Klasifikace pomocí samoorganizujících se map s implementací GUI

Bc. Jiří Šubík

Diplomová práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav informatiky a umělé inteligence

Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jiří Šubík**
Osobní číslo:	**A16233**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Klasifikace pomocí samoorganizujících se map s implementací GUI**
Téma práce anglicky:	**Classification by Means of Self-organising Maps with GUI Implementation**

Zásady pro vypracování

1. Nastudujte a popište Kohonenovy samoorganizující se mapy (SOM).
2. Navrhněte a vhodně implementujte aplikaci pro SOM.
3. Navrhněte a zpracujte metodiku přípravy dat.
4. Hotovou aplikaci otestujte na úlohách rozpoznávání obrazu a zvuku.
5. Zhodnoťte dosažené výsledky.

Rozsah diplomové práce:
Rozsah příloh:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. KOHONEN T.: Self-Organizing Maps, 2nd extended ed., Berlin, 1997, Springer, ISBN 978-3-642-56927-2.
2. OJA E., KASKI S.: Kohonen Maps. Elsevier, 1999, ISBN 9780080535296.
3. KŘIVAN, Miloš. Úvod do umělých neuronových sítí. Vyd. 3., přeprac. Praha: Oeconomica, 2014, 44 s. ISBN 978-80-245-2024-7.
4. KOHONEN T. and HONKELA T.: Kohonen network. Scholarpedia, 2(1):1568, [online] 2007. URL http://www.scholarpedia.org/article/Self-organizing_map.
5. ROJAS R.: Neural networks – a systematic introduction, Springer-Verlag, Berlin,New-York, 1996.
6. VOLNÁ, E. Neuronové sítě I. Ostrava: Ostravská univerzita, 2002. 85 s. Elektronický text.
7. KOHONEN T.: Speedy SOM, Report A33, Laboratory of Computer and InformationScience, Helsinki University of Technology, 1996.
8. WANG, Chao-Huang, 2009. Variants of Self-Organizing Maps: Applications in Image Quantization and Compression. Saarbrücken, Germany: LAP Lambert Academic Publishing. ISBN 978-3-8383-2436-4.

Vedoucí diplomové práce: **doc. Ing. Zuzana Komínková Oplatková, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **28. listopadu 2019**
Termín odevzdání diplomové práce: **15. května 2020**

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Ve Zlíně dne 9. prosince 2019

Prohlašuji, že

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

Prohlašuji,

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

Bc. Jiří Šubík v.r.

Ve Zlíně 1.5.2020 .....

podpis autora

ABSTRAKT

Tato diplomová práce se zabývá Kohonenovou samoorganizující se mapou. Tvorbou aplikace realizující tvorbu a učící proces a využitím samoorganizující se mapy při klasifikaci obrazu a zvuku.

Klíčová slova: Samoorganizující se mapy, Kohonenova mapy, klasifikace obrazu a zvuku

ABSTRACT

This diploma thesis deals with Kohonen's self-organizing map. By creating an application that implements the creation and learning process and using a self-organizing map in the classification of image and sound.

Keywords: Self-organizing Maps, Kohonen map, picture and sound classification

Chtěl bych především poděkovat vedoucí práce doc. Ing. Zuzaně Komínkové-Oplatkové, Ph.D. za velice přínosné rady, pomoc a připomínky při tvorbě diplomové práce. Dále bych chtěl poděkovat rodině za podporu po celou dobu studia.

OBSAH

ÚVOD

Tato práce se zabývá klasifikací obrazu a zvuku pomocí Kohonenových samoorganizujících se map (SOM). V práci je popsána podrobně struktura, princip činnosti a implementace SOM včetně aplikace. V dalších kapitolách se autor věnuje metodice přípravy dat a jejich zpracování. Vytvořená aplikace bude otestována na úlohách klasifikace obrazu a zvuku.

Motivací k této práci byla snaha pochopení fungování rozpoznávání obrazu a zvuku, snaha pochopit základní principy a naprogramovat si vlastní aplikaci, která umí rozpoznávat obraz a zvuk. Především v dnešní době domácích hlasových asistentů, chatovacích aplikacích velkých firem, které nahrazují telefonního operátora umělou inteligencí, chytrých spotřebičů a chytrých aut, u kterých kreslením prstem na touchpadu vybíráme kontakty z telefonního seznamu.

První kapitola shrnuje historii vývoje neuronových sítí, podobnost biologického a umělého neuronu a popisuje jeho matematický model.

Druhá kapitola popisuje Kohonenovy samoorganizující se mapy, jejich topologii, topologické sousedství, princip učení a hlavní oblasti jejich použití.

Třetí kapitola popisuje metodiku přípravy obrazových a zvukových dat a jejich transformací do formy potřebné při trénování a testování SOM.

Čtvrtá kapitola popisuje implementaci aplikace SOM, prerekvizity pro vývoj aplikací, použitý programovací jazyk a vývojové prostředí. Dále popisuje naprogramované aplikace a jejich funkcionalitu a jejich určení v procesu klasifikace pomocí SOM.

Pátá kapitola zahrnuje testování vytvořené aplikace na úlohách rozpoznávání obrazu a zvuku a prezentaci výsledných testů.

Šestá kapitola hodnotí dosažené výsledky.

I. TEORETICKÁ ČÁST

1 Umělé neuronové sítě

Historie vzniku neuronových sítí spadá do poloviny 20. století díky rozvoji informatiky, jako vědního oboru. Jako počátek vzniku oboru neuronových sítí je považována práce Warrena McCullocha a Waltera Pittse z roku 1943, kteří vytvořili velmi jednoduchý matematický model neuronu, což je základní buňka nervové soustavy. Ukázali, že nejjednodušší typy neuronových sítí mohou v principu počítat libovolnou aritmetickou nebo logickou funkci. Ačkoliv nepočítali s možností bezprostředního praktického využití svého modelu, jejich článek měl velký vliv na ostatní badatele. V roce 1949 napsal Donald Hebb knihu *The Organization of Behaviour*, ve které navrhl učící pravidlo pro synapse neuronů, inspirováno myšlenkou, že podmíněné reflexy, které jsou pozorovatelné u všech živočichů, jsou vlastnostmi jednotlivých neuronů.

Dalším mezníkem by rok, kdy 1957 Frank Rosenblatt představil perceptron, který je zobecněním McCullochova a Pittsova modelu neuronu pro reálný číselný obor parametrů. Roku 1958 byl sestrojen první úspěšný neuropočítač pro rozpoznávání znaků. Frank Rossenblatt je proto dodnes některými odborníky považován za zakladatele tohoto nového oboru. Krátce po objevu perceptronu Bernard Widrow se svými studenty vyvinul další typ neuronového výpočetního prvku, který nazval *ADALINE* (ADAptive LInear NEuron). Tento model byl vybaven novým výkonným učícím pravidlem, které se dodnes nezměnilo. V následujících letech dochází k úspěšnému rozvoji neurovýpočtů v oblasti návrhu nových modelů neuronových sítí a jejich implementací. Od roku 1969 do 1982 se výzkum neuronových sítí takřka zastavil. Předcházela tomu kampaň Marvina Minského a Seymoura Paperta, kdy došlo k diskreditaci výzkumu neuronových sítí díky známému triviálního faktu, že jeden perceptron nemůže počítat jednoduchou logickou funkci, tzv. XOR problém. Po tomto období se neurovýpočty začali zabývat další badatelé (např. Shun Ichi Amari, James Anderson, Kunihiko Fukushima, Stephen Grossberg, Harry Klopf, Teuvo Kohonen a David Willshaw), kteří přispěli svými objevy k renesanci neuronových sítí [6].

1.1 Biologický neuron

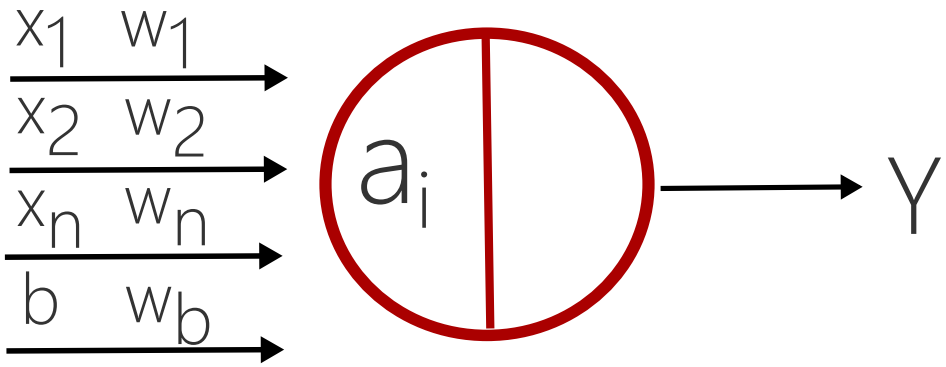
Nervová soustava člověka je velmi složitý systém, který je stále předmětem zkoumání. Uvedené velmi zjednodušené neurofyziologické principy nám však v dostatečné míře stačí k formulaci matematického modelu neuronové sítě. Základním stavebním funkčním prvkem nervové soustavy je nervová buňka, neuron. Neurony jsou samostatné specializované buňky, určené k přenosu, zpracování a uchování informací, které jsou nutné pro realizaci životních funkcí organismu.

Neuron je přizpůsoben pro přenos signálů tak, že kromě vlastního těla (somatu), má i vstupní a výstupní přenosové kanály: dendrity a axon. Z axonu odbočuje řada

větví (terminálů), zakončených blánou, která se převážně stýká s výběžky (trny), dendritů jiných neuronů. K přenosu informace pak slouží unikátní mezineuronové rozhraní, synapse.

Z funkčního hlediska lze synapse rozdělit na excitační, které umožňují rozšíření vzruchu v nervové soustavě a na inhibiční, které způsobují jeho útlum. Paměťová stopa v nervové soustavě vzniká pravděpodobně zakódováním synaptických vazeb na cestě mezi receptorem (čidlem orgánu) a efektorem (výkonným orgánem). Šíření informace je umožněno tím, že soma i axon jsou obaleny membránou, která má schopnost za jistých okolností generovat elektrické impulsy. Tyto impulsy jsou z axonu přenášeny na dendrity jiných neuronů synaptickými branami, které svou propustností určují intenzitu podráždění dalších neuronů. Takto podážděné neurony při dosažení určité hraniční meze, tzv. prahu, samy generují impuls a zajišťují tak šíření příslušné informace. Po každém průchodu signálu se synaptická propustnost mění, což je předpokladem paměťové schopnosti neuronů. Také propojení neuronů prodělává během života organismu svůj vývoj: v průběhu učení se vytváří nové paměťové stopy nebo při zapomínání se synaptické spoje přerušují.

Základem matematického modelu neuronové sítě je formální neuron. Formální neuron má obecně n reálných vstupů, které modelují dendrity a určují vstupní vektor $x = (x_1, ..., x_n)$. Tyto vstupy jsou ohodnoceny reálnými synaptickými váhami tvořícími vektor $w = (w_1, ..., w_m)$. Synaptické váhy mohou být i záporné, pro vyjadření inhibičního charakteru [6].



Obr. 1.1 Umělý neuron

Obrázek (Obr. 1.1) zobrazuje model umělého neuronu, kde x jsou vstupy, w jsou váhy, b je práh, y je výstup, a_i je vnitřní potenciál/aktivační funkce a $f(a)$ je přenosová funkce.

Vnitřní potenciál je definovaný vztahem 1.1

$$a = \sum_{i=1}^{n} x_i w_i + b w_b \tag{1.1}$$

Výstup z přenosové funkce $f(a)$ je definovaný vztahem 1.2

$$f(a) = TF(\sum_{i=1}^{n} x_i w_i + b w_b) \tag{1.2}$$

Přenosová funkce (TF) může být logistická sigmoida, hyperbolický tangens, funkce Perceptron, lineární, lineární omezená, skoková, funkce RBF (radiální báze). Klasifikace může být binární nebo vícehodnotová v závislosti na použité přenosové funkci.

Neuronové sítě můžeme dělit podle počtu vrstev na jednovrstvé a vícevrstvé. Dále dle algoritmu učení na učení s učitelem a bez učitele. Podle stylu učení na deterministické a stochastické. Podle typu vstupu na binární a spojité. Podle struktury a topologie.

1.2 Neuronová síť

Neuronová síť je složena z formálních neuronů, které jsou vzájemně propojeny tak, že výstup jednoho neuronu je vstupem do dalších neuronů. Počet neuronů a jejich vzájemné propojení v síti určuje topologie neuronové sítě. Rozlišujeme v síti vstupní, skryté a výstupní neurony. Stavy všech neuronů v síti určují stav neuronové sítě a synaptické váhy všech spojů představují konfiguraci neuronové sítě. Neuronová síť v čase mění stav neuronů, adaptují se váhy.

Dynamiku neuronové sítě dělíme do tří dynamik a třech režimů práce sítě: organizační (změna topologie), aktivní (změna stavu) a adaptivní (změna konfigurace). Organizační dynamika specifikuje architekturu neuronové sítě a její případnou změnu. Změna topologie se většinou uplatňuje v rámci adaptivního režimu tak, že síť je v případě potřeby rozšířena o další neurony a příslušné spoje. Rozlišujeme dva typy architektury: cyklická (rekurentní) a acyklická (dopředná) síť. Aktivní dynamika specifikuje počáteční stav sítě a způsob jeho změny v čase při pevné topologii a konfiguraci. V průběhu výpočtu je podle daného pravidla aktivní dynamiky vybrán jeden neuron nebo více neuronů, které aktualizují svůj stav na základě svých vstupů, tj. stavů sousedních neuronů, jejichž výstupy jsou vstupy aktualizovaných neuronů. Adaptivní dynamika neuronové sítě specifikuje počáteční konfiguraci sítě a způsob, jakým se mění váhové hodnoty na spojeních mezi jednotlivými neurony v čase. V adaptivním režimu se tedy na začátku nastaví váhy všech spojů v síti na počáteční konfiguraci (např. náhodně). Po inicializaci konfigurace sítě probíhá vlastní adaptace. Cílem adaptace je nalézt takovou konfiguraci sítě ve váhovém prostoru, která by v aktivním režimu realizovala předepsanou funkci. Jestliže aktivní režim sítě se využívá k vlastnímu výpočtu funkce sítě pro daný vstup, pak adaptivní režim slouží k učení této funkce.

Požadovaná funkce sítě je zadána tréninkovou množinou dvojic vstup/výstup sítě

(tzv. tréninkový vzor). Způsobu adaptace, kdy požadované chování sítě modeluje učitel, který pro vzorové vstupy sítě informuje adaptivní mechanismus o správném výstupu sítě, se nazývá učení s učitelem (supervised learning). Někdy učitel hodnotí kvalitu momentální skutečné odpovědi (výstupu) sítě pro daný vzorový vstup pomocí známky, která je zadána místo požadované hodnoty výstupu sítě (tzv. klasifikované učení).

Jiným typem adaptace je samoorganizace. Kdy tréninková množina obsahuje jen vstupy sítě. To modeluje situaci, kdy není k dispozici učitel, proto se tomuto způsobu adaptace také říká učení bez učitele. Neuronová síť v adaptivním režimu sama organizuje tréninkové vzory do tříd s podobnými vlastnostmi [6].

Tato práce se dále věnuje samoorganizaci, konkrétně Kohonenovým samoorganizujícím se mapám.

2 Kohonenova samoorganizující se mapa

Samoorganizující se mapy (SOM) jsou hlavním představitelem umělých neuronových sítí s učením bez učitele a byly poprvé popsány v roce 1982. Patří do skupiny samoučících se neuronových sítí. Využívají se pro klasifikaci neznámých dat. Objekty s podobnými vlastnostmi se shlukují do skupin. Autorem SOM je profesor Teuvo Kohonen z Helsinské technické univerzity ve Finsku.

