o framework vyvinutý společností Apple. Jeho úkolem je vytváření modelů neuronových sítí, které budou běžet na zařízeních s operačním systémem iOS a specializovaném hardwaru a architektuře, kterou mají zařízení od společnosti Apple. Core ML poskytuje řadu funkcí, které usnadňují integraci strojového učení do aplikací pro iOS, mezi které patří podpora různých formátů modelů, automatická optimalizace modelu pro mobilní zařízení a akcelerace prostřednictvím Neural Engine společnosti Apple. [32]

Kromě frameworků, které poskytují různé možnosti integrace a optimalizace modelů do mobilních aplikací, jsou k dispozici i nástroje a knihovny pro vývoj těchto modelů pro mobilní zařízení. Mezi tyto nástroje patří například scikit-learn, Kares nebo PyTorch. Tyto nástroje taktéž poskytují řadu funkcí pro vývoj, trénování, nasazování a optimalizaci modelů neuronových sítí pro mobilní zařízení. Tyto nástroje nejsou jediné a s rostoucí popularitou strojového učení a AI jako takového, se budou tyto nástroje zlepšovat a budou vznikat nové pro řešení aktuálních problémů a výzev.[33–35]

Přestože jsou tyto nástroje dostupné a existuje řada návodů, tutoriálů a videí jak s těmito frameworky nebo nástroji pracovat, může být vývoj takovýchto aplikací stále náročný a tento vývoj doprovází řada limitací, které musí vývojář mít na paměti. Jedná se například o omezení výpočetních zdrojů, omezenou paměť a úložiště a problémy spojené se spotřebou energie a výdrží baterie. Vždy je nutné mít na paměti konkrétní použití a možnosti dostupného hardwaru, pro řešení specifických problémů a na základě dobré analýzy je potřeba vybrat správné nástroje, které umožní některá omezení odstranit a díky tomu vytvořit vysoce výkonné aplikace, které budou využívat strojové učení a neuronové sítě a poběží efektivně přímo na mobilních zařízeních. Mimo používání nástrojů je možné využívat různé techniky a dodržovat určité přístupy, které zlepší výkonnost a sníží nároky na prostředky pro běh aplikací, které využívají strojové učení.

4. Osvědčené postupy

Protože roste zájem o aplikace, které využívají strojové učení, mají v sobě model neuronové sítě a běží nativně na mobilním zařízení, je dobré představit osvědčené postupy, které slouží k tomu, aby aplikace byly efektivnější a úspornější na potřebné prostředky.

1. Algoritmus: Výběr správného algoritmu podle konkrétního problému, který bude aplikace řešit a podle hardwaru, který je dostupný na mobilním zařízení je jednou z důležitých věcí, které mají vliv na výsledný výkon aplikace. Měly by být vybrány algoritmy, které jsou optimalizované pro běh na mobilním zařízení nebo by měly být tyto algoritmy optimalizovány pro běh na mobilním zařízení. Obecné algoritmy mohou plýtvat prostředky a na mobilním zařízení nebudou efektivní.
2. Předtrénované modely: Trénování modelů je velice náročná operace, která vyžaduje velké množství dat, které zabírají hodně paměti. Tato operace je také náročná na výpočetní výkon a často se vyplácí použít specializovaný hardware, který urychlí celý proces. Použití předem natrénovaných modelů, nebo trénování modelů na výkonném zařízení značně urychlí čas potřebný k nasazení aplikace.
3. Úspora energie: Jelikož mobilní zařízení jsou napájeny z baterie, je pro aplikace, které na nich běží důležité, aby byly optimalizované co se týče spotřeby energie a neplýtvaly s ní. Proto by měly aplikace pro mobilní zařízení minimalizovat počet požadovaných výpočtů a množství dat, které je třeba přenést. Toho lze dosáhnout například zmenšením modelů nebo pomocí dalších technik, které sníží výpočetní náročnost za cenu snížení přesnosti výpočtů, která však v určitých případech není prioritní.
4. Soukromí a bezpečnost: Tyto aplikace budou mít přístup k citlivým datům na mobilním zařízení, proto je nutné dbát na tuto skutečnost a aplikace zabezpečit a využívat principy, které zamezí zneužití. K tomu je možné využít anonymizaci nebo jiné přístupy a toto je nutné v případě, že aplikace bude pracovat s osobními daty uživatele. Je třeba zajistit, že budou dokonale chráněna před zneužitím. [36, 37]

Jedná se pouze o obecná doporučení, jak k vývoji těchto aplikací přistupovat a jsou zde popsány body, které by měl vývojář brát v potaz při vývoji aplikace, která využívá strojové učení. Veškeré otázky je potřeba řešit podle konkrétního zaměření a použití aplikace, ale tyto principy jsou obecné a je dobré se nad nimi vždy zamyslet. Díky tomu bude aplikace efektivní a bezpečná.

5. Qualcomm® Neural Processing SDK

Společnost Qualcomm se proslavila v oboru telekomunikace a je tvůrcem patentu CDMA (code division multiple access) technologie, která je používána v mnoha 2G a 3G komunikačních systémech. Společnost se zabývá především těmito oblastmi:

1. Bezdrátové modemy a procesory pro mobilní zařízení.
2. CDMA technologie (bezdrátová technologie, která umožnuje více uživatelům sdílet stejné frekvenční pásmo).
3. 5G technologie
4. IoT: Qualcomm vyvíjí technologie, produkty a platformy pro IoT.
5. Automobilový průmysl (infotainment, asistenty řízení).
6. Bezdrátové nabíjení.
7. Licence (velkou část příjmů tvoří poskytování patentů a licencí).
8. Služby (technická podpora, služby specializovaných inženýrů, řízení dodavatelského řetězce).

1. Co je to SNPE

Jedním z produktů této firmy je i SDK (software development kit), které můžeme najít pod názvem SNPE (Snapdragon Neural Processing Engine) nebo jako Qualcomm Neural Processing SDK. Toto SDK umožnuje vývojářům nasadit modely neuronových sítí na Qualcomm Snapdragon platformy. Toto SDK obsahuje nástroje a knihovny, které umožňují spouštět tyto modely na mobilních zařízeních (s odpovídající technologií Snapdragon procesoru) a provádět optimalizační operace jako například měření výkonu pro identifikaci tzv. bottlenecks neboli míst, kde dochází ke zpomalení. [38, 39]

Toto SDK podporuje nejznámější frameworky hlubokého učení jako například TensorFlow, Caffe nebo Caffe2 a umožnuje vývojářům provádět různé operace jako například rozpoznávání obrazu, detekce objektů nebo zpracování přirozeného jazyka. Součástí je také sada před optimalizovaných modelů hlubokého učení, které umožnují běžné úkoly jako klasifikace obrazu, detekce objektů a segmentace obrazu. [38, 39]

SNPE umožnuje vývojářům využít výkon mobilního Snapdragon procesoru a provádět různé úkony hlubokého učení na mobilních zařízeních bez potřeby dalšího výkonného počítače. To umožňuje provádět výpočty přímo na zařízení a díky

tomu není potřeba cloudových nebo serverových technologií nebo internetového připojení. To je velkou výhodou, protože díky tomu se značně zvýší výkon, neboť zde není latence a zpomalení kvůli síťové komunikaci. SNPE je také navrženo tak, aby bylo snadné ho použít a integrovat do existující mobilní aplikace. Také obsahuje nástroje, díky kterým mohou vývojáři optimalizovat modely právě pro Snapdragon platformy. [38, 39]

2. Výhody SNPE oproti jiným přístupům

Mezi výhody, které SNPE přináší, patří optimalizace pro Snapdragon, podpora populárních frameworku, před optimalizované modely, jednoduchost integrace, podpůrné zdroje, energetická úspora a to, že je vše prováděno na zařízení bez nutnosti internetového připojení.

1. Optimalizace pro Snapdragon: SNPE je navrženo tak, aby využívalo výkon, který Snapdragon mobilní procesory poskytují. Díky tomu je provádění modelů neuronové sítě rychlé a efektivní i na mobilním zařízení. Běh je rychlý a zároveň má malou spotřebu energie. Malá spotřeba energie je u zařízení, která jsou napájena z baterie zásadní, proto je to jedna z nejvýraznějších výhod.
2. Podpora populárních frameworků: SNPE podporuje populární frameworky jako TensorFlow nebo Caffe, díky této podpoře mohou vývojáři používat známé frameworky pro tvorbu neuronových sítí. Díky této podpoře je také mnohem snazší přejít z dosavadního systému a používat SNPE.
3. Optimalizace modelů neuronových sítí: SNPE umožňuje optimalizovat modely neuronových sítí tak, aby bylo dosaženo většího výkonu. SNPE podporuje hardwarovou akceleraci a díky tomu, že procesory Snapdragon jsou od stejné firmy jako SNPE, modely jsou optimalizovány pro běh na těchto procesorech.
4. Jednoduchost integrace: SNPE je navrženo tak, aby bylo jednoduché ho přidat do existující mobilní aplikace. Díky tomu je pro vývojáře jednodušší na SNPE přejít a přidat tak výhody a možnosti, které strojové učení a neuronové sítě poskytují, do své již existující aplikace.
5. Podpůrné zdroje: K SNPE existuje poměrně dobrá dokumentace, tutoriály a ukázky kódu. Díky této dokumentaci je pro vývojáře ještě jednodušší začít a optimalizovat modely na Snapdargon procesoru. K této dokumentaci je k dispozici i fórum, přímo na stránkách Qualcomm, kde se řeší nejčastější problémy nebo dotazy, které vývojáři mají. Kromě fóra je k dispozici blog,

kde Qualcomm zveřejňuje články, které mohou pomoci při vývoji nebo optimalizaci. Jedinou nevýhodou je to, že některá dokumentace není aktuální a některé přístupy se od doby sepsání dokumentace změnily. Problém s fórem je ten, že není dost aktivních uživatelů jako na jiných vývojářských fórech, ale popularita SNPE se postupně zvyšuje. Další problém je, že dokumentace je hodně rozsáhlá a je poněkud složitější se v ní vyznat. To ale pouze ze začátku a po nějakém čase se dá v dokumentaci pro konkrétní produkt od Qualcomm kvalitně orientovat.

