
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INTELIGENTNÍCH SYSTÉMŮ**
**DEPARTMENT OF INTELLIGENT SYSTEMS**


# URČOVÁNÍ POČASÍ PODLE SNÍMKŮ OBLAKŮ

**WEATHER ESTIMATION BASED ON IMAGES OF CLOUDS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** — **TOMÁŠ KUKAŇ**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **Ing. FILIP ORSÁG, Ph.D.**
**SUPERVISOR**



**BRNO 2019**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

# Zadání bakalářské práce


Student: **Kukaň Tomáš**
Program: Informační technologie
Název: **Určování počasí podle snímků oblaků**
**Weather Estimation Based on Images of Clouds**
Kategorie: Umělá inteligence

Zadání:

1. Prostudujte algoritmy zpracování obrazu a základy neuronových sítí, zaměřte se na konvoluční neuronové sítě. Seznamte se s druhy oblaků, s teorií popisující vzhled, jas a barvu oblaku a se vztahem vizuální podoby oblaků k nadcházejícímu počasí.
2. Vytvořte vlastní množinu snímků oblak za různého počasí. Mraky manuálně kategorizujte dle jejich vzhledu.
3. Navrhněte mobilní aplikaci, která na základě snímku oblaků bude uživatele informovat o změnách počasí.
4. Navrženou aplikaci implementujte v programovacím jazyce Java.
5. Aplikaci otestujte v reálných podmínkách a proveďte experimenty s množinou testovacích dat. Zhodnoťte, zda je navržené řešení vhodné jako prostředek přibližné, krátkodobé předpovědi počasí.

Literatura:

- SZEGEDY, Christian, et al. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015. p. 1-9.
- KHAN, Salman; RAHMANI, Hossein; SHAH, Syed Afaq Ali; BENNAMOUN, Mohammed. A Guide to Convolutional Neural Networks for Computer Vision. Morgan & Claypool Publishers. 2018. ISBN: 978-1681730219.
- DAY, John A. The Book of Clouds. Sterling. 2005. ISBN: 978-1402728136.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Orság Filip, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 27. února 2019

# Abstrakt

Hlavním cílem této práce bylo vytvořit aplikaci schopnou předpovědět nastávající počasí na základě fotografie oblak s využitím konvolučních neuronových sítí. V této práci jsou popsané kategorie oblak a jejich odpovídající předpověď. Dále zde jsou výsledky experimentů s různými architekturami sítí a datasetů s přihlédnutím na jejich úspěšnost v rozeznání typu oblak. Nakonec je tu krátce popsána tvorba finální aplikace a řešení problémů, kterým jsem čelil při její implementaci.



# Abstract

The main purpose of this thesis is a creation of a simple mobile application that would be able to give weather predictions based on a cloud photo through the usage of convolutional neural networks. I have analyzed all types of clouds and joined them with weather prediction. Then there are the results of experiments with different neural networks architectures and different datasets. In the end of this thesis I have described the creation of the Android application as well as the problems I had to solve.



# Klíčová slova

umělá inteligence, strojové učení, klasifikace, klasifikační algoritmy, hluboké učení, neuronové sítě, konvoluční neuronové sítě, rozpoznávání obrazu, počítačové vidění, klasifikace oblak, předpověď počasí, mobilní aplikace, Android

# Keywords

artificial intelligence, machine learning, classification, classification algorithms, deep learning, neural networks, convolutional neural networks, image recognition, computer vision, clouds classification, weather prediction, mobile application, Android



# Citace

KUKAŇ, Tomáš. *Určování počasí podle snímků oblaků*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Filip Orság, Ph.D.

# Určování počasí podle snímků oblaků

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Filipa Orsága, Ph.D. Další informace mi poskytl Ing. Tomáš Goldmann. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Kukaň
30. července 2019


## Poděkování

Tímto bych rád poděkoval Ing. Filipovi Orságovi Ph.D. za poskytnutý čas a cenné rady. Dále bych chtěl poděkovat Ing. Tomášovi Goldmannovi za odbornou pomoc a rady s implementací.

# Obsah

# Kapitola 1

# Úvod

Každý se občas podíváme z okna a říkáme si, jaké asi bude počasí. Samozřejmě, vždy si můžeme na internetu vyhledat předpověď, ale co když nemáme přístup k internetu, nebo co když chceme vyzkoušet nějaký alternativní přístup. Slyšeli jsme různé příběhy o většinou starších lidech, kteří dokázali předpovědět přicházející déšť podle pocitu. Tato práce se ale bude zabývat přístupem, který může fungovat pro všechny i bez nevšedních schopností.

Není novinkou, že oblaka nám mnoho o počasí řeknou. Je to samozřejmě proto, že oblaka jsou přímým důsledkem pohybu vzduchu v atmosféře, jeho vlhkosti, tlaku, teplot v různých výškách... Oblaka nám ale neříkají pouze o aktuálním počasí, s trochou výzkumu nám mohou dobře odhadnout i počasí nastávající. V této práci se budeme zabývat právě tímto alternativním přístupem.

Nejdříve si jednotlivá oblaka rozdělíme do kategorií podle jejich vzhledu a ke každé kategorii si řekneme pravděpodobné nastávající počasí. Následně budeme řešit klasifikaci oblaků na základě jejich fotografií. K tomu použijeme poměrně novou a v posledních letech velmi oblíbenou metodu nazývající se konvoluční neuronové sítě.

K naučení neuronové sítě budeme potřebovat dataset, velké množství fotek oblaků, které již budou správně klasifikovány. Jeden z bodů této práce se bude vytvářením tohoto datasetu zabývat. Vzhledem k tomu, že oblaka řadíme do deseti kategorií, budeme potřebovat dataset opravdu rozměrný, vzhledem k omezenému času a prostředkům na vypracování této práce, se budu snažit tvorbu tohoto datasetu optimalizovat.

Po tvorbě dostatečně velkého datasetu se můžeme pak zabývat architekturou a samotným učením neuronové sítě. Existuje mnoho frameworků a jazyků, ve kterém to můžeme udělat, já použiji velmi oblíbený jazyk Python a stejně tak oblíbený framework pro hluboké učení zvaný Tensorflow. Výsledkem tohoto učení bude naučená neuronová síť schopna kategorizovat oblak.

Tato síť je samozřejmě nepoužitelná pro laiky a širokou veřejnost, a proto další velká a konečná část této práce bude vývoj jednoduše použitelné aplikace na telefon. Tato aplikace bude implementována pro systém Android v jazyce Java a jejím účelem bude, aby nám rychle předpověděla počasí na základě fotografie oblaků.

Nakonec se budu zabývat testováním této aplikace, zejména testováním uživatelským. Pokusím se během toho zhodnotit správnost rozeznávání oblaků a také zkontrolovat pravost předpovědi podle typu oblaků.

# Kapitola 2

# Neuronové sítě

Umělé neuronové sítě, které v počítačích modelujeme, byly inspirovány biologickými neuronovými sítěmi v tělech organismů 2.1, kde je jejich funkcí celý organismus ovládat. Biologické neurony přijímají signály, které vznikly různými vzruchy, signály zpracují a jejich výsledkem je opět signál, vedoucí k nějaké reakci. Pomáhají tedy organismu reagovat na různé podněty. Sítě neuronů mají důležitou schopnost se učit díky měnícím se spojením mezi neurony.

V počítačích se snažíme tuto komplexní a složitou funkci zjednodušit a využít k vytvoření podobné sítě, která se bude skládat z neuronů, které budou mít určitou funkci, a pak parametrizovatelných spojení mezi nimi, aby se daná neuronová síť mohla učit stejně jako ta v tělech organismů.

## 2.1 Neuron

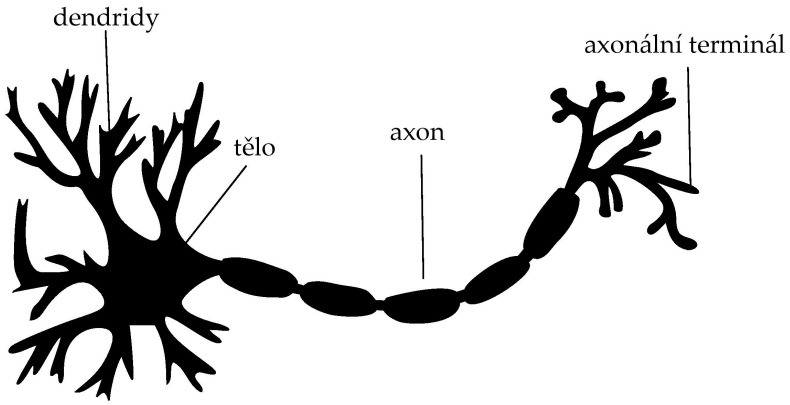
### 2.1.1 Biologický neuron

Uvedené informace jsou částečně převzaty z [13] a [2].

Biologický neuron se skládá z dendridů, které jsou vstupem neuronu a ústí do těla, z kterého dále vede axon. Spojení mezi dvěma neurony, tedy axonu prvního a dendridy druhého nazýváme synapsí. Synapse může mít různou váhu a tím ovlivňuje velikost přenášeného signálu. Váha může být buďto posilující nebo potlačující.

Neurony si předávají informace pomocí buďto elektrických, nebo chemických signálů. Vnitřkem neuronu prochází signál převážně jako slabý elektrický impuls, kdežto přes synapsi se přesouvá díky chemickým podnětům.

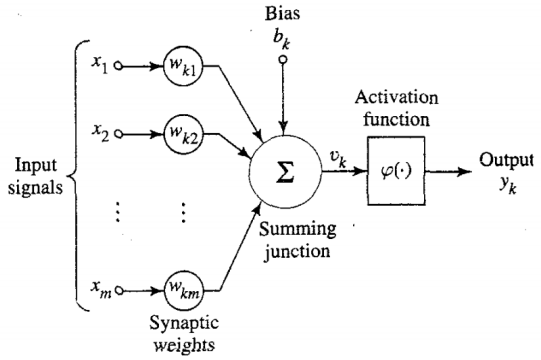
Všechny popsané části neuronu můžeme vidět na obrázku 2.1.




Obrázek 2.1: Vyobrazení [2] pyramidového neuronu z nejčastějších neuronů. Na obrázku vidíme dva druhy dendridů, dlouhý axon a axonální zakončení které je vstupem synapse.

### 2.1.2 Umělý neuron

Na základě biologického neuronu popsaného výše byl vytvořen matematický model graficky vyobrazený na obrázku 2.2.




Obrázek 2.2: Matematický model umělého neuronu [1].

Na obrázku 2.2 vidíme:

- neuron $k$,
- vstupní signál $x_1, x_2 ... x_m$, také můžeme zapsat jako vektor $\vec{x}$,
- synapse $1...m$.
- váhy synapsí $w_{km}$,
- bias $b_k$ sloužící k posunu aktivační funkce,
- sumu $\sum$, která sčítá vážené vstupní signály,
- aktivační funkci $\varphi$ pracující se sumou a vytvářející výstupní hodnotu neuronu.

Matematicky můžeme umělý neuron popsat následujícími rovnicemi:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

a

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

kde $x_1, x_2, ..., x_m$ jsou vstupní signály, $w_{k1}, w_{k2}, ...w_{km}$ jsou synaptické váhy neuronu $k$, $u_k$ je suma vážených vstupních signálů, $b_k$ je bias, $\varphi$ je aktivační funkce a $y_k$ je výstup neuronu.

### 2.1.3 Aktivační funkce

Aktivační funkce (známy taky jako přenosové) v neuronových sítích slouží kromě omezení výstupní hodnoty neuronu ke zrušení linearity výpočtu. Pokud bychom ji neměli, funkce, kterou bychom se snažili neuronovou sítí aproximovat, by byla vždy pouhá lineární regrese a špatně by modelovala složitější funkce.

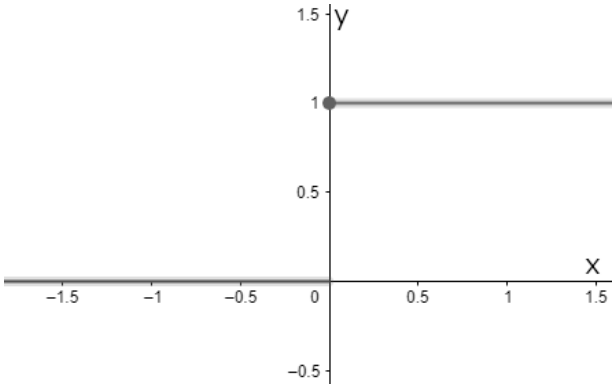
#### Základní druhy aktivační funkce

Aktivační funkce dělíme do třech základních skupin [13]:

**Skoková funkce**, známá taky jako Heavisidova funkce, se dá zapsat následovně:

$$\varphi(x) = \begin{cases} 1, & \text{pro } x \geq 0 \\ 0, & \text{pro } x < 0 \end{cases} \tag{2.3}$$

Model neuronu využívající tuto aktivační funkci nazýváme McCulloch-Pittsův model [13]. V tomto modelu je výstupní hodnota neutronu buďto 1 nebo 0 v závislosti na tom, zdali je jeho vnitřní hodnota nezáporná či záporná. Tato aktivační funkce je vhodná pro binární klasifikaci. V případě klasifikace do více tříd by byla tato aktivační funkce nevhodná.



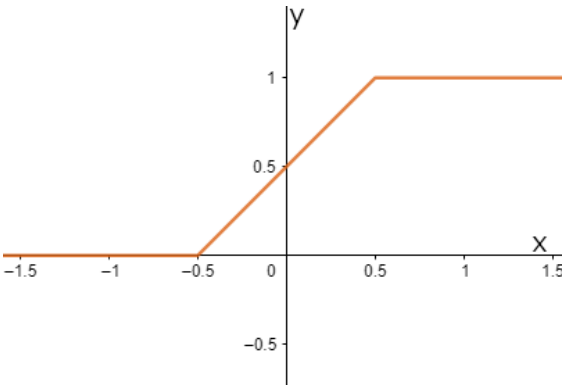

Obrázek 2.3: Skoková funkce.

**Částečně lineární funkce**

Předpis funkce:

$$\varphi(x) = \begin{cases} 1, & \text{pro } x \geq -\frac{1}{2} \\ x + \frac{1}{2}, & \text{pro } -\frac{1}{2} < x < \frac{1}{2} \\ 0, & \text{pro } x \leq -\frac{1}{2} \end{cases} \tag{2.4}$$

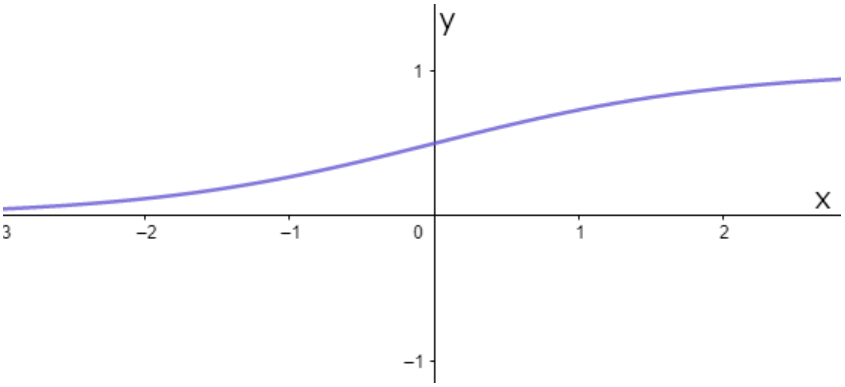
Tuto funkci můžeme považovat za lineární aproximaci jiných nelineárních funkcí jako je např. sigmoida [13]. Oproti předchozí skokové funkci už dokáže klasifikovat do více než dvou skupin díky svému oboru hodnot. Díky tomu ji můžeme použít v nebinární klasifikaci. Nevýhoda této funkce je, že její derivace na všech definičních oborech je konstantní, tedy nezávislé na $x$. Díky tomu při učení nemůžeme použít zpětnou propagaci, protože při něm se používá právě derivace této funkce.




Obrázek 2.4: Částečně lineární funkce.

**Sigmoida** je jedna z nejčastěji používaných aktivačních funkcí v konstrukci umělých neuronových sítí. Je to striktně rostoucí a spojitá funkce ve tvaru S, která má spojitou první derivaci pro všechna $x \in R$. Jako příklad sigmoidy uvedu speciální případ logistické funkce, avšak sigmoid je nekonečně mnoho, je to třída funkcí splňující dříve uvedená pravidla.

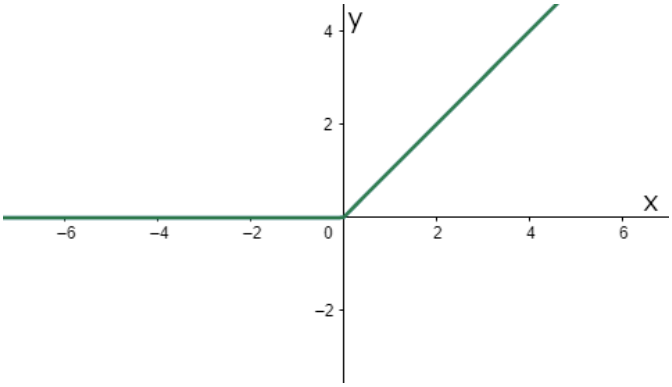
$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$




Obrázek 2.5: Funkce sigmoid.