Síť sama rozhodne, která odezva je pro přeložený vstupnímu vektoru nejlepší a podle toho nastaví své váhy. Pravidlo vítěz bere vše. Výpočet probíhá na základě kompetitivního chování, kdy pro daný vstupní vektor je aktivní právě jeden vítězný neuron. V kompetici vítězí neuron, který je nejblíže předloženému vstupnímu vektoru. Vítězný neuron bude nejaktivnější a bude potlačovat aktivitu ostatních neuronů. Pro rozhodnutí, zda bude neuron aktivní nebo ne, je nutná globální informace o stavu všech neuronů v síti.

Aktivita neuronu signalizuje příslušnost předloženého vstupu ke shluku vektorů reprezentovaných tímto neuronem. Jsou-li si tedy dva libovolné vzory blízké ve vstupním prostoru způsobují v síti odezvu na neuronech, které jsou si fyzicky blízké ve výstupním prostoru. Vítězný neuron, spolu s neurony, které jsou v jeho okolí, adaptuje své váhy směrem k předloženému vzoru. V průběhu učení se aktualizují váhy příslušných neuronů i jejich sousedů v závislosti na poloměru okolí.

Učící faktor $0 < \alpha \leq 1$ definuje míru změny vah. Na počátku učení je obvykle blízký jedné a postupně se zmenšuje až na definovanou hodnotu rovné ukončovací podmínce a ukončí tím proces adaptace. Zároveň taky snižujeme velikost poloměru okolí, kdy na začátku adaptace je okolí obvykle velké (např. polovina velikosti sítě) a na konci učení potom zahrnuje jen jeden samotný vítězný neuron ($R = 0$) [9], [6].

Klasifikace pomocí SOM má vždy dvě fáze:

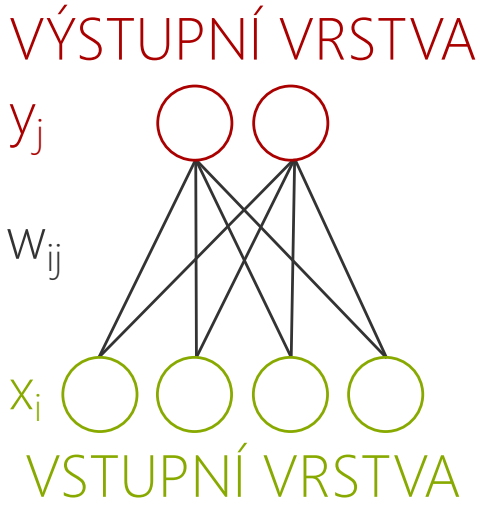
- Trénování - změna synaptických vah odpovídajících vstupnímu vektoru
- Testování - testování sítě - synaptické váhy se nemění

2.1 Topologie

SOM je dvouvrstvá neuronová síť, která se skládá ze vstupní a výstupní vrstvy. Počet neuronů vstupní vrstvy udává dimenzi vstupních dat. Počet neuronů výstupní vrstvy udává počet tříd, které chceme klasifikovat. Topologie výstupní vrstvy může být jednodimenzionální (1D) nebo dvoudimenzionální (2D). Vstupní vrstva je plně propojena s výstupní vrstvou synaptickými váhami. Čili každý neuron vstupní vrstvy je propojen se všemi neurony vrstvy výstupní [1], [6].

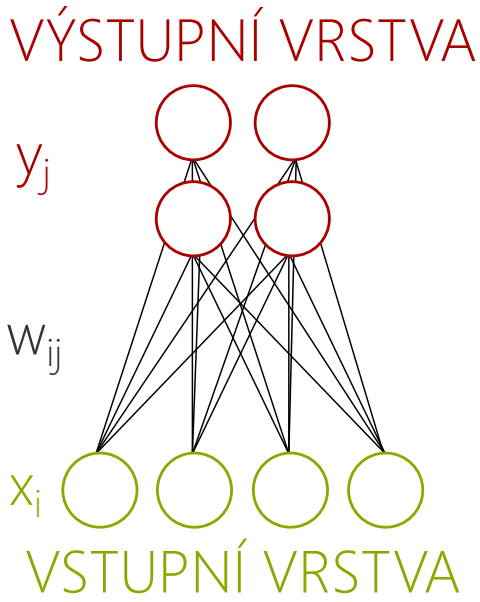
Vstupní vrstvu značíme $X = (x_1, x_2...x_n)$, kde n je počet vstupních neuronů udávající dimenzi vstupních dat. Výstupní vrstvu značíme $Y = (y_1, y_2...y_m)$, kde m je počet výstupních neuronů udávající počet klasifikovaných tříd.

Např. 1D výstupní vrstva (Obr. 2.1) se dvěma neurony pro klasifikaci do dvou tříd. Vstupní vrstva obsahuje čtyři neurony.



Obr. 2.1 1D topologie SOM

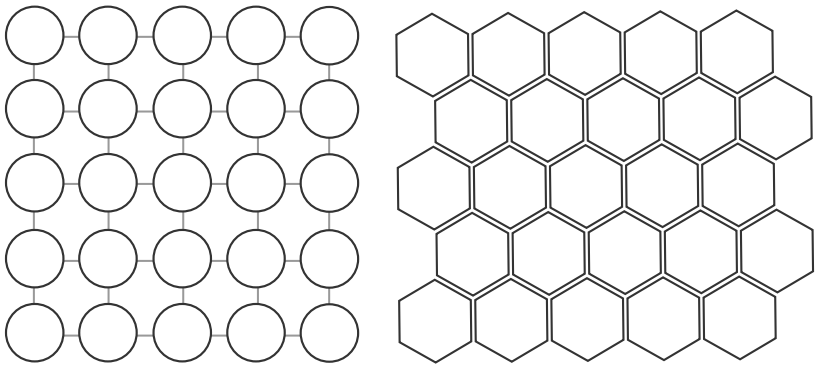
Vstupní neuron x_1 je propojený synaptickými váhami w_{11} a w_{12} s výstupními neurony y_1 a y_2. Obecně je tedy každý vstupní neuron x_i propojený synaptickými váhami w_{ij} se všemi výstupními neurony y_j, kde $j = 1...m$.



Obr. 2.2 2D topologie SOM

2.2 Výstupní vrstva

Výstupní vrstva 2D topologie je uspořádaná do mřížky, může být čtvercová/obdélníková (Obr. 2.3 (a)) nebo hexagonální (Obr. 2.3 (b)). Rozdíl mezi uvedenými typy mřížky spočívá v odlišném okolí neuronů. Neuron čtvercové/obdélníkové mřížky s poloměrem okolí $R = 1$ má 8 sousedních neuronů, zatímco neuron hexagonální mřížky s poloměrem okolí $R = 1$ má 6 sousedních neuronů. Uvedené počty sousedních neuronů jsou platné pro symetrické okolí.



(a) Čtvercová/obdelníková (b) Hexagonální

Obr. 2.3 Mřížka 2D topologie

2.3 Okolí neuronu

Každý neuron výstupní vrstvy má své sousední neurony v daném okolí. Poloměr okolí neuronu značíme R. Okolí R zároveň zahrnuje i okolí $R - 1$. Okolí neuronu může být symetrické nebo asymetrické. Maximální velikost okolí neuronu R_{max} závisí na topologii. Doporučená velikost okolí je v intervalu $R_{max}/2 \leq R \leq R_{max}$. Minimální poloměr okolí by tedy měl být nejméně polovina maximálního okolí. Při nižší hodnotě by při trénování SOM mohlo dojít k situaci, že nepokryjeme celou mřížku. Vítězný neuron značíme BMU (best match unit) [6], [9].

2.3.1 Okolí 1D topologie

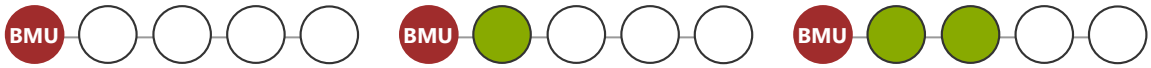
Neurony výstupní vrstvy 1D topologie jsou uspořádány do lineární struktury. Maximální okolí je dáno vztahem 2.1, kde X je počet neuronů výstupní vrstvy.

$$R_{max} = X - 1 \tag{2.1}$$

Např. pro $X = 5$ je $R_{max} = X - 1 = 4$.
V 1D topologii patří do okolí neuronu j s poloměrem okolí $R = 1$ neurony $j - 1$ a $j + 1$.

(a) R=0 (b) R=1 (c) R=2

Obr. 2.4 Okolí vítězného neuronu pro 1D topologii



(a) R=0 (b) R=1 (c) R=2

Obr. 2.5 Asymetrické okolí vítězného neuronu pro 1D topologii

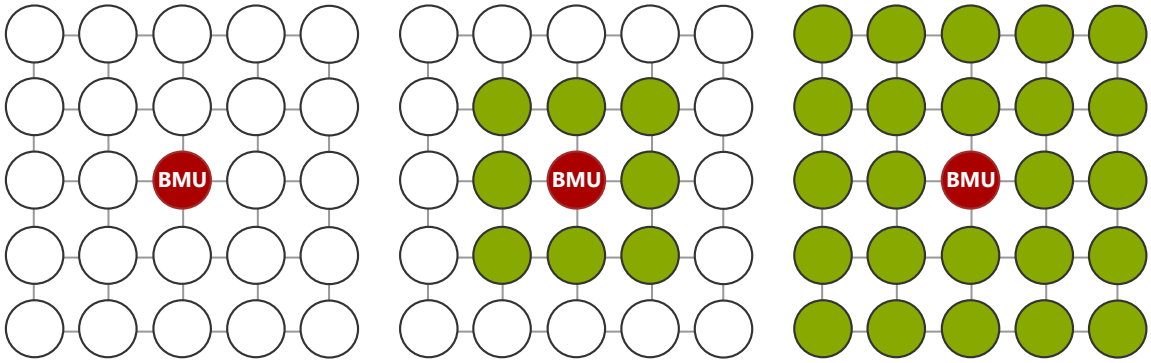
Obecně tedy do okolí neuronu j patří všechny neurony vzdálené od j o R indexů vlevo a vpravo [9].

2.3.2 Okolí čtvercové/obdelníkové mřížky

Neurony výstupní vrstvy 2D topologie jsou uspořádány do čtvercové nebo obdélníkové struktury. Maximální okolí je dáno vztahem 2.2, kde X a Y jsou počty neuronů výstupní vrstvy udávající počet sloupců a řádků mřížky.

$$R_{max} = max(X, Y) - 1 \tag{2.2}$$

Např. pro $X = 3$ a $Y = 4$ je $R_{max} = max(3, 4) - 1 = 3$.

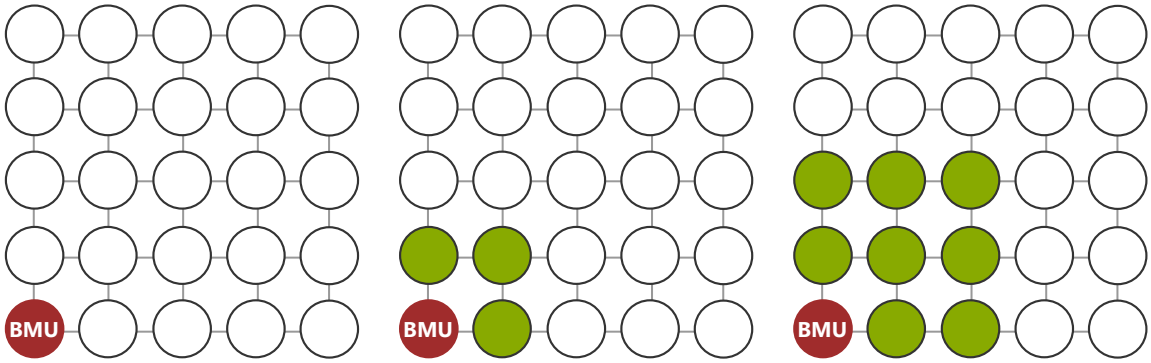


(a) R=0 (b) R=1 (c) R=2

Obr. 2.6 Okolí vítězného neuronu pro 2D topologii

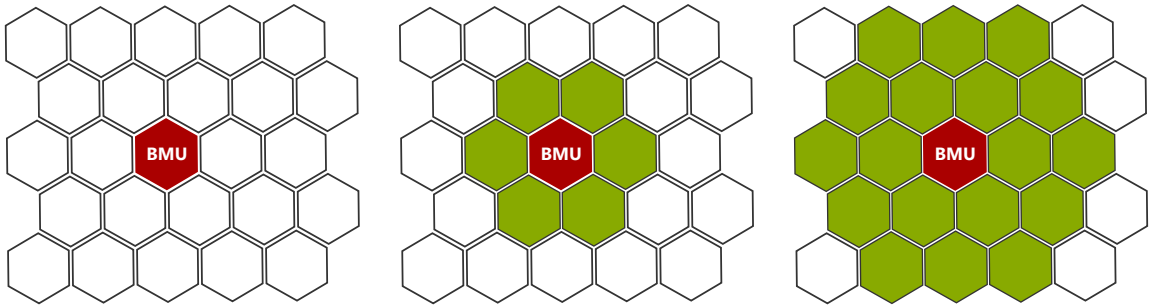
2.3.3 Okolí hexagonální mřížky

Neurony výstupní vrstvy 2D topologie jsou uspořádány do hexagonální struktury. Maximální okolí je dáno vztahem 2.2, kde X a Y jsou počty neuronů výstupní vrstvy udávající počet sloupců a řádků mřížky.



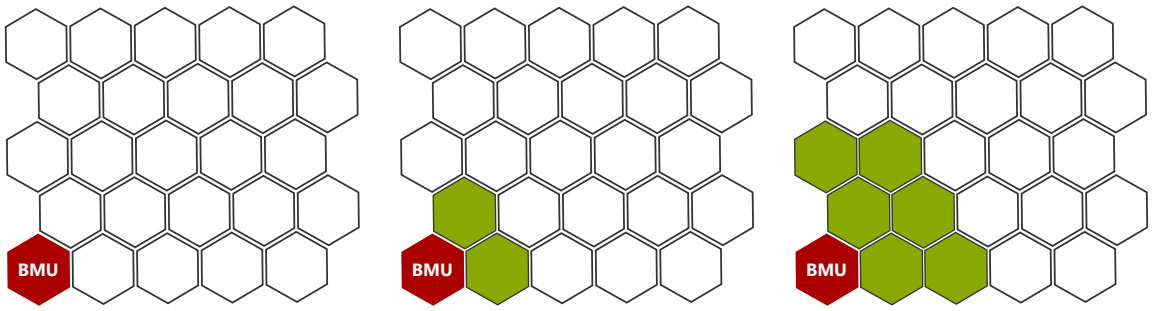
(a) R=0 (b) R=1 (c) R=2

Obr. 2.7 Asymetrické okolí vítězného neuronu pro 2D topologii



(a) R=0 (b) R=1 (c) R=2

Obr. 2.8 Okolí vítězného neuronu pro hexagolnální topologii



(a) R=0 (b) R=1 (c) R=2

Obr. 2.9 Asymetrické okolí vítězného neuronu pro hexagolnální topologii

2.4 Konfigurace SOM

Konfigurací rozumíme sadu parametrů, které určují strukturu SOM, definují počáteční hodnotu pro učící faktor, počáteční okolí a ukončovací podmínku. Vztah 2.3 definuje ukončovací podmínku jako minimální hodnotu faktoru učení, zatímco vztah 2.4 ukončovací podmínku definuje jako počet epoch, čili počet cyklů algoritmu.

$$K = (X \times Y, R, \alpha, \alpha_{\min}) \tag{2.3}$$

Např. konfigurace $K = (2 \times 1, R = 0, \alpha = 0,6, \alpha_{\min} = 0,01)$ definuje 1D topologii se dvěma výstupními neurony, poloměrem okolí $R = 0$ a počáteční hodnotu faktoru učení $\alpha = 0,6$ a ukončovací podmínku $\alpha_{\min} = 0,01$.

$$K = (X \times Y, R, \alpha, epochs) \tag{2.4}$$

Např. konfigurace $K = (5 \times 5, R = 2, \alpha = 0,9, epochs = 1000)$ definuje 2D topologii s výstupními neurony uspořádanými do mřížky s pěti sloupci a pěti řádky, poloměrem okolí $R = 2$ a počáteční hodnotu faktoru učení $\alpha = 0,9$ a ukončovací podmínku $epochs = 1000$.