6. Energetická úspora: SNPE umožňuje provádění modelů na zařízení tak, aby bylo energeticky efektivní. Díky tomu je výdrž baterie větší, a to je na zařízeních, která jsou na napájení z baterie závislé, důležité.
7. Běh na zařízení: SNPE umožňuje běh pouze na zařízení, díky tomu se zlepšuje zabezpečení a snižuje latence, a to proto, že není nutná síťová komunikace. Mimo tyto výhody běh přímo na zařízení umožňuje zpracovávat data na tomto zařízení. To přináší výhody v situacích, kdy není dostupné, nebo je špatné internetové připojení, nebo pokud je potřeba brát ohled na vyšší bezpečnost a ochranu dat.[38, 39]

Pro lepší představu toho, jaké reálné výhody a výsledky použití SNPE přinese, je na stránkách dokumentace k dispozici srovnání výkonosti modelu Inception-v3, který je na mobilním zařízení spuštěn jednou s použitím SNPE a jednou bez SNPE pouze jako TensorFlow model. Na obrázku číslo 11 je vidět porovnání výkonosti s různým přístupem.

Obrázek 11 – porovnání výsledků SNPE a TensorFlow, zdroj:[40]

Oba přístupy korektně identifikovaly objekt jako počítačovou myš. Avšak rozdíl je hlavně u jistoty, s jakou tento model vyhodnotil objekt jako myš. Rozdíl je i v počtu FPS. To udává, kolik snímků je aplikace schopna zpracovat za sekundu. Zde je rozdíl trojnásobný. Výsledky jsou více jak dvakrát lepší s použitím SNPE oproti TensorFlow. Pro připomenutí jedná se o stejný model neuronové sítě, který objekt identifikoval. Lepších výsledků bylo dosaženo díky použití SNPE a optimalizačních nástrojů, které poskytuje. Přehlednější srovnání i s dalším AI frameworkem je vidět na obrázku 12.

	TensorFlow Inception-v3	Qualcomm NPE Inception-v3	Qualcomm NPE AlexNet
Maximum performance with CPU	1x	1.5x	3 - 3.5x
Maximum performance with GPU	3x	8x	14x

Obrázek 12 – porovnání výkonosti Qualcomm SNPE s populárními frameworky, zdroj: [40]

Na obrázku 12 je patrné, že díky optimalizacím, které SNPE nabízí, lze dosáhnout zlepšení výkonosti u Inception-v3 to je 1.5x. Porovnán s AlexNet optimalizovanou pomocí SNPE nástrojů je to 3–3.5x.

3. Limitace SNPE

Přestože SNPE framework přináší řadu výhod, má i jisté limitace. Tyto limitace je dobré vzít v potaz při rozhodování, jaký framework použít pro vývoje aplikace, protože v určitých případech mohou být tyto limitace zásadní.

Ačkoliv lze se SPNE frameworkem použít řadu frameworků pro vytváření neuronových sítích, je výběr omezen. V současné době jsou podporované frameworky Caffe, Caffe2, ONNX, PyTorch, TensorFlow nebo TFLite. Výběr se postupně rozšiřuje a další frameworky mohou být přidány, proto je důležité zkontrolovat si podporované frameworky na oficiálních stránkách Qualcommu. Před použitím SNPE frameworku je tedy důležité vědět, jaký framework pro vytváření neuronové sítě bude použit a pokud je již model vytvořený, zda je SNPE frameworkem podporovaný.[42]

Další limitací může být typ zařízení a jeho operační systém. SNPE je optimalizováno pro běh na zařízeních s Qualcomm Snapdragon procesorem, což může znamenat, že jeho běh na jiných typech procesorů může být problémový nebo nebude možný vůbec. Framework nabízí možnost běhu na více procesorech, ty jsou však omezeny na procesory od firmy Qualcomm. Mimo limitaci na typ procesoru je zde i limitace na operační systém zařízení. SNPE podporuje běh pouze na zařízeních s Android nebo Linux operačním systémem. Proto i zde je nutné dopředu promyslet na jakém zařízení aplikace poběží, jaký bude jeho procesor a operační systém. [42]

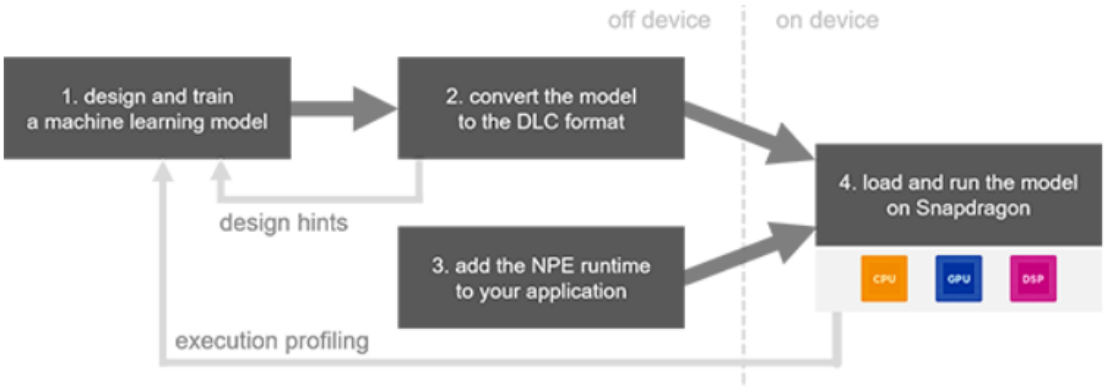
Jako další omezení je dobré zmínit absenci některých operací. Některé specifické operace, které mohou být nutné pro optimalizaci nebo specifické potřeby konkrétních aplikací, mohou ve frameworku SNPE chybět. Pokud je tedy dopředu známo, že v rámci fungování nebo optimalizace aplikace nebo modelu bude potřeba specifických operací, je nutné jejich podporu ověřit v oficiální dokumentaci k tomuto frameworku. SNPE framework nabízí určité operace pro optimalizaci modelu během jeho konverze na .dlc formát jako optimalizace pro určitý typ procesoru nebo například pro model inveption_v3 ve scriptu setup_inceptionv3.py je možné nastavit kvantizaci modelu, která zmenší velikost modelu až o 75 %. Tato operace zmenší přesnost jednotlivých vah v rámci sítě z 32 bitového float čísla na 8 bitové float číslo, čímž se ušetří paměť. [41, 42]

Jednou z limitací, která může nastat spíše pro některé začínající programátory je to, že používání SNPE frameworku může být poměrně komplexní. Dokumentace, která je poskytována k tomuto frameworku je velice rozsáhlá a ve většině případů není psána podrobně pro začínající vývojáře, ale počítá s dalšími znalostmi vývojáře. Další nevýhodou může být absence větší komunity, která by byla kolem tohoto frameworku a v případě potíží je vývojář odkázán pouze na své schopnosti. Podobně jsou na tom i tutoriály od komunity, které kromě oficiálních videí a dokumentace víceméně neexistují. I toto může být limitace a začít s používáním tohoto frameworku nemusí být jednoduchá záležitost.

Protože existují určité limitace, které SNPE framework má, je nutné se podívat na aktuální verzi, zda nebyly tyto limitace vyřešeny. Tyto limitace mohou také být vyřešeny za pomoci dalších nástrojů, které nejsou součástí SNPE frameworku, hlavně v případě modelů neuronových sítí. Například při optimalizaci modelu neuronové sítě lze použít přímo metody, které nabízí framework, a díky tomu odpadne část limitace na chybějící specifické operace.

4. SNPE workflow

Jednou z výhod, které jsou představeny v předchozí kapitole Výhody SNPE oproti jiným přístupům, je podpora populárních frameworků. SNPE nemá vlastní knihovnu pro práci s neuronovými sítěmi, místo toho si mohou vývojáři sami zvolit jeden z několika frameworků (TensorFlow, Caffe/Caffe2, ONNX, PyTorch), kde si mohou vytvořit a natrénovat neuronové sítě. Poté, co je síť natrénována, stačí model převést do formátu „.dlc“ (Deep Learning Container), který je možné použít v SNPE. Pro tento převod jsou v SNPE připraveny nástroje, které mimo převod modelu na formát .dlc, ještě poskytnou statistiky převodu, které mohou pomoci při návrhu a trénování modelu. Tento proces je zobrazen na obrázku 13, kde je vidět rozdíl operací, které se provádí na zařízení a které mimo něj. [39]



Obrázek 13 – workflow práce s SNPE, zdroj: [39]

5. Začínáme se SNPE

Detailní popis toho, jak nastavit, zprovoznit a používat SNPE, bude popsán v praktické části této práce. V této kapitole budou popsány obecné principy a také požadavky, které musí být před používáním SNPE splněny.

