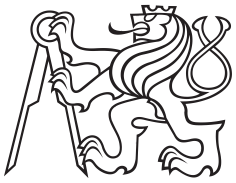
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Využití neuronové sítě při adaptaci uživatelského rozhraní

Jiří František

Vedoucí: Ing. Jiří Šebek
Obor: Otevřená informatika
Studijní program: Softwarové systémy
Květen 2018

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Jiřímu Šebkovi za jeho ochotu, pomoc a rady, které mi pomohly s dokončením této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 25. května 2018

Abstrakt

Tato bakalářská práce se zabývá úlohou vytvoření frameworku využívajícího neuronové sítě k adaptaci uživatelského rozhraní a zjištění, nejvýhodnější konfigurace neuronové sítě pro konkrétní typy vstupních a výstupních dat. Pomocí aplikace, kterou byl framework otestován bylo zjištěno, že ne všechny konfigurace jsou vhodné pro určité typy vstupních a výstupních data setů neuronových sítí. Při řešení úlohy byl použit programovací jazyk Java 8, framework byl vyvíjena v prostředí IntelliJ Idea na operačním systému Windows 10.

Klíčová slova: adaptivní uživatelské rozhraní, umělá neuronová síť, emoce, senzory, Java, Spring framework

Vedoucí: Ing. Jiří Šebek

Abstract

The following bachelor thesis is focused on the implementation of the framework that uses neural networks for user interface adaptation and finding the best neural networks configuration for specific input/output data sets. Testing this framework, using an application, showed that not all neural network configurations are suitable for specified type input and output data sets. Programming language used for implementation of this task was Java 8, framework was developed in IntelliJ Idea IDE on Windows 10.

Keywords: adaptive user interface, artificial neural network, emotions, senzors, Java, Spring framework

Title translation: Use of Artificial Neural Network for adaptive user interface

Obsah

Přílohy

Obrázky

Tabulky

Kapitola 1

Úvod

1.1 Motivace

V dnešní době má téměř každý člověk k dispozici počítač, notebook, chytrý telefon nebo tablet a používání těchto přístrojů je na jeho denním pořádku. Každé zařízení i aplikace v moderních přístrojích má nějaké uživatelské rozhraní, přes které se ovládá, a právě ovládání musí být pro uživatele co nejpříjemnější.

Každé zařízení by mělo být připraveno, že bude používáno v jiných než běžných situacích a běžných podmínkách. Je rozdíl, jestli člověk čte z mobilu za normálního osvětlení nebo pod přímým slunečním svitem, jestli člověk používá zařízení v klidu svého domova nebo pod stresem, když nestíhá přijet včas na důležitou schůzku. A právě i za těchto podmínek musí zařízení (aplikace) poskytnout uživateli co nejpřívětivější rozhraní.

Další v dnešní době velmi používanou technologií je umělá inteligence. Lidé se snaží převést co nejvíce práce na stroje v zájmu usnadnit si svůj vlastní život. Představou, že přístroj je schopen se z předchozích akcí učit a využívat tyto znalosti při řešení následujícího problému, se zabývá podoblast umělé inteligence s názvem strojové učení.

Spojením oborů strojového učení [GOL] a uživatelského rozhraní [OPP] nacházíme řešení problému, aby aplikace byla schopna sama rozhodovat na

základě předchozích situací o nejvhodnějším uživatelském rozhraní v dané situaci.

1.2 Cíle práce

Cílem práce je nastudovat možnosti adaptivního uživatelského rozhraní a následné využití neuronové sítě s problémy s nimi souvisejícími.

Dále vytvoření vhodného data modelu vybraných okruhů adaptivního uživatelského rozhraní, který bude vhodný jako vstup a výstup pro neuronové sítě. S tím je spjaté vytvoření data setů, které budou sloužit k učení neuronové sítě.

Následovat bude návrh a implementace frameworku využívající neuronové sítě ke generování uživatelského rozhraní v závislosti na datech zvolených uživatelem. A nakonec, bude tento framework otestován Java Spring aplikací ve více konfiguracích neuronových sítí, které budou vzájemně porovnány.

Část I

Teoretická část

Kapitola 2

Rešerše

V rešerši si popíšeme všechny pojmy, kterých se tato práce týká a se kterými budeme pracovat. Tato kapitola slouží jako teoretický základ k řešení problémům této práce.

