|  |  |
| --- | --- |
|  | |
| **Mobilní aplikace pro rozpoznávání  zoologických zvířat** | |
|  | |
| Jiří Daberger | |
|  | |
|  |  |
| 2023 | Popis: fai_logo_cz |
|  |  |
|  | |

\*\*\*Do tištěné verze zde vložte oficiální zadání práce, **do PDF verze, která se nahrává do IS/STAG vložte zadání bez podpisů!**\*\*\*

**Prohlašuji, že**

* beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
* beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
* byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
* beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
* beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo –bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
* beru na vědomí, že pokud bylo k vypracování bakalářské práce  
  využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými  
  subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu  
  využití), nelze výsledky bakalářské práce využít ke komerčním  
  účelům;
* beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

* + že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
  + že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …………………….

podpis studenta

ABSTRAKT

Tato bakalářská práce popisuje vývoj mobilní aplikace určené jako nástroj pro rozpoznávání několika druhů zoologických zvířat. Cílem bylo vytvořit aplikaci na mobilní platformu   
Android za účelem zpříjemnění návštěv zoo a vzdělávání uživatelů aplikace. Hlavním pilířem aplikace je model konvoluční neuronové sítě, který se stará o vyhodnocení obrázku z kamery mobilního telefonu a zjišťuje, zda se na obrázku vyskytuje některé z předem definovaných zvířat. K naprogramování aplikace byl využitý programovací jazyk Kotlin spolu s nový frameworkem Jetpack Compose a pro natrénování modelu sítě se využily knihovny   
Tensorflow. Výsledná práce se zaobírá testováním a celkovou úspěšností v přesnosti rozpoznávání zvířat, ale také porovnáním existujících nebo podobných řešení.

Klíčová slova: mobilní aplikace, neuronová síť, rozpoznávání zvířat, Jetpack Compose,   
Tensorflow

ABSTRACT

Abstrakt ve světovém jazyce

Keywords:mobile application, neural network, animals recognition, Jetpack Compose,   
TensorflowPoděkování, motto a čestné prohlášení, že odevzdaná verze bakalářské práce a verze elektronická, nahraná do IS/STAG jsou totožné ve znění:

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

* Nastudujte a popište problematiku spojenou s detekcí objektů v obraze.
* Zvolte vhodné technologie a prostředky k implementaci aplikace.
* Navrhněte mobilní aplikaci pro rozpoznávání vybraných zvířat pomocí fotoaparátu na platformě Android.
* Zvolte vhodná zvířata a vytvořte jejich dataset pro rozpoznání v obraze.
* Implementujte vámi navrženou aplikaci.
* Výslednou implementaci vhodně otestujte a popište výsledky.

OBSAH

[Úvod 8](#_Toc122466081)

1. [TEORETICKÁ ČÁST 10](#_Toc122466082)

[1 konvoluční Neuronové sítě 11](#_Toc122466083)

[1.1 Počítačové vidění 12](#_Toc122466084)

[1.1.1 Rozpoznávání tvarů v obraze 12](#_Toc122466085)

[1.1.2 Detekce objektů v obraze 13](#_Toc122466086)

[1.1.3 Klasifikace obrazu 13](#_Toc122466087)

[1.1.4 Segmentace obrazu 13](#_Toc122466088)

[2 Klasifikace obrazu 14](#_Toc122466089)

[2.1 Využití v reálných aplikacích 14](#_Toc122466090)

[2.1.1 Mobilní systémy 14](#_Toc122466091)

[2.1.2 Embedded systémy 14](#_Toc122466092)

[2.2 Srovnání existujících modelů 14](#_Toc122466093)

[3 Vývoj Mobilních aplikací 15](#_Toc122466094)

[3.1 Druhy vývoje mobilních aplikací 15](#_Toc122466095)

[3.1.1 Platformě závislé 15](#_Toc122466096)

[3.1.1.1 Nativní vývoj 15](#_Toc122466097)

[3.1.2 Platformě nezávislé – multiplatformní 16](#_Toc122466098)

[3.1.2.1 Webový vývoj 16](#_Toc122466099)