Nejprve je třeba splnit systémové požadavky. Pro práci s modelem a jeho převedení do .dlc formátu bude potřeba prostředí s Linuxem a operačním systémem Ubuntu 18.04LTS. Tato verze je doporučena, ale vyplatí se verze u všeho dodržet. Novější verze Ubuntu obsahují novější verze například python knihoven a to působí komplikace při práci se SNPE (lze s nimi pracovat, ale je potřeba dělat určité kroky navíc), proto je doporučeno dodržovat verze a buildy, které jsou na stránkách s návodem doporučeny. Dále na tomto systému některý z podporovaných frameworků pro práci s modelem neuronové sítě. Také je potřeba Android SDK a Android NDK (android-ndk-r19c-linux-x86_64) a Android studio, které není nutné, ale je to jedna z možností, jak vyvíjet aplikaci a používat knihovny

a API z SNPE SDK. Android zařízení se Snapdragon procesorem, na kterém aplikace poběží a na kterém bude možné aplikaci testovat. Dále je postup následující:

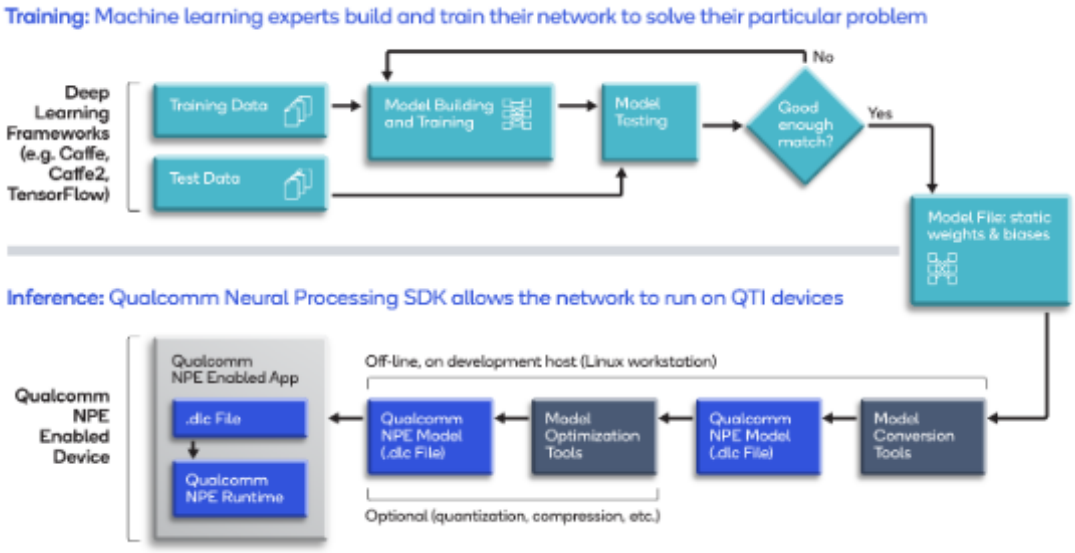
1. Stažení SDK
2. Nainstalování potřebných dependencies (knihoven potřebných pro správný běh SNPE)
3. Nastavení SNPE prostředí a vybraného frameworku
4. Převedení modelu do .dlc formátu
5. Build a spuštění aplikace[39]

Toto je obecný postup pro nastavení prostředí, build a spuštění aplikace na zařízení se Snapdragon procesorem. Není zde zahrnuto vytváření ani testování modelu a to proto, že lze využít už předem vytvořené a natrénované modely jako například AlexNet v Caffe frameworku nebo inception_v3 v TensorFlow frameworku.

6. Strojové učení na zařízení uživatele

S rostoucím výkonem, který nyní poskytují zařízení jako smartphony nebo drony, je možné přesunout výpočetně náročné operace z cloudu rovnou na toto zařízení. Nejpopulárnější aplikace na těchto zařízeních dost závisí na tom, jak si uživatel práci s nimi užívá. Například úprava fotografií v nějaké aplikaci, která používá strojové učení pro ulehčení práce a úpravy fotek, může být pro většinu uživatelů zábava, ale latence, která s tím může být spojená při provádění těchto operací v cloudu, může tento zážitek zničit. V dnešním smartphonu jsou čipy, které mají takový výkon, že tyto operace mohou provádět také, a to velice rychle. SNPE je software umožňující vývojářům využívat výpočetní výkon Snapdragon procesorů, které umožňují spouštět natrénované neuronové sítě na koncových zařízeních, a to bez nutnosti připojení do cloudu. [43]

SNPE umožňuje vývojářům vytvářet aplikace, které využívají modely neuronových sítí vytvořených v některém z populárních frameworků (Caffe/Caffe2, ONNX, TensorFlow) a ty následně spouštět na zařízeních se Snapdragon procesorem a to tak, že tento model převede do .dlc (Deep learning container) formátu. Celý tento proces je zobrazen na obrázku číslo 14.



Obrázek 14 – postup převodu natrénovaného modelu na dlc formát, zdroj:[44]

Na tomto obrázku je vidět, že proces vytváření, trénování a testování modelu, je oddělen a může být prováděn jakýmkoliv preferovaným způsobem. Připravený model se poté jen zkonvertuje do požadovaného .dlc formátu, může se provést optimalizace, a pak je tento soubor předán na zařízení, kde se model využívá. [44]

6. Vývoj aplikace

1. Postup zprovoznění ukázkové aplikace

Nejprve je potřeba připravit prostředí, ve kterém se bude daná aplikace vyvíjet. Na oficiálních stránkách Qualcomm https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/getting-started je návod na přípravu prostředí, ten je však neaktuální, popis některých kroků je vynechaný a některé věci nejsou dostatečně do detailu vysvětleny pro začínající vývojáře. Některý chybějící popis lze najít na dalších stránkách, ale ve velkém množství dokumentace je to velice nepřehledné. Tento návod se tedy snaží zaktualizovat některé informace, vysvětlit detailněji některé kroky a celý proces popsat více do detailu.

1. Systémové požadavky

- Linux x86_64 s Ubuntu 18.04
    - Ubuntu i jeho verze jsou doporučeny, ale při dodržení verzí se vývojář vyhne budoucím komplikacím, nebo extra krokům, které by musel podnikat, pokud zvolí jinou verzi. Pro začínající vývojáře je silně doporučeno držet se verzí i distribuce. Například při použití novější LTS verze ubuntu 20.04 LTS nastane problém s verzí pythonu, která je na této verzi odlišná od verze, která pracuje se SNPE.
- Framework pro práci s neuronovými sítěmi (jeden z níže vypsaných je dostačující)
    - Caffe a Caffe2
    - TensorFlow
    - ONNX
    - PyTOrch
    - TensorFlow Lite
- Android Studio
    - tento požadavek je volitelný, ale značně usnadňuje proces vývoje aplikace a přináší další výhody.
- Android SDK
- Android NDK
    - verze android-ndk-r19c-linux-x86_64

Nově je také možné vývoj provádět na Windows 11, tato práce se však této možnosti nevěnuje, proto bude tato možnost z tohoto návodu vynechána.
(zdroj:https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/getting-started)

2. Proces nastavení prostředí pro vývoj

Prvním krokem je stažení a rozbalení samotného SNPE SDK. To je dostupné na oficiálních stránkách Qualcomm a je nutné si zde vytvořit účet, který je však zdarma. V nabídce je aktuální verze, ale i verze starší. V rámci této práce byla použita verze 2.5.0.4052, která byla v době vytváření této práce nejvyšší dostupnou verzí.

Vzhledem k tomu, že práce s SDK vyžaduje specifické verze pythonu a knihoven, stojí za zvážení využívání virtuálního python environmentu, kde budou odděleny systémové balíčky od těch nově nainstalovaných. Tímto způsobem se lze vyhnout budoucím komplikacím a také komplikacím při updatu systému, který by mohl změnit nainstalované verze. Nutné to však není a lze pracovat i bez virtuálního environmentu.

Nyní je potřeba nainstalovat potřebné systémové závislosti, které jsou spojené s SDK. Základní závislosti lze nainstalovat pomocí příkazu: sudo apt-get install python3-dev python3-matplotlib python3-numpy python3-protobuf python3-scipy python3-skimage python3-sphinx wget zip

Tyto závislosti jsou potřebné pro používání samotného SDK, další mohou být potřeba nainstalovat v průběhu podle používaných frameworků a nástrojů třetích stran.

Dalším krokem je ověřit, zda jsou všechny potřebné závislosti nainstalované, a to pomocí scriptu, který je součástí SNPE SDK. Tento script najdeme ve složce, kde je rozbaleno SDK v podsložce bin s názvem dependencies.sh. Pro spuštění tohoto scriptu bude nejspíše potřeba udělit práva pro spuštění, a to příkazem chmod +x dependencies.sh, za předpokladu spuštění tohoto příkazu z adresáře /snpe-2.5.0.4052/bin/.

Při správném nainstalování všech závislostí, výsledek scriptu vypadá jako na obrázku číslo 15. V případě, že některé závislosti budou chybět, tento script je vypíše a poté stačí pouze nainstalovat ty, které byly vypsány jako výstup tohoto scriptu, dokud nebudou všechny závislosti nainstalovány.

Obrázek 15 – výsledek scriptu dependencies.sh, zdroj: vlastní zpracování

Následujícím krokem je ověření, zda jsou nainstalovány všechny python závislosti ve správných verzích, otestované verze jsou numpy v1.16.5, sphinx v2.2.1, scipy v1.3.1, matplotlib v3.0.3, skimage v0.15.0, protobuf v3.6.0, pyyaml v5.1. To lze provést pomocí scriptu, který nalezneme ve stejné složce jako při předchozím kroku a to /snpe-2.5.0.4052/bin/. I tentokrát bude nejspíše potřeba přidělit práva pro spuštění tohoto scriptu příkazem chmod +x check_python_depends.sh. Opět platí stejné jako u předchozího scriptu. Pokud nějaká závislost chybí, je vypsána varovná hláška a stačí tuto závislost doinstalovat. Další varovnou hláškou, kterou tento script může vypsat je to, že python modul je nainstalovaný pomocí příkazu apt-get a pip zároveň, což by mohlo způsobovat v budoucnu errory, proto je lepší mít tyto moduly nainstalované pouze jednou, ale není to nutná podmínka, jedná se pouze o varování. V předchozích verzích byl tento script neaktuální a padal na erroru, který se dal vyřešit přepsáním příkazu python na python3 na řádce číslo 12. Tento problém se také dá vyřešit nastavením python alternativ tak, aby ukazovaly na python3. Toho lze docílit pomocí příkazů:

```
$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1
$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.6 2
```

Tyto příkazy přidají nainstalované python verze do listu alternativ. Poslední číslo určuje prioritu, pokud není manuálně vybraná žádná z alternativ. Jelikož se jedná o prioritu, tak vyšší je brána za prioritnější, proto defaultně je nyní používán python 3.6. $ update-alternatives --list python

Tento příkaz vypíše dostupné alternativy python verzí, uložených v listu alternativ. Pomocí dalšího příkazu můžeme mezi těmito alternativami přepínat.

$ sudo update-alternatives --config python + číslo alternativy, na kterou se chceme přepnout.