**Rectified Linear Unit (ReLU)** Funkce **ReLu** je v poslední době nejpoužívanější nejoblíbenější aktivační funkce[3]. V práci *Deep Sparse Rectifier Neural Networks (2011)* [30] bylo ukázáno, že je vhodnější pro učení hlubokých neuronových sítí, než dříve zmiňované funkce patřící do třídy sigmoid. Jedním z důvodů je i ten fakt, že její výpočet je jednodušší. Její předpis je následující:

$$\varphi(x) = x^{+} = max(0, x) \tag{2.6}$$




Obrázek 2.6: ReLU.

## 2.2 Učení

Učení v umělé inteligenci je proces, který zlepšuje výsledné hodnoty určitého mechanizmu. Ve smyslu neuronových sítí pod pojmem učení rozumíme úpravy hodnoty vah jednotlivých spojení neuronů tak, aby se nám zvýšila úspěšnost vyhodnocování touto sítí dle očekávaných výsledků.

Učení neuronových sítí můžeme rozdělit do třech kategorií: s učitelem, bez učitele a posilované učení.

### 2.2.1 Učení s učitelem

V tomto případě je potřeba trénovací množina vstupů a k ním očekávané výstupy. Síť tyto vstupy zpracuje, převezme se její výsledek, porovná se s očekávaným výsledkem a pomocí **chybové funkce** se vypočítá chyba. Tu potom převezme algoritmus učení a upraví hodnoty sítě tak, aby se chyba snížila. Učení **s učitelem** se nejčastěji používá k regresi (odhadu) nebo ke klasifikaci. Jako příklad regrese můžeme uvést předpověď počasí a pro klasifikaci třeba rozeznávání písmen na základě jejich fotografie.

### 2.2.2 Učení bez učitele

Pro učení neuronové sítě bez učitele potřebujeme vstupní množinu prvků, která nemusí být nijak klasifikovaná. Namísto změny atributů na základě chyby se v tomto případě snaží neuronová síť hledat ve vstupních datech podobnosti a na základě jich je klasifikovat tak, aby na podobné vstupy odpovídala podobnými výstupy. Učení bez učitele se používá zejména na shlukování.

### 2.2.3 Posilované učení

Při tomto učení se snažíme vytvořit neuronovou síť, která dokáže správně reagovat na prostředí, které jí vytváří vstupy, které ovlivní prostředí, ve kterém se nachází. To děláme tak, že na základě její reakce (výstupu) vrátíme síti buďto odměnu nebo trest. Síť se potom přizpůsobuje tak, aby získala co nejvíce odměn a naopak co nejméně trestů.

## 2.3 Architektura neuronových sítí

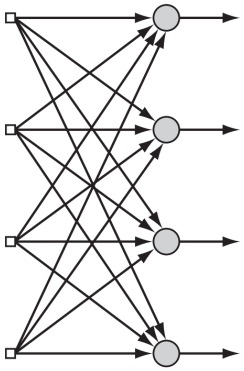
Neuronové sítě můžeme dělit do architektonických kategorií podle různých kritérií: počet vrstev sítě, typ propojení neuronů, výpočetní úlohy a podobně.

**Vstupní vrstva** Za vstupní vrstvu považujeme právě ty neurony, do kterých předáváme vstup neuronové sítě, jako například jednotlivé pixely obrázku.

**Výstupní vrstva** Analogicky pro vstupní vrstvu: za výstupní vrstvu považujeme ty neurony, ze kterých čteme hodnoty po vyhodnocení neuronové sítě a porovnáváme je s očekávanými hodnotami.

**Skrytá vrstva** Skrytá vrstva, je taková vrstva, která není vstupní ani výstupní. Aby tedy taková vrstva existovala, síť musí mít alespoň 3 vrstvy: vstupní, skrytou a výstupní.

Jedny z nejjednodušších neuronových sítí jsou sítě jednovrstvé a dopředné. Taková síť se skládá pouze ze vstupních uzlů a výstupní vrstvy neuronů [13]. Aby daná síť byla dopředná, tak musí mít spojení mezi neurony pouze ve směru na výstup. Takováto jednoduchá síť je vyobrazena na obrázku 2.7.



Obrázek 2.7: Dopředná jednovrstvá neuronová síť. Nachází se zde pouze vstupní a výstupní vrstva. Převzato z [13].

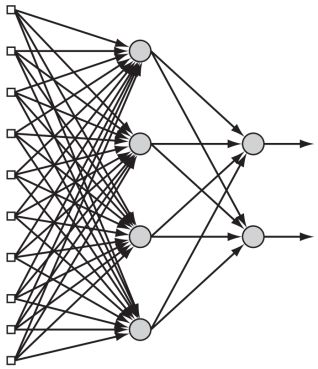
Tří a více vrstvé neuronové sítě nazýváme **hlubokými neuronovými sítěmi**. Musí tedy obsahovat alespoň jednu skrytou vrstvu 2.7.

Dalším významným typem jsou **rekurentní neuronové sítě**. Jsou to takové sítě, kde je alespoň jeden výstup neuronu zapojen do nějakého neuronu ve vrstvě bližší vstupní vrstvě. Jedná se tedy o opak dopředné neuronové sítě.

Důležitá je zejména tehdy, když je potřeba si udržovat nějaký záznam o předchozích vstupech. Tento typ sítí se často využívá při rozpoznávání hlasu nebo rozpoznávání písma [4].

## 2.4 Zpětná propagace

Jedním ze základních algoritmů používaných při učení neuronových sítích s učitelem je zpětná propagace (anglicky **backpropagation**) [26]. Tento algoritmus nám dokáže vypočítat vhodnou úpravu váhy pro každou synapsi v neuronové síti na základě výpočtu gradientu ztrátové funkce 2.4.1.



Obrázek 2.8: Dopředná třívrstvá neuronová síť. Vrstvu neronů, která není výstupní nazýváme skrytou - v tomto případě první vrstva neuronů. Této síti říkáme plně propojená, protože každý uzel v grafu má spojení s každým uzlem v následující vrstvě. Převzato z [13].

### 2.4.1 Ztrátová funkce

Ztrátová funkce (anglicky **loss function**) je v neuronových sítích důležité měřidlo, podle kterého určujeme, jak moc správné výsledky nám neuronová síť vrací. Její hodnota je vždy nezáporná a čím vyšší hodnotu má, tím hůř máme natrénovanou neuronovou síť. Při trénování sítě se snažíme najít minimum této funkce [7]. Jednou z často používaných ztrátových funkcí je střední kvadratická chyba (anglicky Mean Squared Error (MSE)), jinak také používaná k lineární regresi. Její předpis je:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2 \tag{2.7}$$

kde $y^{(i)}$ je očekávaná výstupní hodnota (vektor) neuronu, $\hat{y}^{(i)}$ je skutečná výstupní hodnota (vektor), $i$ je index příkladu z datasetu a $n$ je počet příkladů.

Když nám síť vrátí špatný výsledek pro daný příklad, zajímá nás, jaké váhy mají největší dopad na aktuální výsledek a tedy jaké váhy máme změnit (Hebbianova teorie[24]). Na co se tedy ptáme, je jak moc nám změna váhy změní výsledek ztrátové funkce. To si můžeme vyjádřit jako parciální derivaci:

$$\Delta E = \frac{\partial E}{\partial w_{ij}} \tag{2.8}$$

kde $E$ je ztrátová funkce, $w$ je $j$. váha $i$. neuronu.

Pokud si chybovou funkci zapíšeme jako $E = E(y)$, kde $y$ je skutečná hodnota výstupního neuronu. A pokud pro každý neuron $j$ můžeme vypočítat jeho výstupní (ang. *output*) hodnotu tímto vzorcem[26]:

$$o_j = \varphi(net_j) = \varphi(\sum_{k-1}^{n} w_{kj}o_k) \tag{2.9}$$

kde $o_k$ je výstupní hodnota předchozího neuronu (v první vrstvě neuronů to bude $x_k$, tedy vstup), $\varphi$ je aktivační funkce neuronu a $w_{kj}$ je váha synapse mezi neuronem $k$ a $j$. Z předchozích vzorců si můžeme všimnout že pro výpočet parciální derivace $\Delta E$ bude potřeba využít pravidlo pro derivování vnořených funkcí, protože $\varphi$ je vnořená do $o_j$ a $o_j$ je pak vnořená do $E$.

(Následující výpočty jsou převzaty z [26].) Pro vyjádření parciální derivace chybové funkce $E$ na základě změny váhy $w_{ij}$ pak získáváme vzorec

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} . \frac{\partial o_j}{\partial net_j} . \frac{\partial net_j}{\partial w_{ij}} \tag{2.10}$$

kde vidíme, že výsledek parciální derivace chybové funkce pro váhu $w_{ij}$ je závislý na výstupní hodnotě pravého neuronu $net_j$, jeho hodnotě $o_j$ před aplikací aktivační funkce a také na $E$ pravého neuronu.

Poslední derivaci si můžeme obecně vyjádřit jako

$$\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \varphi(\sum_{k-1}^{n} w_{kj} o_k) = \frac{\partial}{\partial w_{ij}} w_{ij} o_i = o_i \tag{2.11}$$

Ostatní derivace už musíme dopočítat vždy v závislosti na tom, jakou použijeme aktivační funkci a jaký vzorec použijeme pro výpočet chyby. Pro tento příklad použiji jako vzorec pro výpočet chyby $E = \frac{1}{2}(t-y)^2$, kde $t$ je aktuální výstupní hodnota a $y$ je hodnota očekávaná. A jako aktivační funkci použiji dříve zmíněnou logistickou funkci $\varphi(x) = \frac{1}{1+e^{-x}}$, jejíž derivace je $\varphi(z)(1-\varphi(z))$.

Teď už si můžeme vyjádřit ostatní parciální derivace:

$$\frac{\partial E}{\partial o_j} = \begin{cases} o_j - t, & \text{pokud j je výstupní neuron} \\ \sum_{l \in L} (\frac{\partial E}{\partial o_l} \frac{\partial o_l}{\partial net_l} w_{jl}), & \text{pokud j je vnitřní neuron} \end{cases} \tag{2.12}$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial}{\partial net_j} \varphi(net_j) = \varphi(net_j)(1 - \varphi(net_j)) \tag{2.13}$$

Po dosazení parciálních derivací do původního vzorce dostáváme:

$$\Delta E = \frac{\partial E}{\partial w_{ij}} = \delta_j o_i \tag{2.14}$$

kde

$$\delta_j = \begin{cases} (o_j - t_j) o_j (1 - o_j), & \text{pokud j je výstupní neuron} \\ (\sum_{l \in L} w_{jl} \delta_l) o_j (1 - o_j), & \text{pokud j je vnitřní neuron} \end{cases} \tag{2.15}$$

Výsledná hodnota se použije v optimalizačním algoritmu 2.4.2 ke zjištění vhodné úpravy argumentů.

### 2.4.2 Optimalizační algoritmus

V kontextu umělých neuronových sítí je optimalizační algoritmus algoritmem, použitý ke zjištění lokálního minima chybové funkce. Snažíme se upravit parametry sítě tak, aby nám vracely co nejmenší chybu, tedy co nejlepší výsledek. Jedním z nejpoužívanějších z těchto algoritmů je **gradientní sestup** (známý taky jako slézání z kopce). Jeho princip je takový, že si vypočítáme gradient pomocí zpětné propagace 2.4 a uděláme změny parametru ve směru největšího sestupu. Velikost změny závisí na tom, jak nastavíme parametr $\gamma$ - míra učení (anglicky **learning rate**). Vzorec pro gradientní sestup je $a_{n+1} = a_n - \gamma \Delta E(a_n)$ [5]. V naší síti si to můžeme představit tak, že $a_n$ je aktuální hodnota váha nebo bias, $a_{n+1}$ je hodnota následující a $\Delta E$ je gradient vypočítaný pomocí zpětné propagace. Protože nevíme jak se funkce chová, musíme zvolit dostatečně malou míru učení, protože gradient

určuje jak funkce klesá/roste pouze v jednom daném bodě, a pokud bychom se posunuli příliš, mohli bychom se posunout nesprávným směrem - tedy ne k minimu. U gradientního sestupu je důležité, že nejdříve vypočítáme změnu argumentů přes celý dataset a až poté upravíme hodnotu parametrů sítě. Tato metoda ale může být velmi náročná na zdroje. Proto se vytvořily i jisté modifikace. Jednou z nich je **stochastický gradientní sestup** u kterého měníme parametry neuronové sítě po každém vzorku z datasetu. Další známou variantou je malý dávkový gradientní sestup (anglicky **Mini Batch Gradient Descent**), kde vzorky z datasetu náhodně zamícháme, vytvoříme z celého datasetu několik dávek a měníme hodnoty parametrů po každé takovéto dávce.

## 2.5 Hluboké neuronové sítě

Jak je ukázáno v práci *When and Why Are Deep Networks Better than Shallow Ones?* [14] hluboké neuronové sítě jsou lepší pro aproximaci komplexnějších funkcí. Sice můžeme podobnou aproximaci vytvořit i pomocí jen jedné vrstvy neuronů, ale většinou tato síť bude mít mnohem více parametrů než několikavrstvá, méně široká síť.

Jednotlivé vrstvy nám můžou dobře abstrahovat komplexní problém a rozdělit ho na mnoho primitivních problémů. Například při detekci objektů na obrázku nám první vrstva sítě může zpracovat hrany, další už tyto hrany bude spojovat do nějakých složitějších objektů jako třeba krychle a další vrstva už může detekovat ještě komplexnější objekty.

### 2.5.1 Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou poddruhem hlubokých neuronových sítí používané zejména v zpracování zvuku a obrazu. Konvoluční neuronové sítě jsou navržené tak, aby dobře rozpoznávaly vzory ve vstupních signálech a aby byly nezávislé vůči jejich posunu, změně velikosti nebo rotaci. V těchto sítích jsou důležité následující vlastnosti[13]:

- **Extrakce vlastností**. Síť extrahuje vlastnosti z obrazu lokálně bez ohledu na celkovou pozici v obrazu. Ta je méně důležitá, dokud je uchována přibližná poloha vzhledem k ostatním vlastnostem v obrazu.
- **Mapování vlastností**. Každá z výpočetních vrstev neuronové sítě se skládá z několika aktivačních vrstev (anglicky *feature map*) a pro každou tuto mapu má více neuronů sdílené váhy synapse. Touto vlastností sítě si zajistíme invarianci vůči posunu a redukci počtu parametrů.
- **Vzorkování (supersampling)**. Každá konvoluční vrstva je následována výpočetní vrstvou, která vykoná lokální zprůměrování a vzorkování tak, aby měla výsledná mapa vlastností menší rozměry. Tato operace má za výsledek, že zmenší citlivosti sítě na různé transformace objektu v obraze.

Konvoluční neuronová síť se typicky skládá ze čtyř druhů vrstev: plně propojené, konvoluční, ReLU a spojovací (anglicky pooling).

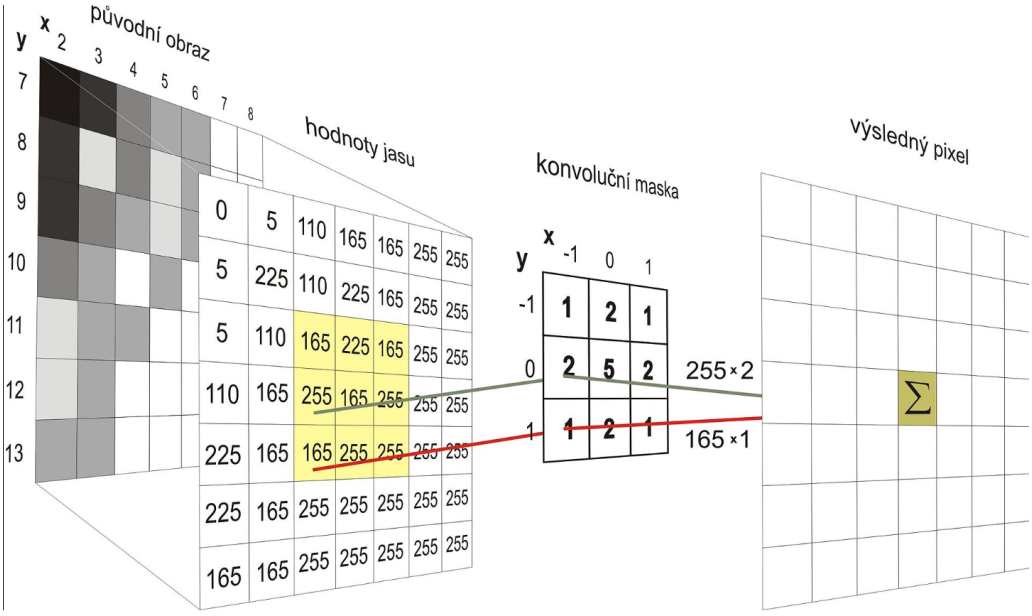
#### Konvoluční vrstva

První vrstvou v konvolučních neuronových sítích je vždy konvoluční vrstva. Dle názvu poznáme, že se tu provádí konvoluce 2.5.1 vstupního signálu.