[3.1.2.2 Hybridní vývoj 16](#_Toc122466100)

[3.2 Architektonické vzory 16](#_Toc122466101)

[3.2.1 Model-View-Controller 17](#_Toc122466102)

[3.2.2 Model-View-Presenter 18](#_Toc122466103)

[3.2.3 Model-View-ViewModel 18](#_Toc122466104)

[3.3 Android 19](#_Toc122466105)

[3.4 Framework Jetpack Compose 20](#_Toc122466106)

1. [Praktická část 21](#_Toc122466107)

[4 Popis aplikace 22](#_Toc122466108)

[4.1 První koncept 22](#_Toc122466109)

[4.2 Funkce mobilní kamery 22](#_Toc122466110)

[5 příprava 23](#_Toc122466111)

[5.1 Návrh aplikace 23](#_Toc122466112)

[5.2 Výběr technologií 23](#_Toc122466113)

[6 Tvorba modelu 24](#_Toc122466114)

[6.1 Test dostupných modelů 24](#_Toc122466115)

[6.2 Script pro stahování obrázků 24](#_Toc122466116)

[6.3 API pro vytvoření vlastních modelů 24](#_Toc122466117)

[6.3.1 Model pro klasifikaci 24](#_Toc122466118)

[6.3.1.1 TensorFlow lite model maker 24](#_Toc122466119)

[6.3.2 Model pro detekci 24](#_Toc122466120)

[6.3.2.1 YOLO v5 24](#_Toc122466121)

[6.3.2.2 TensorFlow 2 Object Detection API 24](#_Toc122466122)

[6.3.3 Zhodnocení výsledků a finální výběr 24](#_Toc122466123)

[7 realizace 25](#_Toc122466124)

[8 Testování 26](#_Toc122466125)

[8.1 Srovnání s dostupným řešením 26](#_Toc122466126)

[Závěr 27](#_Toc122466127)

[seznam použité literatury 28](#_Toc122466128)

[Seznam použitých symbolů a zkratek 30](#_Toc122466129)

[Seznam obrázků 31](#_Toc122466130)

[Seznam tabulek 32](#_Toc122466131)

[Seznam Příloh 33](#_Toc122466132)

Úvod

Chytré mobilní telefony a jejich aplikace jsou trendem dnešní společnosti a většina z nás, si život bez chytrého mobilního telefonu nedokáže ani představit. Dnešní telefony již neslouží pouze jako prostředek ke komunikaci s našimi přáteli pomocí telefonování a SMS, ale zejména k využívání chytrých aplikací, které mohou udělat náš život jednodušší.

Mobilní telefony dnes obsahují širokou škálu hardwaru, jako je GPS sloužící pro zjištění aktuální polohy, čip NFC, který mimo jiné umožňuje provádět bezkontaktní platby, kvalitní fotoaparát pro zachycení neopakovatelných okamžiků a v hlavní řadě výkonný procesor, který se o všechny operace dokáže postarat. Výše uvedené možnosti ale vyžadují jednu důležitou věc, kterou je správná mobilní aplikace, díky které je možné využít mobilní telefon prakticky k čemukoliv. Takových aplikací existuje celá řada, některé slouží pro zabavení ve volném čase, některé jako důležitá zdravotní pomůcka nemocným lidem, jiné zase jako pomocník při cestování nebo nakupování a v neposlední řadě ty, které se snaží člověka něco přiučit, naučit nebo mu pomoct vyhledat si další informace. Mezi poslední množinu z výčtu aplikací se může zařadit i aplikace vznikající v této bakalářské práci, jelikož se snaží uživateli usnadnit návštěvu zoologické zahrady a udělat ji více interaktivní za pomocí   
mobilního fotoaparátu. Aplikací využívající mobilní fotoaparát je nespočet, převážně se ale jedná o aplikace sloužící jako komunikační prostředek s možností sdílení fotografií, jako jsou sociální sítě Instagram nebo Snapchat, popřípadě aplikace, které dokážou pomocí umělé inteligence detekovat tvář uživatele a zaměnit ji za něco jiného, například za hlavu zvířete, popřípadě tvář digitálně zkrášlit.