2.5 Princip učení

Výstupní vrstvě postupně předkládáme vstupní vektory (tzv. trénovací vzory) a pro každý vektor porovnáme rozdíl vah každého neuronu výstupní vrstvy s hodnotami vstupního vektoru. Počet neuronů výstupní odpovídá předpokládanému počtu klasifikovaných tříd. Ke zjištění rozdílu použijeme výpočet euklidovské vzdálenosti $D(j)$ pro každé $j = 1...m$ podle vztahu 2.5.

$$D(j) = \sum_i (w_{ij} - x_i)^2 \tag{2.5}$$

Neurony počítají vzdálenost mezi předloženým vzorem a svým váhovým vektorem.

Faktor učení α po každém cyklu upravujeme podle vztahu 2.6 s počáteční hodnotou $\alpha(0)$ v intervalu $0 < \alpha < 1$.

$$\alpha(t+1) = \alpha(t)/2 \tag{2.6}$$

Další vztah 2.7 pracuje se zadaným počtem epoch.

$$\alpha(t+1) = 0,9(1 - t/epochs) \tag{2.7}$$

Využití vztahu s definováním počtu epoch se využívá například pro klasifikaci zvuku, kdy Kohonen [1] doporučuje 1000–10000 epoch.

Synaptické váhy w_{ij} pro vítězný neuron aktualizujeme po každém předložení vstupního vektoru podle vztahu 2.8.

$$w_{ij}(new) = ((1 - \alpha) \times w_{ij}(old)) + (\alpha \times x_i) \tag{2.8}$$

Zároveň taktéž aktualizujeme synaptické váhy pro sousední neurony vítězného neuronu

podle vztahu 2.9.

$$w_{ij}(new) = ((1 - NF) \times w_{ij}(old)) + (NF \times x_i) \tag{2.9}$$

NF je funkce sousedství a je dána vztahem 2.10.

$$NF = \alpha \times \exp(-(DFW^2)/(2 \times R^2)) \tag{2.10}$$

Funkce závisí na aktuální hodnotě poloměru okolí R a hodnotě funkce DFW, která udává vzdálenost aktualizovaného neuronu od vítězného neuronu.

Algoritmus učení sestává z několika kroků:

Krok 0 :

Definice konfigurace K podle vztahu 2.3 nebo 2.4. Inicializace váhové matice náhodně vygenerovanými hodnotami z intervalu $0 < w_{ij} < 1$.

Krok 1 :

Předložení a adaptace vstupního vektoru.

Krok 2 :

Výpočet euklidovské vzdálenosti D(j) pro každý výstupní neuron.

Krok 3 :

Zjištění indexu vítězného neuronu j, pro který platí, že $D(j) = min$.

Krok 4 :

Aktualizace vah vítězného neuronu podle vztahu 2.8 a v případě, že poloměr okolí $R > 0$, pro neurony v okolí podle vztahu 2.9. Kroky 1–4 opakujeme pro všechny vstupní vektory.

Krok 5 :

Aktualizace učícího faktoru α a pokračování v dalším cyklu algoritmu dokud $\alpha > \alpha_{min}$ nebo do vyčerpání počtu epoch, jinak ukončení algoritmu. V případě, že poloměr okolí $R > 0$, zároveň nastavíme hodnotu poloměru okolí na $R = R - 1$. Po každém cyklu algoritmu tedy zmenšujeme poloměru topologického sousedství.

Krok 5 tedy aktualizuje učící faktor α a poloměr okolí R a zároveň provádí kontrolu ukončovací podmínky běhu algoritmu.

2.6 Příklad

Mějme čtyři vstupní vektory $x_1 = (1, 1, 0, 0)$, $x_2 = (0, 0, 0, 1)$, $x_3 = (1, 0, 0, 0)$ a $x_4 = (0, 0, 1, 1)$ a 1D topologii se dvěma výstupními neurony [6].

Krok 0 :

Definice konfigurace $K = (2 \times 1, R = 0, \alpha = 0,6, \alpha_0 = 0,01)$

Inicializace váhové matice náhodně vygenerovanými hodnotami z intervalu $0 < w_{ij} < 1$.

$$\begin{bmatrix} 0,2000 & 0,8000 \\ 0,6000 & 0,4000 \\ 0,5000 & 0,7000 \\ 0,9000 & 0,3000 \end{bmatrix}$$

Krok 1 : Adaptace vektoru $x = (1, 1, 0, 0)$

Krok 2 : Výpočet vzdálenosti

$$D(0) = (0,2 - \mathbf{1})^2 + (0,6 - \mathbf{1})^2 + (0,5 - \mathbf{0})^2 + (0,9 - \mathbf{0})^2 = 1,86$$
$$D(1) = (0,8 - \mathbf{1})^2 + (0,4 - \mathbf{1})^2 + (0,7 - \mathbf{0})^2 + (0,3 - \mathbf{0})^2 = \mathbf{0,98}$$

Krok 3 : Zjištění vítězného neuronu
Index vítězného neuronu $j = \mathbf{1}$

Krok 4 : Aktualizace vah vítězného neuronu

$$w_{i1}(new) = ((1 - 0,6) \times w_{i1}(old)) + (0,6 \times x_i)$$

$$\begin{bmatrix} 0,2000 & \mathbf{0,9200} \\ 0,6000 & \mathbf{0,7600} \\ 0,5000 & \mathbf{0,2800} \\ 0,9000 & \mathbf{0,1200} \end{bmatrix}$$

Krok 1 : Adaptace vektoru $x = (0, 0, 0, 1)$

Krok 2 : Výpočet vzdálenosti

$$D(0) = (0,2 - \mathbf{0})^2 + (0,6 - \mathbf{0})^2 + (0,5 - \mathbf{0})^2 + (0,9 - \mathbf{1})^2 = \mathbf{0,66}$$
$$D(1) = (0,92 - \mathbf{0})^2 + (0,76 - \mathbf{0})^2 + (0,28 - \mathbf{0})^2 + (0,12 - \mathbf{1})^2 = 2,2768$$

Krok 3 : Zjištění vítězného neuronu
Index vítězného neuronu $j = \mathbf{0}$

Krok 4 : Aktualizace vah vítězného neuronu

$$w_{i0}(new) = ((1 - 0,6) \times w_{i0}(old)) + (0,6 \times x_i)$$

$$\begin{bmatrix} \mathbf{0,0800} & 0,9200 \\ \mathbf{0,2400} & 0,7600 \\ \mathbf{0,2000} & 0,2800 \\ \mathbf{0,9600} & 0,1200 \end{bmatrix}$$

Krok 1 : Adaptace vektoru $x = (1, 0, 0, 0)$

Krok 2 : Výpočet vzdálenosti

$$D(0) = (0,08 - \mathbf{1})^2 + (0,24 - \mathbf{0})^2 + (0,2 - \mathbf{0})^2 + (0,96 - \mathbf{0})^2 = 1,8656$$

$$D(1) = (0,92 - \mathbf{1})^2 + (0,76 - \mathbf{0})^2 + (0,28 - \mathbf{0})^2 + (0,12 - \mathbf{0})^2 = \mathbf{0,6768}$$

Krok 3 : Zjištění vítězného neuronu

Index vítězného neuronu $j = \mathbf{1}$

Krok 4 : Aktualizace vah vítězného neuronu

$$w_{i1}(new) = ((1 - 0,6) \times w_{i1}(old)) + (0,6 \times x_i)$$

$$\begin{bmatrix} 0,0800 & \mathbf{0,9680} \\ 0,2400 & \mathbf{0,3040} \\ 0,2000 & \mathbf{0,1120} \\ 0,9600 & \mathbf{0,0480} \end{bmatrix}$$

Krok 1 : Adaptace vektoru $x = (0, 0, 1, 1)$

Krok 2 : Výpočet vzdálenosti

$$D(0) = (0,08 - \mathbf{0})^2 + (0,24 - \mathbf{0})^2 + (0,2 - \mathbf{1})^2 + (0,96 - \mathbf{1})^2 = \mathbf{0,7056}$$

$$D(1) = (0,968 - \mathbf{0})^2 + (0,304 - \mathbf{0})^2 + (0,112 - \mathbf{1})^2 + (0,048 - \mathbf{1})^2 = 2,7243$$

Krok 3 : Zjištění vítězného neuronu

Index vítězného neuronu $j = \mathbf{0}$

Krok 4 : Aktualizace vah vítězného neuronu

$$w_{i0}(new) = ((1 - 0,6) \times w_{i0}(old)) + (0,6 \times x_i)$$

$$\begin{bmatrix} \mathbf{0,0320} & 0,9680 \\ \mathbf{0,0960} & 0,3040 \\ \mathbf{0,6800} & 0,1120 \\ \mathbf{0,9840} & 0,0480 \end{bmatrix}$$

Krok 5 : Aktualizace učícího faktoru $\alpha = 0,6 \times 0,5 = \mathbf{0,3}$
Pokračování v dalším cyklu algoritmu.

Zobrazili jsme si první cyklus algoritmu učení. Kompletní výpočet učení, pro uvedený příklad klasifikace daných vektorů, je součástí přílohy **P I.** (v umístění *Priklad > Priklad.pdf*) a zahrnuje postup výpočtu pro všechny cykly algoritmu až do dosažení ukončující podmínky.

Po ukončení algoritmu, dosažením hodnoty α_{min} postupným snižováním hodnoty učícího faktoru α, finální váhová matice konverguje k matici:

$$\begin{bmatrix} \mathbf{0,0} & 1,0 \\ \mathbf{0,0} & 0,5 \\ \mathbf{0,5} & 0,0 \\ \mathbf{1,0} & 0,0 \end{bmatrix}$$

Hodnoty prvního sloupce odpovídajících průměrným hodnotám složek obou vektorů přiřazeným prvnímu neuronu výstupní vrstvy, tj. vektoru $x_2 = (0, 0, 0, 1)$ a vektoru $x_4 = (0, 0, 1, 1)$.
Hodnoty druhého sloupce odpovídajících průměrným hodnotám složek obou vektorů přiřazeným druhému neuronu výstupní vrstvy, tj. vektoru $x_1 = (1, 1, 0, 0)$ a vektoru $x_3 = (1, 0, 0, 0)$.

Po naučení sítě (získáním váhové matice pro trénovací data) lze předložit neznámou instanci a otestovat, do kterého klastru patří. Otestujeme vektory x_1 =(0, 0, 0,9, 0,9) a x_2 =(0,9, 0,9, 0, 0).

Klasifikace vektoru x_1 =(0, 0, 0,9, 0,9):
Výpočet vzdálenosti D_j pro všechny neurony výstupní vrstvy.

$$D(0) = (0 - \mathbf{0})^2 + (0 - \mathbf{0})^2 + (0,5 - \mathbf{0,9})^2 + (1 - \mathbf{0,9})^2 = \mathbf{0,17}$$
$$D(1) = (1 - \mathbf{0})^2 + (0,5 - \mathbf{0})^2 + (0 - \mathbf{0,9})^2 + (0 - \mathbf{0,9})^2 = 2,87$$

Index vítězného neuronu výstupní vrstvy $j = \mathbf{0}$.

Klasifikace vektoru $x_2 = (0, 9, 0, 9, 0, 0)$:

Výpočet vzdálenosti D_j pro všechny neurony výstupní vrstvy.

$$D(0) = (0 - \mathbf{0{,}9})^2 + (0 - \mathbf{0{,}9})^2 + (0,5 - \mathbf{0})^2 + (1 - \mathbf{0})^2 = 2,87$$
$$D(1) = (1 - \mathbf{0{,}9})^2 + (0,5 - \mathbf{0{,}9})^2 + (0 - \mathbf{0})^2 + (0 - \mathbf{0})^2 = \mathbf{0{,}17}$$

Index vítězného neuronu výstupní vrstvy $j = \mathbf{1}$.

Oba předložené vektory x_1 a x_2 byly správně klasifikovány.

2.7 Použití SOM

SOM mají velice široké možnosti použití [1]. Hlavní oblasti použití jsou shrnuty v následujícím přehledu.

- Počítačové vidění a analýza obrazu
  - Kódování a komprese obrazu
  - Segmentace obrazu
  - Analýza obrazu v lékařství
- Analýza a rozpoznávání řeči
  - Rozpoznávání izolovaných slov
  - Rozpoznávání souvislé řeči
  - Identifikace mluvčího
- Zpracování signálů a telekomunikace
- Robotika a návrh elektronických obvodů
  - Robotická paže
  - Navigace robotů
- Biomedicínské aplikace (EEG)
- Zpracování a analýza dat

II. ANALYTICKÁ ČÁST

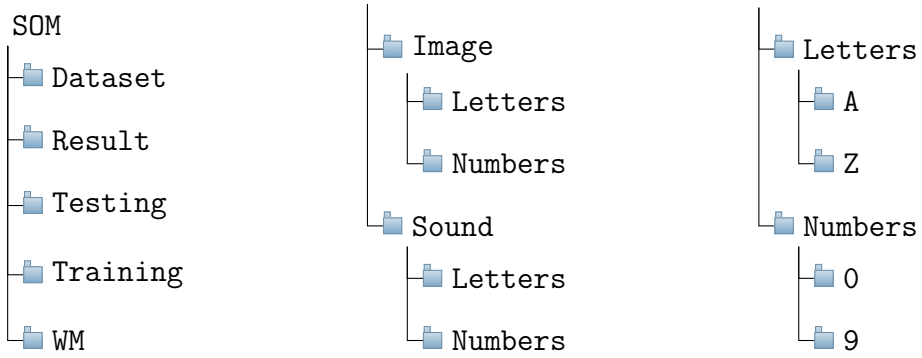
3 Příprava dat

Tato část se zabývá návrhem a zpracováním metodiky přípravy dat. Vzhledem k tomu, že naším cílem je pomocí SOM klasifikovat obrazová a zvuková data, je nutné si vytvořit obrazový a zvukový dataset. Takto získaná data následně vhodným způsobem transformujeme tak, abychom je mohli použít k trénování a testování SOM.

Dataset rozdělíme na trénovací a testovací v uživatelsky definovaném poměru. Vektory trénovacího datasetu, získané transformací originálních dat obrazu nebo zvuku, budou uloženy do souboru *training.vectors* v adekvátním adresáři. Vektory testovacích datasetů, získané transformací originálních dat obrazu nebo zvuku, budou uloženy do souboru *x_ testing.vectors* v adresáři dané třídy, kde x odpovídá znaku dané třídy. Čili pro klasifikaci čísel $x \in \{0-9\}$ a pro klasifikaci písmen $x \in \{A-Z\}$.

3.1 Adresářová struktura

Do uživatelem specifikovaného umístění se automaticky vytvoří adresáře (Obr. 3.1 (a)) a do každého z těchto adresářů se vytvoří další podadresáře (Obr. 3.1 (b)). Do podadresářů *Image* a *Sound* v adresáři *Dataset*, se navíc vytvoří adresáře A–Z a 0–9 (Obr. 3.1 (c)), kam si uložíme originální datové soubory.



(a) (b) (c)

Obr. 3.1 Adresářová struktura

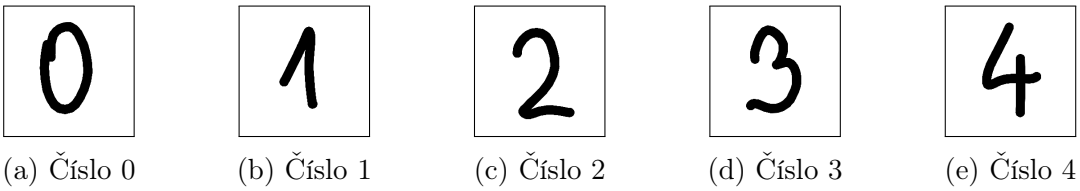
- Adresář **Dataset** k ukládání originálních obrazových a zvukových soubory.
- Adresář **Result** k ukládání váhové matice.
- Adresář **Testing** k ukládání transformovaných testovacích souborů.
- Adresář **Training** k ukládání transformovanných trénovacích souborů.
- Adresář **WM** k ukládání finální váhové matice.