**Konvoluce** V matematice je konvoluce operace nad dvěma funkcemi. Můžeme si ji vyjádřit jako $(f * g)(x) = \int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)\, d\alpha$ [6], kde $f$ a $g$ jsou vstupní funkce. Tento vzorec platí pro spojité funkce. Jelikož se v počítačích signál vzorkuje a ukládá se diskrétně, vzorec je třeba trochu upravit, vzniká nám:

$$f(x,y) * g(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x-i, y-j)g(i,j) \tag{2.16}$$

Tento vzorec se může těžko představovat, ale jeho princip si můžeme lehce zobrazit na obrázku 2.9



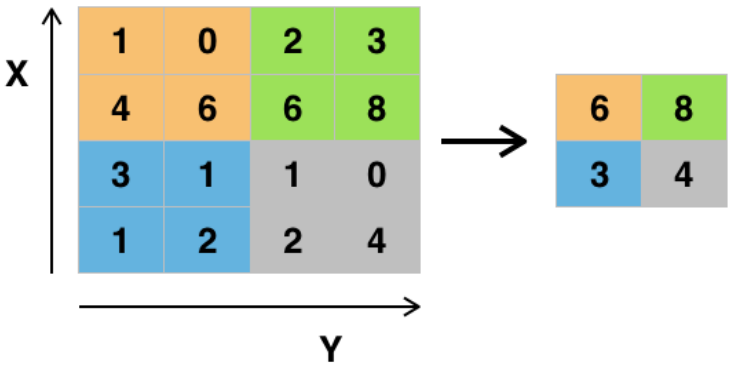

Obrázek 2.9: Grafická ukázka výpočtu konvoluce na černobílém obrázku. Původní obraz se reprezentuje do dvourozměrného pole v hodnotách 0-255, poté se přiloží konvoluční maska na vytvořené pole a vynásobíme každou hodnotu z konvoluční masky s hodnotou obrazu. Výsledek sečteme a získáváme tím výslednou hodnotu pixelu [28].

Pokud takto aplikujeme konvoluční masku na celém vstupu, vzniká nám tzv. **aktivační vrstva**, pokud použijeme více filtrů, vzniká nám více aktivačních vrstev. Ve většině architektur konvolučních neuronových sítí jsou tyto vrstvy převzorkovány spojovací (*pooling*) vrstvou 2.5.1.

### Spojovací (pooling) vrstva

Pooling vrstva je většinou po konvoluční vrstvě. Její hlavní účel je zmenšit vstupní signál (u 2D tedy jeho $x$ a $y$ rozměry) pro další konvoluční vrstvu. Činnost, kterou tato vrstva vykonává, se nazývá převzorkování. S tím přichází i ztráta informací, ale i to může být v neuronových sítích žádoucí, protože to zejména snižuje výpočetní náročnost dalších vrstev (zmenšení neuronové sítě) a také to redukuje **přeučení** (*over-fitting*) - síť je schopna si vstupní vzorky zapamatovat a negeneralizuje, poté funguje hůře nebo vůbec na vzorcích, ze kterých se neučila [9].

Podobně jako v předchozím případě přikládáme na vstupní signál jakousi mřížku, ze které vybíráme maximum a to nám vytvoří převzorkovaný výstup.




Obrázek 2.10: Na obrázku vidíme graficky vyobrazené převzorkování pomocí max-poolingu. Pro každou nepřekrývající se možnost vybereme z původního signálu největší hodnotu(pokud není velikost okénka n-násobkem velikosti vstupního obrazu nebo signálu, můžeme mít i variantu, kde bereme v potaz překrývající se možnosti). Převzato z [27].

Je více implementací samotného převzorkování, ale v poslední době je nejoblíbenější tzv. **max-pooling**, jak z názvu vychází, vybíráme vždy prvek s největší hodnotou. Mimo to existuje ještě varianta, která vezme průměr z hodnot v okně. Max-pooling se využívá díky lepšímu výkonu [19].

### Aktivační vrstva

Aktivační vrstva se používá k zavedení nelinearity do konvoluční neuronové sítě pomocí nelineárních aktivačních funkcí. Většinou se nachází vždy za vrstvou konvoluční. Jak je uvedeno výše, existuje mnoho funkcí které se dají použít, v praxi bylo ale ukázáno že ReLU podává velmi dobré výsledky v poměru výkon a zatížení počítače [18].

### Plně propojená vrstva

Na konci každé konvoluční neuronové sítě se nachází plně propojená neuronová síť, která na základě rozpoznaných vlastností v signálu rozhoduje, jaké složitější celky se ve vstupu nacházejí, a její výstup je výstup celé konvoluční neuronové sítě [17].

### Příklad konvoluční neuronové sítě

Jako příklad známé konvoluční neuronové sítě uvedu **GoogleNet** (také známá jako Inception V1) od společnosti Google. V roce 2014 vyhrála prestižní soutěž ILSVRC a se svou chybou pouze 6,67 % se velmi přiblížila výkonu samotného člověka. Síť byla inspirovaná sítí LeNet, ale byl u ní předělán tzv. inception modul. Tento modul je založený na několika velmi malých konvolucích, který drasticky sníží počet parametrů. Podařilo se vytvořit 22 vrstvou výslednou síť se snížením parametrů z 60 milionů na pouhé 4 miliony [23].

# Kapitola 3

# Oblaky

## 3.1 Motivace

Často se díváme na oblohu a přemýšlíme nad tím, jaké by mohlo být počasí. Poměrně intuitivně se můžeme podívat na čistou modrou oblohu a říct, že v následujících chvílích bude hezké počasí, nebo nebude pršet. Horší to ale je, pokud je obloha zatažená, nebo se na ní nachází oblaka pro nás neznámého typu. V tom případě už může pouze zkušený člověk nebo meteorolog odhadnout, jaké bude počasí.

## 3.2 Druhy oblak

Světová meteorologická organizace dělí oblaka na 10 základních druhů [22]. Liší se tvarem, výškou, podmínkami, za kterých vznikají apod. Mimo základních 10 druhů se dají oblaky dělit do jednotlivých odrůd, ale to nebudu brát v potaz, protože to pro náš účel předpovědi počasí není podstatné.

U následujícího výpisu druhů jsou obrázky převzaty z [29] a popisy jsou inspirovány z Atlasu Mraků [21].

### 3.2.1 Cirrus

Obrázek 3.1: Cirrus.

Cirrus je oblak připomínající lehké, tenké nitky, nebo průsvitná vlákna, bez vznikajícího stínu [29]. Tato vlákna nebo nitky můžou být přímočaré, ale i zakřivené. Musí ale vytvářet patrnou nehomogenní strukturu. Občas tato vlákna končí zaoblenými háčky, které nám pomůžou tento typ oblak jednoduše rozpoznat. Pokud se díváme na oblohu blíže k obzoru, může se nám zdát, že se oblaka zdánlivě sbíhají a vzniká oblak Cirrostratus, ale jedná se jen o optický klam. Tento oblak má velmi světlou, bílou barvu a při západu nebo východu slunce mění svoji od žluté přes růžovou až k červené. Na tomto oblaku můžeme pozorovat halové jevy, ale vzhledem k tomu, že mají malou rozlohu, halové jevy vidíme jako neúplné.

Při rozpoznávání si musíme dávat pozor na podobnost s oblaky Cirrocumulus, Cirrostratus, Altocumulus a Altostratus. Cirrocumulus má narozdíl od Cirrusu menší jednotlivé části, u Cirru se jednotlivé vlákna můžou protahovat na několik procent prostorového úhlu, ale jednotlivé chuchvalce Cirrocumulu by měly mít pod 1 stupeň prostorového úhlu.

Cirrus se řadí mezi vysoké oblaky, vzniká v 8 - 13 km ve velmi nízkých teplotách a skládá se z malých krystalků vody [29]. Cirrus často vzniká z Cirrocumulů, Altocumulů nebo nejvyšších částí Cumulonimbů. Občas taky může vzniknout z Cirrostratu, který má různou tloušťku, a části s menší tloušťkou se vypaří. Zaoblené chomáčky na konci Cirru vznikají nejčastěji na bezoblačné obloze.

**Předpověď -** dá se předpokládat hezké počasí, ale musíme si dávat pozor na výskyt oblak druhu Cirrostratus nebo Altostratus [11].

### 3.2.2 Cirrocumulus

Obrázek 3.2: Cirrocumulus.

Cirrocumulus připomíná drobounké vločky nebo obláčky. Vločkovitá struktura oblaku je více nebo méně pravidelná [29]. Vločky a obláčky se na obloze vyskytují v menších nebo větších skupinách a jejich důležitá vlastnost je, že stejně jako Cirrus nevytvářejí vlastní stín. Jednotlivé obláčky by neměly přesahovat jeden stupeň prostorového úhlu. Cirrocumulus se většinou vyskytuje jako rozsáhlé plochy složené z menších oblačných částí, které mají podobu zrn nebo vln. Tyto plochy můžou mít řasnaté okraje a tak nám můžou připomínat oblak Cirrus. U těchto oblak můžeme díky jejich velké průhlednosti jednoduše rozpoznat polohu slunce, nebo měsíce. Stejně jako u Cirru, i zde můžeme pozorovat částečné halové jevy.

Stejně jako ostatní oblaky v nejvyšší vrstvě se skládá výhradně z ledových krystalků, ale můžeme v tomto oblaku občasně najít i velmi přechlazené vodní kapičky, které rychle přecházejí v ledové krystalky.

U rozpoznávání si opět musíme dávat pozor na oblaka podobného vzhledu jako Cirrus, Cirrostratus a Altocumulus. Oproti Cirru a Cirrusratusu tu musí valně převažovat malé, téměř průhledné obláčky. Sice se tu občasně může objevit i Cirrus, nebo větší jednolitá plocha podobná Cirrostratusu, pokud převažují obláčky, můžeme tento oblak považovat za Cirrocumulus. Altocumulus je vzhledově velmi podobný zejména v obláčkovém vzhledu (jak napovídá Cumulus v názvu). Oproti Cirrocumulu se ale nachází v nižších výškách a zpravidla vytváří výrazný vlastní stín. Také jednotlivé obláčky jsou u Cirrocumulu dosti menší než u Altocumulu.

**Předpověď -** dá se předpokládat pozvolný příchod srážek v následujících 12 - 24 hodinách, zejména pokud je doprovázen oblaky Cirrostratus a Altostratus [11].

### 3.2.3 Cirrostratus

Obrázek 3.3: Cirrostratus.

Cirrostratus se dá popsat jako oblak připomínající téměř průhledný bělavý závojem, může ale nemusí mít zjevnou strukturu. Často lze na něm pozorovat halové jevy. [29] Uvnitř Cirrostratu můžeme pozorovat jemné žebrování, ne ale příliš výrazné. Cirrostratus má díky své poměrně velké transparenci a malé tloušťce velmi světlou barvu a také díky tomu nevrhá výrazný stín. Občas je Cirrostratus tak tenký, že je téměř nepozorovatelný a pouze halové jevy prozrazují jeho existenci. Objekty na zemi při zakrytém slunci Cirrostratem stále vrhají výrazné stíny. Jeho kraje můžou mít výraznou hranici, ale většinou je závoj ohraničen a roztřásněn Cirry.

Cirrostratus stejně jako Cirrus a Cirrocumuls patří do nejvyšší vrstvy a vzniká ve výšce 5 až 13 km [29]. Stejně tak se skládá zejména z ledových krystalek. Tento oblak může vzniknout tím, že se rozsáhlé vrstvy vzduchu zvedají až do velkých výšek, často také třeba spojením Cirrů nebo částí Cirrocumulů. Další způsob jeho vzniku je zmenšení tloušťky Altostratu nebo rozšířením vysokých částí Cumulonimbu.

Při identifikaci toho oblaku si musíme dávat opět pozor na několik vzhledově podobných oblak: Cirrus, Cirrucomulus, Altocumulus, Altostratus a Stratus. Oproti Cirrusu má Cirrustratus mnohem větší horizontální rozsah a má více homogenní strukturu. Oproti

Altocumulu a Cirrocumulu se u Stratusu nevyskytují žádné kupy, které jsou podstatou Cumulů. Velmi těžko se z fotografie rozeznává rozdíl Altostratem, Stratem a Cirrostratem, Cirrostratus můžeme poznat zejména podle menší tloušťky a tedy i světlejší barvy. Také často má oproti Altostratu a Stratu vláknitou strukturu.

**Předpověď -** Můžou se objevit srážky v následujících 12 - 24 hodinách, zejména když oblaka stoupají [11].

### 3.2.4 Altocumulus

Obrázek 3.4: Altocumulus.

Altocumulus jsou oblaky připomínající oblázky, bílé až šedé barvy majících vlastní stín. Jednotlivé oblázky bývají oddělené, ale můžou i souviset a tak vytvářet objekty podobající se vlnám, které při spojení můžou připomínat včelí plást nebo síť. Oblak tedy můžeme považovat za Altocumulus i přes to, že nevidíme oblohu, kvůli spojení jednotlivých oblázků. Důležité ale je, aby byly oblázky výrazné, ve středu tmavší než u kraje díky jejich tloušťce. Jednotlivé plochy se můžou dělit i bezoblačnou ostře ohraničenou oblastí. Velikost jednotlivých pravidelně uspořádaných oblázků je asi jeden až pět stupňů prostorového úhlu. Slunce prosvítá, jeho kotouč je nejasně ohraničený a kolem se vytváří malý kruh [29]. Často je ale Altocumulus rozdílné výšky, a proto v některých místech může být slunce zcela zastíněno a někde může prosvítat. Pokud z oblaku vypadávají krystalky, můžou se tu vyskytovat i některé halové jevy.

Tento typ oblak patří do středního výškového pásma a vzniká ve výšce 1,5 až 7 km, skládá se z kapiček vody, avšak za nízkých teplot se může skládat i z ledových krystalů [29]. Oblak vzniká na okraji kondenzující rozsáhlé vzduchové vrstvy, nebo také při turbulenci (nerovnoměrný, střídavý pohyb vzduchu ve vertikálním směru) nebo konvekci (uspořádaný pohyb vzduchu ve vertikálním směru).

Při identifikaci si opět musíme dávat pozor na několik vzhledově podobných oblak: Cirrus, Cirrocumulus, Altostratus, Stratocumulus a Cumulus. Někdy z Altocumulu vybíhají směrem dolů pruhy jako vlečka vláknité struktury (virga) a můžou nám tím připomínat Cirrus. I v tomto případě ale musíme označovat tento oblak za Altocumulus a to do té doby, dokud nemá vláknitou strukturu a výrazně světlou barvu. Altocumulus je lehce rozeznatelný od Cirrocumulu díky jeho menší transparenci a i očividně menší výšce. Altocumulus si

můžeme lehce splést s Altostratem, pokud má ale oblak viditelnou balvanovitou strukturu, považujeme ho za Altocumulus. Stratocumulus můžeme od Altocumulu rozeznat tak, že změříme velikost jednotlivých balvanů. V případě, že jsou mezi jedním a pěti stupni prostorového úhlu, považujeme oblak za Altocumulus. Podobně to je i s Cumulusem. Malé oblaka Cumulus také oproti Altocumulu nemají vláknité pruhy a celistvou vláknitou strukturu.

**Předpověď -** stejná jako u Cirrocumulu 3.2.2.

### 3.2.5 Altostratus

Obrázek 3.5: Altostratus.

**Popis vzhledu -** Altostratus je většinou velmi rozměrný, horizontálně se může táhnout až tisíce kilometrů a na výšku v řádek stovek metrů. Na pohled to je šedavá nebo namodralá oblačná plocha s vláknitou nebo žádnou strukturou [29]. Plocha Altostratu může být poměrně tenká, a proto přes ni můžeme vidět Slunce jako přes matné sklo, většina plochy je ale z pravidla tlustší a slunce zcela zakrývá tak, že nejsme schopni jednoduše určit polohu slunce a objekty na zemi nevrhají stín. Plocha je homogenní a nevyskytují se na ní žádné výraznější tvary. V některých případech může Altostratus tvořit více horizontálních vrstev oblak, které spolu na některých místech souvisí. Z Altostratu můžou vypadávat srážky, které se projevují jako pruhy pod základnou oblaku (virga), a spodní vrstva oblaku pak vypadá roztrhaně.

Stejně jako Altocumulus, i oblak Altostratus patří do středního výškového pásma a nachází se ve výšce 2 až 6 km. Skládá se jak z vodních kapek, tak z krystalů, občasně i ze sněhových vloček[29]. V počátečním vývoji tohoto oblaku na obloze můžeme vidět několik menších a osamocených Altostratů, vyskytujících se poměrně nízko. Poté se zvyšuje jejich tloušťka a šířka, které se postupně spojují do větších útvarů, až vzniká jedna velká souvislá vrstva. Mimo to může vzniknout i zesílením vrstvy Cirrostratu nebo naopak zeslabením vrstvy nimbostratu. Dalším možným způsobem vzniká z Altocumulu, pokud z něho vypadávají ve větší míře ledové krystalky. Stejně jako Cirrostratus, i tento oblak může vzniknout rozšířením střední nebo vrchní vrstvy Cumulonimbu.

Tento oblak je podobný několika dalším: Cirrus, Cirrostratus, Altocumulus, Stratocumuls, nimbostratus a Stratus. Při svém vzniku nebo rozpadu můžou jednotlivé plošky se strukturou připomínat Cirrus, ale Cirrus má oproti nim mnohem světlejší barvu a nevysky-

tuje se ve větších horizontálních plochách. Od Cirrostratu můžeme Altostratus rozpoznat absencí halových jevů, více odstínů do šedé barvy díky větší výšce a také jen málo viditelným sluncem. Od Altocumulu a Stratocumulu můžeme Altostratus rozpoznat absencí jakýchkoliv kupovitých struktur a tudíž větší vzhledovou homogenností. Od nimbostratu se liší lépe procházejícím světlem a tudíž světlejší barvou, mimo to má Altostratus méně jednotvárnou strukturu. Altostratus a Stratus jsou téměř identické, krom toho že Stratus bývá o mnoho světlejší proti slunci.