Výše zmíněná umělá inteligence je tvořena za pomoci umělých neuronových sítí, které jsou hlavním prvkem úspěšnosti této práce, jelikož díky nim by nebylo možné vytvořit model, který se naučí rozpoznat zvířata jenom podle dodaných fotografií.

Mezi takovou technologii můžeme řadit i neuronovou síť, která může dopomáhat v určitých situacích, které lze obtížně naprogramovat neboli algoritmizovat.

, která je díky své schopnosti učit se, vhodná pro řešení komplikovaných úloh v oblastech, jako je například klasifikace dat.

Díky neuronovým sítím se můžeme jako lidstvo dále posouvat a s jejich využitím objevovat data, které bychom bez nich nalezli jen stěží.

Proto se v této práci budeme zabývat vývojem mobilní aplikace sloužící jako prostředek   
k detekci zvířat s využitím konvolučních neuronových sítí, které se běžně používají právě pro potřeby zpracování obrazových dat.

V teoretické části si objasníme pojem konvoluční neuronové sítě, jejich využití a rozdělení do možných kategorií. Zjistíme reálné možnosti existujících řešení pro rozpoznávání objektů s využitím mobilních a embedded systémů. Dále si také ukážeme teoretický vývoj mobilních aplikací na platformě Android za pomocí nového frameworku Jetpack Compose.

Praktická část bude popisovat samotný vývoj této aplikace od začátku až do konce. Autor představí celkový koncept a přípravu aplikace, jak vytvořil potřebný dataset obrázků pro naučení modelu neuronové sítě, aby byl schopen rozpoznat vybraná zvířata a jak tento model implementoval do vytvořené aplikace. Dozvíme se, jak si autor poradil s problémy během vývoje, jak výsledek testoval, ale taky porovnání s již existujícím řešením daného problému.

|  |  |
| --- | --- |
|  | TEORETICKÁ ČÁST |

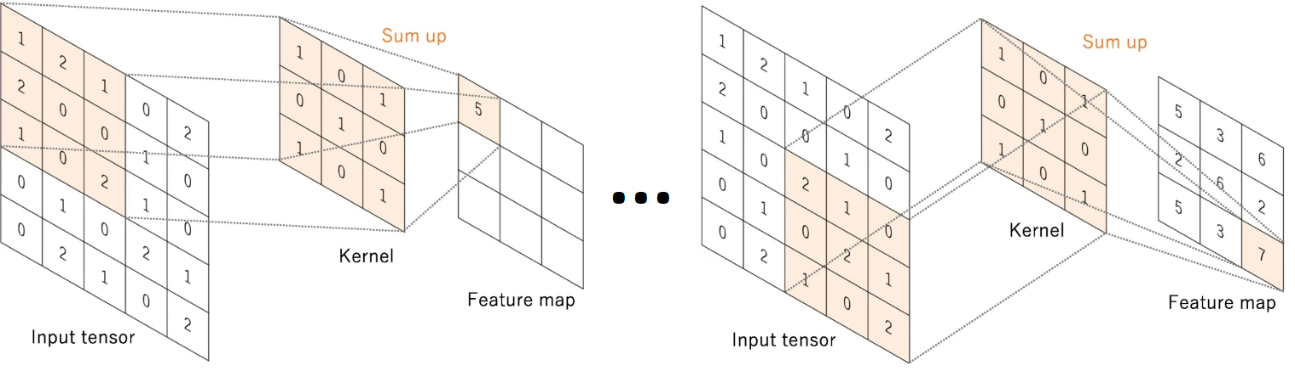
# konvoluční Neuronové sítě

Konvoluční neuronové sítě (anglicky Convolutional Neural Network, **CNN**) jsou rozšířenou variantou klasických neuronových sítí o speciální vrstvu, která dopomáhá elegantně eliminovat problém s velkými daty, jako jsou například obrázky. Tímto elegantním řešením se zabývá tzv. konvoluční vrstva (od toho poté název Konvoluční neuronové sítě), která značně redukuje vstupní parametry obrazu. [1]