2.1 Uživatelské rozhraní

Uživatelské rozhraní (anglicky user interface, UI) slouží ke komunikace nějakého systému s uživatelem. Hlavním účelem uživatelského rozhraní je co nejvíce zjednodušit uživateli ovládání systému, to znamená, aby byl systém co nejpřehlednější a nejintuitivnější i pro lidi, kteří nemají zkušenosti s podobnými systémy.

Uživatelské rozhraní se dělí na typy [oui]:

1. **Command Line Interface (CLI)**

   Rozhraní příkazového řádku, které se ovládá sérií příkazů zadávaných z klávesnice. CLI funguje jako vlastní jazyk, vyžaduje po uživateli, aby byl seznámen se syntaxí daného rozhraní a pamatoval si příkazy, které je většinou možno použít jen v určitém pořadí po sobě. Příklad takového rozhraní je třeba terminál v UNIX systémech.

2. **Graphical User Interface (GUI)**

Grafické uživatelské rozhraní je založeno na práci s okny, rolovacími menu nabídkami a tlačítky. Uživatel okamžitě vidí reakce rozhraní na jeho akce. Takovéto rozhraní se ovládá nejčastěji klávesnicí, myší nebo prstem (dotykové obrazovky, dotykové plochy jako třeba touchapad u notebooků).

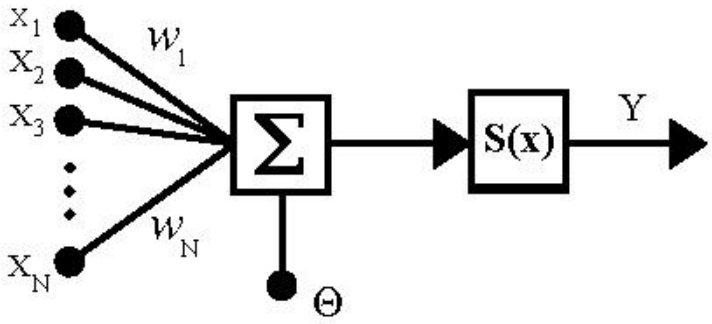
3. **Adaptivní uživatelské rozhraní**

Adaptivní uživatelské rozhraní je takové rozhraní, které je schopno se samo (dynamicky) měnit dle preferencí uživatele. Systém sbírá informace o uživateli a o prostředí a je schopen na základě napsaných programových instrukcí nebo umělé inteligence vytvořit optimální rozhraní pro daného uživatele.

2.2 Neuronové sítě

Umělá neuronová síť (anglicky Artificial neural network) je matematický model inspirovaný biologickými neurony v lidském mozku. Biologický neuron přijímají signály přicházející na synapse neuronu, pokud jsou signály dostatečně silné, neuron se aktivuje a vyšle přes axon signál. Ten může směřovat do dalších neuronů [BAS].

Při modelování umělé neuronové sítě je složitost reálných neuronů silně abstrahována. Umělé neurony (též nazývané perceptrony) se skládají ze vstupů (jako synapse u biologického neuronu), které jsou násobeny (synaptickou) váhou signálu (dostáváme sílu signálu). Dále matematická funkce vypočítá ze vstupů vynásobených jejich vahami, zda se neuron aktivuje. Další funkce vypočítá výstup neuronu [JAI].



Obrázek 2.1: Model umělého neuronu [ns].

kde:

- X_i jsou vstupy neuronu
- W_i jsou synaptické váhy
- θ je práh, označující prahovou hodnotu aktivace neuronu, též označováno jako bias
- $S(x)$ je přenosová funkce neuronu (někdy nazývána též aktivační funkce)
- Y je výstup neuronu

Umělé neuronové sítě se využívají například pro rozpoznávání vzorů, diagnóza onemocnění, komprese dat, předpovědi (počasí, vývoje trhu, signálů) umělá inteligence, detekce poruch, optimalizace a mnoho dalších příkladů [BAJ].

Výhody umělé neuronové sítě jsou:

- Může provádět úlohy, které lineární program nemůže
- Selhání jednoho elementu y neuronové sítě neznamená selhání celého systému
- Učí se a nemusí být přeprogramována
- Může být použita v každé aplikaci

Nevýhody umělé neuronové sítě jsou:

- Musí se před použitím naučit
- Architektura neuronové sítě je jiná než architektura mikroprocesoru, a proto musí být emulována
- Vyžaduje delší čas pro zpracování větších sítí

2.2.1 Učení neuronových sítí

Aby mohly neuronové sítě fungovat a dávat co nejpřesnější výsledky, tak se musí nejdříve naučit. Učení (nebo také trénování) umělé neuronové sítě znamená změnit váhy vstupů jednotlivých neuronů podle nějakého pravidla. To se nejčastěji děje tak, že máme nachystaná a podle nich necháme síť, aby si sama podle zvoleného pravidla nastavila váhy vstupů. Fáze učení sítě se nazývá adaptivní, a po naučení se neuronová síť nachází ve fázi vybavování.