**Předpověď -** předpověď podobná jako u Cirrostratu 3.2.3.

### 3.2.6 Stratocumulus

Obrázek 3.6: Stratocumulus.

Oblaky druhu Stratocumulus mají šedavý nebo bělavý vzhled s tmavými místy. Oblak se jako v jiných případech podobá valounům, dlaždicím nebo oblázkům. Ve většině případů nemá žádné známky vláknité struktury. Stejně jako Altocumulus, jednotlivé balvany můžou být propojené nebo oddělené. Velikost balvanů přesahuje pět stupňů prostorového úhlu, jsou tedy výrazně větší než u Altocumulu. Samotný tvar balvanů se ale výrazně liší, někdy se může Stratocumulus skládat ze zdánlivě rovnoběžných valounů, jindy se zase může vyskytovat jako jeden velký podlouhlý valoun. Plochy Stratocumulů se můžou vyskytovat i ve více vrstvách. Průsvitnost může být různorodá právě v závislosti na tvaru a velikosti valounů. Měla by se tu ale v nejširších místech vyskytovat tmavá místa.

Stratocumulus se řadí do oblak nízkého patra a vyskytuje se do výšky 2 km. Vzniká většinou rozpadem jiného druhu oblaku díky turbulentnímu proudění , většinou z oblaku typu Cumulus. Skládá se z vodních kapiček, které jsou občasně doprovázeny dešťovými kapkami a zřídka i ledovými krystalky a vločkami [29]. Stratocumulus může vzniknout z větších jednotlivých balvanů Altocumulu. Také často vzniká pod Altostratem nebo Nimbostratem, občas také vzniká rozpadem nimbostratu. Vzniká také ze Stratu, buďto jeho zvýšením nebo zvlněním bez změny výšky. Mimo to vzniká i rozšířením střední nebo horní partie Cumulonimbů nebo zploštěním Cumulů.

Rozpoznání Stratocumulu bývá složité, je podobný několika dalším: Cirrostratus, Altocumulus, Altostratus, Nimbostratus, Stratus, Cumulus. Od Cirrostratu ho rozpoznáme podle jeho tmavší barvy, menší transparenci a balvanovité struktuře. Jak už bylo řečeno

výše, od Altocumulu ho můžeme rozeznat pomocí velikosti jednotlivých balvanů, Stratocumulus je má i hodně přes pět stupňů prostorového úhlu. Od oblak typu Altostratus, Nimbostratus a Stratus jej rozeznáme opět podle výrazné balvanovité struktury a absence vláken. Od Cumulu ho rozeznáme pomocí jeho zploštělého tvaru. Vrch Cumulu má tvar kupole, ale Stratocumulus má spíše rovný vrch.

**Předpověď -** nastává přetrvávající zatažená obloha nebo její pomalé vyjasňování [11].

### 3.2.7 Stratus

Obrázek 3.7: Stratus.

Stratus se vzhledově téměř neliší od mlhy [29], je to šedá, až velmi tmavá oblačná vrstva s poměrně jednotvárnou strukturou. Pokud přes něj jde vidět Slunce, má jasný obrys, na rozdíl od Altostratu. Často ale Slunce i Měsíc zcela zakrývá. Halové jevy se u něho krom speciálních případů za velmi nízké teploty nevyskytují.

Všeobecně se skládá z malých vodních kapiček, při nižších teplotách se ale může skládat i z ledových částic. Je-li velmi hutný, často obsahuje vodní kapičky, které způsobují mrholení nebo v nižších teplotách ledové jehličky nebo zrna.

Stratus vzniká velmi nízko, často jen několik metrů nad zemí ochlazením nejspodnější části atmosféry. Nejčastěji vzniká z mlhy [29]. Roztrhaný Stratus taky může vzniknout následkem turbulence ve spojení s padáním srážek z Altostratu, Nimbostratu, Cumulonimbu nebo Cumulu. V přímořských oblastech často vzniká mlhou vznikající na moři, která se díky větru přiblíží ke břehu, kde díky zahřátí zemského povrchu pevniny stoupá a stává se zní oblak Stratus.

Při rozpoznávání si můžeme nejvíce plést s následujícími oblaky: Altostratus, Nimbostratus a Stratocumulus. Na rozdíl od Altostratu, u kterého je slunce rozmazané na mnohonásobně větší plochu, u Stratu je slunce tmavší, ale zachovává si stejnou velikost. Oproti Nimbostratu je Stratus skoro nerozeznatelný, jediným vizuálním rozdílem je, že Stratus má mnohem jednotvárněji ohraničenou spodní hranu. Oproti Stratocumulu se Stratus odlišuje absencí menších oblačných, valounovitých částí.

**Předpověď -** žádné výrazné srážky, maximálně mrholení nebo v nižších teplotách lehké sněžení, Stratus se pomalu vytratí nebo zůstane zataženo [11].

### 3.2.8 Nimbostratus

Obrázek 3.8: Nimbostratus.

Nimbostratus může horizontálně dosáhnout velikosti až tisíce kilometrů, vypadá jako šedá nebo namodralá plocha. Je to typický srážkový oblak, ze kterého vypadává trvanlivý déšť [29]. Vrstva oblak je všude tak hustá, že nemůžeme určit polohu Slunce. Pod touto hustou vrstvou se často vyskytují další nízké roztrhané oblaky. Občas lze pozorovat trhání Nimbostratu na několik samostatných vrstev, které se zase spojují.

Tento oblak se skládá z vodních kapiček a dešťových kapek, za menší teploty pak z ledových krystalů a sněhových vloček, nebo směsi těchto kapalných a pevných částic. Tyto částice z tohoto oblaku většinou padají ve formě deště, mrholení nebo sněžení.

Skládá se jak z vodních kapek, tak z krystalků [29]. Nimbostratus se řadí mezi oblaka spodního patra, může mít ale i znatelný vertikální růst a dosahovat až středního patra. Co se týče jeho vzniku, tak se nejdříve oblaka skládají z menších roztrhaných částí až do jedné velké jednolité plochy.

I tento oblak je velmi podobný několika dalším: Altostratus, Altocumulus, Stratocumulus, Stratus a Cumulonimbus. Lehce si ho spleteme s Altostratem, jediným výrazným rozdílem je Nimbostratova tmavší barva a neprostupnost slunce. Oproti Altocumulu a Stratocumulu nemá zřetelně ohraničené části, nemá zřetelnou základnu ani balvanovitou strukturu. Stratus je vzhledově téměř identický, jen může být trochu světlejší. Hlavní rozdíl je, že ze Stratu většinou nepadají žádné srážky, maximálně mrholení nebo za nižších teplot sněhové zrnka. Pokud se nacházíme přímo pod Comulonimbem, opět ho pouze vizuálně nedokážeme odlišit. Pokud padají z oblak kroupy nebo slyšíme hřmění a vidíme blesky, jedná se konvenčně o Comulonimbus.

**Předpověď -** velmi podobná jak u Altostratus 3.2.5.

### 3.2.9 Cumulus

Obrázek 3.9: Cumulus.

Oblaky typu Cumulus jsou velké, husté a osamocené obláčky, které mají tmavší a většinou vodorovné základy díky stínu, který vrhají. Vrchní část je jasně ohraničená a jeví se jako čistě bílá [29]. Jednotlivé Cumuly se od sebe liší různým stupněm vertikálního vývoje. Mohou mít jen malou výšku a malé oblé vrcholky. Mohou ale také mít velký vertikální rozsah a vrchní část rozdělenou do mnohých rostoucích výběžků vizuálně připomínajících květák. Často se také tato oblaka vyskytují s roztrhanými oblaky, které se díky vzdušnému pohybu rychle mění.

Cumulus vzniká za vzestupných proudů v atmosféře, které ve větší výšce (2-7 km nad povrchem země) kondenzuje do vodních kapiček [29]. Tyto proudy vzduchu nejčastěji vznikají díky zahřívání zemského povrchu slunečním zářením. Mimo to můžou Cumuly vzniknout z Altocumulů nebo Stratocumulů nebo přeměnou Stratocumulů nebo Stratu.

Stejně jako u ostatních oblak, i tento se lehce splete s některými podobnými oblaky: Altocumulus, Stratocumulus a Cumulonimbus. Malé Cumuly nám mohou připomínat Altocumuly nebo Stratocumuly, zvláště pokud jsou od nás vzdáleny. Tato oblaka klasifikujeme jako Cumuly, pokud jejich vrcholek stále připomíná kopuli a jejich základny se nespojily. Protože Cumulonimbus vzniká z Cumulu, je těžké ho klasifikovat pouze z fotografie. Pokud z něho neprší nebo pokud není zjevná vláknitá struktura, považujeme oblak za Cumulus.

**Předpověď -** můžeme předpokládat že bude hezké počasí [11].

### 3.2.10 Cumulonimbus

Obrázek 3.10: Cumulonimbus.

Cumulonimbus připomíná svým vzhledem Cumulus, ale je mnohem vyšší. Jeho horní část připomíná tvarem kovadlinu [29]. Jeho vertikální rozsah většinou protíná všechny výškové patra a kdežto jeho spodní část připomíná Cumulus, jeho vrchní část bývá hladká, vláknitá a téměř vždy zploštělá. Základna je díky jeho obrovské výšce a hustotě zejména velmi tmavá a můžou se tu vyskytovat další roztrhané oblaky. Tyto jednotlivé Comulonimby se vyskytují buďto samy, nebo také v řadě, tvořící vysokou zeď. Pokud se nacházíme přímo pod tímto oblakem, vidíme pouze jeho spodní část, protože většinou nabývá velkých horizontálních rozměrů.

Jak už bylo řečeno, tento oblak roste do velkých výšek kolem 5 nebo až 20 km [29]. Skládá se z vodních kapiček a ve vyšších částech zejména z ledových krystalů. Často ale obsahují i sněhové vločky, zmrzlý déšť, krupky či kroupy.

Cumulonimbus se postupně tvoří z horizontálně i vertikálně vyvinutých Cumulů postupným vertikálním růstem. Stejně jako Cumulus i Cumulonimbus potřebuje ke svému vzniku silné stoupavé proudy. Občas se mohou vytvořit i z Altocumulu nebo Stratocumulu, jejichž vrchní části mají výběžky ve tvaru věží.

Tento oblak si můžeme vizuálně splést pouze s oblaky typu Nimbostratus nebo Cumulus. Jak už bylo řečeno, Cumulus rozeznáme zejména absencí srážek a absencí nejvyšších částí Cumulonimbu - kovadlinovým vrchem. Nimbostratus a Cumulonimbus je jasně rozeznatelný na delší vzdálenost, pokud se ale nacházíme přímo pod daným oblakem, jsou vizuálně nerozeznatelné. Konvenčně mluvíme o Cumulonimbu, pokud padají kroupy nebo se vyskytují blesky.

**Předpověď -** V případě výskytu oblak tohoto typu můžeme očekávat déšť, bouřky a občasně i tornádo [11].

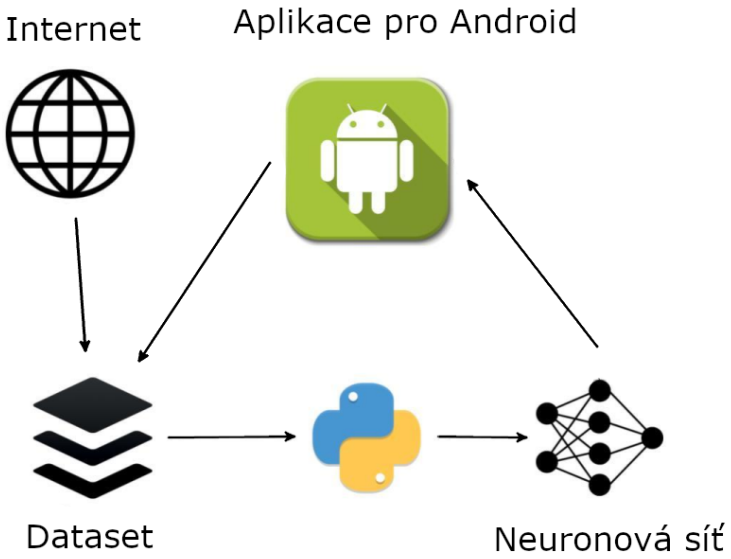
# Kapitola 4

# Návrh a implementace

Tato kapitola se bude věnovat konečné implementaci výsledného koncového produktu - mobilní aplikace předpovídající počasí. Nebudu se věnovat pouze implementaci této aplikaci, ale také tvorbě její důležité části, a to naučeným neuronovým sítím.

## 4.1 Vysokoúrovňový návrh

Samotné fungování výsledné aplikace bude výsledek několika důležitých součástí systému, které jsou vyobrazené na následujícím diagramu:




Obrázek 4.1: Grafické znázornění vysokoúrovňové architektury

Na obrázku 4.1 vidíme několik zásadních věcí: dataset se bude tvořit z volně dostupných zdrojů na internetu, navíc ho ale v budoucnu budou obohacovat fotografie uživatelů. Toto jsem navrhl kvůli budoucímu zkvalitňování rozeznávání oblak. Dataset bude použit jako vstup programu napsaného v Pythonu, jehož výstup bude naučená neuronová síť. Ta se pak vloží a bude využívat v cílové aplikaci.

## 4.2 Kategorizace oblak pro dataset a samotný klasifikátor

Některé druhy oblak mají jak podobný vzhled, tak i podobnou nebo i stejnou předpověď. Na základě prozkoumání těchto vlastností můžeme některé druhy oblak sloučit do jedné kategorie, což bude zejména výhodné při přípravě datasetu. Jednotlivé kategorie budou mít více vzorků a kategorií bude méně. Z předchozích kapitol o jednotlivých kategorií a z praktických poznatků při prvním pokusu o tvorbu datasetu jsem došel k několika závěrům.

- Musíme vytvořit samostatnou kategorii pro čistou oblohu.
- Oblak typu Cumulonimbus se vyskytuje velmi zřídka a také je jeho identifikace velice těžká, pokud není ve správném úhlu. Proto tuto kategorii budeme ignorovat.
- Oblaka typu Stratus, Altostratus a Nimbostratus jsou na fotografiích na pohled identické. V realitě můžeme oblak rozeznat podle srážek, které z tohoto oblaku vycházejí, na fotografii nám ale tato informace chybí. Spojíme proto tyto kategorie do jedné.
- Oblaka typů Cirrus, Cirrostratus, Altocumulus, Cirrocumulus, Stratocumulus a Cumulus můžeme s trochou snahy z fotografie rozpoznat. Můžeme tedy každému tomuto typu přiřadit samostatnou kategorii.

Vzniká nám tedy těchto 7 klasifikačních skupin:

#### Čistá obloha

Poměrně častý jev při hezkém počasí je čistá obloha. V tomto případě nemáme žádný oblak, který bychom mohli kategorizovat a můžeme předpokládat hezké počasí. Vzhledově se jedná o jednolitou plochu v modrých odstínech.

**Předpověď -** můžeme odhadnout přetrvávající hezké počasí.

#### Cirrus

Velmi lehce rozeznatelný oblak. Na fotografii musí převládat odstíny modré, jakožto oblohy a mimo to tu vidíme podélné bíle křivky tvořící svým tvarem jakousi vlnu, u konce často tvořící útvar připomínající háček.

**Předpověď -** můžeme předpokládat hezké počasí, ale musíme si dávat pozor na výskyt oblak druhu Cirrostratus nebo Altostratus.

#### Cirrocumulus, Altocumulus

Cirrocumulus oblak bude mít podobné barevné spektrum jako Cirrus, narozdíl od něj je ale tvořen velmi malými bělavými obláčky ve velké výšce, které nepřesahují jeden prostorový stupeň. Často se vyskytuje ve větších skupinách, které mají jasně viditelnou strukturu. Altocumulus se skládá stejně jako Cirrocumulus z velkého množství menších obláčků a v méně častých případech, když je Altocumulus tenčí a v menších obláčkách, může vypadat i stejně jako Cirrocumulus. Vzhledem k tomu že sdílejí stejnou předpověď, můžeme je zařadit do jedné kategorie.

**Předpověď -** můžeme očekávat pozvolný příchod srážek.

#### Stratocumulus

Oproti Altocumulu se jedná o horizontálně velmi větší oblaka, oproti němu se také skládá z menšího počtu oblázků a může nám připomínat Stratus.

**Předpověď -** Žádné srážky, oblaka se vyjasní nebo zůstane zataženo.

#### Cumulus

Oblak velmi podobný Stratocumulu, oproti němu se ale vyskytuje v menším počtu, s většími mezerami mezi jednotlivými balvany. Má také světlejší barvu a vrchní část zaoblenou do tvaru kopule.

**Předpověď -** Bude přetrvávat hezké počasí.

#### Cirrostratus

Fotografie tohoto oblaku jsou většinou světle šedo-modré s často se vyskytujícím sluncem, jež je trochu rozmazané.

**Předpověď -** Můžou se objevit srážky v následujících 12 - 24 hodinách, zejména když oblaka stoupají [11].

#### Stratus, Altostratus a Nimbostratus

Fotografie všech těchto oblaků vypadají jako poměrně homogenní šedavá plocha s občasnou nevýraznou strukturou.

**Předpověď -** Ve všech třech případech můžeme předpokládat nepříznivé počasí, buďto stávající nebo nadcházející.

## 4.3 Tvorba datasetu

Vzhledem k tomu, že už máme vytvořené jednotlivé třídy, můžeme se pustit do tvorby datasetu. Protože chci, aby moje koncová mobilní aplikace byla odolná, rozhodl jsem se přidat ještě jednu extra třídu a to **ne-obloha**, bude to reprezentovat fotografii, na které není obloha, nebo nezabírá dostatečně velkou část fotografie. Může se jednat třeba o fotografii oblohy, kde je na polovině fotografie nějaká budova, či strom. Takováto fotografie by nebyla klasifikátorem správně vyhodnocena a mohla by poskytnout špatné výsledky.