Jak již bylo naznačeno, konvoluční neuronová síť se skládá z několika vrstev, mezi které spadá:

1. **Konvoluční** vrstva

Jedná se o vrstvu, která v určitém směru prochází obraz pomocí několika filtrů, které zachycují různé informace obrázku, jako jsou hrany, světlost, popř. barva a jiné. Tento filtr si můžeme představit jako matici N×N (obvykle 3×3 nebo 5×5) navíc s doplněním barevného kanálu obrázku (×3 pro barevný RGB režim nebo ×1 pro režim šedi). Při procházení, které má navíc nastavení udávající krok (obvykle 1), se generuje tzv. Feature mapa obsahující součet „aktivovaných“ pixelů ze vstupního obrázku a filtru, tak, jak můžeme vidět na obrázku (Obrázek 1.)[[1]](#footnote-1). [2]



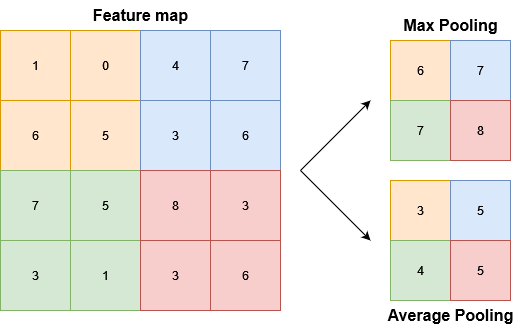
Obrázek . Aplikace filtru na vstupní obrázek v konvoluční vrstvě. [2]

1. **Pooling** vrstva

Pooling vrstva neboli vrstva sdružující, se stará o další zredukování vstupních parametrů pomocí jednoho z existujících filtrů. Tato operace se podobá operacím v konvoluční vrstvě, s výjimkou, že vstupní maticí v Pooling vrstvě je výstup z konvoluční vrstvy, tzn. Feature mapa.

Nejčastějším filtrem je operace nazvaná **Max Pooling**, kterou je obvykle matice 2×2 s krokem 2, aby se operace nepřekrývali a tím dosáhli největšího zredukování vstupních parametrů. Tento filtr má za následek to, že se z Feature mapy přečte pole o velikosti 2×2 a do výstupu se z daného pole uloží **maximum**, a tak pokračuje až do konce, vždy s nastaveným krokem 2.

Druhým nejčastějším filtrem je **Average Pooling**, který do výstupu vkládá průměrnou hodnotu z hodnot vstupní matice. [2]



Obrázek . Ukázka použití filtrů v Pooling vrstvě.

Z obrázku (Obrázek 2.) je patrné, že se mapa velikostně zredukuje o ¼ díky odstranění redundantních pixelů. To zapříčiní snížení celkového rozlišení obrázku, ale také snížení potřebného výkonu pro další potřebné výpočty a samotné trénování sítě.

1. **Plně propojená** vrstva

Doposud se vždy pracovalo s 2D maticemi…

## Počítačové vidění

### Rozpoznávání tvarů v obraze

OCR

Edge detection

### Detekce objektů v obraze

### Klasifikace obrazu

### Segmentace obrazu

# Klasifikace obrazu

## Využití v reálných aplikacích

### Mobilní systémy

### Embedded systémy

## Srovnání existujících modelů

Uceni

Problémy při uceni

# Vývoj Mobilních aplikací

V dnešním světě rozlišujeme 2 hlavní hráče na poli operačních systémů, pro které je možné vyvíjet mobilní aplikace. Jedná se o zařízení s operačním systémem iOS nebo Android. Z grafu (Graf 1.) lze vidět, že ve světě převládá operační systém Android a jelikož je tato práce zaměřena na vývoj mobilní aplikace na platformě Android, bude velká část této kapitoly věnovaná právě tomuto operačnímu systému.

Graf . Zastoupení mobilních operačních systémů za rok 2022. [3]

## Druhy vývoje mobilních aplikací

### Platformě závislé

#### Nativní vývoj

Nativní vývoj aplikací v praxi znamená, že vyvíjená aplikace bude spustitelná pouze na příslušné platformě operačního systému, pro který je aplikace naprogramovaná. Tento způsob vývoje převládá u takových aplikací, které se neobejdou bez veškerého výkonu a hardwaru mobilního telefonu, jako je například fotoaparát, čipy NFC nebo GPS, popřípadě přímá práce s paměťovým adresářem.