3.2 Příprava obrazových dat

Obrazová data si vytvoříme pomocí mobilní aplikace **Dataset Creator** (Obr. 4.2) kreslením na obrazovku mobilního zařízení pomocí prstu. Pro každý znak, číslo nebo písmeno, si vytvoříme požadované množství dat. Pro účely klasifikace obrazu čísel a písmen pomocí SOM, použijeme dataset vytvořený autorem této práce, který má 450 obrazů pro každý znak. Celkem tedy 4500 obrazů čísel a 11700 obrazů písmen. Obrazová data, nebo-li bitmapová grafika, budou ukládána v bezeztrátovém formátu PNG v rozlišení 256×256 pixelů. V aplikaci **SOMGUI** (Kap. 4.4) je možné obrazová data vhodně oříznout na požadované rozměry a tak si následně zvolit finální rozměry obrazových dat, platných pro celý dataset.

3.2.1 Dataset čísel 0–9

Dataset čísel (Obr. 3.2) obsahuje čísla 0–9 psaná arabskými číslicemi. Máme tedy dataset obsahující obrazy pro 10 znaků čísel.



(a) Číslo 0 (b) Číslo 1 (c) Číslo 2 (d) Číslo 3 (e) Číslo 4

(f) Číslo 5 (g) Číslo 6 (h) Číslo 7 (i) Číslo 8 (j) Číslo 9

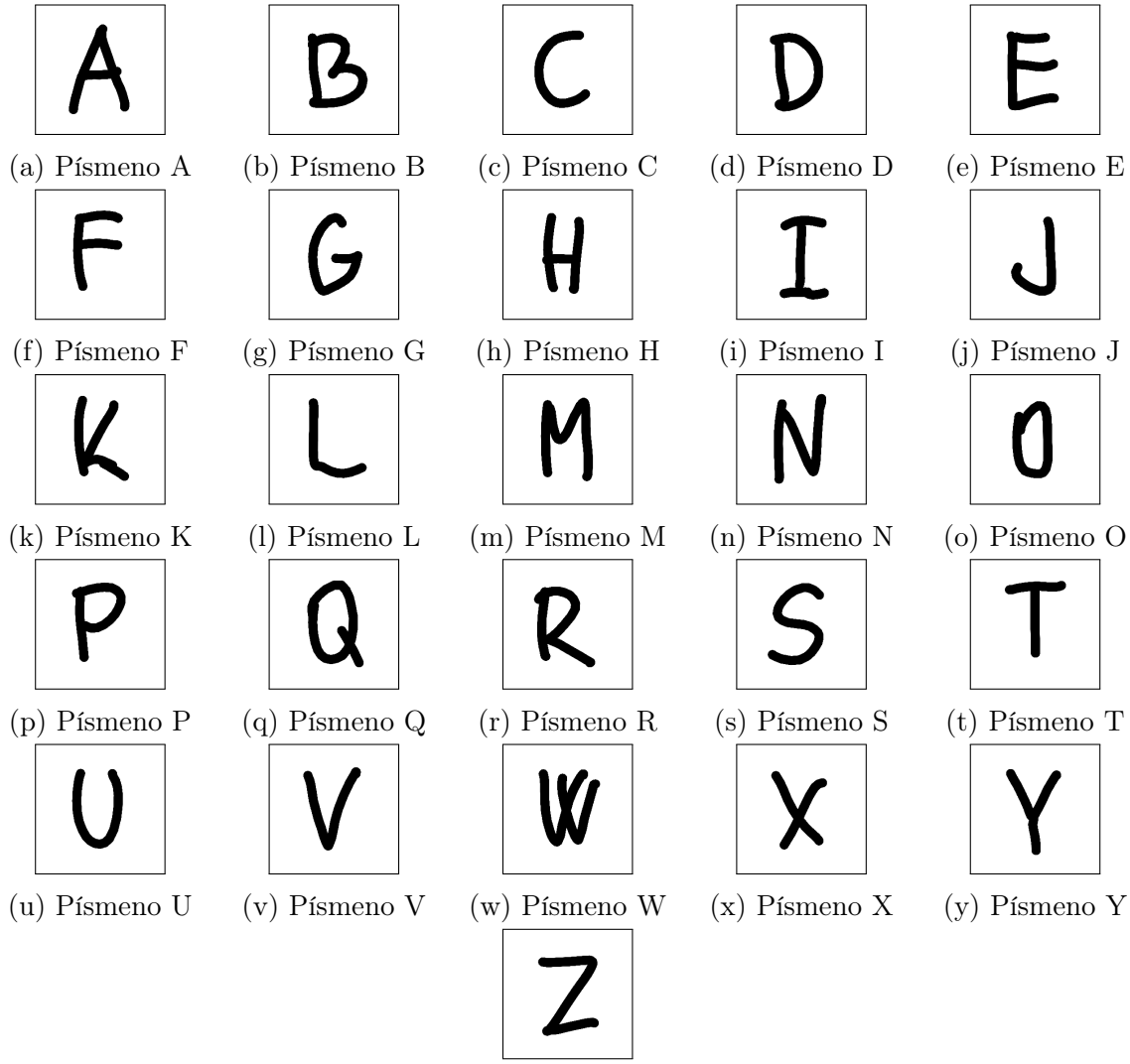
Obr. 3.2 Obrazy čísel 0–10

3.2.2 Dataset písmen A–Z

Dataset písmen (Obr. 3.3) obsahuje písmena anglické abecedy A–Z psaná latinkou. Máme tedy dataset obsahující obrazy pro 26 znaků písmen, který neobsahuje písmena s diakritikou a neobsahuje písmeno *Ch*, což je specifikum českého jazyka a v kontextu této práce bychom jej považovali za dva znaky.

3.2.3 Transformace obrazových dat

Obraz se skládá z jednotlivých pixelů a je specifikován svým rozlišením, čili šířkou a výškou, a barvou daného pixelu. Pro účely této práce pracujeme s černobílým obrazem, kdy pixel obrazu může být pouze černý nebo bílý. Počet pixelů obrazu závisí na



(a) Písmeno A (b) Písmeno B (c) Písmeno C (d) Písmeno D (e) Písmeno E

(f) Písmeno F (g) Písmeno G (h) Písmeno H (i) Písmeno I (j) Písmeno J

(k) Písmeno K (l) Písmeno L (m) Písmeno M (n) Písmeno N (o) Písmeno O

(p) Písmeno P (q) Písmeno Q (r) Písmeno R (s) Písmeno S (t) Písmeno T

(u) Písmeno U (v) Písmeno V (w) Písmeno W (x) Písmeno X (y) Písmeno Y

(z) Písmeno Z

Obr. 3.3 Obrazy písmen A–Z

jeho rozlišení. Obraz si můžeme představit jako matici $N \times M$, kde N je počet sloupců (šířka obrazu) a M je počet řádků (výška obrazu).

Obraz je potřeba transformovat z bitmapové grafiky na vektor V.

$$V = (v_1, v_2 ... v_n), v_i \in (0, 1)$$

Délka vektoru V je rovna $N \times M$, kde každá složka může nabývat hodnot $\{0, 1\}$. Pokud má tedy pixel černou barvu, ve vektoru V je zapsaná hodnota 1. Pokud má pixel barvu bílou, zapsaná hodnota je 0. Délka vektoru V pro obraz o rozlišení 128 bude tedy 16384.

Jak již bylo zmíněno, lze obrazová data vhodným způsobem oříznout a pracovat tak následně s vektory menší délky. Při ořezávání obrazu nám jde o to, abychom ořízli obraz tak, abychom se zbavili prázdných ploch (typicky na okrajích obrazu) a zachovali jen signifikantní data.

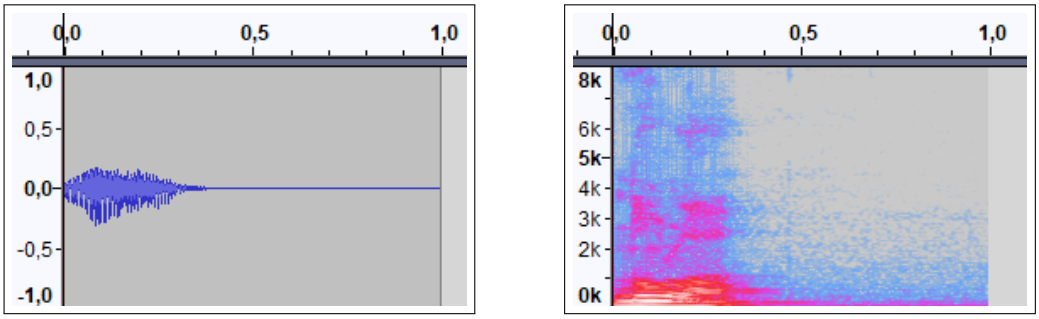
3.3 Příprava zvukových dat

Zvuková data si vytvoříme pomocí aplikace **SOMGUI** (Obr. 4.1). Pro každý znak, číslo nebo písmeno, si vytvoříme požadované množství dat. Pro účely klasifikace zvuku čísel a písmen pomocí SOM, použijeme dataset namluvený autorem této práce, který má 330 zvuků pro každý znak. Celkem tedy 3300 zvuků čísel a 8580 zvuků písmen.

Obrázek (Obr. 3.4) ukazuje průběh zvukového signálu a spektrogram pro číslo nula, získané pomocí programu Audacity. Délka zvukového souboru je jedna sekunda (1000 ms) a zobrazuje se na ose X. Osa Y zobrazuje u spektrogramu hodnoty v rozsahu 0–8 kHz.

Pro všechny znaky si lze stanovit maximální dobu trvání zvuku a zvolit si optimální dobu trvání pro všechny zvukové soubory. Můžeme tedy zvolit dobu trvání 500 ms a zvukové soubory při zpracování zkrátit na tuto hodnotu. Jak si totiž můžeme všimnout na obrázku (Obr. 3.4), zhruba po 500 ms již nemáme k dispozici signifikantní data, ale pouze nahrané ticho po vysloveném znaku.

Projdeme si tedy dataset zvuku a pro každého typického představitele jedné třídy detekujeme konec vysloveného znaku. Tyto představitele tříd následně porovnáme a vezmeme z nich nejdelší dobu trvání a tuto hodnotu stanovíme jako dobu trvání zvuku a při zpracování datasetu soubory na tuto hodnotu zkrátíme.

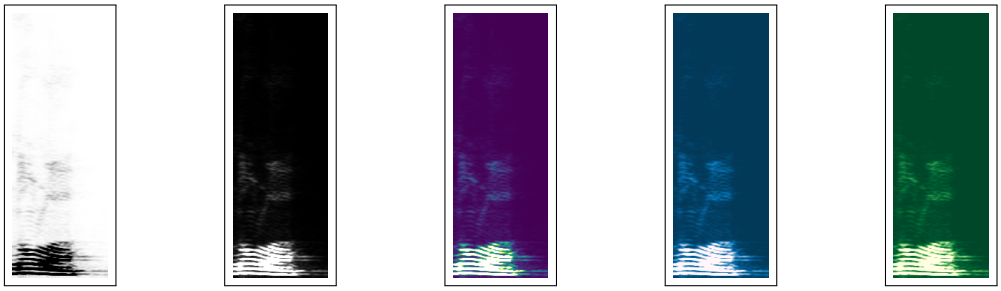


(a) Průběh zvukového signálu (b) Spektrogram

Obr. 3.4 Zvukový soubor

Spektrogram se typicky zobrazuje pomocí teplotní mapy. Obrázek (Obr. 3.5) ukazuje použití teplotních map, pro číslo nula s délkou 500 ms a v rozsahu 0–8 kHz, které používá knihovna .NET Spectrogram[1)]. V této práci použijeme teplotní mapu Inverted (Obr. 3.5 (a)).

[1)] https://github.com/swharden/Spectrogram



(a) Inverted (b) Grayscale (c) Viridis (d) Blue (e) Green

Obr. 3.5 Teplotní mapa

3.3.1 Výslovnost znaků

Tabulka (Tab. 3.1) ukazuje použitou výslovnost čísel. Tabulka (Tab. 3.2) ukazuje použitou výslovnost písmen. Je samozřejmě možné používat jinou výslovnost (jeden/jedna/raz, dva/dvě), ale v této práci použijeme uvedené výslovnosti. Dále je možné různé výslovnosti téhož znaku přidat do datasetu a po natrénování namapovat na daný znak.

Tab. 3.1 Výslovnost čísel

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
nula	jedna	dva	tři	čtyři	pět	šest	sedm	osm	devět

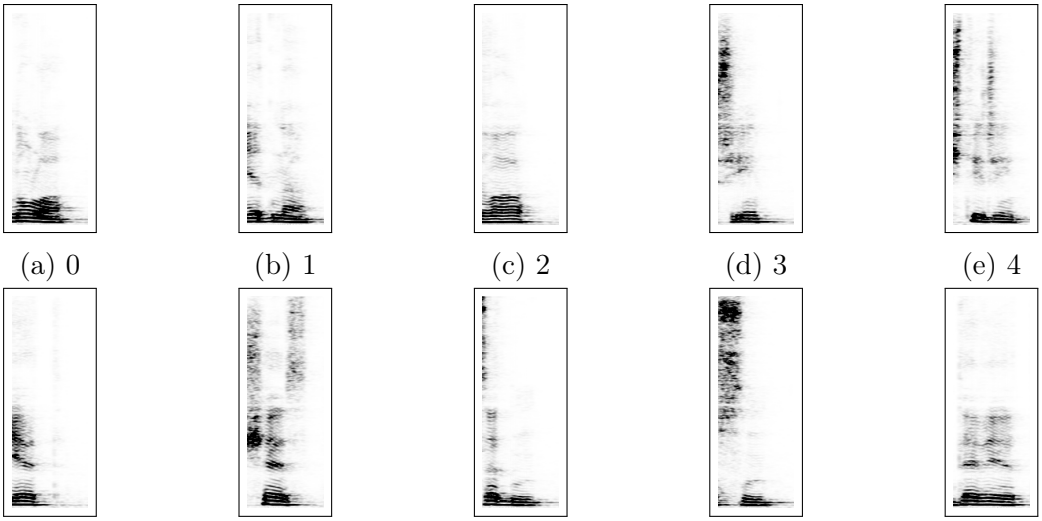
Tab. 3.2 Výslovnost písmen

A	á
B	bé
C	cé
D	dé
E	é
F	ef
G	gé
H	há
I	í
J	jé
K	ká
L	el
M	em

N	en
O	ó
P	pé
Q	kvé
R	er
S	es
T	té
U	ú
V	vé
W	dvojité vé
X	iks
Y	ypsilon
Z	zet

3.3.2 Dataset čísel 0–9

Dataset namluvených zvuků čísel (Obr. 3.6) obsahuje zvuky čísel 0–9. Máme tedy dataset obsahující zvuky pro 10 znaků čísel.



(a) 0 (b) 1 (c) 2 (d) 3 (e) 4

(f) 5 (g) 6 (h) 7 (i) 8 (j) 9

Obr. 3.6 Spektrogramy čísel 0–10

3.3.3 Dataset písmen A–Z

Dataset namluvených zvuků písmen (Obr. 3.7) obsahuje zvuky písmen anglické abecedy A–Z. Máme tedy dataset obsahující zvuky pro 26 znaků písmen, který neobsahuje písmena s diakritikou a neobsahuje písmeno *CH*.