Tvorba datasetu se skládá ze sehnání velkého množství fotografií a jejich následné anotace. V tomto případě jsem se jednotlivé fotografie rozhodl anotovat jednoduchým vložením fotografie do příslušné třídy dle typu oblak na fotografii.

Během samotného vývoje aplikace a učení neuronové sítě jsem z důvodů popsaných dále vyzkoušel několik rozdílných způsobů tvorby datasetu. V následujících podkapitolách je popíši a zhodnotím jejich úspěšnost.

### 4.3.1 Web scraping

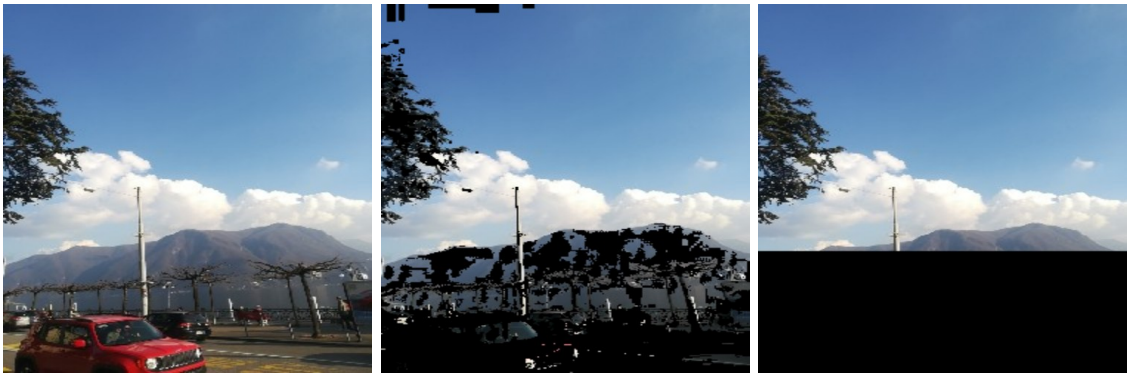
První technika, kterou jsem se rozhodl použít byl web scraping. V jazyce Python 3 jsem implementoval jednoduchý skript, který do vyhledávače google.com zadal název oblaku jako klíčové slovo a poté stáhl maximální počet výsledků. Tato technika se zdála být celkem úspěšná, ale po detailnější analýze stažených fotografií jsem zjistil, že jsem stáhl mnoho duplikátů, mnoho fotografií s vodoznakem nebo s jinak narušenou fotografií. V mnoha

případech oblak na fotografii neodpovídal klíčovému slovu, podle kterého byl vyhledán. Po manuální filtraci těchto chybných fotografií mi zbylo průměrně asi jen 150 fotografií ke každému typu oblak (ne tedy ke každé třídě v datasetu). I přesto se tento dataset ukázal jako užitečný při učení neuronové sítě k rozeznání fotografie oblohy oproti fotografie čehokoliv jiného, kde dosahovala asi 92% úspěšnosti.

### 4.3.2 Samostatné pořizování fotografií

Neuronovou síť naučenou na rozeznaní fotografie oblohy oproti čemukoliv jinému jsem se rozhodl vyžít na prototyp mé cílové aplikace, který by mi zároveň usnadnil vytvořit můj vlastní dataset. Prototyp aplikace poskytoval uživateli prostředí velmi podobné typickému fotoaparátu, kde se po vyfocení fotografie fotka zanalyzovala, vypsala uživateli s jakou pravděpodobností je objekt na fotografii obloha a poté se zaslala do cloudového úložiště s příznakem, zdali to je obloha či jiný objekt. Každých několik dní jsem pak tyto fotografie manuálně stáhl a anotoval.

Při anotaci jsem narazil na problém, který by pravděpodobně velmi ztížil učení neuronové sítě. Byly to fotografie, na kterých byl mimo oblohy i jiný objekt, většinou nějaká budova nebo strom. Na základě rady od Ing. Goldmanna jsem se pokusil vytvořit jednoduchý skript, který by tyto obrázky detekoval a ořízl je. Hlavní myšlenka spočívala v převedení obrázku do HSV formátu, u kterého bych vyfiltroval jen barvy vyskytující se na obloze.



(a) Originální fotografie (b) Po vyfiltrování pixelů (c) Po definování oblasti k ořezu

Obrázek 4.2: Ořezávání fotografií s rušivými objekty na fotografii

První krok spočíval v průchodu přes všechny pixely a odstranění těch, které nespadaly do definovaného spektra barev. Výsledek můžeme vidět na obrázku 4.2b. Dále bylo potřeba definování oblasti ořezu, to jsem uskutečnil průchodem obrázku ve sloupcích a řádcích ze všech stran. Pokud v daném řádku nebo sloupci bylo více jak polovina pixelů vyřazených, rozhodl jsem se tento řádek nebo sloupec vymazat celý, vzniká tak obrázek 4.2c.

Při testování tohoto prototypu jsem zjistil, že velmi často musím měnit rozsah přípustných barev. Nepodařil se mi najít takový, který by mohl být použit u valné většiny fotografií. Často také tento algoritmus, právě díky nevhodnému rozsahu barev, kazil i dobře pořízené fotografie. Rozhodl jsem se proto tento prototyp zanechat a přizpůsobil jsem aplikaci tak, aby umožňovala uživateli vytvořit dobrou fotografii pouze oblohy - toho jsem docílil pomocí implementace přibližování.

Mimo to se mi tato technika zpočátku zdála velmi úspěšná, fotografie byly dobré kvality a díky manuální anotaci i kvalitně rozdělené do tříd. Po několika prvních pokusech o naučení

klasifikátoru oblak, jsem zjistil, že budu potřebovat alespoň tisíc fotografií ke každé kategorii a na to byl tento způsob žalostně pomalý. Během dvou měsíců se mi podařilo vytvořit celkově asi 2500 fotografií, což bylo málo a se zbývajícím časem bych nebyl schopný udělat dostatek dalších snímků.

### 4.3.3 Google Street View API

Společnost Google v aktuální době provozuje službu Google Street View[1], která uživateli poskytuje možnost si prohlédnout velké množství míst na mapě v podobě 360° panoramatických snímků. Tato služba se nejčastěji využívá přes prohlížeč pro virtuální procházku míst. Google také poskytuje API[2] pro získání těch snímků za poměrně rozumnou cenu a ze začátku zcela zdarma.

Pro získání jedné fotografie potřebujeme několik údajů: API klíč (získán při registraci), souřadnice odkud fotografii chceme, vertikální a horizontální úhel fotoaparátu. Po několika testech jsem nastavil pevné úhly a posledním problémem byly souřadnice. Pro efektivnější anotaci jsem chtěl, aby na sebe měly po sobě jdoucí fotografie návaznost, nemohl jsem tedy volit náhodné souřadnice. Po kratším zamyšlení jsem se rozhodl v Google mapách vytvořit trasu, kterou jsem konvertoval do formátu GPX pomocí volně dostupného online nástroje mapstogpx.com [20] vytvořeného Sverrirem Sigmundarsonem. GPX [10] (GPS Exchange Format) je formát používaný k serializaci tras nebo bodů globálního pozičního systému - GPS. Jedná se o XML soubor obsahující pro můj účel nejpodstatnější elementy - jednotlivé body trasy.

Pomocí těchto nástrojů a krátkého skriptu opět implementovaného v jazyce Python 3 jsem stáhl několik desítek tisíc fotek z tras zejména mezi velkými městy, protože tam většinou vedly dálnice, na kterých nic nestálo ve výhledu na oblohu. Po jejich stáhnutí jsem ještě musel uříznout spodní část s Google logem, aby nepoškozovalo fotografii oblohy. To jsem udělal pomocí volně dostupného nástroje pro příkazovou řádku ImageMagick [16].

Některé fotografie bylo velmi těžké anotovat díky jejich špatné kvalitě způsobené např. špatným úhlem světla. Vzhledem k obrovskému množství dostupných fotografií jsem je mohl přeskočit a použít pouze ty v dobré kvalitě.

[1] https://www.google.com/streetview/

[2] https://developers.google.com/maps/documentation/streetview/intro

Obrázek 4.3: Fotografie oblaku typu Cirrus, získaná pomocí Google Street View API.

Takto získané fotografie jsem opět manuálně anotoval a staly se tak finálním datasetem použitým pro učení klasifikačních sítí.

## 4.4 Tvorba a učení neuronových sítí

Konvoluční neuronové sítě budou logickým jádrem výsledné aplikace. I přes to, že jejich výsledná podoba je v podstatě jednoduchá - jen několik matic čísel, jejich tvorba a zajištění jejich funkčnosti je netriviální proces. Pro zjednodušení tvorby potřebných neuronových sítí budu využívat jazyk Python 3, framework Tensorflow a na Tensorflow postavený framework Keras. Naučené modely pak uložím, transformuji je do přenositelného formátu a použiji ve výsledné aplikaci.

V naší aplikaci budeme potřebovat dvě neuronové sítě:

- Rozeznávání jakkoliv vypadající oblohy k zajištění větší spolehlivost funkčnosti aplikace a navíc odlehčením dalším neuronovým sítím se vstupy, které nedávají smysl. Účelem této neuronové sítě tedy bude vyfiltrování všech fotografií, na kterých není pořádná fotografie oblohy zabírající většinu plochy (asi 90 %).
- Klasifikace oblak do kategorii dle tříd datasetu - konečná neuronová síť použitá pro předpověď počasí.

Další možnost je vytvořit síť pro každou klasifikační třídu. To bude velmi výhodné, protože v mnoha případech se na obloze nevyskytuje pouze jeden oblak. Tento návrh nám dává možnost tento případ detekovat a reagovat na tuto situaci jinak, s větším detailem. Vzhledem k velké složitosti anotace fotografií tuto možnost vynechám.

### 4.4.1 Rozeznání fotografie oblohy

U architektury této sítě jsem postupoval experimentálně. Začal jsem s typickým návrhem [15], kde se na začátku vyskytuje několik konvolučních vrstev, po kterých vždy následuje max-pooling a po nich následuje plně propojená vrstva s výstupními uzly.

Vstup této sítě jsem nastavil na tenzor o velikosti $200 \times 200 \times 3$, což bylo experimentálně ověřeno jako velikost fotografie, na které jde ještě jasně rozeznat, jaký typ oblaka se na ní nachází. Výstup této sítě má pouze jeden uzel, který pokud se blíží jedničce, značí, že považuje vstup za oblohu, naopak pokud se blíží nule, považuje vstup za něco jiného.

#### První experiment

Jako vstup jsem použil první verzi datasetu - vyfiltrované obrázky získané pomocí web scrapingu. Zkusil jsem použít několik obdobných architektur, u kterých jsem měnil několik parametrů a vyzkoušel všechny permutace.

- Dimenze konvoluční sítě - počet konvolučních filtrů v jednotlivých vrstvách.
- Počet konvolučních vrstev.
- Počet plně propojených vrstev.

Knihovna Keras postavená na frameworku TensorFlow nám poskytuje velmi příjemný, vysokoúrovňový přístup k tvorbě architektury sítě. Na následujícím příkladu uvedu několik nejzásadnějších volání této knihovny, které jsem využil.

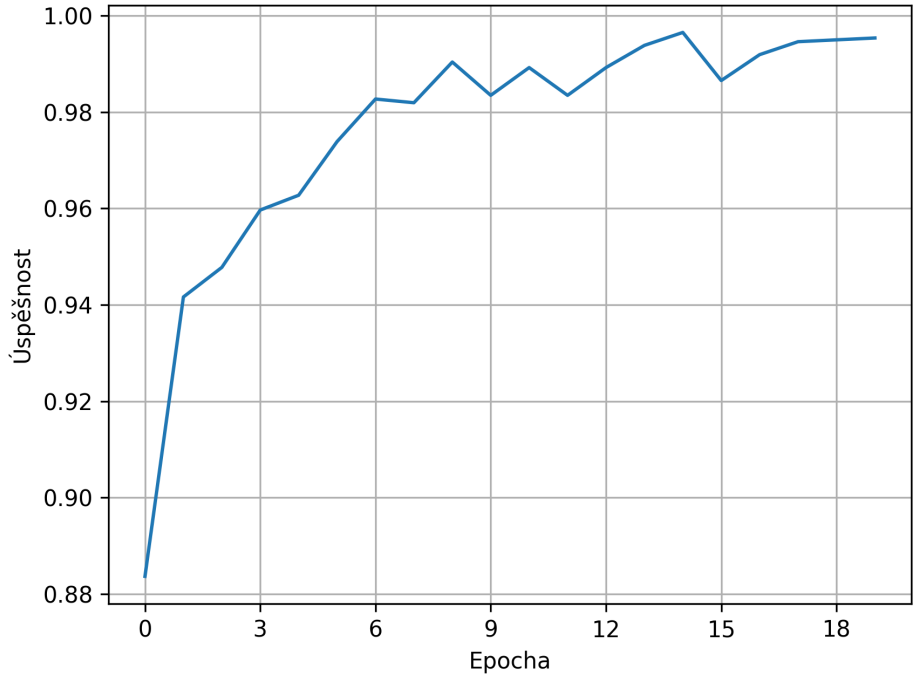
```
layer_size = 42
for _ in range(2):
    model.add(Conv2D(layer_size, (3, 3), input_shape=X.shape[1:]))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

for _ in range(dense_layer):
    model.add(Dense(layer_size))
    model.add(Activation('relu'))

model.add(Dense(1))
model.add(Activation('sigmoid'))
```

Výpis 4.1: Vytvoření sekvenčního modelu a nadefinování jeho vrstev pro první verzi neuronové sítě na rozeznávání fotografie oblak.



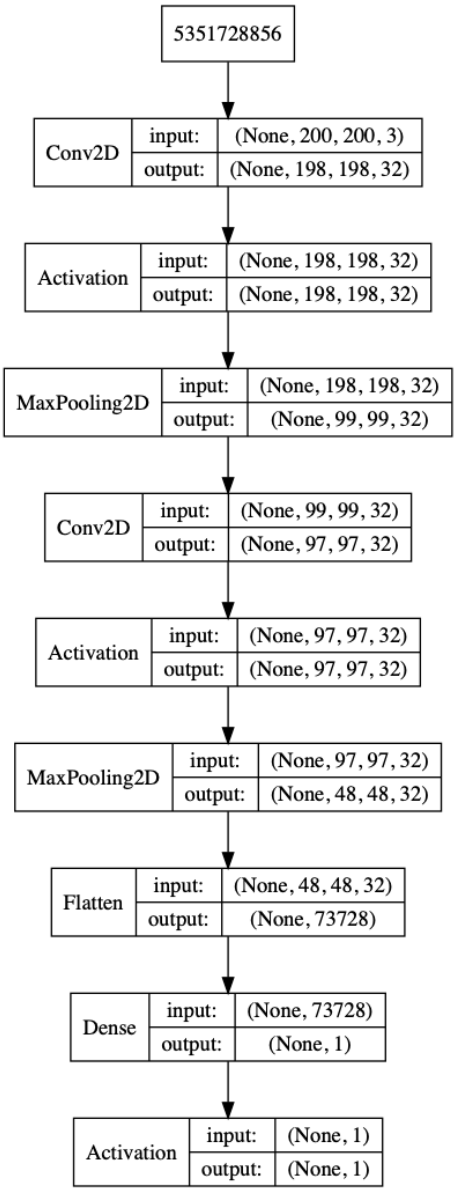

Obrázek 4.4: Graf úspěšnosti nejúspěšnější sítě pro rozpoznání fotografie oblohy během prvního pokusu.

Každý testovaný model byl sekvenční a měl podobnou strukturu. Jednu a více konvoluční síť společně s pooling vrstvou, poté variabilní počet plně propojených vrstev a nakonec výstupní plně propojenou vrstvu s jedním neuronem. Jak je popsáno výše, během experimentování jsem měnil počet konvolučních vrstev a jejich počet konvolučních filtrů.