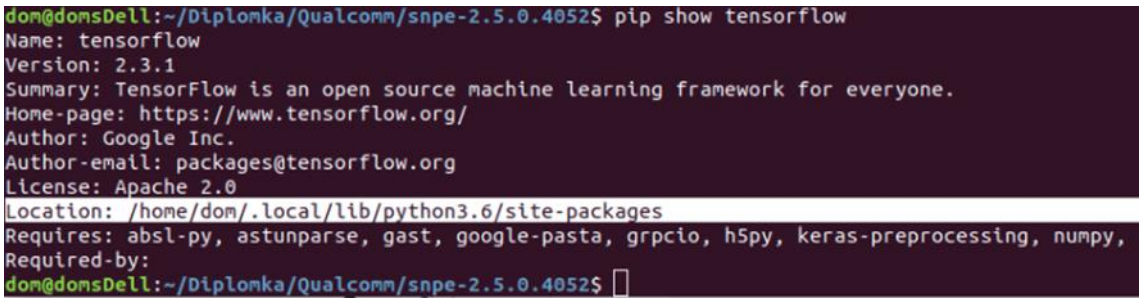
Jako další je potřeba nainstalovat některý z frameworků, ve kterém budou zpracovávány modely neuronových sítí. Na výběr je hned z několika populárních frameworků a to Caffe, Caffe2, TensorFlow, TensorFlow Lite, ONNX a PyTorch. V rámci tohoto tutoriálu byl vybrán framework TensorFlow. Jeho instalace je popsána na oficiálních stránkách https://www.tensorflow.org/install. Otestované verze jsou v1.6 a v2.3 [45]. Tuto verzi lze nainstalovat pomocí nástroje pip, a to příkazem pip install tensorflow==2.3.1. Opět je doporučeno držet se verze 2.3, ale lze použít i jiné verze. Existuje pak ale riziko, že se mohou objevit problémy navíc, které bude nutné vyřešit.

Pro snadnější vývoj a práci s mobilními aplikacemi je třeba nainstalovat Android Studio. Návod na instalaci lze najít na oficiálních stránkách https://developer.android.com/studio/install. Díky nainstalovanému Android Studiu lze snadno nainstalovat Android SDK a NDK. Podle oficiální dokumentace je NDK potřeba pouze v případě, že build aplikace bude prováděn v nativním C++. SDK je také nepovinné a je potřeba pro vývoj Android aplikace. V dokumentaci jsou uvedeny verze NDK android-ndk-r19c-linux-x86_64 a SDK verze 23 a build tool verze 23.0.2, avšak pro novější aplikace lze použít novější verze. Je možné, že s novějšími verzemi bude potřeba udělat pár úprav v kódu aplikace, které je součástí SNPE SDK. Android NDK a SDK lze nainstalovat v Android Studiu a to tak, že se zvolí File -> Settings -> Appearance & Behavior -> System Settings -> Android SDK. Na této kartě jsou záložky SDK Platforms a SDK Tools. V SDK Platforms můžeme stáhnout požadovanou verzi Android SDK a v záložce SDK Tools dostahovat SDK Tools a NDK a další potřebné nástroje pro vývoj aplikace.

Dalším nepovinným krokem je nastavení proměnné ANDROID_NDK_ROOT na cestu k složce s NDK. To lze provést pomocí příkazu export ANDROID_NDK_ROOT=~/Android/Sdk/ndk-bundle. Cesta k souboru se může lišit podle toho, kde je NDK umístěno. V případě, že tato proměnná není nastavena, script, který s ní pracuje, bude automaticky hledat složku ndk-build.

Jako poslední krok při přípravě prostředí je nutné nastavit framework neuronových sítí. V rámci tohoto návodu byl vybrán framework TensorFlow. Postup pro jiné frameworky je však podobný a návod je dostupný na https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/getting-started. V kořenové složce, kde je rozbaleno SNPE SDK, je třeba spustit script, který je v bin složce s názvem envsetup.sh a to pomocí příkazu source bin/envsetup.sh -t $TENSORFLOW_DIR, kde místo $TENSORFLOW_DIR je

potřeba zadat cestu k instalaci TensorFLow. Umístění instalace TensorFlow lze zjistit příkazem pip show tensorflow, který vypíše informace, mezi kterými je i umístění této instalace pod Location jak je zvýrazněno na obrázku 16.



Obrázek 16 – Umístění instalace frameworku tensorflow, zdroj: vlastní zpracování

Předchozí příkaz pro nastavení frameworku by tedy v této konkrétní situaci vypadal source bin/envsetup.sh -t ~/.local/lib/python3.6/site-packages, za předpokladu spuštění příkazu z kořenového adresáře SNPE. Tento script nastaví potřebné proměnné, zkopíruje knihovny a upraví Android AAR soubor.

Tímto je připraveno prostředí pro vývoj a práci se SNPE SDK. Na Linuxu s Ubuntu 18.04LTS jsou nainstalovány potřebné systémové balíčky a python moduly. Byl nainstalován TensorFlow framework a spuštěny potřebné scripty pro nastavení prostředí na tento framework. Dále je zde Android SDK a NDK spolu s Android Studiem pro vytváření a úpravu aplikací.

3. Příprava modelu a spuštění ukázkové aplikace

Pro potřeby ukázkové aplikace je součástí SNPE SDK script, který stáhne a překonvertuje předtrénovaný model inception_v3 v TensorFlow formátu na .dlc model. Pro stáhnutí a překonvertování tohoto modelu s ukázkovými daty slouží příkaz python models/inception_v3/scripts/setup_inceptionv3.py -a ../../inception -d, který je spuštěn z kořenového adresáře SNPE. Přepínač -a (--assets_dir) je povinný a jako argument má cestu ke složce, kde jsou staženy potřebné soubory. Tento adresář může být prázdný a pomocí přepínače -d (--download) se stáhnou potřebné soubory do tohoto nově vytvořeného adresáře. Dále jsou dostupné přepínače -h pro nápovědu a popis tohoto scriptu. V této nápovědě jsou popsány další přepínače, které se mohou hodit pro specifické situace. Ukázkový předtrénovaný model spolu s ukázkovými obrázky je dostupný ke stažení na adrese https://storage.googleapis.com/download.tensorflow.org/models/inception_v3_2016_08_28_frozen.pb.tar.gz. Přepínačem -r (--runtime) může script provést kroky navíc, aby optimalizoval model pro běh na konkrétním zařízení. Na výběr jsou cpu, gpu,

dps nebo aip (HTA). Defaultně je tento přepínač nastaven na cpu, proto je možné ho vynechat. Po dokončení tohoto scriptu, jsou nakopírovány předpřipravené soubory modelu v dlc formátu spolu s ukázkovými obrázky do složky /models/inception_v3. Tyto soubory jsou také optimalizované pomocí tohoto scriptu.

S takto předpřipravenými soubory, lze začít s přípravou samotné aplikace. Nejprve je potřeba překopírovat snpe-release.aar soubor, který je součástí SNPE SDK a je ve složce android, do /app/libs. V tomto případě do složky /examples/android/image-classifiers/app/libs, kde jsou knihovny potřebné pro běh aplikace a využívání SNPE SDK v rámci aplikace. Následně je potřeba spustit script, podle zvoleného frameworku, v tomto případě to provede příkaz bash ./setup_inceptionv3.sh. Tento příkaz je nutné spustit ze složky /examples/android/image-classifiers, protože cesty, které jsou v něm uvedené, jsou relativní. Tento script zkopíruje potřebné soubory a ukázkové obrázky a vloží je do aplikace jako resource soubory.

Nyní je čas spustit Android Studio a otevřít projekt ve složce /examples/android/image-clasifiers. Při prvním spuštění je potřeba povýšit verzi gradle na novější, v případě této ukázky byla zvolena verze 7.2. Tato verze se dá nastavit automaticky pomocí nabídky Android Studia, které nabídne aktualizace nejen pro gradle, ale i pro další moduly, pokud je potřeba. Manuálně se tento update dá provést pomocí změny distributionUrl v gradle-wrapper.properties. Dále je třeba zkontrolovat, zda je přidána závislost v app/build.gradle (ne v image-classifiers/build.gradle), kde je potřeba přidat nejprve do repositories flatDir { dirs 'libs' } a do části dependencies implementation(name: 'snpe-release', ext:'aar'), zde bylo původní compilation, které je deprecated nahrazeno implementation (více informací je dostupných v dokumentaci Gradle). Díky takto přidané závislosti nyní lze využívat SNPE SDK v rámci kódu aplikace. V tomto stavu lze provést build aplikace, a pokud je dokončen bez problémů, lze pokračovat. V opačném případě je potřeba vyřešit errory, které při buildu nastaly.

Pokud proběhlo vše bez problémů, lze aplikaci distribuovat na zařízení. Toto zařízení musí mít Snapdragon procesor minimální verze 855. Seznam komerčních zařízeních, které mají Snapdragon procesor, který umožňuje spouštět aplikaci spolu se SNPE SDK a spouštět neuronové sítě, je dostupný na adrese https://www.qualcomm.com/snapdragon/device-finder.

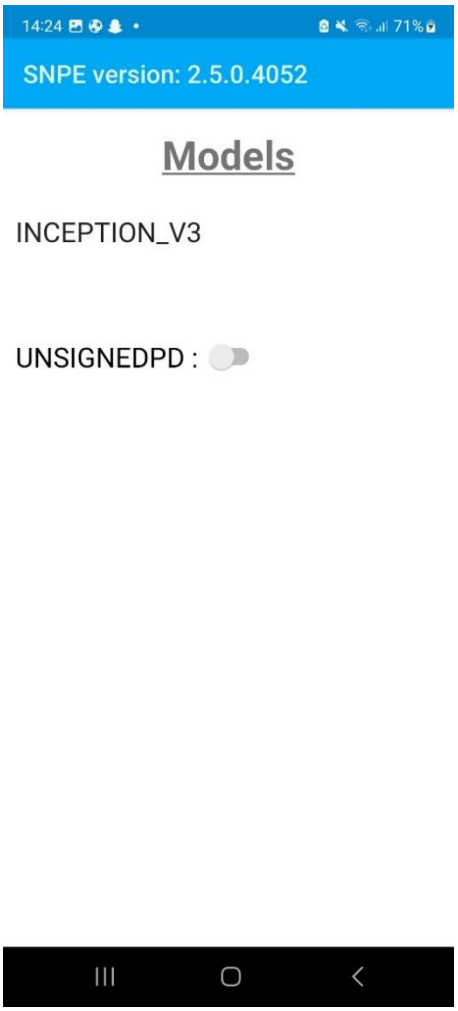
Pro spuštění aplikace na android zařízení přímo z android studia je potřeba nejprve nastavit telefon. První krok, který se musí na cílovém zařízení provést, je zapnutí vývojářských možností. Toho lze docílit tak, že v nastavení telefonu přejdeme do „o telefonu“ dále zvolíme „informace o software“ a přesuneme

se na položku „číslo sestavení“. Nyní je zapotřebí 10x kliknout na toto číslo sestavení. Měla by se zobrazit hláška s tím, že x kliknutí vás dělí od toho, abyste se stal vývojářem. Následně se zobrazí hláška, že jste vývojářem a tím jsou vývojářské možnosti zapnuty. Druhý krok je povolení ladění přes usb. V nastavení nyní nalezneme „vývojářské možnosti“. Po jejich rozkliknutí najdeme možnost ladění přes usb a ta musí být zapnuta, aby bylo možné naši aplikaci na mobilním zařízení spustit pomocí android studia na mobilu, který má tuto možnost zapnutou a je připojen pomocí usb k počítači. Android studio využívá adb shell pro naistalování a spuštění aplikace na cílovém zařízení.