3.3.4 Transformace zvukových dat

Zvukové soubory musíme transformovat do formy použitelné k jejich rozpoznávání pomocí SOM. K tomu se používá spektrogram[2)](hlasový otisk), což je vizuální znázornění spektra z frekvencí signálu. Spektrogram je tedy vizualizací zvukového soubor, formátem je graf se dvěma osami (čas a frekvence) a třetím rozměrem je amplituda konkrétní frekvence v určitém čase.

Transformace zvuku na spektrogram používá FFT[3)](rychlá Fourierova transformace). Pro tvorbu spektogramu použijeme knihovnu .NET Spectrogram[4)], jehož autorem je Scott W Harden, která nám umožní transformovat zvukové soubory do formy, kterou je již možno použít pro klasifikaci zvuku.

[2)]https://cs.qwe.wiki/wiki/Spectrogram
[3)]https://cs.wikipedia.org/wiki/Rychlá_Fourierova_transformace
[4)]https://github.com/swharden/Spectrogram

(a) A (b) B (c) C (d) D (e) E (f) F (g) G (h) H

(i) I (j) J (k) K (l) L (m) M (n) N (o) O (p) P

(q) Q (r) R (s) S (t) T (u) U (v) V (w) W (x) X

(y) Y (z) Z

Obr. 3.7 Spektrogramy písmen A–Z

Vstupem pro knihovnu je zvukový soubor nebo zvukový stream a výstupem je obraz spektrogramu [10], který převedeme na vektor a použijeme trénování SOM a následně k vlastní klasifikaci.

Spektrogram je potřeba transformovat na vektor.

$$V = (v_1, v_2 ... v_n), v_i \in \langle 0, 255 \rangle$$

Délka vektoru V je rovna $N \times M$ a odpovídá rozlišení získaného obrazu spektrogramu, kde každá složka může nabývat hodnot z intervalu $\langle 0, 255 \rangle$.

III. PROJEKTOVÁ ČÁST

4 Implementace

Tato část se zabývá implementací aplikace pro SOM. Implementujeme několik aplikací. Mobilní aplikace slouží ke tvorbě obrazového datasetu a k vlastnímu testování SOM na úlohách rozpoznávání obrazu. Desktopová aplikace slouží k trénování a testování SOM (pro obraz i zvuk) a tvorbě zvukového datasetu a k vlastnímu testování SOM na úlohách rozpoznávání zvuku. V rámci této práce je vytvořena aplikace SOMGUI (zajištuje tranformace datasetů, trénování, testování SOM a klasifikaci zvuku), aplikace SOM (zajišťuje klasifikaci obrazu) a aplikace Dataset Creator (pro tvorbu obrazových dat). Další podrobnosti jsou uvedeny v příslušných kapitolách. Mobilní aplikace pro operační sytém Android jsou určeny pro zařízení s šířkou displeje 1080 pixelů.

4.1 Prerekvizity

4.1.1 Software

Použité vývojové prostředí, verze běhového prostření .NET, vývojové prostředí pro programování aplikací na GPU, verze ovladače GPU a další software použitý při vlastním vývoji.

- Windows 10 Pro - verze 2004
- Microsoft Visual Studio Community 2019 - verze 16.6.5
- .NET Framework 4.7.2
- NVIDIA CUDA Toolkit - verze 11.0
- Ovladač grafické karty - verze 451.67
- ROG RAMDisk - verze 2.03.00
- Audacity 2.3.3
- Equalizer APO 1.2.1
- Peace 1.5.4.3

4.1.2 Hardware

K nahrávání zvukových dat použijeme USB mikrofon s vysokou přesností zvuku a s nízkou hladinou šumu na pozadí. Při použití USB mikrofonu může dojít k situaci, že zvuk bude málo hlasitý a je třeba použít další software[1)] (Equalizer APO, Peace), který tuto chybu napravuje. Doporučuje se nastavit hodnotu zesílení na +15 dB.

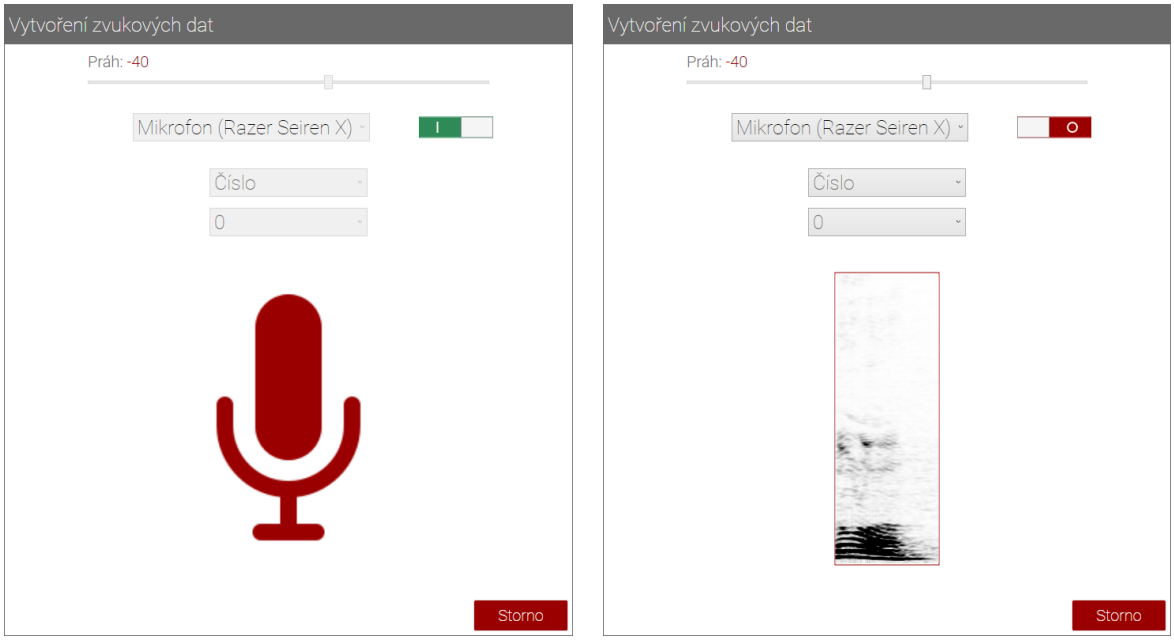
[1)] https://www.pcsteps.com/15528-windows-10-driver-causes-low-volume-usb-microphone/

Grafickou kartu GeForce RTX 2080 Ti použijeme k paralelizaci výpočtů s využitím jejich 4352 CUDA jader a tím ke zrychlení doby běhu algoritmu SOM. Míra zrychlení výpočtů závisí na použité grafické kartě a procesoru PC sestavy. Na základě vlastního testování autorem této práce lze říci, že výpočet bude 10–15 krát rychlejší na grafické kartě.

- Mikrofon Razer Seiren X, Mercury Edition
- MSI NVIDIA GeForce RTX 2080 Ti VENTUS 11G OC

4.2 Vytvoření zvukových dat

Nahrávání zvukových dat (Obr. 4.1) je součástí aplikace **SOMGUI** . Nejprve vybereme vstupní zvukové zařízení a následně vybereme, jestli zda nahrajeme jako zvuk písmeno nebo číslo. Zapnutí vstupního zvukového zařízení potvrdíme posunutím spínače do pozice 1. Nyní již můžeme nahrávat zvuk daného znaku.



(a) Nastavení prahu, výběr zvukového zařízení a výběr znaku

(b) Spektrogram nahraného zvukového souboru

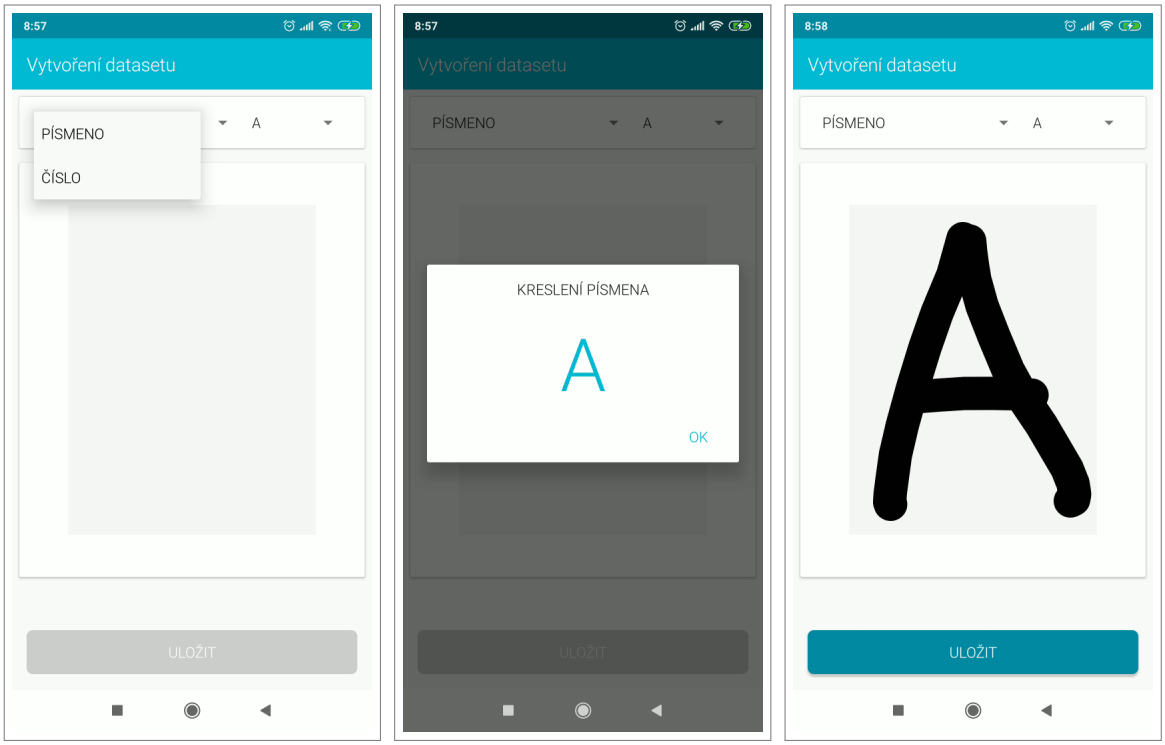
Obr. 4.1 Nahrávání zvukových dat

Doba trvání nahrávání je 1000 ms. Po uplynutí doby nahrávání je zvukový soubor uložen. Nahrávání se automaticky spustí, jakmile je detekovaný zvuk ze vstupního zvukového zařízení.
Červená ikona mikrofonu signalizuje, že je nahráváno ze vstupního zvukového zařízení. Šedá ikona signalizuje, že zvuk nebyl detekovaný.

Zvukové soubory jsou nahrávány do vybraného umístění do složky *Dataset* > *Sound*.

4.3 Aplikace DATASET CREATOR

Aplikace **Dataset Creator** (Obr. 4.2) pro vytvoření obrazových dat je naprogramována pomocí Xamarin.Android v jazyce C#. Jedná se o aplikaci pro mobilní telefony s operačním systémem Android 7.0 a vyšším. Důvodem použití mobilní aplikace pro tvorbu obrazových dat bylo snadné kreslení znaků pomocí prstu, oproti kreslení znaků pomocí myši na PC. Aplikace má jednoduché ovládání. Nejprve vybereme, jestli na-



(a) Výběr typu znaku (b) Výběr znaku (c) Kreslení znaku

Obr. 4.2 Aplikace DATASET CREATOR

kreslíme písmeno nebo číslo, dále dle předchozí volby vybereme požadovaný znak A–Z pro kreslení písmen nebo znak 0–9 pro kreslení čísel. Následně již můžeme prstem nakreslit požadovaný znak. Po stisknutí tlačítka **ULOŽIT** dojde k uložení obrazu do souborového systému mobilního telefonu.

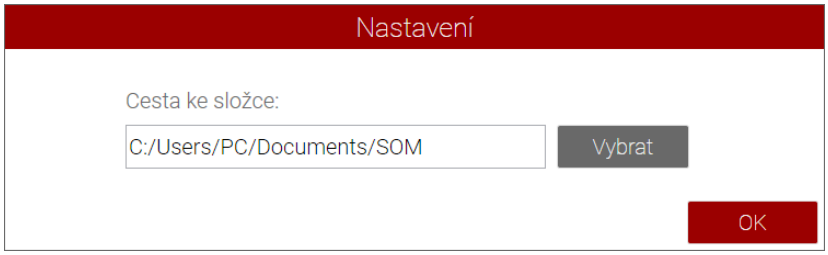
Obrazové soubory jsou ukládány do složky *Vnitřní úložiště* > *Pictures* > *Dataset*. Po vytvoření obrazových dat si tato data zkopírujeme do PC do složky *Dataset* > *Images* ve vybraném umístění, abychom mohli získaná data použít v další aplikaci.

4.4 Aplikace SOMGUI

Aplikace je naprogramována v jazyce C# s frameworkem WPF. Slouží k vytváření zvukového datasetu, transformaci obrazových a zvukových dat, trénování a testování

SOM a nakonec k vlastní klasifikaci zvuku na úlohách rozpoznávání zvuku písmen a čísel.

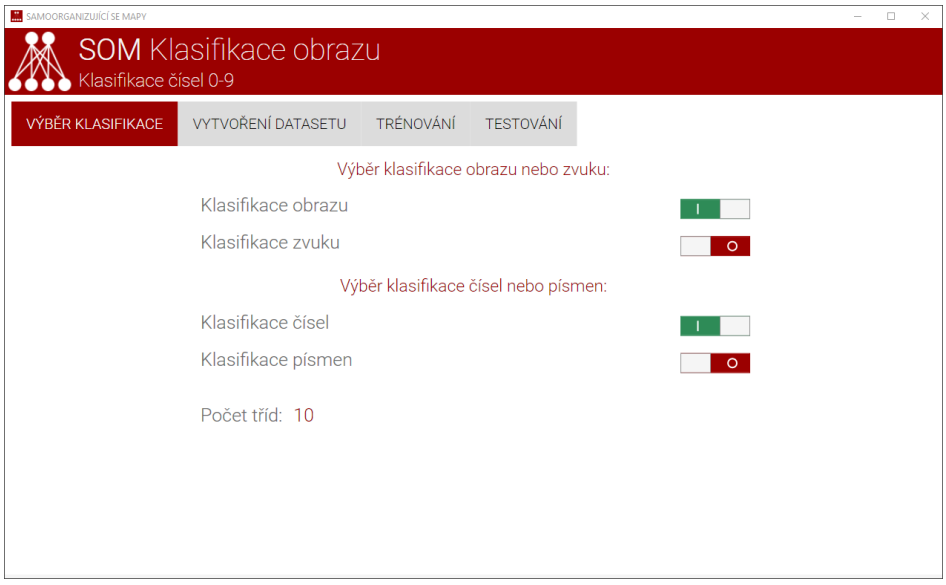
Při prvním spuštění aplikace se zobrazí dialogové okno s výběrem umístění na pevném disku (Obr. 4.3), kam budou ukládány soubory potřebné k práci s aplikací (obrazová/zvuková data, transformovaná data, finální váhové matice).