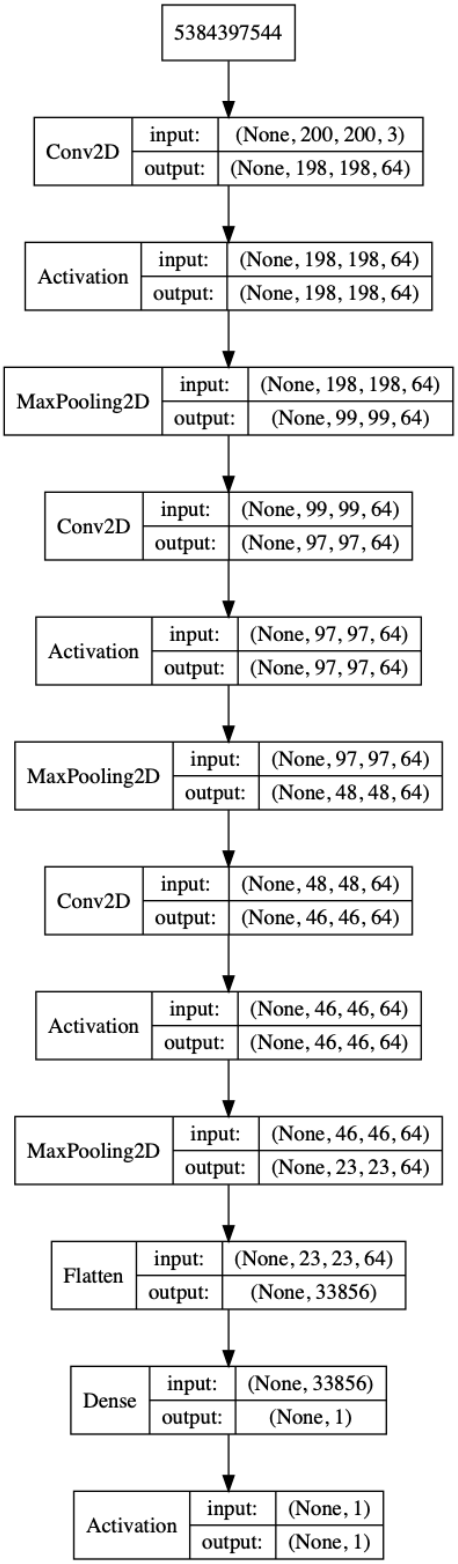
Po učení jsem analyzoval výsledné grafy a zjistil jsem, že všechny architektury sítě poskytovaly přibližně stejnou úspěšnost. Na testování jsem vybral z nich nejúspěšnější 4.4, která měla 3 konvoluční vrstvy a 32 konvolučních filtrů pro každou vrstvu. Na jejím grafu učení můžeme vidět, že se po asi devíti epochách úspěšnost ustaluje kolem hodnoty 0,99 a nadále se nezvyšuje. *Úspěšnost* epochy je průměrná hodnota úspěšnosti pro každou dávku (*batch*), kterou můžeme zadefinovat jako $\frac{\text{Počet správně ohodnocených fotografií v dávce}}{\text{Počet fotografií v jedné dávce}}$.

Během testování této sítě se ukázalo, že velice špatně rozeznává za umělého světla a vracela velké množství falešně pozitivních hodnot. Během přírodního světla rozeznávala poměrně dobře, ale musel jsem manuálně nastavit hranici výsledné hodnoty, při které považuji fotografii za fotografii oblak na 0,8. Očekával jsem, že tato hodnota bude 0,5, ale při této hodnotě síť zhodnotila mnoho fotografií falešně pozitivně.

Na obrázcích 4.5 můžeme vidět architekturu dvou testovaných sítí. Na levé straně vidíme síť s dvěma konvolučními vrstvami a jejich dimenze je 32. V knihovně Keras se mluví o aktivační funkci jednotlivých vrstev jako samostatné vrstvě, proto má v grafu samostatný uzel. Hned za každou konvoluční vrstvou následuje spojovací (*pooling*) vrstva 2.5.1 pro převzorkování jejího výstupu. Jako výstupní vrstva obou modelů je plně propojená vrstva s jedním výstupním neuronem, který díky jeho aktivační funkci sigmoid nabývá hodnot mezi jedničkou a nulou reprezentující pravděpodobnost výskytu oblohy na fotografii.




(a) Model neuronové sítě s dvěma konvolučními vrstvami.

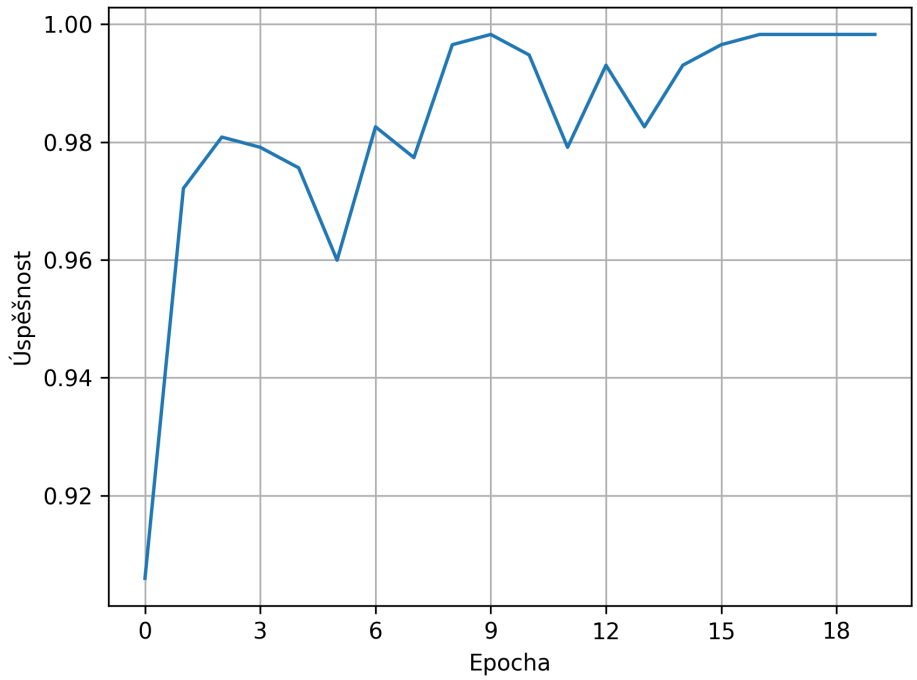



(b) Model neuronové sítě se třemi konvolučními vrstvami a jejich výstupní dimenzí 64.

Obrázek 4.5: Grafické porovnání architektury neuronových sítí.

**Druhý experiment**

Protože jsem s funkčností první verze nebyl spokojen, pokusil jsem se o naučení neuronové sítě ještě jednou s datasetem obohacený o nově pořízené, vlastní fotografie. Testoval jsem opět různou architekturu a k použití jsem opět vybral tu nejúspěšnější, i přes to, že jejích výsledná úspěšnost byla dosti podobná. Jednalo se o síť s dvěma konvolučními vrstvami a šířkou 32 uzlů.




Obrázek 4.6: Graf úspěšnosti nejúspěšnější sítě pro rozpoznání fotografie oblohy během druhého pokusu.

I přes to, že její hodnoty úspěšnosti nejsou příliš odlišné od prvního pokusu, při praktickém testování jsem si všiml, že síť rozhodovala s velkou jistotou - většinou výsledky blížící se nule nebo naopak jedničce. Mimo to se síť hůř nechala oklamat různými objekty, které oblohu jenom připomínaly. Toto chování dle mého názoru je následkem lepšího datasetu, který přímo odpovídal snímkům pořízených v aplikaci.

Funkčnost této verze sítě jsem uznal jako vhodnou pro daný účel a dále jsem ji neměnil.

### 4.4.2 Rozeznání jednotlivých tříd oblak

Cílová aplikace se pokusí o rozeznání jednotlivé třídy oblak až poté, co se rozhodne, že na fotografii opravdu jsou oblaka. Tím konečné neuronové síti pomůžeme a zjednodušíme její práci, stejně jako její učení.

### První experiment

Při prvním pokusu o vytvoření této sítě jsem použil stejnou strategii jako při rozeznávání fotografie oblohy. Zkoušel jsem tedy na stejném datasetu různé architektury sítě.

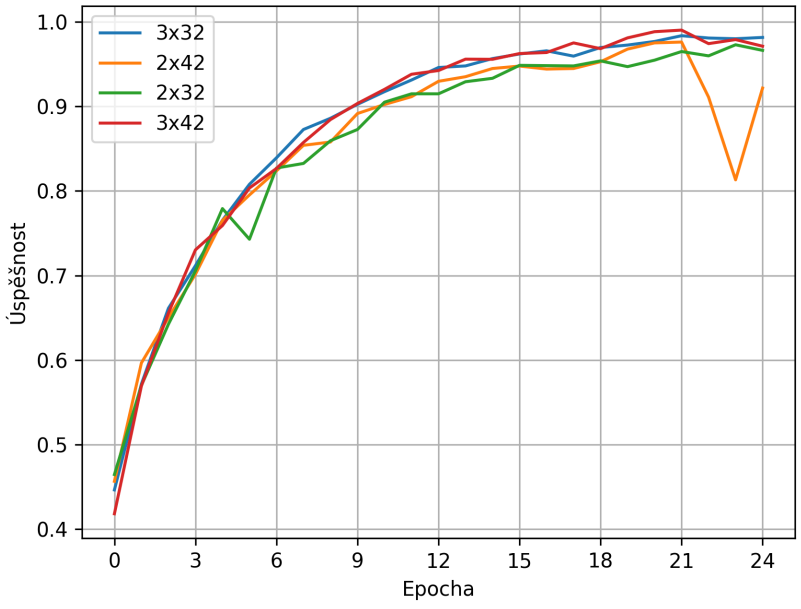
Neuronová síť se se svou úspěšností ve většině případech dostala jen málo nad hodnotu danou plně náhodným chováním. Pravděpodobně se naučila rozeznávat pouze některé nejjednodušší případy - pravděpodobně čistou oblohu.

Tento neúspěšný výsledek jsem dal za vinu datasetu. I jednotlivé obrázky ve stejné kategorii byly často velmi odlišné díky tomu, že byly vyfoceny jiným fotoaparátem, za jiných podmínek a podobně. Mimo to mohly být i upraveny, aby vypadaly lépe. Mezi těmito odlišnými vzorky síť při učení nenašla žádnou nebo téměř žádnou spojitost a její výsledná funkce byla tedy spíše náhodná. Dalším důvodem neúspěchu mohl být i velmi malý počet vzorků.

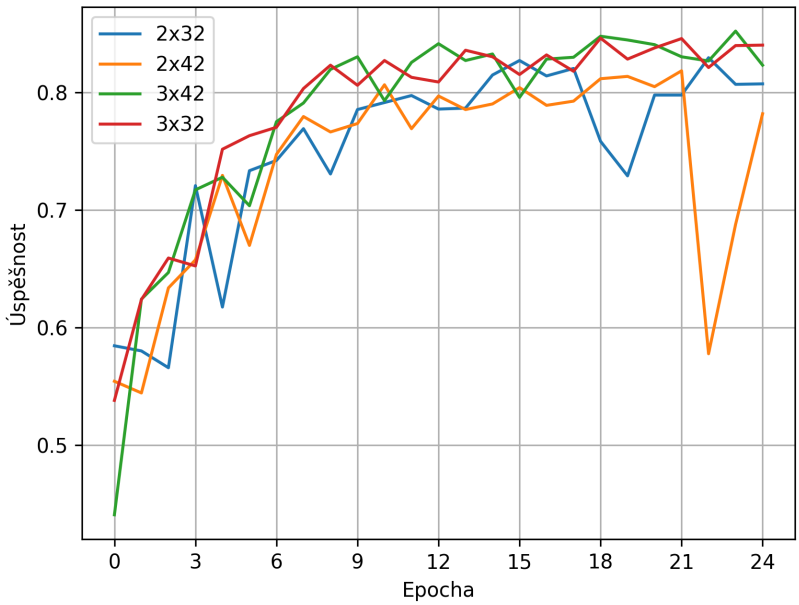
Vzhledem k nízké úspěšnosti jsem tuto síť dále neprozkoumával a rozhodl se upravit dataset.

### Druhý experiment

Při druhém pokusu o naučení neuronové sítě jsem opět vyzkoušel několik rozdílných architektur, u kterých jsem měnil parametry. Tentokrát jsem měl na vstupu vlastní dataset obohacený o snímky z Google Street View. Počet snímků byl nevyrovnaný, u některých tříd dosahoval 2000 a u některých měl pouhých 500 snímků. Aby byl dataset vyrovnaný při trénování, rozhodl jsem se nasadit pevný počet na 1000, což znamenalo, že u objemnějších kategorií se část snímků nepoužila a u ostatních se jeden snímek použil maximálně dvakrát.




Obrázek 4.7: Graf úspěšnosti sítí s různými architekturami, které jsou vypsány v pravém horním rohu ve formátu počet konvolučních vrstev X jejich dimenze (počet konvolučních filtrů).




Obrázek 4.8: Graf validační úspěšnosti jednotlivých sítí.

Na obrázku 4.7 vidíme přehled úspěšnosti učení neuronových sítí s různou architekturou. Můžeme si všimnout, že úspěšnost sítí roste poměrně rychle až do asi patnácté epochy, po které už zůstává na podobné hodnotě. Můžeme tedy o těchto modelech říct, že jejich váhy konvergují.

Dále můžeme vidět nenadálý pokles úspěšnosti u sítě s dvěma konvolučními sítěmi a dimenzí 42. Může tomu být díky malé velikosti dávky (anglicky *batch*). Vzhledem k tomu, že se toto chování vyskytlo pouze u jednoho modelu, nadále jsem to neřešil.

Poslední významné věci, které si můžeme všimnout je, že výsledky pro všechny modely jsou přibližně stejné. Tomu tak ale není v případě validační úspěšnost viz. obrázek 4.7. Na tomto grafu si můžeme všimnout, že vrstvy se třemi konvolučními vrstvami měli průměrně vyšší úspěšnost asi o 4 %, a proto jsem se rozhodl použít tuto síť.

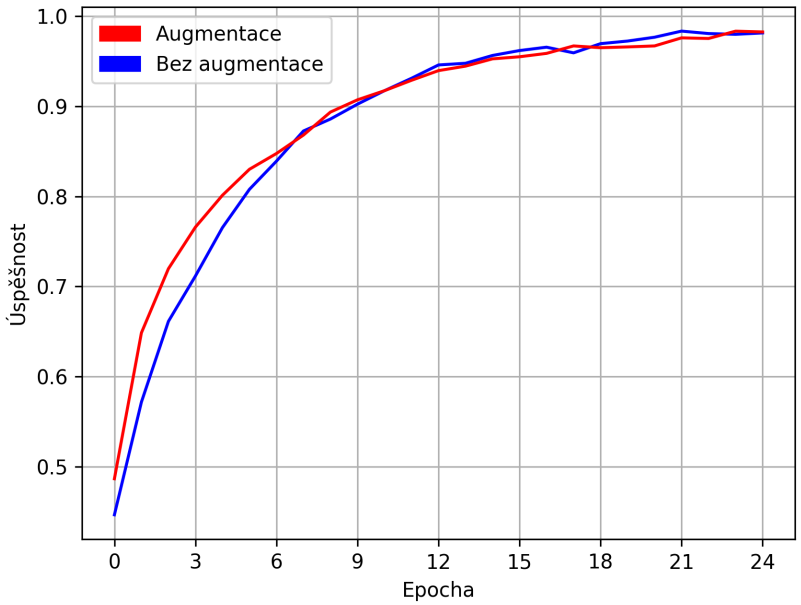
Mimo to si na grafu validační úspěšnosti můžeme všimnout, že se hodnoty ustalují po už asi devíti epochách. Vzhledem k tomu, že úspěšnost se ustaluje až po patnácti epochách, může tu nastávat lehké přeučení (*overfitting*).

### Třetí experiment

V tomto pokusu jsem experimentoval s navýšením dimenze konvolučních vrstev a epoch. Mimo to jsem se rozhodl vyzkoušet *augmentaci* (rozšíření) mého datasetu pomocí rotace fotografie o 180 stupňů. Nakonec jsem vyzkoušel dát na vstup neuronové sítě snímky převzorkované na čtverec o straně 300 pixelů (v minulých případech se snímky z datasetu převzorkovaly na čtverec o straně 200 pixelů).

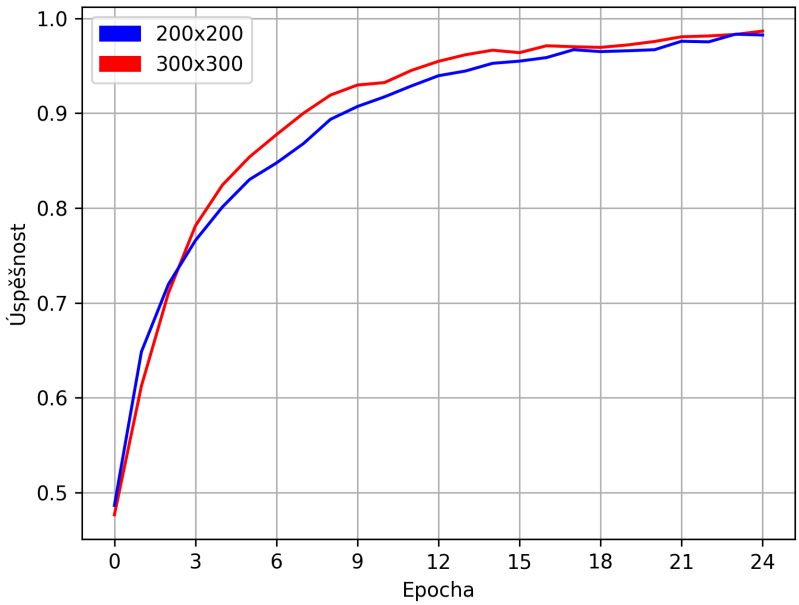
Při zapnuté augmentaci se počet snímků pro epochu zdvojnásobil, neb obsahoval vždy originál snímku a poté jeho otočenou kopii. Na obrázku 4.9 si můžeme všimnout, že v prvních šesti epochách je síť s povolenou augmentací výrazně úspěšnější, než síť učené na datasetu bez augmentace. Domnívám se, že to je právě kvůli většímu množství snímků, na

kterých byly trénovány během jedné epochy. Vzhledem k tomu, že se v dalších epochách dostáváme k ideálnímu stavu sítě - naše síť tedy konverguje, její úspěšnost už se nezvedá a po 25 epochách končí sítě ve velmi podobných hodnotách.




Obrázek 4.9: Porovnání úspěšnosti při jednoduché augmentaci.