Při takovém vývoji je ale nutné vytvořit zvlášť aplikaci pro platformu Android a zvlášť aplikaci pro platformu iOS a díky tomu je celkový vývoj mnohdy časově i finančně náročnější.

### Platformě nezávislé – multiplatformní

#### Webový vývoj

Přesným opakem nativního vývoje mobilních aplikací je webový vývoj. Jedná se o responzivní webovou stránku využívající webové technologie, jako je standard HTML 5 a JavaScriptové frameworky. Tyto aplikace většinou nemají přímý přístup k hardwaru mobilního telefonu, s výjimkou GPS nebo základní funkčnosti kamery. Jelikož takto vyrobené aplikace využívají pro svůj běh pouze internetový prohlížeč, není možné je samostatně nainstalovat do zařízení, ale podporují uložení na plochu zařízení.

#### Hybridní vývoj

Hybridní vývoj je dosti podobný k vývoji webovém. Opět se jedná o aplikaci využívající webové rozhraní internetového prohlížeče, které se nazývá WebView [4]. Toto rozhraní je „zabaleno“ v nativní části aplikace, díky kterému je možné aplikaci nainstalovat na libovolné zařízení. K výše zmíněnému „zabalení“ se využívá tzv. wrapper technologie, například Ionic Capacitor, která se postará, aby webová aplikace dokázala komunikovat s hardwarem mobilního telefonu a tím využívat jeho funkce.

## Architektonické vzory

Slouží pro nastolení přístupu a pravidel při vývoji mobilní aplikace. Díky dodržení všech pravidel některého z vybraného vzoru, má aplikace pevně danou strukturu, ve které se dá snadno orientovat díky rozdělení do určitých celků a v budoucnosti rozšiřovat o novou implementaci.

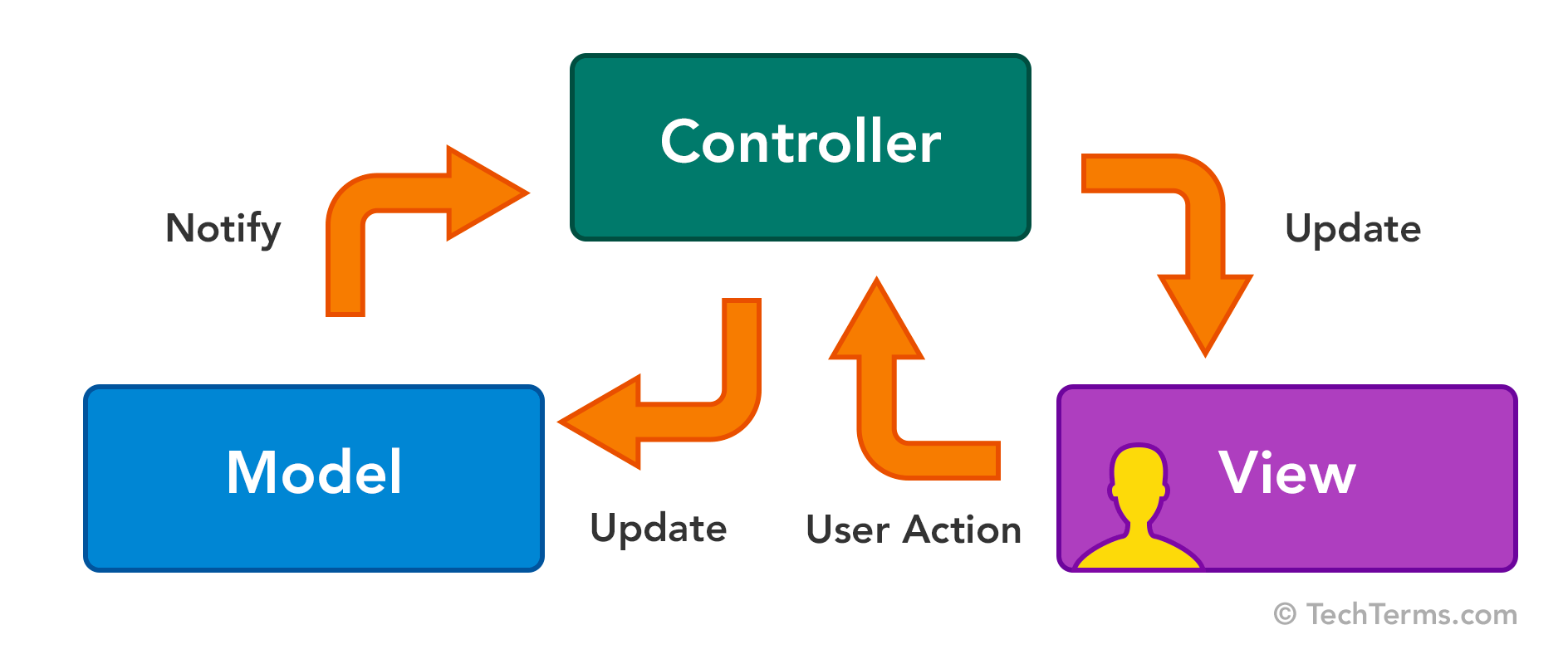
Následující příklady vzorů spadají pod tzv. třívrstvou architekturu, která se již podle názvu vyznačuje třemi základními vrstvami:

1. **Prezentační vrstva** se stará o prezentování/zobrazení uživatelského rozhraní uživateli dané aplikace a umožňuje mu tím aplikaci ovládat. Jedná se o platformě závislou vrstvu – webová, desktopová, mobilní.
2. **Aplikační vrstva** se stará o veškeré operace a výpočty v běhu aplikace. Zprostředkovává komunikaci mezi Datovou a Prezentační vrstvou. Zpracovává vstupy od uživatele, nebo naopak posílá data do Prezentační vrstvy pro zobrazení uživateli.
3. **Datová vrstva** je vrstva, která se stará o ukládání a práci s daty aplikace. Může se jednat o lokální nebo serverové úložiště nebo databázi. Tato data jsou vždy dostupná pro Aplikační vrstvu.

### Model-View-Controller

**M**odel-**V**iew-**C**ontroller neboli **MVC**, je jeden z nejčastějších architektonických vzorů při tvorbě webových aplikací. Jeho rozdělí do tří vrstev je následující: [5]

* Model – Tato část představuje logiku a práci s daty aplikace. Stará se o načítání, ukládání a zpracování dat z databáze a při tom nemá tušení o tom, jak jsou data na aplikaci závislá a zobrazená uživateli.
* View – View můžeme z angličtiny přeložit jako *pohled*, a to je přesně tím, čím je. Prakticky se jedná o grafické zobrazení aplikace s daty, které View čerpá z Modelu. Tato vrstva by nikdy neměla být schopná měnit data aplikace a vykonávat jinou logiku, než je zobrazování šablony s UI.
* Controller – Kontrolér většinou reaguje na události uživatele nebo samotného systému. Komunikuje s Modelem a na základě vstupu od uživatele rozhoduje, které View aplikace se načte, popř. s jakými daty.



Obrázek . MVC diagram. [6]

Z tohoto rozdělení je patrné, že všechny vrstvy jsou na sobě nezávislé a rozdělené na bloky, které souvisí pouze spolu. Díky tomu je aplikaci možné snadno škálovat a modifikovat bez obav porušení některé z existující funkcionality aplikace.

### Model-View-Presenter

Architektonický vzor MVP vychází ze vzoru MVC a díky tomu mají téměř identické chování. Model opět reprezentuje datovou vrstvu, View zobrazuje data uživateli a také reaguje na požadavky uživatele, které posílá na svůj Presenter, který se navíc stará o komunikaci mezi vrstvami Model a View.