Pokud je aplikace spuštěna touto cestou, jednou z výhod je, že v android studiu je konzole, ve které můžeme vidět loggy, které se při běhu aplikace vypisují. Mimo základní informace jsou zde i errory a díky nim lze v případě problémů odhalit, kde problém nastal a jeho vyřešení bude snazší. Tento způsob také umožňuje provádět debugging přímo za běhu aplikace na zařízení, což může být také nápomocné. Jelikož tato aplikace nebude fungovat v emulátoru, kvůli chybějícímu Snapdragon procesoru, lze takto zjistit stav aplikace na konkrétním zařízení v jednotlivých fázích.

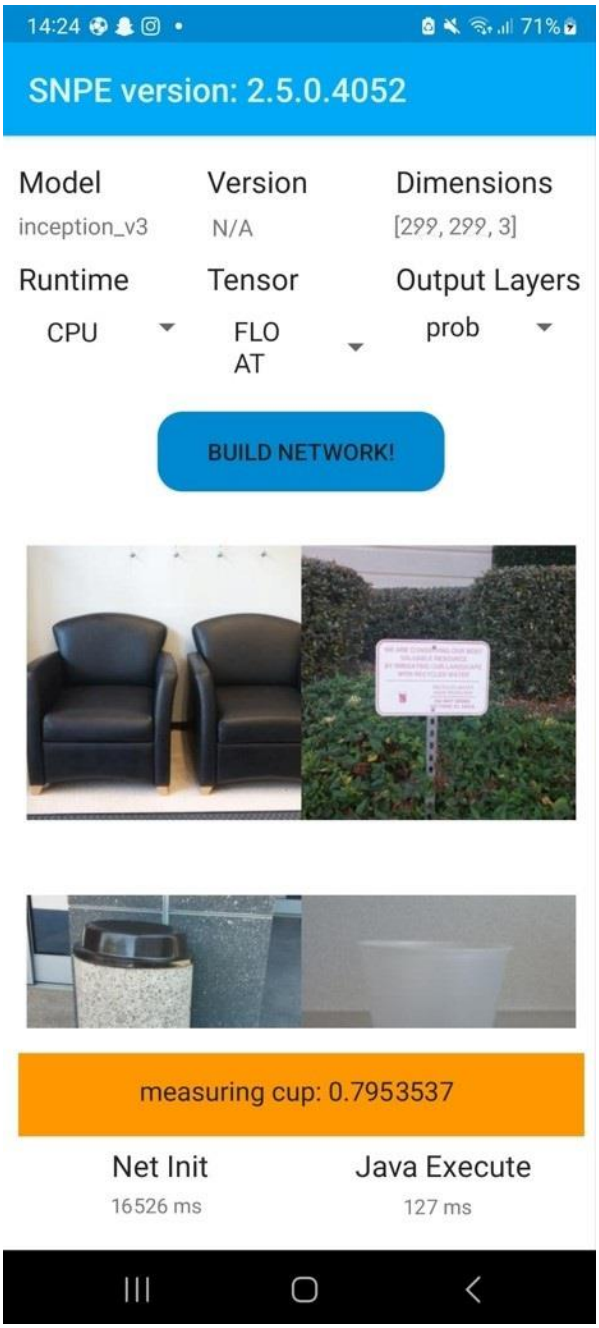
4. Práce s aplikací

Po spuštění aplikace se na hlavní obrazovce zobrazí seznam dostupných modelů neuronových sítí a „UNSIGNEDPD“ přepínač. Tento přepínač určuje zda model, který se bude používat, má platný soukromý klíč. Při převodu modelu z originálního formátu na .dlc formát je tento model skriptem podepsán privátním klíčem, aby se zaručila pravost modelu. Při vývoji a testování modelů však lze používat i modely, které tento podpis nemají či jsou ze spolehlivého zdroje. Tyto modely a jejich bezpečnost jsou ověřovány pomocí jiného mechanismu. Model, který je použit v rámci této ukázkové aplikace, byl konvertován pomocí originálního skriptu, který je součástí SNPE sdk, a proto by měl mít podpis v pořádku a tento přepínač může zůstat vypnutý. Tato obrazovka s modelem a přepínačem je vidět na obrázku 17 níže.



Obrázek 17 – úvodní obrazovka aplikace, zdroj: vlastní zpracování

Dalším krokem je kliknutí na název modelu, a díky tomu se přesuneme na další obrazovku.



Obrázek 18 – práce s modelem a klasifikace obrázků, zdroj: vlastní zpracování

Na obrázku 18 výše, jsou vidět informace o modelu, jeho název, verze a „dimensions“, které udává rozměry vstupních dat, tedy obrázku. Čísla 299, 299 určují rozměr obrázku a číslo 3 počet kanálů barev (červená, zelená a modrá). Na dalším řádku je poté runtime, tensor a output layers. Runtime určuje, který procesor bude zpracovávat požadavky. Dostupných je více procesorů, ale pro některé je potřeba učinit speciální kroky navíc během přípravy aplikace a modelu. Některé skripty například potřebují přidat přepínač pro správné fungování

s daným procesorem či je zapotřebí úprava makefilu a další kroky. Defaultně bez nadstandartních kroků mimo ty, které jsou popsány výše v této práci, by měly fungovat CPU a GPU runtime. Pokud chceme docílit lepších výsledků pomocí GPU, je zapotřebí malých úprav v procesu vytváření aplikace, které optimalizují model pro běh speciálně na GPU. Float hodnota v tensor značí datatype čísla, která jsou použita v rámci jednotlivých tensorů, díky tomu mohou tato čísla mohou být velice přesná. Poslední je zde output layers. Hodnota prob zde reprezentuje typ výstupní vrstvy. Model inception_v3 má softmax výstupní vrstvu a ta generuje pravděpodobnost třídy pro daný obrázek (prob = probability). Výstupem je tedy pravděpodobnost, s jakou vstupní obrázek spadá do konkrétní třídy.

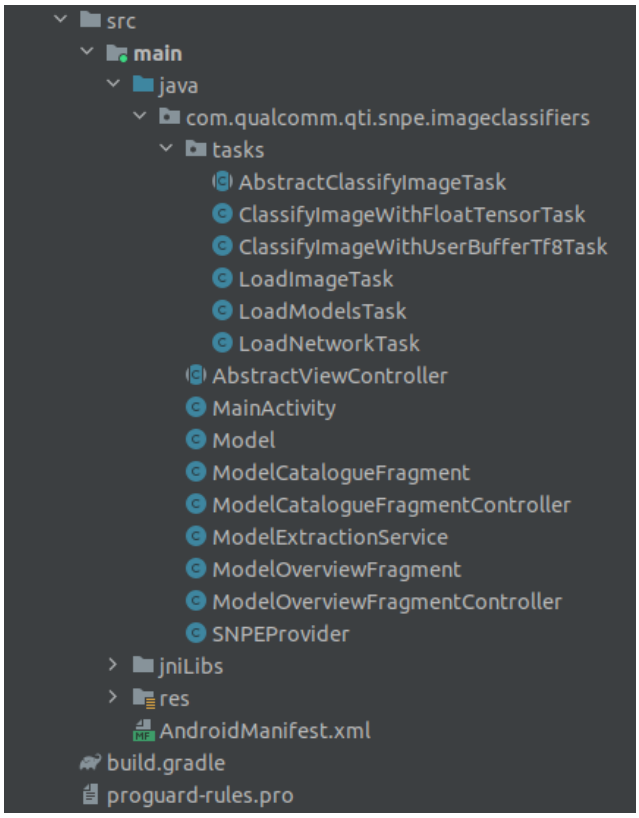
Po správném nastavení těchto hodnot je zapotřebí nejdříve sestavit neuronovou síť, než bude možné přejít ke klasifikaci obrázků. K tomu slouží tlačítko „build network!“. Po kliknutí na toto tlačítko se na pozadí vytvoří potřebné objekty pro běh aplikace a nastaví se hodnoty do proměnných. Díky tomu je následně možné provádět klasifikaci obrázků, které následují po tomto tlačítku. Pokud by nebyla síť správně sestavena, aplikace spadne. Pokud klikneme na obrázek, který chceme klasifikovat předtím, než je síť sestavena, objeví se upozornění, že je nejdříve potřeba síť sestavit, aby bylo možné klasifikovat obrázky.

Pokud je neuronová síť správně sestavena, tak po kliknutí na obrázek, který chceme klasifikovat a který je ve správném formátu a splňuje podmínky vstupních dat pro neuronovou síť, je pod obrázky do oranžového pole vypsána patřičná třída a k ní desetinná hodnota, která reprezentuje, na kolik procent tento obrázek podle neuronové sítě spadá do této kategorie.

Na konci obrazovky jsou pak statistiky, které ukazují rychlost sestavení neuronové sítě a rychlost provádění klasifikace. Přesto, že je aplikace pouze ukázková, model, který klasifikuje jednotlivé obrázky je plnohodnotný. Jedná se o model inception_v3 od společnosti Google.

2. Kód aplikace

Kód aplikace není složitý. Na obrázku 19 lze vidět celková struktura projektu. V dalších složkách jsou pouze knihovny a soubory potřebné pro běh aplikace jako model neuronové sítě, obrázky a další soubory.