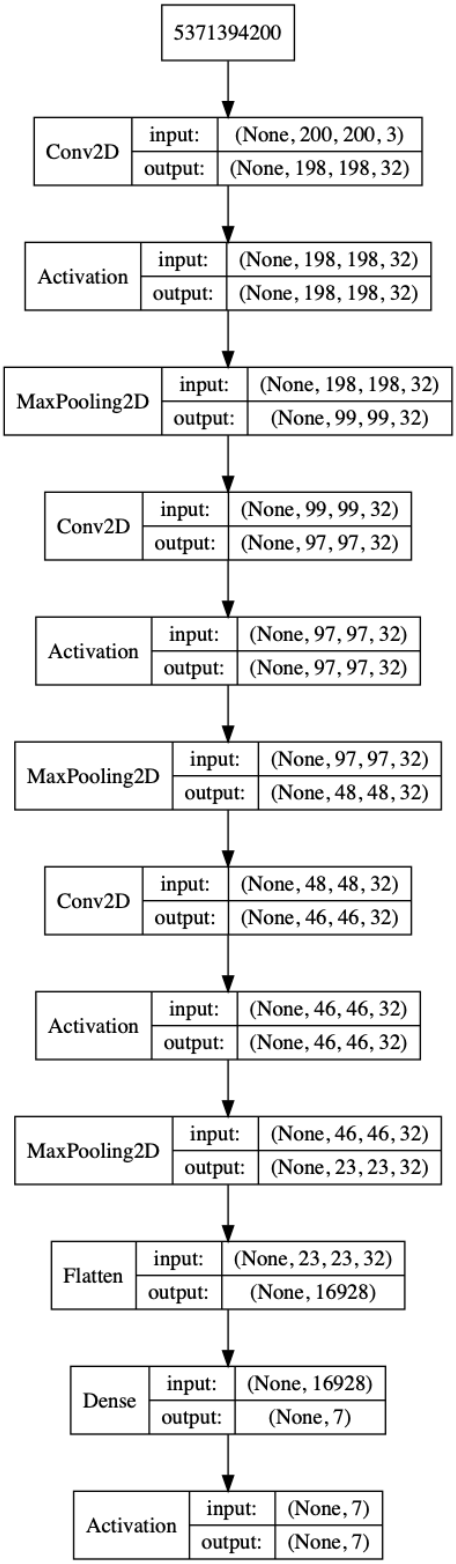
Pouze velmi malá změna nastala i při změně vstupního rozměru vzorku 4.10, opět ale jen v několika epochách a ve výsledku obě sítě konvergovaly ke stejným hodnotám asi 0,97. Na druhou stranu se tímto zvětšením vstupní vrstvy a tím i zvětšením počtu konvolucí výrazně zvětšila náročnost učení i velikost sítě. Rozhodl jsem se tedy zůstat u původního rozlišení $200 \times 200$ pixelů.




Obrázek 4.10: Porovnání úspěšnosti při změně velikosti vstupního snímku. Červeně jsou vyznačeny grafy úspěšnosti sítí, kde byl vstup čtverec o straně 300 px, modře pak značí sítě se vstupem o velikosti 200 px.

Po zhodnocení výsledků a porovnání úspěšnosti s přihlédnutím k validační úspěšnosti jsem se rozhodl pro svoji finální aplikaci použít síť naučenou ve dvaceti pěti epochách na augmentovaném datasetu se třemi vrstvami konvolučních vrstev o šířce 32 uzlů, která dosahovala přibližně 97% úspěšnosti a 83% validační úspěšnosti.

Její architekturu můžeme vidět na grafu 4.11. Za povšimnutí stojí poslední vrstva se šířkou 7 (jeden neuron pro každou kategorii). V této vrstvě jsem musel použít aktivační funkci 2.1.3 *softmax* [31], která nám za výsledek dá pravděpodobnost každé kategorie. Zprvu jsem používal funkci sigmoid, ale ta po několika epochách způsobovala nahodilé chování neuronové sítě.




Obrázek 4.11: Architektura finálního modelu využitého v aplikaci.

## 4.5 Aplikace pro Android

V této sekci stručně popíši tvorbu této aplikace. Při její tvorbě jsem se nesnažil rovnou vytvořit cílový produkt, ale postupoval jsem s využitím modifikované metodiky prototypování. V každé iteraci jsem nastavil určitou nedokonalou verzi aplikace, tu jsem otestoval, získal zpětnou vazbu od testovacích uživatelů, na jejímž základě postavil plán pro další iteraci.

### 4.5.1 Grafický návrh

Rozhodl jsem, že se aplikace bude skládat z pouze dvou různých obrazovek - aktivit. První bude sloužit pro pořízení fotografie, druhá pak pro její zobrazení s její analýzou.

Důležitým aspektem hlavní aktivity je živý náhled fotografie. Sice existují jednodušší způsoby jak pořídit fotografii, např. využitím vestavěné aplikace fotoaparátu, ale vzhledem k tomu, že by uživatel musel přepínat mezi dvěma aplikacemi, bylo by to špatné řešení z pohledu jednoduchosti a rychlosti použití aplikace, a proto jsem toto řešení zavrhl.


(a) Hlavní aktivita aplikace - fotoaparát.

(b) Aktivita pro analýzu a představení jejích výsledků uživateli.

Obrázek 4.12: Grafický návrh aplikace.

Náhled fotografie 4.12a bude zabírat celý rozsah obrazovky a ve spodní části bude tlačítko na vytvoření fotografie. Tento nápad jsem převzal z typické implementace aplikace fotoaparátu na mobilních zařízeních, a proto by měl být pro hrubou většinu uživatelů intuitivní.

Pod zanalyzovanou fotografií se vypíše rozeznaný typ oblak a na jeho základě předpověď počasí 4.12b. Typ oblak by se mohl zdát zbytečný, ale bude dobrý pro verifikaci funkce neuronové sítě pro rozeznání typu oblak. Mimo to se může uživatel přiučit názvům různých typů oblak.

### 4.5.2 Implementační návrh

Vytvořil jsem velmi jednoduchý návrh aplikace, který bude pokrývat veškerou potřebnou funkcionalitu. Aplikace se bude skládat z celkově tří aktivit:

- Hlavní aktivita náhledu a pořízení fotografie,
- aktivita pro analýzu fotografie,
- vedlejší aktivita pro nastavení.

V hlavní aktivitě budu využívat knihovnu *android.hardware.camera2* [8], pomocí které budu promítat aktuální pohled fotoaparátu na objekt *TextureView*. Dále bude na hlavní aktivitě tlačítko pro pořízení fotografie, které uživatele přesune do aktivity pro analýzu fotografie. Pokud nebude zapnutý rychlý mód (rychlé odeslání fotografie na *cloudové* úložiště včetně analýzy), fotografie se vytvoří jednoduchým uložením aktuálního snímku promítaného na objekt *TextureView* a cesta k fotografii se předá jako argument pro následující aktivitu. Pokud bude zapnutý rychlý mód, fotografie se pouze zanalyzuje a na pozadí pošle na *cloudové* úložiště.

V druhém kroku se obrázek zanalyzuje pomocí naučené neuronové sítě a pomocí knihovny TensorFlow přizpůsobené pro android. Výsledek analýzy se vypíše pod fotografii. Posledním elementem na této aktivitě bude tlačítko *Back*, které uživatele přesune zpět do hlavní aktivity. Mimo to se obrázek po zpracování zašle společně s výsledky analýzy na *cloudové* úložiště pro obohacování datasetu.

Pro různé nastavení bude ještě existovat třetí aktivita. Implementace této aktivity bude velmi jednoduchá, protože bude dědit od třídy *PreferenceActivity*, která nám poskytne jednoduché rozhraní pro ukládání nebo načítání konfigurace přetrvávající i po vypnutí a zapnutí aplikace. Pro jednotlivé nastavení pak budeme přistupovat pomocí metody *PreferenceManager.getDefaultSharedPreferences* Uživatelské rozhrané této aktivity se bude skládat jen z jednoduchých přepínačů mezi stavy zapnuto a vypnuto.

### 4.5.3 Implementace aplikace

Svým prvním prototypem jsem se snažil udělat tři základní věci: připravit si aplikaci, do které bude jednoduché zakomponovat výsledné neuronové sítě, usnadnit si tvoření mé druhé verze datasetu a otestovat funkčnost grafického návrhu.

#### Náhled fotografie

Pro živý náhled fotografie a samotné zachycení snímku jsem použil aktuální knihovnu pro práci s fotoaparátem *android.hardware.camera2* [8]. A při psaní kódu jsem se inspiroval ukázkou použití této knihovny [12].

Vytvoření živého náhledu s pomocí *android.hardware.camera2* se skládá z několika kroků:

- vytvoření a připravení *TextureView* objektu, sloužící jako plátno, na které se promítá obraz,
- získání *CameraManager* objektu, který použijeme pro přístup k dostupné konfiguraci kamery,

- vybrání kamery, protože většina mobilních zařízení má většinou více kamer, v tomto prototypu jsem zvolil první dostupnou,
- zavolání *CameraManager.openCamera* metody s připravenou funkcí na zpětné volání - použije se pro informování o změně stavu vyžadované kamery,
- při úspěšném otevření kamery vytvoříme žádost o snímek pomocí volání metody *CameraDevice.createCaptureRequest* a nastavíme mu jako jeho cíl naši připravenou texturu.

### Analýza fotografie

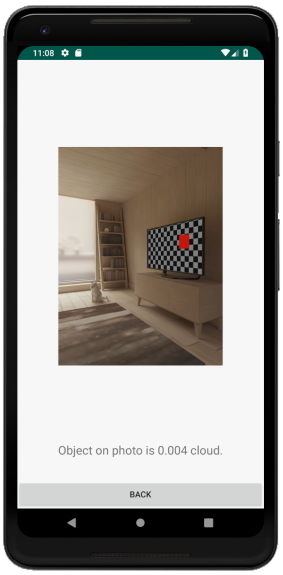
Dalším důležitým implementačním úkolem bylo využití naučené neuronové sítě vytvořené v Pythonu pomocí knihovny nad knihovnou TensorFlow zvanou Keras. Protože aktuálně neexistuje žádná volně dostupná knihovna, která by nám umožnila použít takto vytvořený model, musel jsem model převést do tvaru, který by mohla využít existující knihovna pro TensorFlow. Využil jsem pro to volně dostupný skript od Amir Abdiho [25].

S takto připravenou neuronovou sítí jsem mohl využít knihovnu TensorFlow a třídu poskytující operace s naučeným modelem - *TensorFlowInferenceInterface*. Tato třída poskytuje příliš široké možnosti, její použití je pro můj jednoduchý účel poměrně složité, a proto jsem si vytvořil třídu *TensorflowImageClassifier*, podle návrhové vzoru adaptér, která toto rozhraní zjednodušuje. Tato třída implementuje rozhraní *ImageClassifier*:

```
public interface ImageClassifier {
    float[] classifyImage(String imagePath);
    float[] classifyImage(Resources resources, int imgId);
}
```

Výpis 4.2: Jednoduché rozhraní poskytující jednu přetíženou metodu classifyImage. Tato metoda na vstupu přijímá fotografii buďto jako soubor specifikovaný jeho cestou, nebo jako jeho identifikační číslo ve zdrojích. Tato metoda navrací pole čísel typu *float*, což jsou hodnoty výstupní vrstvy neuronů.


(a) Hlavní aktivita - fotoaparát.




(b) Aktivita analýzy fotografie.

Obrázek 4.13: Aplikace v konečné fázi prvního prototypu.

**Výsledky testování**

Po samostatném testování a opravení několik menších chyb jsem tuto aplikaci rozšířil mezi mé známé a požádal je o otestování a zpětnou vazbu. Výsledkem bylo několik zásadních problémů:

- Pevně daný rozměr náhledu fotografie - tato vlastnost *TextureView* objektu zapříčinila nezamýšlené roztahování náhledu, to se projevovalo zejména na zařízeních s velmi vysokým displayem jako např. Pixel 2 XL.
- Spuštěná aplikace na pozadí byla velmi náročná a v konečném důsledku rychle vyprázdnila baterii daného zařízení.
- Aktivita analýzy fotografie byla vzhledově nepřívětivá.
- Funkce přepínaní do druhé aktivity zpomalovala vytváření datasetu, uživatel pak ztrácel čas vracením se zpět do aktivity náhledu fotografie.
- Fotografie se musely často před přidáním do datasetu ořezat, kvůli nechtěným objektům
- Rozeznávání oblohy nefungovalo dle očekávání, pokud na fotografii byla obloha, výstupní hodnota se sice blížila číslu jedna (100 %), ale často i naprosto náhodné jiné objekty vykazovaly vysokých hodnot. Po otestování jsem rozhodl, že za fotografii oblak nebo oblohy se bude považovat fotografie s výslednou hodnotou nad 0,8.

Tyto problémy jsem vyřešil v druhém prototypu.

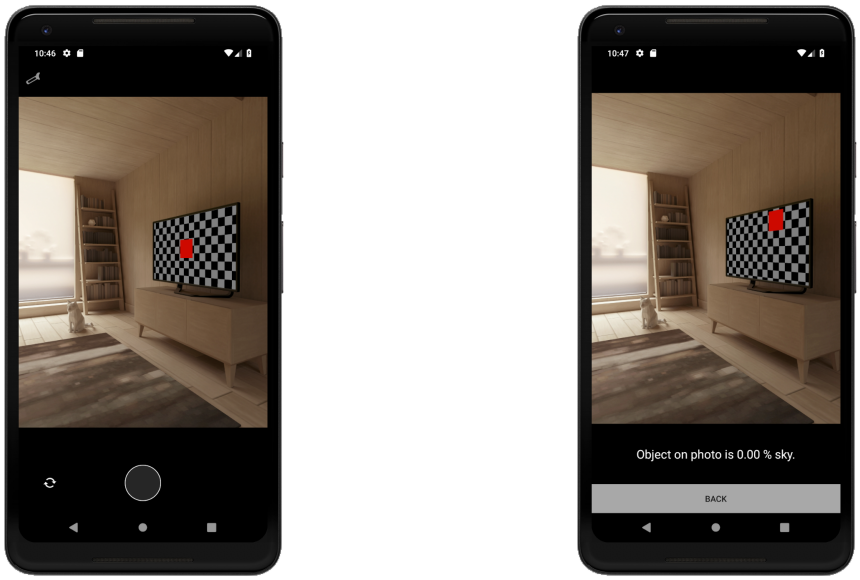
### 4.5.4 Upravená aplikace

Mimo opravení chyb prvního prototypu jsem se rozhodl udělat aplikaci vhodnější pro rychlé tvoření datasetu. Všiml jsem si, že zejména při chůzi by bylo výhodnější na fotografie použít

fotoaparát na přední straně telefonu. Mimo to jsem musel vytvořit přepínač, pomocí kterého by si uživatel zvolil, jestli chce tvořit fotografie rychle - tedy jestli se nechce přepínat do aktivity analýzy.

Přepínač na výběr fotoaparátu jsem vložil do hlavní aktivity vedle tlačítka pro pořízení fotografie, stejně jako to bývá u vestavěné aplikace.

Pro přepínač na změnu režimu do rychlého jsem vytvořil celou novou aktivitu nastavení, která by se mohla hodit v budoucích verzích aplikace. Jelikož se fotografie analyzovala i v tomto případě, abych mohl správně nastavit příznak před odesláním na cloudové úložiště, rozhodl jsem se uživatele informovat o výsledku analýzy prostřednictvím třídy *Toast* - vytvoří se malý informační text, který po pár vteřinách zmizí.




(a) Hlavní aktivita - fotoaparát.

(b) Aktivita analýzy fotografie.

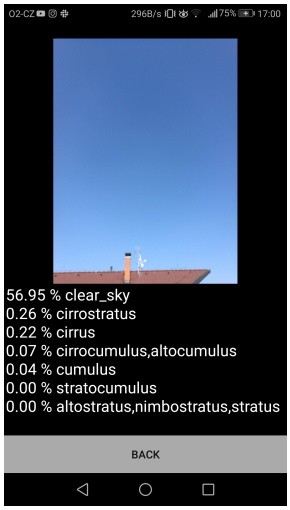
Obrázek 4.14: Aplikace v konečné fázi druhého prototypu.

Na obrázku 4.14a si můžeme všimnout několika drobných změn, kterými jsem se pokusil přiblížit vzhledu typické aplikace fotoaparát - například změnou barvou nebo úpravou vzhledu hlavního tlačítka. V levém vrchním rohu můžeme vidět ikonu nastavení.

I v aktivitě analýzy fotografie 4.14b jsem změnil barvu, aby byla aktivita vzhledově přívětivější. Mimo to, díky přeučené neuronové síti byly výsledky mnohem přesnější. Většinou více než 98 % pro fotografii oblohy nebo jinak méně než 1 % v případě fotografie jiného objektu.

### 4.5.5 Aplikace s přidaným klasifikátorem oblak

V této verzi aplikace jsem vyladil několik dalších méně významných chyb a zejména jsem zakomponoval první funkční verzi finálního klasifikátorů oblak. Z testovacích důvodů jsem namísto nastávajícího počasí vypisoval výstup z neuronové sítě, tedy typy oblak a jejich výskyt na fotografii.




(a) Zanalyzová fotografie čisté oblohy.


(b) Zanalyzová fotografie ne-oblaku.

Obrázek 4.15: Snímky obrazovky z testování třetího prototypu aplikace.