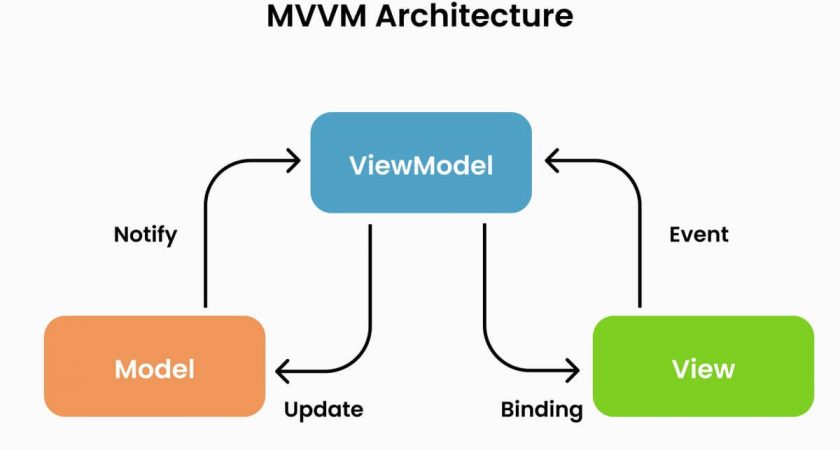
Změna od MVC přichází u samotné komunikaci mezi View a Modelem, které oproti MVC nejsou nyní napřímo propojeny, a tudíž musí Presenter předat data z Modelu do View sám a tím samotný View naformátovat. To umožňuje větší flexibilitu při automatizovaném testování. [7]



Obrázek . MVP diagram. [7]

### Model-View-ViewModel

**M**odel-**V**iew-**V**iew**M**odel (MVVM) je silným architektonickým vzorem pro tvorbu rozsáhlých, nejen, mobilních aplikací. Spočívá ve vytvoření vazeb mezi daty a uživatelským rozhraním. Tato vazba probíhá za pomocí binding třídy ve ViewModelu a stará se o držení dat, jejich aktualizování a přeposílání na View. To znamená, že z UI je zcela odstraněn kód aplikace a data jsou pevně vázána právě na třídu ve ViewModelu, kde si neustále uchovává aktuální stav.



Obrázek . MVVM diagram. [8]

* **Model** představuje identickou funkci jako u vzorů MVC a MVP, to je, že se stará o získávání dat z databáze aplikace.
* **View** popisuje uživatelské rozhraní, stará se o komponenty, které jsou ovlivněné vstupy od uživatele. Nemělo by obsahovat aplikační logiku.
* **ViewModel** – nejdůležitější část tohoto arch. vzoru. Obsahuje veškerou aplikační logiku, stará se o udržování stavu a aktualizování dat podle potřeb View.

Jelikož ViewModel nemusí být pevně vázaný na konkrétní View, může být tak automaticky použito ve více případech a tím značně redukovat celý kód aplikace, kterou je pak možné snadno testovat a rozšiřovat. [9]

## Android

Jedná se o open-source projekt založený na Linuxovém jádru vyvíjeným společností Google od roku 2008 [10]. Dá se považovat za krále mobilních operačních systémů, jelikož se jedná o nejvíce používaný operační systém v mobilních zařízení a před druhým nejčastějším operačním systémem iOS si každoročně drží odstup přibližně 45 % využívaných zařízení [11].

Aplikace vyvíjené pro tento systém se programují v jazyku Java nebo v moderním jazyce Kotlin. Vývojářům jsou dostupné různé nástroje v podobě SDK balíčků (neplést s API), které hrají důležitou roli při vývoji jakékoliv aplikace. Tyto SDK nástroje jsou běžnou součástí oficiálního vývojového prostředí Android Studio.

## Framework Jetpack Compose

Jedná se o novou moderní technologii pro tvorbu nativních mobilních aplikací pro platformu Android s pomocí programovacího jazyka Kotlin. Jeho hlavní předností je rychlejší vykreslování prvků, intuitivní a rychlejší tvorba UI elementů aplikace a tím i plynulejší chod celé aplikace.

|  |  |
| --- | --- |
|  | Praktická část |

# Popis aplikace

Jak již bylo mnohokrát napsáno, jedná se o aplikaci schopnou rozpoznat několik zoologických zvířat, v praxi to ale znamená mnohem víc.