Obr. 4.3 Výběr adresáře

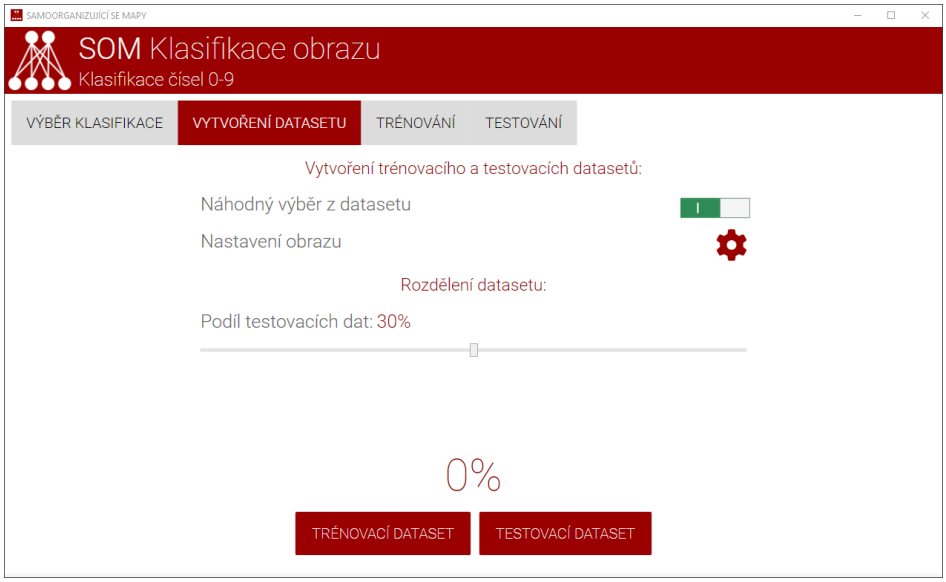
Na umístění víceméně nezáleží, je možné ponechat výchozí adresář v dokumentech PC. Pokud ale chceme použít ROG RAMDisk, jako autor této práce, změníme výchozí umístění na disk namapovaný do RAM pomocí aplikace RAMDisk. Princip RAMDisku spočívá k namapování operační paměti jako další pevný disk. Operace čtení a zápisu z paměti jsou rychlejší ve srovnání s SSD nebo s plotnovým pevným diskem. Nevýhodou je, že po odpojení od napájení se data z operační paměti smažou. Musíme si tedy soubory se kterými chceme později pracovat zálohovat na jiné umístění.

Po výběru umístění se nám aplikace nabídne výběr klasifikace. Pomocí přepínačů si zvolíme druh klasifikace (obraz nebo zvuk) a typ (čísla nebo písmena). V aplikaci se navigujeme podle záložek.



Obr. 4.4 Výběr klasifikace

Po výběru klasifikace přepneme na záložku *Vytvoření datasetu*, kde si můžeme nastavit podíl testovacích dat, náhodný výběr z datasetu a zejména nastavení obrazu. Dále zde máme tlačítka pro vytvoření trénovacího a testovacích datasetů.



Obr. 4.5 Vytvoření datasetu

V případě klasifikace zvuku se zobrazí tlačítko *VYTVOŘENÍ ZVUKOVÝCH DAT*, které po kliknutí zobrazí okno s nahráváním zvukových dat (Obr. 4.1).

Nastavení obrazu (Obr. 4.6) je platné pro celý dataset, čili všechny obrazové soubory budou před transformací oříznuty na zvolené rozlišení. Rozlišení je v násobcích čísla 8 z důvodu velikosti transformovaného vektoru, který by měl mít tuto velikost při provádění výpočtu na GPU kvůli organizaci vláken při výpočtu.

(a) Obraz bez oříznutí

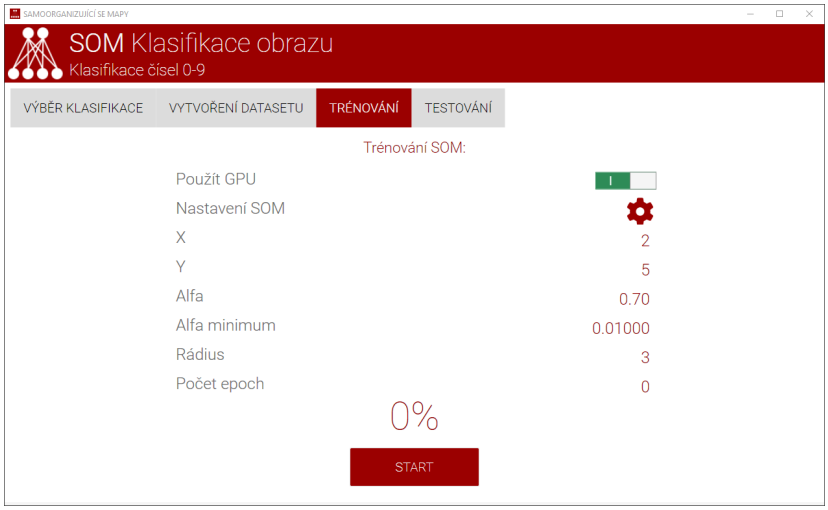
(b) Oříznutí obrazu

Obr. 4.6 Nastavení obrazu

Pomocí šipek na klávesnici si můžeme měnit zobrazené znaky jednotlivých tříd datasetu. SOM tedy můžeme trénovat a testovat pro různé rozlišení obrazových dat. Pro

každé rozlišení můžeme dostat jiné výsledky testování, jelikož pracujeme s rozdílými vektory, tzn. s vektory s různou velikostí.

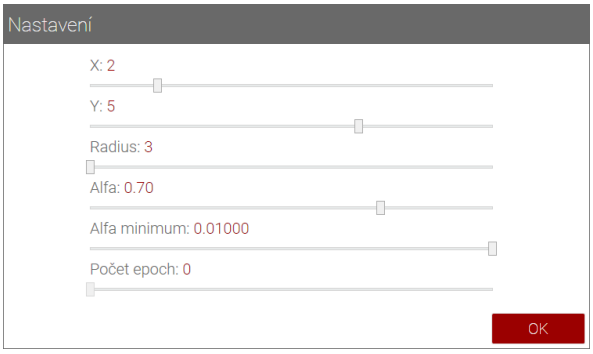
Při změně rozlišení je nutné vytvořit nové trénovací a testovací datasety. Dalším krokem, který je společný pro klasifikaci obrazu i zvuku, je trénování SOM, umístěné na záložce *TRÉNOVÁNÍ* (Obr. 4.6), kde nastavujeme parametry trénování (konfiguraci SOM) a pomocí přepínače volíme, zda výpočet poběží na GPU nebo CPU.



Obr. 4.7 Trénování SOM

Nastavíme velikost mřížky, tzn. počet sloupců a počet řádků, faktor učení α, poloměr okolí R a ukončovací podmínku α_{min} nebo počet epoch (Obr. 4.8). Počet epoch lze nastavit pouze když $\alpha_{min} = 0$.

α_{min} je zdola omezena na hodnotu $0,00001$ a shora omezena na hodnotu $0,01$. Maximální počet epoch je omezen na 10000 a minimální a maximální hodnota poloměru okolí se automaticky nastaví, v intervalu $R_{max}/2 \leq R \leq R_{max}$, podle velikosti mřížky, čili podle hodnot X a Y.

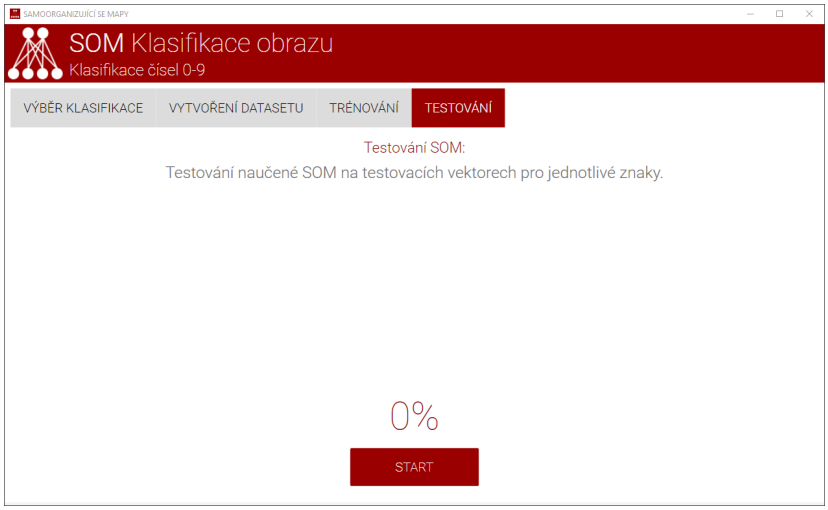


Obr. 4.8 Nastavení SOM

Po nastavení konfigurace SOM trénování spustíme tlačítkem *START*. Po dokončení testování se přepneme na záložku *TESTOVÁNÍ* (Obr. 4.9) a vlastní testování spustíme

tlačítkem *START*.

Testování SOM funguje tak, že aplikace všechny testovací vektory dané třídy postupně prochází (podle zvoleného typu a druhu klasifikace) a předkládá SOM ke klasifikaci a pamatuje si do kterého neuronu výstupní vrstvy byl daný vektor klasifikován a aplikace si ukládá tyto informace do tabulky.



Obr. 4.9 Testování SOM

Po dokončení testování se nám zobrazí okno s výsledkem testování. Výsledek testování SOM je prezentován v tabulce na obrázku (Obr. 4.10).

Obr. 4.10 Výsledek testování SOM

Sloupce prezentují neurony výstupní vrstvy a řádky jednotlivé znaky. V tabulce se v buňkách zobrazuje počet klasifikovaných vektorů daného znaku v daném výstupním neuronu. Tučným písmem zobrazuje fakt, že daná buňka obsahuje ve sloupci nejvyšší počet klasifikací a zapíše daný znak do mapovací tabulky pro daný index.

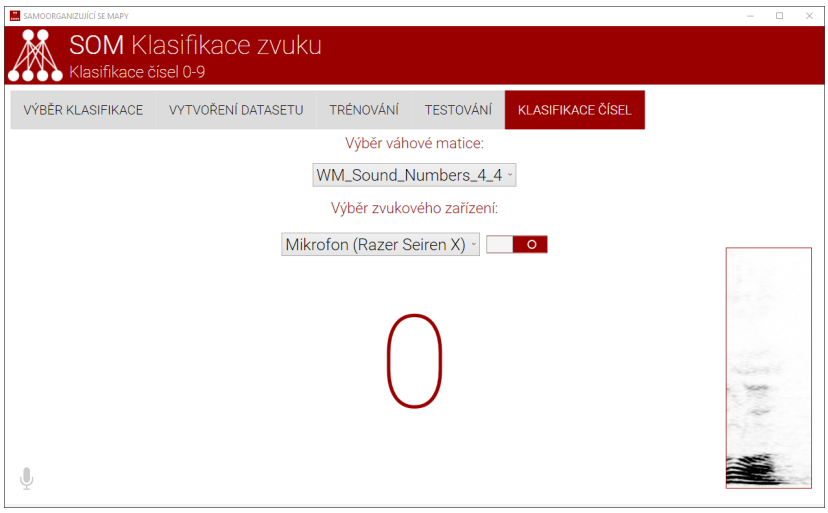
Naším cílem je tedy natrénovat SOM tak, aby při testování byly všechny klasifikace daného znaku ve stejném výstupním neuronu a zároveň aby v tomto neuronu nebyly klasifikovány další znaky.

Např. výsledek testování (Obr. 4.10) mám ukazuje, že všechny vektory obrazu pro třídy $1 - 9$ jsou klasifikovány do samostatného výstupního neuronu. Pro znak nula je 130 vektorů klasifikováno do samostatného neuronu s indexem 2 a 5 vektorů do neuronu s indexem 0, který obsahuje všechny vektory znaku šest. V tomto případě je počet špatně klasifikovaných vektorů znaku nula nízký a situaci nijak řešit nemusíme.

Pokud máme po testování v tabulce vyšší počty špatně klasifikovaných znaků, musíme změnit konfiguraci SOM a začít znovu od trénování SOM. Typicky pokud k jednomu neuronu výstupní vrstvy jsou přiřazeny vektory více tříd. Naopak pokud máme vektory jedné třídy rozmístěné samostatně do více výstupních neuronů, tak tuto situaci nepovažujeme za špatnou klasifikaci.

Po stisku tlačítka *Uložit* dojde k uložení finální váhové matice (*.som*), ve vybraném umístění, do složky *WM* > *Images* > *Numbers*, v případě klasifikace čísel nebo do složky *WM* > *Images* > *Letters*, v případě klasifikace písmen a obdobně pro klasifikaci zvuku.

Tuto uloženou finální váhovou matice, v případě klasifikace obrazu, můžeme následně použít v aplikaci **SOM** , kdy si matici uložíme do úložiště mobilního zařízení.



Obr. 4.11 Klasifikace zvuku

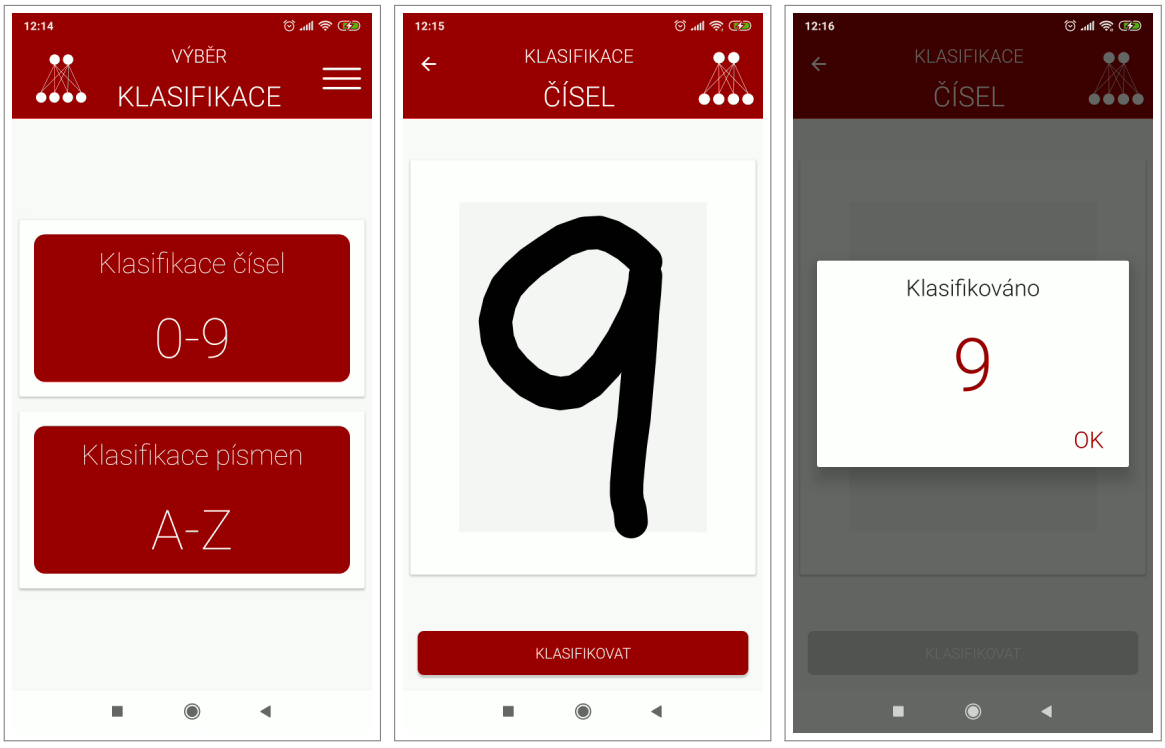
Klasifikaci zvuku provádíme pomocí aplikace *SOMGUI* (Obr. 4.11). Nejprve vybe-

reme finální váhovou matici z uložených. Následně zvolíte vstupní zvukové zařízení na nahrávání zvuku. Pak stačí pomocí přepínače nebo mezerníku klávesnice zapnout nahrávání zvuku ze vstupního zařízení a pak vyslovíme znak. Aplikace obsahuje detekci ticha a při vyslovení znaku dojde k nahrání zvuku, který je pak klasifikován.

Výsledek klasifikace zvuku vidíme v aplikaci zároveň se spektrogramem zaznamenaného zvuku. Po každé klasifikaci dojde k vypnutí nahrávání vstupního zvukového zařízení.

4.5 Aplikace SOM

Aplikace **SOM** (Obr. 4.12) je naprogramována pomocí Xamarin.Android v jazyce C#. Jedná se o aplikaci pro mobilní telefony s operačním systémem Android 7.0 a vyšším. Aplikace slouží k otestování SOM na úlohách rozpoznávání obrazu, kdy si můžeme vybrat, zda chceme klasifikovat čísla nebo písmena.