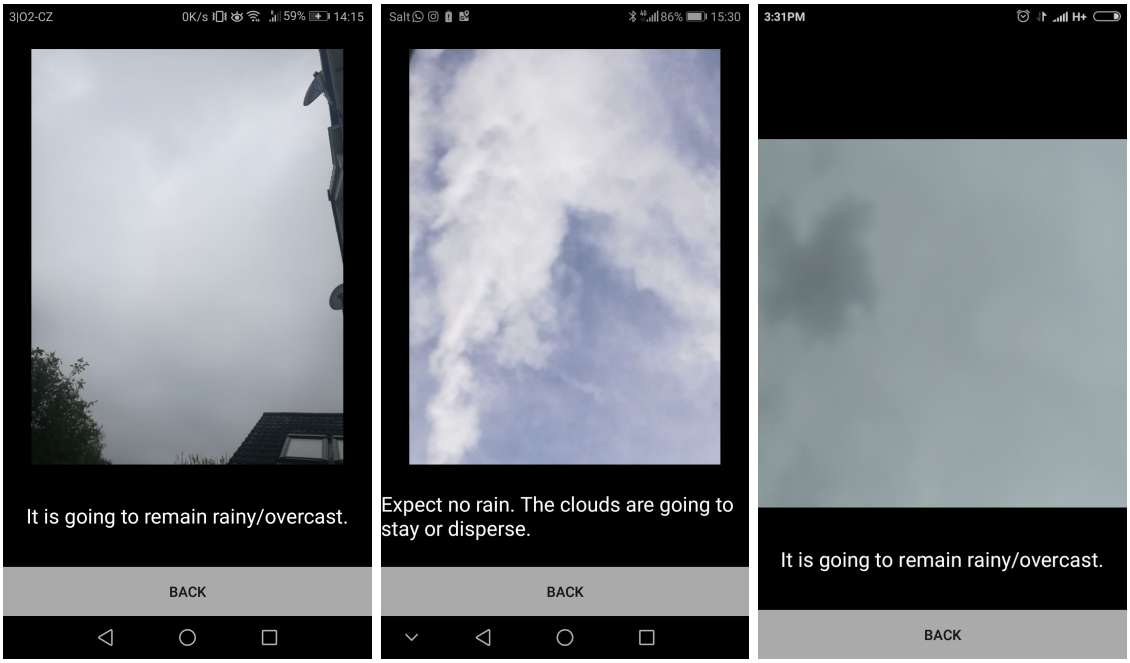
Při testování jsem zjistil, že síť nerozhoduje příliš s velkou pravděpodobností, většinou rozhodla správně, které kategorie oblak na fotografii nejsou, ale ty, které se tam vyskytovaly, neměly hodnoty blížící se jedničce, jak jsem očekával. Podobné chování jsem pozoroval i při rozeznávání fotografie oblohy v prvním prototypu. Vzhledem k tomu, že se při předpovědi počasí bude vždy vybírat kategorie s maximální hodnotou, bez ohledu na to o kolik je větší oproti ostatním, rozhodl jsem se tento problém ignorovat.

### 4.5.6 Finální verze

V této finální verzi jsem zakomponoval konečnou funkcionalitu - předpověď počasí. Také přidal možnost v nastavení zapnout ladící mód, který umožňuje uživateli vidět výpis všech kategorií a jejich výslednou hodnotu stejně jako v předchozí verzi 4.15a.

Mimo to jsem i změnil neuronovou síť ke klasifikaci oblak, jejíž architektura a učení je popsáno v sekci 4.4.2. Během uživatelského testování jsem nezpozoroval žádnou větší změnu proti předchozí verzi. Myslím, že pro stejnou změnu, která nastala u poslední verze sítě rozeznávání oblohy, by bylo třeba zkvalitnění datasetu.

Funkčnost této verze jsem otestoval na více zařízeních, testoval jsem rozložení aktivity analýzy fotografie při různých rozlišeních jak displaye, tak fotografie 4.16. Všechny byly přijatelné.




(a) Huawei P9 (b) Huawei P10 Lite (c) Xiaomi Redmi Note 4

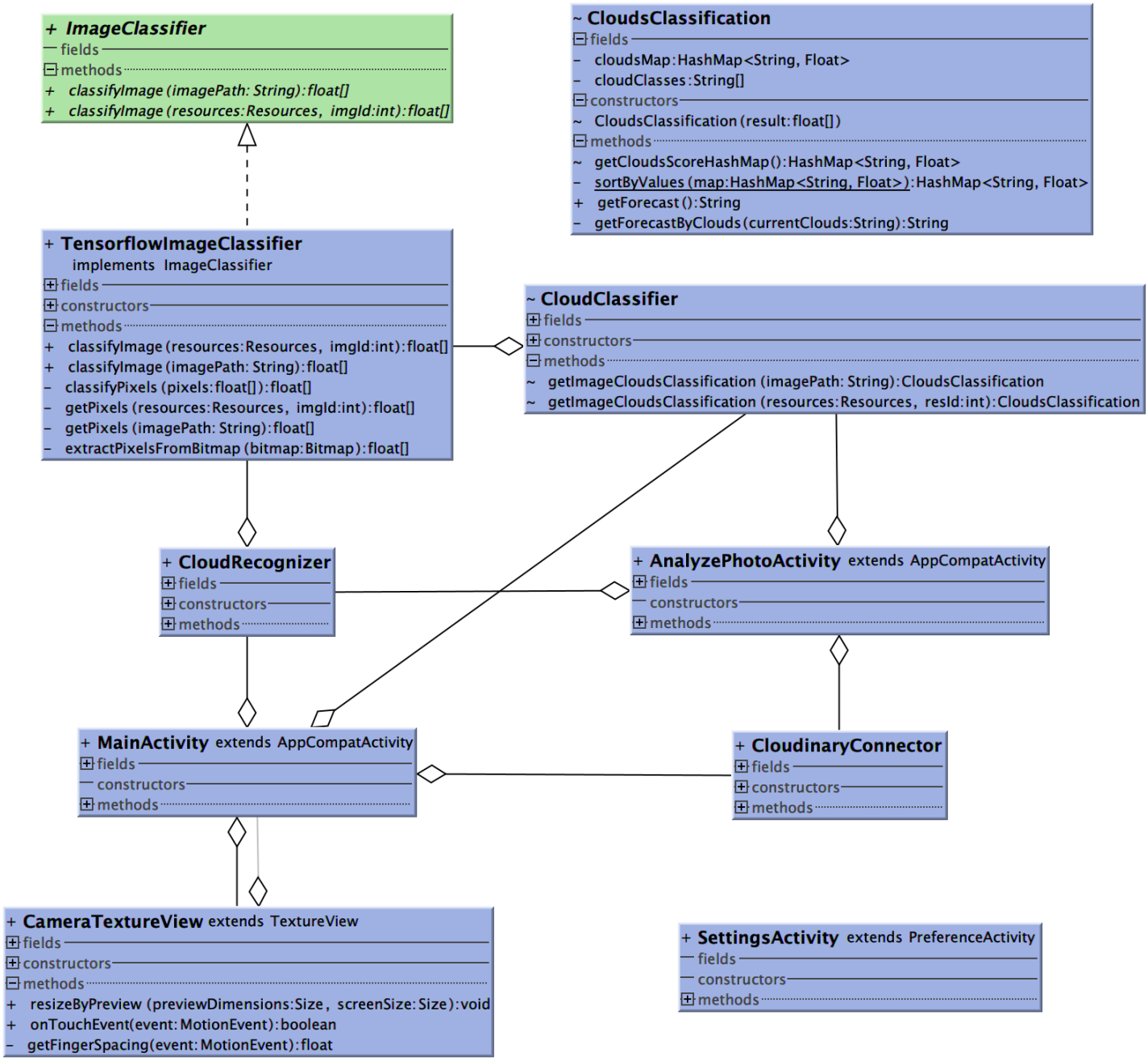
Obrázek 4.16: Snímky obrazovky aktivity analýzy fotografie finální verze na různých zařízeních.

### 4.5.7 Diagram tříd

Na obrázku 4.17 můžeme vidět implementovaný diagram tříd. V horní části vidíme rozhraní *ImageClassifier*, které je implementováno ve třídě *TensorflowImageClassifier*, což je tzv. *wrapper* neboli adaptér, obalující třídu *TensorFlowInferenceInterface*. Tato samotná třída z knihovny *tensorflow-android* má příliš složité rozhraní a v tomto adaptéru jsem její funkcionalitu zjednodušil na jediný účel, který budu potřebovat - klasifikace.

Logicky jsem tuto třídu opět využil při vytvoření adaptéru pro detektor oblak a jejich klasifikaci. Vznikají nám tedy dva další adaptéry: *CloudRecognizer* a *CloudClassifier*. Ty už se dají nadále jednoduše využívat v hlavních aktivitách aplikace: *MainActivity* a *AnalyzePhotoActivity*. V *MainActivity* se využívá pouze kvůli rychlému módu, u kterého se nepřechází do aktivity analýzy fotografie.

Za zmínku ještě stojí třída *CloudsClassification*, která je výsledkem metody *getImageCloudsClassification*. Nese v sobě výsledky analýzy a poskytuje jednoduché rozhraní pro jejich dotazování.




Obrázek 4.17: Diagram tříd finální aplikace. Vyobrazené jsou pouze klíčové metody.

# Kapitola 5

# Závěr

Cílem této práce bylo vytvořit aplikaci pro mobilní zařízení použitelnou v reálném světě, která je schopna předpovědět nastávající počasí na základě snímků aktuálních oblak.

Během této práce jsem analyzoval jednotlivé kategorie oblak, popsal jsem jejich vzhled, strukturu, podmínky vzniku a předpokládané nastávající počasí. Vzhledem k velmi podobnému vzhledu několika kategorií jsem vytvořil možné třídy oblak, kde do každé třídy může náležet více kategorií.

Dále jsem v několika iteracích tvořil dataset. Musel jsem vyzkoušet více metod tvorby datasetu, protože některé byly nedostatečně kvalitní, nebo za daný čas nebyly schopny poskytnout dostačující počet vzorků. Vzhledem k velmi malé četnosti výskytu a k těžké klasifikaci oblaku typu Comulonimbus jsem musel tuto kategorii vynechat a brát ji jako možnou nedetekovatelnou podmnožinu oblaku Cumulus.

Dataset jsem poté využil pro učení neuronových sítí, jejichž teoretické základy a princip jsem vysvětlil v první části práce. Na každém datasetu jsem experimentoval s různými architekturami a porovnával jejich úspěšnost na základě různých parametrů. Mimo neuronové sítě pro kategorizaci oblak jsem implementoval a naučil síť pro rozhodnutí, zda-li je na fotografii obloha, nebo nějaký jiný objekt, zejména pro ulehčení práce klasifikátoru, stejně tak ale i pro přívětivější reakci na uživatelský vstup.

Nakonec jsem implementoval aplikaci pro mobilní zařízení s operačním systémem Android, která využívala naučené neuronové sítě. Při její tvorbě jsem postupoval v jednotlivých iteracích, ve kterých jsem ji vždy testoval a na základě testování měnil. Vracel jsem se i k úpravě datasetu a přeučení neuronových sítí. Při testování jedné z prvních verzí jsem se pokusil vyřešit problém se snímky, na kterých velkou část plochy zabíral jiný objekt než obloha. Vzhledem k tomu, že se mi nepodařilo tento problém vyřešit automatickým ořezem, poskytl jsem uživatelovi možnost fotoaparát přiblížit a tím zlepšit snímek pro klasifikaci.

Protože dataset byl jedním ze základních kamenů při tvorbě této práce, vytvořil jsem spojení mezi aplikací a online úložištěm, kam se fotografie vždy po analýze pošle s příznakem výstupu analýzy. Tuto vlastnost jsem pak využil pro tvorbu druhé iterace datasetu.

Výsledkem byla plně funkční, jednoduše použitelná aplikace, která velmi dobře dokázala rozpoznat oblak a na základě toho s jistou pravděpodobností předpovědět počasí. Vzhledem k tomu, že nastávající počasí závisí i na různých dalších faktorech, které z jedné fotografie oblak nevyčteme, musíme brát předpověď s nadhledem. Aplikace nám ale může dát o nastávajícím počasí velmi dobrou představu, a to zejména v situacích, kdy nemáme přístup k internetu.

# Literatura


[1] Artificial neuron — Wikipedia, The Free Encyclopedia. 2018, [Online; accessed 21-December-2018].
URL https://en.wikipedia.org/wiki/Artificial_neuron

[2] Neuron — Wikipedia, The Free Encyclopedia. 2018, [Online; accessed 21-December-2018].
URL https://en.wikipedia.org/wiki/Neuron

[3] Rectifier (neural networks) — Wikipedia, The Free Encyclopedia. 2018, [Online; accessed 21-December-2018].
URL https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

[4] Recurrent neural network — Wikipedia, The Free Encyclopedia. 2018, [Online; accessed 24-December-2018].
URL https://en.wikipedia.org/wiki/Recurrent_neural_network

[5] eon Bottou, L.: Stochastic Gradient Learning in Neural Networks. 1991, [Online; accessed 23-July-2019].
URL https://leon.bottou.org/publications/pdf/nimes-1991.pdf

[6] Cernocky, J.: Systemy. 2010, [Online; accessed 23-July-2019].
URL https://www.fit.vutbr.cz/study/courses/ISS/public/pred/05_syst_konv/syst_konv.pdf

[7] Changhau, I.: Loss functions in neural networks. 2018, [Online; accessed 2-December-2018].
URL https://isaacchanghau.github.io/post/loss_functions/

[8] developer.android.com: android.hardware.camera2. 2019, [Online; accessed 15-March-2019].
URL https://developer.android.com/reference/android/hardware/camera2/package-summary

[9] EliteDataScience.com: Overfitting in Machine Learning: What It Is and How to Prevent It. 2019, [Online; accessed 23-July-2019].
URL https://elitedatascience.com/overfitting-in-machine-learning

[10] Foster, D.: GPX: the GPS Exchange Format. 2019, [Online; accessed 26-December-2019].
URL https://www.topografix.com/gpx.asp

[11] Goldstein, M.: *The Complete Idiot's Guide to Weather (2nd Edition)*. Alpha, 2002, ISBN 0028643410.
URL https://www.amazon.com/Complete-Idiots-Guide-Weather-2nd/dp/0028643410?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0028643410

[12] Google: Android Camera2Basic Sample.
https://github.com/googlesamples/android-Camera2Basic, 2019.

[13] Haykin, S.: *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, jul 1998, ISBN 0132733501.
URL https://www.xarg.org/ref/a/0132733501/

[14] Hrushikesh Mhaskar, T. P., Qianli Liao: When and Why Are Deep Networks Better than Shallow Ones? 2018, [Online; accessed 24-December-2018].
URL https://pdfs.semanticscholar.org/f594/f693903e1507c33670b89612410f823012dd.pdf

[15] Keras.io: Getting started with the Keras Sequential model. 2019, [Online; accessed 24-July-2019].
URL https://keras.io/getting-started/sequential-model-guide/

[16] LLC, I. S.: ImageMagick. 2019, [Online; accessed 15-March-2019].
URL https://www.imagemagick.org/

[17] Mathworks: Convolutional Neural Network. 2018, [Online; accessed 26-December-2018].
URL https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

[18] Prabhu, R.: Understanding of Convolutional Neural Network (CNN) — Deep Learning. 2018, [Online; accessed 26-December-2018].
URL https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[19] Saxena, A.: Convolutional Neural Networks (CNNs): An Illustrated Explanation. 2016, [Online; accessed 26-December-2018].
URL https://blog.xrds.acm.org/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/

[20] Sigmundarson, S.: Maps to GPX. 2019, [Online; accessed 15-March-2019].
URL https://mapstogpx.com/

[21] Skřehot, D. P.: Atlas oblaků. 2005, [Online; accessed 3-March-2019].
URL http://mraky.astronomie.cz

[22] Skřehot, D. P.: *Velký atlas oblaků*. Computer Press, 2008, ISBN 978-80-251-2015-6.

[23] Szegedy, C.: Going deeper with convolutions. 2015, p. 1-9.
URL https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

[24] Techopedia: Hebbian Theory. 2018, [Online; accessed 26-December-2018].
URL https://www.techopedia.com/definition/33275/hebbian-theory

[25] to TensorFlow, K.: Android Camera2Basic Sample.
https://github.com/amir-abdi/keras_to_tensorflow/, 2019.

[26] Wikipedia: Backpropagation — Wikipedia, The Free Encyclopedia. 2018, [Online; accessed 2-December-2018].
URL https://en.wikipedia.org/wiki/Backpropagation

[27] Wikipedia: Convolutional neural network. 2018, [Online; accessed 26-December-2018].
URL https://en.wikipedia.org/wiki/Convolutional_neural_network

[28] Wikipedia: Konvoluce. 2018, [Online; accessed 26-December-2018].
URL https://cs.wikipedia.org/wiki/Konvoluce

[29] Wikipedia: Oblak. 2018, [Online; accessed 26-December-2018].
URL https://cs.wikipedia.org/wiki/Oblak

[30] Xavier Glorot, Y. B., Antoine Bordes: Deep Sparse Rectifier Neural Networks. 2011, [Online; accessed 21-December-2018].
URL http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf

[31] Zhou, V.: A Simple Explanation of the Softmax Function. 2019, [Online; accessed 24-July-2019].
URL https://victorzhou.com/blog/softmax/

# Příloha A

# Obsah paměťového média

Po otevření paměťového média najdete tyto adresáře:

**src/** Všechny soubory se zdrojovým kódem

**doc/** Zdrojové kódy k této technické zprávě, stejně jako její PDF verze

**bin/** Konečná verze přeložené aplikace na Android