## První koncept

Ještě před samotným programováním aplikace byla potřeba vymyslet její funkčnost,   
přívětivý design a intuitivní ovládání. Z jednoduchého nákresu vznikla první část aplikace, kterou je úvodní obrazovka s kategoriemi zvířat. Dalším rozšířením bylo hlavní menu   
v levém horním rohu, které se dá kdykoliv otevřít táhnutím prstu z kterékoliv levé části   
obrazovky doprava, a tak se dostat na další obrazovky aplikace.

## Funkce mobilní kamery

# příprava

Text

## Návrh aplikace

## Výběr technologií

text

# Tvorba modelu

Tato část popisuje celkový vývoj konvolučního modelu, který byl využití v aplikaci pro rozpoznávání zvířat.

## Test dostupných modelů

## Script pro stahování obrázků

## API pro vytvoření vlastních modelů

### Model pro klasifikaci

#### TensorFlow lite model maker

### Model pro detekci

#### YOLO v5

#### TensorFlow 2 Object Detection API

### Zhodnocení výsledků a finální výběr

# realizace

# Testování

## Srovnání s dostupným řešením

Závěr

text

# seznam použité literatury

|  |  |
| --- | --- |
| [1] | Pokročilé techniky neuronových sítí [online]. In: . s. 1 [cit. 2022-12-20]. Dostupné z: https://course.elementsofai.com/cs/5/3. |
| [2] | Yamashita, R., Nishio, M., Do, R.K.G. et al. Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9. |
| [3] | Zastoupení mobilních operačních systémů. Statcounter [online]. [cit. 2022-12-18]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202111-202211-bar. |
| [4] | VALA, Radek. Programování mobilních aplikací: Metody vývoje mobilních aplikací. FAI, UTB, 2022. Prezentace. Fakulta aplikované informatiky - Univerzita Tomáše Bati ve Zlíně.. |
| [5] | POP, Dragos-Paul a Adam ALTAR. Designing an MVC Model for Rapid Web Application Development [online]. 2014, Pages 1172-1179 [cit. 2022-12-18]. ISSN 1877-7058. Dostupné z: https://www.sciencedirect.com/science/article/pii/S187770581400352X. |
| [6] | MVC diagram. In: TechTerms [online]. [cit. 2022-12-18]. Dostupné z: https://techterms.com/definition/mvc. |
| [7] | MISHRA, Rishu. Difference Between MVC and MVP Architecture Pattern in Android. GeeksForGeeks [online]. 10 Nov 2020n. l., 1 [cit. 2022-12-18]. Dostupné z: https://www.geeksforgeeks.org/difference-between-mvc-and-mvp-architecture-pattern-in-android/?ref=gcs. |
| [8] | MVVM diagram. In: Bach Khoa-npower [online]. [cit. 2022-12-19]. Dostupné z: http://bachkhoa-npower.vn/mvvm-la-gi/. |
| [9] | MISHRA, Rishu. MVVM (Model View ViewModel) Architecture Pattern in Android. GeeksForGeeks [online]. 18 Oct 2022n. l., 1 [cit. 2022-12-19]. Dostupné z: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/. |
| [10] | Android. Wikipedie [online]. [cit. 2022-12-10]. Dostupné z: https://cs.wikipedia.org/wiki/Android\_(opera%C4%8Dn%C3%AD\_syst%C3%A9m). |
| [11] | Příčky v mobilních operačních systémech. Statcounter [online]. [cit. 2022-12-10]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202112-202211. |

Seznam použitých symbolů a zkratek

SDK Software Development Kit

IDE Integrated Development Environment

API

UI

iOS

NFC

GPS

SMS

HTML

Tzv

popř

Seznam obrázků

[Obrázek 1. Ukázkový obrázek 9](#_Toc56699022)

Seznam tabulek

[Tabulka 1. Ukázková tabulka 9](#_Toc56699033)

Seznam Příloh

Příloha P I: Název přílohy

1. Matici **Kernel** z obrázku chápejme jako **filtr**. [↑](#footnote-ref-1)