Obrázek 19- struktura projektu, zdroj: vlastní zpracování

Většina tříd v hlavním balíčku neobsahuje nijak zajímavý kód. Z velké části je zde pouze nastavování proměnných, které jsou později potřebné pro správné výpočty a provádění kódu, který je specifický pro rozdílné prostředí, nebo se liší nastavením například na jakém procesoru budou výpočty prováděny, typ obrázku a další proměnné v kódu. Mimo nastavování proměnných je zde také provolávání metod, které extrahují data z obrázků pro další zpracování těchto dat pomocí neuronové sítě a jejich klasifikace. Také jsou zde metody pro načtení souborů a knihoven potřebných pro běh aplikace a také načtení souboru, kde je uložen model neuronové sítě ve formátu .dlc.

Na obrázku 19 v podsložce tasks, jsou třídy, které slouží k načtení a sestavení objektů, získání dat z obrázků, reprezentace obrázku v číselných hodnotách a provedení samotné klasifikace. Ve třídě AbstractClassifyImgeTask, jsou metody, které jsou využívané ostatními třídami pro získání dat z obrázku. Ve třídě LoadNetworkTask je poté sestavení objektu neuronové sítě a následně ve třídě ClassifyImageWithFloatTensorTask je kód pro vytvoření tensoru se vstupními daty pro vstupní vrstvu neuronové sítě a následné provedení klasifikace neuronovou sítí.

Hlavní částí kódu je vytvoření a nakonfigurování objektu modelu neuronové sítě, který poskytuje právě SNPE framework.

```
NeuralNetwork network = null;
try {
    final SNPE.NeuralNetworkBuilder builder = new SNPE.NeuralNetworkBuilder(mApplication)
            .setDebugEnabled(false)
            .setRuntimeOrder(mTargetRuntime)
            .setModel(mModel.file)
            .setCpuFallbackEnabled(true)
            .setUseUserSuppliedBuffers(mTensorFormat != SupportedTensorFormat.FLOAT)
            .setUnsignedPD(mUnsignedPD);
    if (mUnsignedPD){
        builder.setRuntimeCheckOption(NeuralNetwork.RuntimeCheckOption.UNSIGNEDPD_CHECK);
    }

    final long start = SystemClock.elapsedRealtime();
    network = builder.build();
    final long end = SystemClock.elapsedRealtime();

    mLoadTime = end - start;
} catch (IllegalStateException | IOException e) {
    Log.e(LOG_TAG, e.getMessage(), e);
}
return network;
```

Obrázek 20 – vytvoření objektu neuronové sítě, zdroj: vlastní zpracování

Na obrázku 20 lze vidět vytvoření objektu neuronové sítě, který je následně využívaný pro klasifikaci obrázků. Je zde vidět návrhový vzor „builder“, který nastaví potřebné proměnné jako například na jakém procesoru model poběží, soubor s modelem v .dlc formátu a další. Když je builder správně nastaven je vytvořen objekt network a změřen čas, který byl zapotřebí na vytvoření objektu, kvůli přehledu výkonosti.

Další kód, který stojí za pozornost je vytvoření, konfigurace a načtení vstupních dat v podobě FloatTensor objektu, který je také součástí SNPE frameworku. Tento tensor obsahuje hodnoty, které jsou získané z obrázku, který bude klasifikován a slouží jako vstupní hodnota pro vstupní vrstvu neuronové sítě. Tento kód lze vidět na obrázku 21.

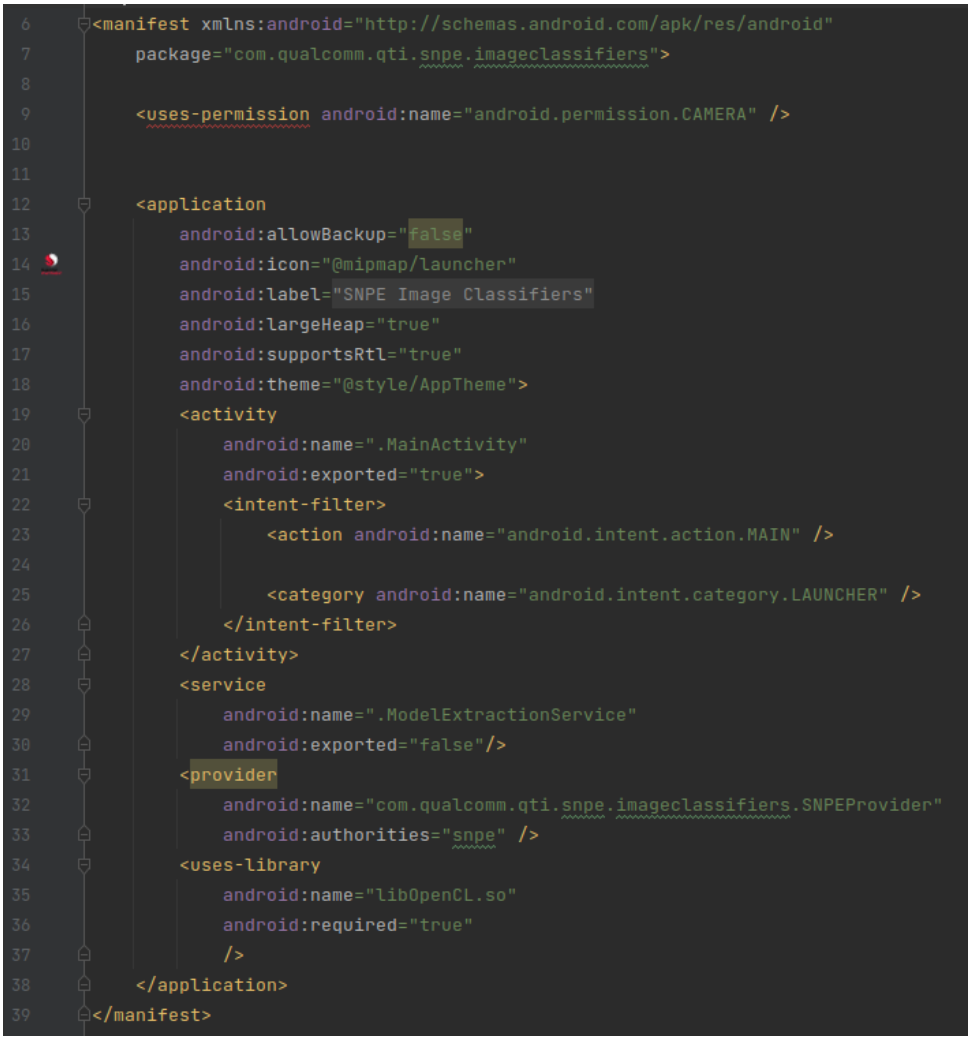
```
final FloatTensor tensor = mNeuralNetwork.createFloatTensor(
        mNeuralNetwork.getInputTensorsShapes().get(mInputLayer));

loadMeanImageIfAvailable(mModel.meanImage, tensor.getSize());

final int[] dimensions = tensor.getShape();
final boolean isGrayScale = (dimensions[dimensions.length -1] == 1);
float[] rgbBitmapAsFloat;
if (!isGrayScale) {
    rgbBitmapAsFloat = loadRgbBitmapAsFloat(mImage);
} else {
    rgbBitmapAsFloat = loadGrayScaleBitmapAsFloat(mImage);
}
tensor.write(rgbBitmapAsFloat,  i: 0, rgbBitmapAsFloat.length);

final Map<String, FloatTensor> inputs = new HashMap<>();
inputs.put(mInputLayer, tensor);

final long javaExecuteStart = SystemClock.elapsedRealtime();
final Map<String, FloatTensor> outputs = mNeuralNetwork.execute(inputs);
final long javaExecuteEnd = SystemClock.elapsedRealtime();
mJavaExecuteTime = javaExecuteEnd - javaExecuteStart;

for (Map.Entry<String, FloatTensor> output : outputs.entrySet()) {
    if (output.getKey().equals(mOutputLayer)) {
        FloatTensor outputTensor = output.getValue();

        final float[] array = new float[outputTensor.getSize()];
        outputTensor.read(array,  i: 0, array.length);

        for (Pair<Integer, Float> pair : topK( k: 1, array)) {
            result.add(mModel.labels[pair.first]);
            result.add(String.valueOf(pair.second));
        }
    }
}

releaseTensors(inputs, outputs);
```

Obrázek 21 – vytvoření tensoru, klasifikace obrázku neuronovou sítí, zdroj: vlastní zpracování

Na obrázku 21 v jeho druhé polovině můžeme vidět příkaz `mNeuralNetwork.execute(inputs)`. Tento příkaz provede zpracování vstupních dat neuronovou sítí. Výsledek tohoto příkazu je následně zpracován a výsledné hodnoty ve výstupní vrstvě (pravděpodobnosti s jakou obrázek spadá do příslušné kategorie), jsou poté uloženy do pole a předány k dalšímu zpracování.

Na obrázcích 20 a 21 je hlavní logika sestavení neuronové sítě a následné použití sítě s využitím FloatTensoru s daty pro vstupní vrstvu neuronové sítě. Data z výstupní vrstvy jsou poté dostupná ve FloatTensor objektu ve výstupní vrstvě.

Dalším souborem, který je důležitý v rámci tohoto projektu, je soubor AndroidManifest.xml, který je zobrazen na obrázku 22.



Obrázek 22 – nastavení v souboru AndroidManifest.xml, zdroj: vlastní zpracování

Na tomto obrázku lze vidět nastavení, které je poměrně běžné pro všechny android aplikace. Oproti kódu, který je poskytován jako součást SNPE frameworku v image-classifier demo app, je zde navíc přidán <provider…/>, který je nutný v novějších verzích androidu. Další velice důležitou změnou, je přidání závislostí <uses-library…/>, která zajistí přístup aplikace ke knihovnám na cílovém zařízení. Bez tohoto nastavení nebude mít aplikace povolení k využívání těchto knihoven. Toto nastavení se bude lišit v závislosti na verzi androidu a zařízení, na kterém bude aplikace běžet. Některé knihovny mohou na zařízení chybět, je tedy potřeba ověřit, že všechny knihovny jsou na zařízení dostupné a přidat je do AndroidManifest.xml,

aby je aplikace mohla využívat. Pokud tyto knihovny nejsou dostupné na zařízení, je potřeba je doinstalovat nebo nakopírovat na cílové zařízení. Některé knihovny jsou součástí SNPE frameworku.