(a) Výběr klasifikace (b) Klasifikace čísel (c) Výsledek klasifikace

Obr. 4.12 Aplikace SOM

Pomocí tlačítka si vybereme druh klasifikace a prstem nakreslíme znak. Následně zmáčkneme tlačítko *KLASIFIKOVAT* a aplikace nám zobrazí výsledek klasifikace.

Další možností je načtení souboru *.som*, ze souborového systému mobilního zařízení, pomocí menu aplikace a otestovat si další váhové matice vytvořené programem *SOMGUI*.

5 Testování SOM

V této části testujeme klasifikaci obrazu a zvuku nejprve jako součást procesu vytváření finální váhové matice pomocí programu SOMGUI. Výstupem bude tabulka přiřazení jednotlivých testovacích vektorů do výstupních neuronů a mapovací tabulka přiřazení znaku do výstupních neuronů, která udává pro daný neuron výstupní vrstvy znak reprezentující danou třídu.

Testování provedeme s danou konfigurací K dle nastavení v aplikaci. Pro klasifikaci obrazu nás bude zajímat i rozlišení vstupního obrazu. Hledáme tedy finální váhovou matici a nejlepším rozdělení vektorů do výstupních neuronů.

Pro tuto matici následně provedeme testování na neznámých datech, kdy klasifikaci obrazu otestujeme pomocí mobilní aplikace *SOM* a pro klasifikaci zvuku pomocí aplikace *SOMGUI*. Každý znak 30-krát otestujeme na úlohách rozpoznávání obrazu a zvuku a výsledky shrneme do tabulky.

Při procesu trénování obrazu můžeme nastavovat rozlišení obrazu a konfiguraci K. Čili pro různá rozlišení obrazu můžeme měnit rozměr mřížky, poloměr okolí nebo ukončovací podmínku. Při procesu trénování zvuku můžeme nastavovat pouze konfiguraci K. Takto natrénovanou SOM následně testujeme a snažíme se nalézt takovou konfiguraci, která dává nejlepší výsledky ve smyslu přiřazení vektorů k neuronům výstupní vrstvy.

Cílem tedy je přiřadit nejlépe všechny vektory jedné třídy do jednoho výstupního neuronu. Takovéto přiřazení je ale pouze idealizované. Při testování SOM se nám nemusí ve všech případech povést takovéto stoprocentní přiřazení do neuronů výstupní vrstvy.

Snažíme se tedy minimalizovat počet nesprávných umístění vstupních vektorů dané třídy do výstupních neuronů, mimo jejich cílový neuron, kam je přiřazena většina vektorů třídy. Pokud je počet těchto vektorů zanedbatelný, nalezli jsme kvalitní řešení. Jinak se musíme vrátit k procesu trénování a upravit konfiguraci SOM. Kvalitních řešení můžeme nalézt samozřejmě více, záleží pak na nás, které si zvolíme.

Nejdůležitějším požadavkem pro kvalitní řešení jsou vstupní data, kterých musí být dostatečné množství a která musejí obsahovat typického představitele dané třídy a podobné jedince blízké hranicím mezi jednotlivými třídami. Např. pro klasifikaci obrazu je dobré mít nakreslená písmena nebo čísla s posunem (vlevo, vpravo, nahoru, dolu) od typického představitele třídy a s jiným sklonem písma.

5.1 Klasifikace obrazu

5.1.1 Klasifikace čísel 0–9

V této části testujeme klasifikaci obrazu čísel 0–9 při různém rozlišení obrazu.

Test 1: Klasifikace obrazu čísel v konfiguraci K=(2×5, R=3, α=0,7, α_{min}=0,01) a rozlišení obrazu 128 × 128.

Tab. 5.1 Klasifikace obrazu - čísla 0–9 K=(2×5, R=3, α=0,7, α_{min}=0,01)

Výstupní vrstva										
Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	5	0	0	0	0	0	1	0	0	**129**
1	0	**135**	0	0	0	0	0	0	0	0
2	0	0	0	0	**134**	0	0	1	0	0
3	0	0	0	0	0	0	**65**	**70**	0	0
4	0	135	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	**134**	0
6	**135**	0	0	0	0	0	0	0	0	0
7	0	0	**135**	0	0	0	0	0	0	0
8	0	0	0	0	0	**135**	0	0	0	0
9	0	2	0	**133**	0	0	0	0	0	0

Mapovací tabulka										
Znak	6	1	7	9	2	8	3	3	5	0

Výsledek testu 1: Tato konfigurace dává špatné výsledky, znaky 1 a 4 jsou klasifikovány do stejného výstupního neuronu.

Test 2: Klasifikace obrazu čísel v konfiguraci K=(2×5, R=3, α=0,7, α_{min}=0,01) a rozlišení obrazu 78 × 120.

Tab. 5.2 Klasifikace obrazu - čísla 0–9 K=(2×5, R=3, α=0,7, α_{min}=0,01)

Výstupní vrstva										
Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**131**	0	0	0	0	0	0	0	4	0
1	0	0	0	0	0	0	0	0	0	**135**
2	0	0	**135**	0	0	0	0	0	0	0
3	0	0	0	**135**	0	0	0	0	0	0
4	0	0	0	0	0	0	0	**135**	0	0
5	0	0	0	0	0	0	**135**	0	0	0
6	0	0	0	0	0	0	0	0	**135**	0
7	0	0	0	0	0	**135**	0	0	0	0
8	0	**135**	0	0	0	0	0	0	0	0
9	0	0	0	0	**135**	0	0	0	0	0

Mapovací tabulka										
Znak	0	8	2	3	9	7	5	4	6	1

Výsledek testu 2: Tato konfigurace dává dobré výsledky, jen 4 znaky čísla 0 jsou špatně klasifikovány do stejného výstupního neuronu jako číslo 6.

Test 3: Klasifikace obrazu čísel v konfiguraci K=(2×5, R=3, Epochs=1000) a rozlišení obrazu 78 × 120.

Tab. 5.3 Klasifikace obrazu - čísla 0–9 K=(2×5, R=3, Epochs=1000)

Výstupní vrstva										
Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	0	5	0	0	**130**	0
1	0	0	0	0	**135**	0	0	0	0	0
2	0	0	0	0	0	0	**135**	0	0	0
3	0	0	**135**	0	0	0	0	0	0	0
4	**135**	0	0	0	0	0	0	0	0	0
5	0	**135**	0	0	0	0	0	0	0	0
6	0	0	0	0	0	**135**	0	0	0	0
7	0	0	0	0	0	0	0	**135**	0	0
8	0	0	0	**135**	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	**135**

Mapovací tabulka										
Znak	4	5	3	8	1	6	2	7	0	9

Výsledek testu 3: Tato konfigurace dává dobré výsledky, jen 5 znaků čísla 0 jsou špatně klasifikovány do stejného výstupního neuronu jako číslo 6.

Test 4: Klasifikace obrazu čísel v konfiguraci K=(2×5, R=4, Epochs=1000) a rozlišení obrazu 78 × 120.

Tab. 5.4 Klasifikace obrazu - čísla 0–9 K=(2×5, R=4, Epochs=1000)

Výstupní vrstva										
Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	**130**	5	0	0	0	0	0	0
1	**135**	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	**135**	0	0	0
3	0	0	0	0	**135**	0	0	0	0	0
4	0	0	0	0	0	**135**	0	0	0	0
5	0	**135**	0	0	0	0	0	0	0	0
6	0	0	0	**135**	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	**135**
8	0	0	0	0	0	0	0	0	**135**	0
9	0	0	0	0	0	0	0	**135**	0	0

Mapovací tabulka										
Znak	1	5	0	6	3	4	2	9	8	7

Výsledek testu 4: Tato konfigurace dává dobré výsledky, jen 5 znaků čísla 0 jsou špatně klasifikovány do stejného výstupního neuronu jako číslo 6.

K otestování klasifikace obrazů čísel vybereme výslednou váhovou matici z testu č. 4

a nakopírujeme matici *WM_Image_Numbers_2_5.som* z umístění *SOM > WM > Image > Numbers* do mobilního zařízení a v aplikaci *SOM* načteme.

Nyní otestujeme klasifikaci obrazů čísel v mobilním zařízením tak, že každý ze znaků $0-9$ 30-krát nakreslíme a necháme klasifikovat. Výsledky shrneme v tabulce (Tab. 5.5), kde uvedeme počet úspěšných klasifikací, z celkového počtu pokusů, a jeho procentuální vyjádření.

Tab. 5.5 Testování klasifikace obrazu čísel

Znak	**Počet**	**Úspěšnost**
0	28	93%
1	30	100%
2	27	90%
3	30	100%
4	30	100%
5	30	100%
6	30	100%
7	30	100%
8	26	87%
9	30	100%

5.1.2 Klasifikace písmen A–Z

V této části testujeme klasifikaci obrazu písmen A–Z při různém rozlišení obrazu. Vzhledem k vyššímu počtu neuronů výstupní vrstvy, při testování klasifikace obrazu písmen (26 znaků), budou tabulky jednotlivých testů dané konfigurace součástí přílohy **P I.** (v umístění *Vysledky > Testovani_obrazu_pismen.pdf*) A to především z důvodu velikosti tabulky. U každé tabulky v příloze je uvedeno, ke kterému z testů náleží.

Test 1: Klasifikace obrazu písmen v konfiguraci K=(2×13, R=3, α=0,7, $\alpha_{\min}$=0,01) a rozlišení obrazu 72×120.
Výsledek testu 1: Tato konfigurace dává špatné výsledky, znaky O a Q, F a P jsou klasifikovány do stejného výstupního neuronu.

Test 2: Klasifikace obrazu písmen v konfiguraci K=(5×6, R=3, α=0,7, $\alpha_{\min}$=0,01) a rozlišení obrazu 72×120.
Výsledek testu 2: Tato konfigurace dává dobré výsledky, několik znaků různých tříd je klasifikováno do stejného výstupního neuronu.

Test 3: Klasifikace obrazu písmen v konfiguraci K=(6×6, R=4, Epochs=100) a rozlišení obrazu 72×120.
Výsledek testu 3: Tato konfigurace dává dobré výsledky, několik znaků různých tříd je klasifikováno do stejného výstupního neuronu.

K otestování klasifikace obrazů písmen vybereme výslednou váhovou matici z testu č. 3 a nakopírujeme matici *WM_Image_Letters_6_6.som* z umístění *SOM > WM > Image > Numbers* do mobilního zařízení a v aplikaci *SOM* načteme.

Nyní otestujeme klasifikaci obrazů písmen v mobilním zařízením tak, že každý ze znaků $A - Z$ 30-krát nakreslíme a necháme klasifikovat. Výsledky shrneme v tabulce (Tab. 5.6), kde uvedeme počet úspěšných klasifikací, z celkového počtu pokusů, a jeho procentuální vyjádření.

Tab. 5.6 Testování klasifikace obrazu písmen

Znak	Počet	Úspěšnost
A	30	100%
B	30	100%
C	26	87%
D	30	100%
E	30	100%
F	27	90%
G	27	90%
H	29	97%
I	28	93%
J	30	100%
K	28	93%
L	29	97%
M	26	87%

Znak	Počet	Úspěšnost
N	29	97%
O	26	87%
P	29	97%
Q	28	93%
R	28	93%
S	29	97%
T	30	100%
U	27	90%
V	29	97%
W	30	100%
X	30	100%
Y	28	93%
Z	30	100%

5.2 Klasifikace zvuku

5.2.1 Klasifikace čísel 0–9

Test 1: Klasifikace zvuku čísel v konfiguraci K=(2×5, R=3, Epochs=100).

Tab. 5.7 Klasifikace zvuku - čísla 0–9 K=(2×5, R=3, Epochs=100)

Výstupní vrstva										
Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	0	0	**99**	0	0	0
1	0	0	0	0	**99**	0	0	0	0	0
2	0	0	0	0	0	0	99	0	0	0
3	0	22	0	0	0	**77**	0	0	0	0
4	0	0	**99**	0	0	0	0	0	0	0
5	0	**99**	0	0	0	0	0	0	0	0
6	**77**	22	0	0	0	0	0	0	0	0
7	0	11	0	0	11	0	0	**77**	0	0
8	0	0	0	0	0	0	0	0	**71**	**28**
9	0	0	0	0	99	0	0	0	0	0

Mapovací tabulka										
Znak	6	5	4		1	3	0	7	8	8

Výsledek testu 1: Tato konfigurace dává špatné výsledky, znaky 0 a 2, 1 a 9 jsou klasifikovány do stejného výstupního neuronu.

Test 2: Klasifikace zvuku čísel v konfiguraci K=(3×4, R=2, Epochs=100).

Tab. 5.8 Klasifikace zvuku - čísla 0–9 K=(3×4, R=2, Epochs=100)

Výstupní vrstva												
Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0	0	0	0	**99**	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	**99**	0	0	0	0	0
2	0	0	0	99	0	0	0	0	0	0	0	0
3	0	0	22	0	0	0	0	0	0	0	0	**77**
4	0	0	0	0	**99**	0	0	0	0	0	0	0
5	0	0	**99**	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	**33**	**66**	0	0
7	0	0	0	0	0	0	22	**77**	0	0	0	0
8	**71**	**28**	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	**99**	0

Mapovací tabulka												
Znak	8	8	5	0	4		1	7	6	6	9	3

Výsledek testu 2: Tato konfigurace dává špatné výsledky, znaky 0 a 2 jsou klasifikovány do stejného výstupního neuronu.

Test 3: Klasifikace zvuku čísel v konfiguraci K=(4×4, R=2, Epochs=100).

Tab. 5.9 Klasifikace zvuku - čísla 0–9 K=(4×4, R=2, Epochs=100)

Výstupní vrstva																
Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**99**
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**99**	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	99
3	0	0	22	**77**	0	0	0	0	0	0	0	0	0	0	0	0
4	**99**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	**99**	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	**64**	0	0	**5**	0	0	**30**	0	0	0	0	0
7	0	0	11	0	0	**70**	0	0	0	0	0	**18**	0	0	0	0
8	0	0	0	0	0	0	0	0	**21**	**48**	0	0	**23**	**7**	0	0
9	0	**99**	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Mapovací tabulka																
Znak	4	9	5	3	6	7		6	8	8	6	7	8	8	1	0

Výsledek testu 3: Tato konfigurace dává špatné výsledky, znaky 0 a 2 jsou klasifikovány do stejného výstupního neuronu.

Test 4: Klasifikace zvuku čísel v konfiguraci K=(4×5, R=3, Epochs=100).

Tab. 5.10 Klasifikace zvuku - čísla 0–9 K=(4×5, R=3, Epochs=100)

Výstupní vrstva																				
Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**99**	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	99	0	0	0	0	0	0
3	0	0	0	0	0	**77**	0	0	0	0	0	0	22	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0
6	**8**	0	0	0	0	0	**27**	0	0	0	**3**	**22**	0	0	0	0	**39**	0	0	0
7	0	**18**	0	0	0	0	0	0	0	**70**	0	0	11	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**60**	0	0	**11**	**28**
9	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0	0	0	0	0

Mapovací tabulka																				
Znak	6	7				3	6		9	7	6	6	5	0	4	8	6	1	8	8

Výsledek testu 4: Tato konfigurace dává špatné výsledky, znaky 0 a 2 jsou klasifikovány do stejného výstupního neuronu.

Test 5: Klasifikace zvuku čísel v konfiguraci K=(5×5, R=4, Epochs=400).

Výsledek testu 5: Tato konfigurace dává již kvalitnější výsledky, menší počety

Tab. 5.11 Klasifikace zvuku - čísla 0–9 K=(5×5, R=4, Epochs=400)

Výstupní vrstva																								
Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	**34**	0	0	0	0	0	0	0	0	11	0	0	0	0	**54**	0	0	0	0	0	0
4	0	0	0	0	0	**61**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**38**	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0	0	0	0	0
6	0	**27**	0	0	0	0	0	0	0	0	**9**	**2**	0	0	0	0	0	0	0	0	0	0	0	**39**	**22**
7	0	0	**18**	0	0	0	**33**	0	0	0	0	0	0	4	**33**	0	0	0	0	**11**	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**59**	**11**	0	0	0	**29**	0	0	0	0
9	0	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Mapovací tabulka																									
Znak		6	7		3	4	7		1	9	6	6	2	5	7	8	8	0	3	7	8		4	6	6

znaků 3 a 7 jsou klasifikovány do výstupního neuronu přiřazenému znaku 5.