3. Problémy při vývoji

Přesto, že se práce se SNPE může zdát jednoduchá, je potřeba si některé věci pohlídat. Nejprve pro správné nastavení prostředí, je nutné dodržet otestované verze, protože verze novější by mohly způsobit problémy při spouštění scriptů nebo neobvyklé chování. Kromě dodržení verzí, na které upozorňují i kontrolní scripty, je třeba vyřešit warningy, které se objeví v těchto scriptech. Ty by mohly v pozdější fázi způsobit errory nebo by další scripty nebo finální aplikace nemusela běžet správně, právě kvůli nějaké maličkosti, která byla vynechána na začátku. Pokud tedy s jistotou nelze použít novější verzi nebo nástroj vynechat, je doporučeno udělat vše podle návodu. Plno nástrojů není potřebných pro spuštění android aplikace, protože spousta z nich je použita pro build a běh C++ aplikace, nebo pro nástroje, které slouží pro další vývoj, ale pro to jsou potřeba hlubší znalosti problému.

Další potencionální nebezpečí, které je třeba eliminovat, je úprava makefilů a dalších scriptů. SNPE dokumentace neposkytuje konkrétní návod na to, jak soubory upravit tak, aby vše fungovalo. Tyto soubory je potřeba upravit vždy s ohledem na konkrétní použití SNPE. Podle toho v jakém prostředí aplikace poběží, jaký procesor bude zpracovávat požadavky a běh neuronové sítě, konkrétní architektura a plno dalších proměnných, které jsou závislé na individuálním použití SNPE. V tomto případě se mohou lišit i použité nástroje třetích stran a dokonce i verze. Například některé scripty umožňují výběr mezi pythonem verze 2 a 3. V tomto případě je potřeba zakomentovat a odkomentovat některé řádky ve scriptu a případně poskytnout doplňující informace jako například cestu k instalaci daného nástroje. Další změny ve scriptech se mohou týkat architektury cílového zařízení.

Dokumentace je v případě SNPE framworku rozsáhlá a ve většině případů i podrobná, ale má i dost nedostatků. V některých případech byla zastaralá a některé informace v dokumentaci již nebyly pravdivé. Dalším problémem v dokumentaci je její rozsáhlost. Qualcomm poskytuje více nástrojů a přístupů a dokumentace není úplně dobře a jasně rozdělena, který článek se čeho týká. Vývojář se v ní lehce ztratí a je potřeba dávat pozor, jakého nástroje se konkrétní dokumentace týká, pro které nástroje platí a pro které ne.

Kromě rozsáhlosti dokumentace je zde ještě jeden problém. Některé postupy jsou pospány na více místech v dokumentaci. Problém nastává ve chvíli, kdy na jednom

místě je dokumentace stručná a jedná se spíše o přehled a jedna její část jde více do hloubky a tím tato stránka může působit, že obsahuje detailní popis, ale na jiném místě je poté dovysvětlena jiná část podrobněji. Takto se vývojář dozví, pokud dohledá příslušnou dokumentaci, některé důležité informace až po projití většího množství dokumentace a může to způsobit mystifikaci, protože vývojář bude pokračovat v domnění, že provedl příslušné kroky podle návodu, ale přitom někde jinde v dokumentaci je ještě další upřesnění nebo změny, které jsou potřeba udělat v jistých případech.

Qualcomm sice nabízí framework, který umožňuje spouštění neuronových sítí přímo na zařízení a usnadňuje developerům práci, ale rozhodně není ideální pro začátečníky. Pro práci s tímto frameworkem jsou potřeba pokročilé znalosti z různých oblastí a znalost různých technologií a nástrojů, jako například práce s Linuxem, správné nastavení a práce s nástroji potřebnými pro práci s umělou inteligencí, různé frameworky neuronových sítí a pokročilé znalosti ve vývoji aplikací pro android zařízení. Obecně je potřeba, aby byl vývojář zkušenější a věděl co dělá, protože technologie, které jsou nutné pro vývoj jsou dost provázané a je zapotřebí znalostí, jak tyto technologie fungují společně a co vše je potřeba udělat, aby spolu fungovaly.

Jedna z nevýhod SNPE frameworku je kombinace různých technologií, ve kterých se musí vývojář dobře vyznat. Poněkud nepřehledná a velice rozsáhlá dokumentace tomu také nenapomáhá. Ostatní přístupy, které umožňují provádět podobné operace na mobilním zařízení mají tu výhodu, že nejsou tolik spojeny s různými technologiemi a je kolem nich větší komunita, která může pomoci. Díky této komunitě také vznikají tutoriály a na fórech lze nalézt řešení běžných problémů. Fórum na Qualcomm webu obsahuje sice poměrně hodně článků, ale většina jsou jen otázky. Pokud je položena nějaká otázka, většinou je velice málo vývojářů, kteří odpoví nebo umějí odpovědět. Další problém je, že když už přijde odpověď na otázku většinou je pouze jednorázová a další diskuse nebo doplňující otázky zůstávají bez odpovědi.

7. Závěr

V této práci byl na začátku představen a vysvětlen princip strojového učení a počítačového vidění, kde byly popsány principy detekce objektů a jejich zařazení do různých tříd neboli klasifikace. Následně byly představeny konvoluční neuronové sítě, u kterých byl popsán a podrobněji vysvětlen jejich princip a fungování. Jako další byl představen framework SNPE od firmy Qualcomm. Bylo popsáno, jaké výhody tento framework přináší oproti klasickým přístupům a jeho architektura a základní principy, jak tento framework funguje a tedy k čemu je vhodné ho použít, aby bylo možné využít jeho výhod. Mimo výhody byly představeny i limitace, a to nejen na jeho použití, ale především hardwarové limitace. Právě kvůli tomu, že firma Qualcomm vyvíjí i procesory, tento framework je s touto technologií spojen a lze ho využívat pouze na určitém hardwaru. Konkrétní seznam procesorů, na kterých lze spouštět neuronové sítě, je uveden na oficiálním webu firmy Qualcomm, kde lze také dohledat komerční zařízení, která obsahují Snapdragon procesory. Právě Snapdragon procesory, které firma Qualcomm vyrábí, jsou podmínkou pro použití SNPE frameworku a to od verze 855 a vyšší.

Po představení principů strojového učení, počítačového vidění, konvolučních neuronových sítí a SNPE frameworku, byla v této práci zdokumentována příprava prostředí pro vývoj a práci se SNPE frameworkem. Pro práci je potřeba Linux x86_64 s Ubuntu 18.04 LTS, jeden z frameworků pro práci s neuronovými sítěmi (v této práci byl použit framework TensorFlow) a Android studio, které není nutně potřeba, ale usnadní práci, vývoj a případnou distribuci aplikace. Dále byl zdokumentován proces přípravy a instalace prostředí pro práci se SNPE frameworkem. Zde byla popsána instalace potřebných nástrojů a knihoven, úprava a spouštění scriptů a další nastavení, které je potřebné pro vývoj a práci se SNPE frameworkem. Poté co bylo prostředí připraveno, byl připraven model konvoluční neuronové sítě. Pro účely této práce byl zvolen model inception_v3 od firmy Google, který je již natrénovaný na velkém množství dat a obsahuje 1000 tříd, do kterých mohou být obrázky klasifikovány. Tento model byl stažen a připraven pomocí scriptů, které jsou součástí SNPE frameworku a součástí tohoto modelu byl i seznam tříd a ukázkové obrázky ve správném rozměru a formátu, pro jejich klasifikaci. Tento model byl převeden na .dlc formát, se kterým umí SNPE framework pracovat, také pomocí již připravených scriptů.

Tímto bylo vše připraveno a dále se tato práce věnovala popisu práce s ukázkovou aplikací, která je součástí SNPE. Byly popsány jednotlivé proměnné a jejich význam

pro běh aplikace. Byl zde také představen a okomentován základní kód této ukázkové aplikace.

Jako další byly sepsány problémy při vývoji a na co je potřeba dát pozor při práci s tímto frameworkem a některé jeho limitace, na které je třeba se připravit.

SNPE framework nabízí poměrně zajímavé urychlení práce a její usnadnění spolu s velice zajímavým výkonem na Snapdragon procesorech. To je však podmíněno dobrou znalostí v ostatních oblastech a dalších technologiích. Mimo znalost těchto technologií samostatně je zapotřebí znalost funkčnosti těchto technologií společně, což v jejich množství může být výzva. Usnadnění vývoje je limitováno tím, že SNPE framework obsahuje vlastní logiku a je nutné naučit se s ním pracovat. Při přípravě modelu a prostředí pro vývoj je velice obtížné cokoliv debugovat a kromě pár kontrolních scriptů téměř neexistuje možnost jak se dozvědět co způsobuje chybu, nebo proč aplikace neběží tak, jak se očekává. Komunita kolem SNPE frameworku je málo aktivní, a proto v případě problému, může jeho vyřešení zabrat opravdu dlouhou dobu a hodně experimentování.

Seznam obrázků

Seznam literatury

[1] SADHAYO, INTESAB HUSSAIN, MUHAMMAD IBREHIM CHANNA, a ASIF ALI LAGHARI. Automated Medication System for Rural and War Affected Area. *Transactions on Machine Learning and Artificial Intelligence 3, no 1*. 2015.

[2] RUSSELL, STUART J., AND PETER NORVIG. *Artificial intelligence: a modern approach*. Malaysia: Pearson Education Limited, 2016.

[3] CAMPBELL, MURRAY, A. JOSEPH HOANE JR, AND FENGHSIUNG HSU. *Deep blue*. 2002.

[4] ABDULLAH AYUB KHAN, ASIF ALI LAGHARI, a SHAFIQUE AHMED AWAN. Machine Learning in Computer Vision: A Review. 2021.

[5] EL NAQA, ISSAM a MURPHY, MARTIN J. *What is Machine Learning ? Machine Learning In Radiation Oncology*. Schwitzerland: Springer International Publishing, 2015.

[6] NASTESKI, Vladimir. An overview of the supervised machine learning methods. *HORIZONS.B* [online]. 2017, 4, 51–62. ISSN 18578578, 18579892. Dostupné z: doi:10.20544/HORIZONS.B.04.1.17.P05

[7] GENTLEMAN, R. a V. J. CAREY. Unsupervised Machine Learning. In: Florian HAHNE, Wolfgang HUBER, Robert GENTLEMAN a Seth FALCON*Bioconductor Case Studies* [online]. New York, NY: Springer New York, 2008 [vid. 2022-11-12], s. 137–157. ISBN 978-0-387-77239-4. Dostupné z: doi:10.1007/978-0-387-77240-0_10

[8] KHAN, Asharul Islam a Salim AL-HABSI. Machine Learning in Computer Vision. *Procedia Computer Science* [online]. 2020, 167, 1444–1451. ISSN 18770509. Dostupné z: doi:10.1016/j.procs.2020.03.355

[9] S. RAJ, Jennifer. A COMPREHENSIVE SURVEY ON THE COMPUTATIONAL INTELLIGENCE TECHNIQUES AND ITS APPLICATIONS. *Journal of ISMAC* [online]. 2019, 01(03), 147–159. ISSN 2582-1369. Dostupné z: doi:10.36548/jismac.2019.3.002

[10] S. RAJ, Jennifer a Vijitha ANANTHI J. RECURRENT NEURAL NETWORKS AND NONLINEAR PREDICTION IN SUPPORT VECTOR MACHINES. *Journal of Soft Computing Paradigm* [online]. 2019, 2019(1), 33–40. ISSN 2582-2640. Dostupné z: doi:10.36548/jscp.2019.1.004

[11] CIPOLLA, Roberto a Alex PENTLAND. *Computer Vision for Human-Machine Interaction*. B.m.: Cambridge University Press, 1998. ISBN 978-0-521-62253-0.

[12] OUAKNINE, Arthur. Review of Deep Learning Algorithms for Object Detection. *Zyl Story* [online]. 5. únor 2018 [vid. 2022-11-21]. Dostupné z: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

[13] Classification, Object Detection and Image Segmentation. *Qualcomm Developer Network* [online]. [vid. 2023-02-03]. Dostupné z: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/image-segmentation-deeplab-neural-processing-sdk/classification-object-detection-segmentation

[14] KULKARNI, Amruta Kiran. *Classification of Faults in Railway Ties Using Computer Vision and Machine Learning* [online]. B.m., 2017 [vid. 2022-11-21]. Thesis. Virginia Tech. Dostupné z: https://vtechworks.lib.vt.edu/handle/10919/86522

[15] LINDSAY, Grace W. Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future. *Journal of Cognitive Neuroscience* [online]. 2021, 33(10), 2017–2031. ISSN 0898-929X, 1530-8898. Dostupné z: doi:10.1162/jocn_a_01544

[16] Introduction to computer vision: what it is and how it works. *DataRobot AI Cloud* [online]. [vid. 2022-12-17]. Dostupné z: https://www.datarobot.com/blog/introduction-to-computer-vision-what-it-is-and-how-it-works/

[17] GEITGEY, Adam. Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks. *Medium* [online]. 24. září 2020 [vid. 2022-12-17]. Dostupné z: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

[18] UJJWALKARN. An Intuitive Explanation of Convolutional Neural Networks. *Ujjwal Karn* [online]. 10. srpen 2016 [vid. 2022-12-17]. Dostupné z: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[19] Understanding Convolutional Neural Networks for NLP. *Denny's Blog* [online]. 7. listopad 2015 [vid. 2022-12-17]. Dostupné z: https://dennybritz.com/posts/wildml/understanding-convolutional-neural-networks-for-nlp/

[20] *Kernel (image processing)* [online]. 2022 [vid. 2022-12-17]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Kernel_(image_processing)&oldid=1124968907

[21] VARSHNEY, Munender a Pravendra SINGH. Optimizing nonlinear activation function for convolutional neural networks. *Signal, Image and Video Processing* [online]. 2021, 15(6), 1323–1330. ISSN 1863-1703, 1863-1711. Dostupné z: doi:10.1007/s11760-021-01863-z

[22] Convolutional Neural Network. *Questions and Answers in MRI* [online]. [vid. 2022-12-17]. Dostupné z: http://mriquestions.com/convolutional-network.html

[23] *CS231n Convolutional Neural Networks for Visual Recognition* [online]. [vid. 2022-12-17]. Dostupné z: https://cs231n.github.io/convolutional-networks/

[24] RASCHKA, Sebastian. *Python Machine Learning*. B.m.: Packt Publishing Ltd, 2015. ISBN 978-1-78355-514-7.

[25] GUBBI, Jayavardhana, Rajkumar BUYYA, Slaven MARUSIC a Marimuthu PALANISWAMI. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* [online]. 2013, 29(7), 1645–1660. ISSN 0167739X. Dostupné z: doi:10.1016/j.future.2013.01.010

[26] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. *Deep learning*. Cambridge, Massachusetts: The MIT Press, 2016. Adaptive computation and machine learning. ISBN 978-0-262-03561-3.

[27] PATEL, Sunil A., Sanjay P. PATEL, Yagna Bhupendra Kumar ADHYARU, Santosh MAHESHWARI, Pankaj KUMAR a Mukesh SONI. Developing smart devices with automated Machine learning Approach: A review. *Materials Today: Proceedings* [online]. 2022, 51, 826–831. ISSN 22147853. Dostupné z: doi:10.1016/j.matpr.2021.06.243

[28] LANE, Nicholas D., Sourav BHATTACHARYA, Petko GEORGIEV, Claudio FORLIVESI a Fahim KAWSAR. An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices. In: *SenSys '15: The 13th ACM Conference on Embedded Network Sensor Systems*: *Proceedings of the 2015 International Workshop on Internet of Things towards Applications* [online]. Seoul South Korea: ACM, 2015, s. 7–12 [vid. 2023-05-06]. ISBN 978-1-4503-3838-7. Dostupné z: doi:10.1145/2820975.2820980

[29] CHEN, Yanjiao, Baolin ZHENG, Zihan ZHANG, Qian WANG, Chao SHEN a Qian ZHANG. Deep Learning on Mobile and Embedded Devices: State-of-the-art, Challenges, and Future Directions. *ACM Computing Surveys* [online]. 2021, 53(4), 1–37. ISSN 0360-0300, 1557-7341. Dostupné z: doi:10.1145/3398209

[30] XU, Jie, Benjamin S. GLICKSBERG, Chang SU, Peter WALKER, Jiang BIAN a Fei WANG. Federated Learning for Healthcare Informatics. *Journal of Healthcare Informatics Research* [online]. 2021, 5(1), 1–19. ISSN 2509-4971, 2509-498X. Dostupné z: doi:10.1007/s41666-020-00082-4

[31] TensorFlow Lite | ML for Mobile and Edge Devices. *TensorFlow* [online]. [vid. 2023-05-06]. Dostupné z: https://www.tensorflow.org/lite

[32] Core ML. *Apple Developer Documentation* [online]. [vid. 2023-05-06]. Dostupné z: https://developer.apple.com/documentation/coreml

[33] *PyTorch* [online]. [vid. 2023-05-06]. Dostupné z: https://www.pytorch.org

[34] *Keras: Deep Learning for humans* [online]. [vid. 2023-05-06]. Dostupné z: https://keras.io/

[35] *scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation* [online]. [vid. 2023-05-06]. Dostupné z: https://scikit-learn.org/stable/

[36] LIU, Jie, Jiawen LIU, Wan DU a Dong LI. Performance Analysis and Characterization of Training Deep Learning Models on Mobile Device. In: *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*: *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)* [online]. Tianjin, China: IEEE, 2019, s. 506–515 [vid. 2023-05-07]. ISBN 978-1-72812-583-1. Dostupné z: doi:10.1109/ICPADS47876.2019.00077

[37] CAI, Ermao, Da-Cheng JUAN, Dimitrios STAMOULIS a Diana MARCULESCU. NeuralPower: Predict and Deploy Energy-Efficient Convolutional Neural Networks. In: *Asian Conference on Machine Learning*: *Proceedings of the Ninth Asian Conference on Machine Learning* [online]. B.m.: PMLR, 2017, s. 622–637 [vid. 2023-05-07]. ISSN 2640-3498. Dostupné z: https://proceedings.mlr.press/v77/cai17a.html

[38] *Snapdragon Neural Processing Engine SDK: Main Page* [online]. [vid. 2023-01-21]. Dostupné z: https://developer.qualcomm.com/sites/default/files/docs/snpe/

[39] Qualcomm Neural Processing SDK for AI. *Qualcomm Developer Network* [online]. [vid. 2023-01-21]. Dostupné z: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk

[40] Benchmarking the Qualcomm Neural Processing SDK for AI vs. TensorFlow on Android. *Qualcomm Developer Network* [online]. [vid. 2023-02-03]. Dostupné z: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/ai-ml-android-neural-processing/benchmarking-neural-processing-sdk-tensorflow

[41] *Neural Network Quantization: What Is It and How Does It Relate to TinyML? - Technical Articles* [online]. [vid. 2023-05-07]. Dostupné z: https://www.allaboutcircuits.com/technical-articles/neural-network-quantization-what-is-it-and-how-does-it-relate-to-tiny-machine-learning/

[42] Qualcomm Neural Processing SDK for AI Getting Started. *Qualcomm Developer Network* [online]. [vid. 2023-05-07]. Dostupné z: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/getting-started

[43] On-Device AI with Snapdragon Neural Processing Engine SDK. *Qualcomm Developer Network* [online]. [vid. 2023-02-03]. Dostupné z: https://developer.qualcomm.com/blog/device-ai-qualcomm-snapdragon-neural-processing-engine-sdk

[44] Introduction to the Qualcomm Neural Processing SDK for AI and its components. *Qualcomm Developer Network* [online]. [vid. 2023-02-03]. Dostupné z: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/ai-ml-android-neural-processing/introduction-neural-processing-sdk-components

[45] *Snapdragon Neural Processing Engine SDK: TensorFlow Setup* [online]. [vid. 2023-02-06]. Dostupné z: https://developer.qualcomm.com/sites/default/files/docs/snpe/setup_tensorflow.html