Test 6: Klasifikace zvuku čísel v konfiguraci K=(5×5, R=4, Epochs=1000).

Tab. 5.12 Klasifikace zvuku - čísla 0–9 K=(5×5, R=4, Epochs=1000)

Výstupní vrstva																									
Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	**99**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	11	0	0	0	0	**54**	0	0	0	0	0	0	0	0	0	0	**34**	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	**38**	0	0	0	0	**61**	0	0	0	0	0	0
5	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	**5**	0	**22**	0	0	0	0	0	0	0	0	0	0	0	0	0	**39**	0	0	**6**	0	0	0	**27**	0
7	0	**33**	0	0	0	0	4	**18**	0	0	0	0	**11**	0	0	0	0	0	0	0	**33**	0	0	0	0
8	0	0	0	0	**28**	0	0	0	0	**71**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**99**	0	0	0	0	0	0	0

Mapovací tabulka																									
Znak	6	7	6		8	1	5	7	2	8		3	7	4	0		6	9	4	6	7		3	6	

Výsledek testu 6: Výsledek pro tuto konfigurace je podobný výsledku přechozího testu.

K otestování klasifikace zvuku čísel vybereme výslednou váhovou matici z testu č. 6. Nyní otestujeme klasifikaci zvuků čísel v aplikaci *SOMGUI* tak, že každý ze znaků $0-9$ 30-krát vyslovíme a necháme klasifikovat. Výsledky shrneme v tabulce (Tab. 5.13), kde uvedeme počet úspěšných klasifikací, z celkového počtu pokusů, a jeho procentuální vyjádření.

5.2.2 Klasifikace písmen A–Z

V této části otestujeme klasifikaci zvuku písmen A–Z. Vzhledem k vyššímu počtu neuronů výstupní vrstvy, při testování klasifikace zvuku písmen (26 znaků), budou tabulky

Tab. 5.13 Testování klasifikace zvuku čísel

Znak	**Počet**	**Úspěšnost**
0	30	100%
1	29	97%
2	27	90%
3	28	93%
4	28	93%
5	30	100%
6	27	90%
7	30	100%
8	30	100%
9	28	93%

jednotlivých testů dané konfigurace součástí přílohy **P I.** (v umístění *Vysledky > Testovani_ zvuku_ pismen.pdf*) A to především z důvodu velikosti tabulky. U každé tabulky v příloze je uvedeno, ke kterému z testů náleží.

Test 1: Klasifikace zvuků písmen v konfiguraci K=(2×13, R=3, α=0,7, $\alpha_{\min}$=0,01).
Výsledek testu 1: Tato konfigurace dává špatné výsledky, vysoký počet různých znaků je klasifikován do stejného výstupního neuronu.

Test 2: Klasifikace zvuků písmen v konfiguraci K=(5×6, R=3, α=0,7, $\alpha_{\min}$=0,01).
Výsledek testu 2: Tato konfigurace dává špatné výsledky, vysoký počet různých znaků je klasifikován do stejného výstupního neuronu.

Test 3: Klasifikace zvuků písmen v konfiguraci K=(6×6, R=4, Epochs=100).
Výsledek testu 3: Tato konfigurace dává špatné výsledky, vysoký počet různých znaků je klasifikován do stejného výstupního neuronu.

Test 4: Klasifikace zvuků písmen v konfiguraci K=(5×7, R=4, Epochs=100).
Výsledek testu 4: Tato konfigurace dává špatné výsledky, vysoký počet různých znaků je klasifikován do stejného výstupního neuronu.

Test 5: Klasifikace zvuků písmen v konfiguraci K=(6×7, R=5, Epochs=300).
Výsledek testu 5: Tato konfigurace dává špatné výsledky, vysoký počet různých znaků je klasifikován do stejného výstupního neuronu.

Test 6: Klasifikace zvuků písmen v konfiguraci K=(6×7, R=5, Epochs=1000).

Výsledek testu 6: Tato konfigurace dává špatné výsledky, vysoký počet různých znaků je klasifikován do stejného výstupního neuronu.

V žádném z testů se nepodařilo rozlišit písmena M a N, důvodem je jejich velice podobná výslovnost, což bude dále rozebráno při zhodnocení výsledků testování. Zároveň některé další dvojice znaků jsou přiřazeny do jednoho výstupního neuronu.

K otestování klasifikace zvuků písmen vybereme výslednou váhovou matici z testu č. 6. Nyní otestujeme klasifikaci zvuků písmen v aplikaci *SOMGUI* tak, že každý ze znaků $A - Z$ 30-krát vyslovíme a necháme klasifikovat. Výsledky shrneme v tabulce (Tab. 5.14), kde uvedeme počet úspěšných klasifikací, z celkového počtu pokusů, a jeho procentuální vyjádření.

Tab. 5.14 Testování klasifikace zvuku písmen

Znak	Počet	Úspěšnost
A	20	67%
B	5	17%
C	4	13%
D	5	17%
E	10	33%
F	30	100%
G	4	13%
H	5	17%
I	30	100%
J	0	0%
K	28	93%
L	10	33%
M	20	67%

Znak	Počet	Úspěšnost
N	0	0%
O	30	100%
P	10	33%
Q	25	83%
R	15	50%
S	30	100%
T	25	83%
U	30	100%
V	25	83%
W	30	100%
X	30	100%
Y	30	100%
Z	28	100%

Poznámka k době trvání algoritmu trénování SOM. Pro *Test 6* s tisícem epoch byl výpočet na GPU dokončen za 46 minut a 23 sekund. V případě spuštění na CPU lze očekávat dobu trvání 10 krát vyšší, čili 7 hodin a 42 minut. Proto doporučujeme použít k trénování SOM grafickou kartu s podporou CUDA, díky níž dojde k násobnému urychlení výpočtu.

6 Zhodnocení dosažených výsledků

V této části zhodnotíme dosažené výsledky na úlohách klasifikace obrazu a zvuku, testovaných v předcházející kapitole.

K nejlepšímu výsledku dané klasifikace jsme se postupně dostávali změnou konfigurace SOM a nebo úpravou vstupních dat datasetu. V případě klasifikace obrazu, změnou rozlišení obrazu.

6.1 Klasifikace obrazu

V klasifikace obrazu čísel 0–9 jsme se dostali na průměrnou úspěšnost pro všechny znaky ve výši 97%, což je vynikající výsledek testování.

V klasifikace obrazu písmen A–Z jsme se dostali na průměrnou úspěšnost pro všechny znaky ve výši 91,7%, což značí velmi dobrý výsledek testování.

Pokud při testování docházelo ke špatné klasifikaci, bylo to způsobenou přílišnou podobností tříd vstupních obrazů, typicky se jedná a písmena O a Q, I a T apod., která se mohou v psané formě velmi podobat. V případě klasifikace obrazu čísel to jsou znaky 3 a 8, 4 a 9 nebo 5 a 6. Jestliže došlo ke špatné klasifikaci, obvykle se jednalo záměnu těchto znaků.

6.2 Klasifikace zvuku

V klasifikace zvuku čísel 0–9 jsme se dostali na průměrnou úspěšnost pro všechny znaky ve výši 96,6%, což je vynikající výsledek testování.

V klasifikace zvuku písmen A–Z jsme se dostali na velmi slabý a neočekávaný výsledek testování. Některé znaky byly sice klasifikovány se 100% úspěšností (F, I, O, S, U, W, X, Y, Z), ale ostatní s nízkou úspěšností a některé znaky nebyly klasifikovány vůbec.

Připomeňme si nyní výslovnost jednotlivých čísel (Tab. 3.1) a výslovnost písmen (Tab. 3.2). Čísla mají 3 až 5 hlásek a nejsou si výrazně podobná při vyslovování. To potvrzuje výsledek testování klasifikace zvuku čísel. Při srovnání spektrogramů zvuku čísel (Obr. 3.6) nevidíme výraznou podobnost.

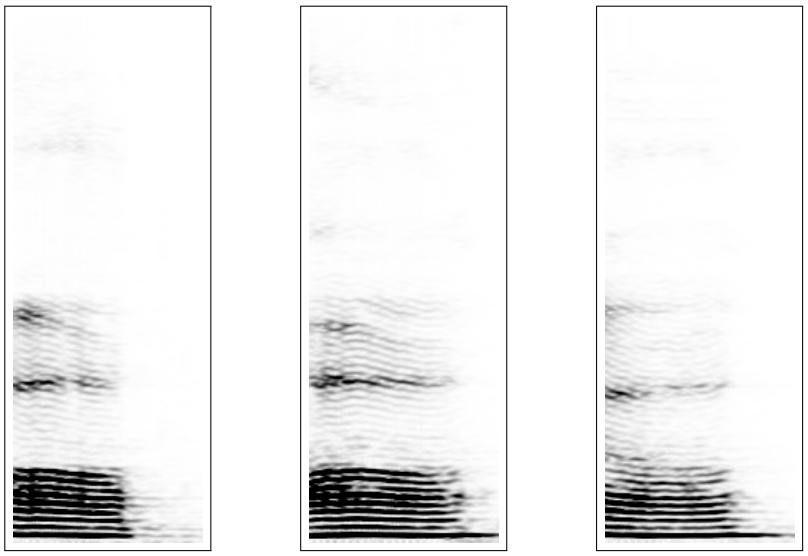
U výslovnosti písmen je situace jiná. Znaky jako Q, W, X, Y a Z mají 3 a více hlásek. Tyto znaky jsou klasifikovány s vysokou úspěšností. Ostatní znaky mají jednu až dvě hlásky a některé byly klasifikovány s nízkou úspěšností.

Rozdělme si tedy písmena do skupin na základě podobnosti výslovnosti.

- Skupina A - obsahuje znaky, jejich výslovnost obsahuje hlásku Á (a, há, ká)
- Skupina E

- obsahuje znaky, jejich výslovnost obsahuje hlásku É na konci (bé, cé, dé, é, gé, jé, pé, té, vé)
- obsahuje znaky, jejich výslovnost obsahuje hlásku E na začátku (ef, el, em, en, er, es)

Srovnejme si tedy např. spektrogramy skupiny A (Obr. 6.1).



(a) A (b) H (c) K

Obr. 6.1 Spektrogramy písmen skupiny A

Podobnost spektrogramů znaků skupiny A je velmi vysoká a z toho tedy vyplývá vzájemná nerozlišitelnost při klasifikaci těchto znaků. Při vyslovení např. H (há) trvá hláska H výrazně menší část délky než následující hláska A.

Totéž tedy platí i pro obě podskupiny skupiny E a jejich znaky jsou taktéž vzájemně nerozlišitelné. Nízká úspěšnost testování klasifikace zvuku je tedy způsobena výše popsanou podobností skupin znaků.

ZÁVĚR

V této práci jsme se zabývali Kohonenovými samoorganizujícími se mapami (SOM). Popsali jsme si jejich topologii, mřížku, okolí neuronu výstupní vrstvy, konfiguraci, princip učení a nejčastější použití. Dále jsem navrhli a zpracovali metodiku přípravy dat pro účely klasifikace obrazu a zvuku. Jak pro obrazová, tak pro zvuková data jsme navrhli metodiku předzpracování, od pořízení dat po tranformaci do vektorů, se kterými jsme dále pracovali.

Implementovali jsme aplikaci pro SOM formou desktopové a mobilní aplikace, kdy obrazová data byla pomocí mobilní aplikace pořízena a zároveň následně testována na úlohách rozpoznávání obrazu. Zvuková data byla pořízena a testována pomocí desktopové aplikace na úlohách rozpoznávání zvuku, aplikace umožňuje tranformaci obrazových a zvukových dat, trénování a testování SOM.

Následovalo otestování hotových aplikací na úlohách rozpoznávání obrazu a zvuku, pro čísla 0–9 a písmena A–Z. Trénování SOM probíhalo výhradně na GPU, aplikace samozřejmě podporuje běh programu i na CPU, nicméně jak bylo zmíněno, výpočet je pak časově náročnější.

Na závěr jsme zhodnotili dosažené výsledky, kdy testování klasifikace obrazu, pro čísla i písmena, mělo vysokou úspěšnost. Taktéž klasifikace zvuku čísel byla úspěšná, narozdíl od klasifikace zvuku písmen, kde jsme narazili na obtíže, které jsme si objasnili. Klasifikace zvuku pomocí SOM se tedy více hodí ke klasifikaci izolovaných slov, než-li ke klasifikaci krátkých posloupností hlásek.

Mezi nevýhody klasifikace pomocí SOM patří citlivost na posun vzoru. V této práci jsme neřešili při předzpracování úpravu obrazu ve smyslu vystředění a změna velikosti.

Mezi možné aplikace, které by SOM představené v této práci byly využitelné, patří například hlasové ovládání aplikací pomocí jednoduchých příkazů nebo hlasové ovládání myši. Další aplikací by byl doplněk do vývojového prostředí, který by na základě hlasových povelů vkládal do editoru zdrojového kódu programové konstrukty. Možných aplikací je však nepřeberné množství.

Cíle diplomové práce tímto byly splněny.

SEZNAM POUŽITÉ LITERATURY

[1] KOHONEN T.: Self-Organizing Maps, 2nd extended ed., Berlin, 1997, Springer, ISBN 978-3-642-56927-2.

[2] OJA E., KASKI S.: Kohonen Maps. Elsevier, 1999, ISBN 9780080535296.

[3] KŘIVAN, Miloš. Úvod do umělých neuronových sítí. Vyd. 3., přeprac. Praha: Oeconomica, 2014, 44 s. ISBN 978-80-245-2024-7.

[4] KOHONEN T. and HONKELA T.: Kohonen network. Scholarpedia, 2(1):1568, [online] 2007. URL `http://www.scholarpedia.org/article/Self-organizing_map`

[5] ROJAS R.: Neural networks - a systematic introduction, Springer-Verlag, Berlin,New-York, 1996.

[6] VOLNÁ, E. Neuronové sítě I. Ostrava: Ostravská univerzita, 2002. 85 s. Elektronický text.

[7] VONDRÁK, I. Neuronové sítě. Ostrava: VŠB-TU, 1994. 56 s. Elektronický text.

[8] KOHONEN T.: Speedy SOM, Report A33, Laboratory of Computer and InformationScience, Helsinki University of Technology, 1996.

[9] MRÁZOVÁ I.: Neuronové sítě. Praha: Univerzita Karlova, 2010. 46 s. Elektronický text.

[10] GEITGEY A.: Machine Learning is Fun Part 6: How to do Speech Recognition with Deep Learning, [online] 2016. URL `https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471`

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SOM	Samoorganizující se mapy
BMU	Best match unit - vítězný neuron
HDD	Pevný disk
SSD	Solid-state drive
FFT	Rychlá Fourierova transformace
GPU	Grafická karta
CPU	Procesor počítače

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. DVD

Struktura adresářů a obsah přiloženého DVD:

- DP
  - apk - složka s instalačními soubory mobilních aplikací
  - bin - složka s aplikací SOMGUI.exe
  - dataset - složka s použitým datasetem obrazu a zvuku
  - latex - zdrojové soubory textu práce včetně ilustrací
  - src
    - složka android - zdrojové soubory mobilní aplikace
    - složka app - zdrojové soubory aplikace SOMGUI
  - readme.txt
  - DP.pdf
- Priklad
  - *Priklad.pdf* - příklad algoritmu učení SOM
- Testovani
  - *Testovani_obrazu_pismen.pdf* - soubor s tabulkami testování klasifikace obrazu písmen
  - *Testovani_zvuku_pismen.pdf* - soubor s tabulkami testování klasifikace zvuku písmen