2.2.2 Druhy učení neuronové sítě

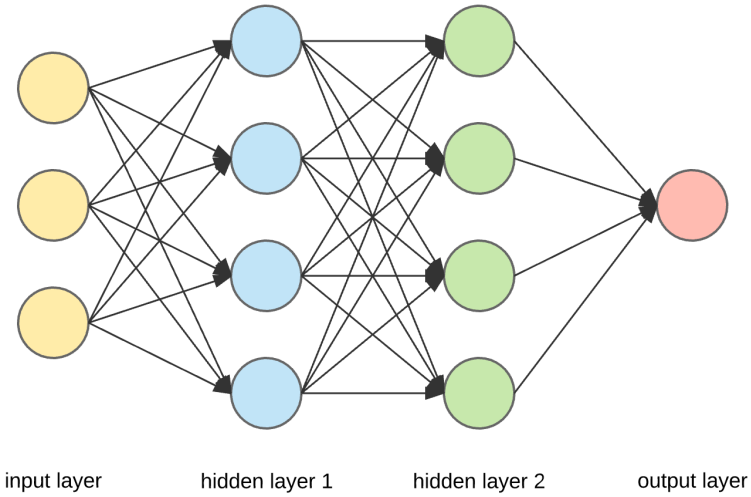
Existují dva druhy učení neuronové sítě [tNN]:

1. Učení s učitelem – síti je poskytnut soubor vstupů a k nim korespondujících výstupů. Síť zpracuje vstup podle svého aktuálního nastavení a její výstup se porovná s očekávaným výstupem a změří se odchylka, podle odchylky se upraví hodnoty vah a prahů u neuronů. Toto pokračuje, dokud se nedosáhne předem nastavené minimální odchylky.
2. Učení bez učitele – síti je poskytnut pouze soubor vstupů, které sama zpracovává a třídí.

2.2.3 Vrstvy neuronové sítě

Neuronová síť se skládá vždy minimálně ze dvou vrstev (vstupní a výstupní). Počet neuronů vstupní a výstupní vrstvy musí být stejný, jako počet prvků očekávaného vstupu a výstupu sítě. Neuronová síť může, a v naprosté většině případů má, víc vrstev. Mezi vstupní a výstupní vrstvou se nacházejí takzvané skryté vrstvy, které se také skládají z perceptronů, ale jejich počet není přesně daný, tyto vrstvy si mezi sebou posílají data směrem od vstupní vrstvy k výstupní a platí, že každý perceptron skryté vrstvy a výstupní vrstvy obsahuje tolik vstupů, kolik se nachází perceptronů v předchozí vrstvě. Jinak řečeno, každý perceptron, kromě vstupní vrstvy, přijímá na svém vstupu data z každého perceptronu z předchozí vrstvy. Perceptrony ze vstupní vrstvy mají pouze jeden vstup, ale posílají data do všech perceptronů následující vrstvy a perceptrony výstupní vrstvy mají pouze jeden výstup [GAR]. Graficky zobrazeno na obrázku 2.2. Neexistuje přesný předpis pro optimální počet skrytých vrstev a počet neuronů v těchto vrstvách, nejlepší variantu pro

neuronové sítě použité v této práci zjistíme testováním a porovnáváním různých konfigurací.



Obrázek 2.2: Ukázka vícevrstvé neuronové sítě [Net].

2.2.4 Aktivační funkce

Aktivační funkce neuronu obměňuje výstupní signál, který se dále šíří do neuronů další vrstvy. Existují funkce lineární a nelineární. Není explicitně dáno, jaká aktivační funkce se používá pro jaký problém, výběr vhodné funkce tedy záleží na zkušenostech nebo testování různých funkcí. Pro neuronové sítě v této aplikaci jsem vybral na testování a porovnávání tři z nejpoužívanějších aktivačních funkcí, jsou to funkce: lineární, hyperbolický tangens (tanh) a sigmoid [KAR].

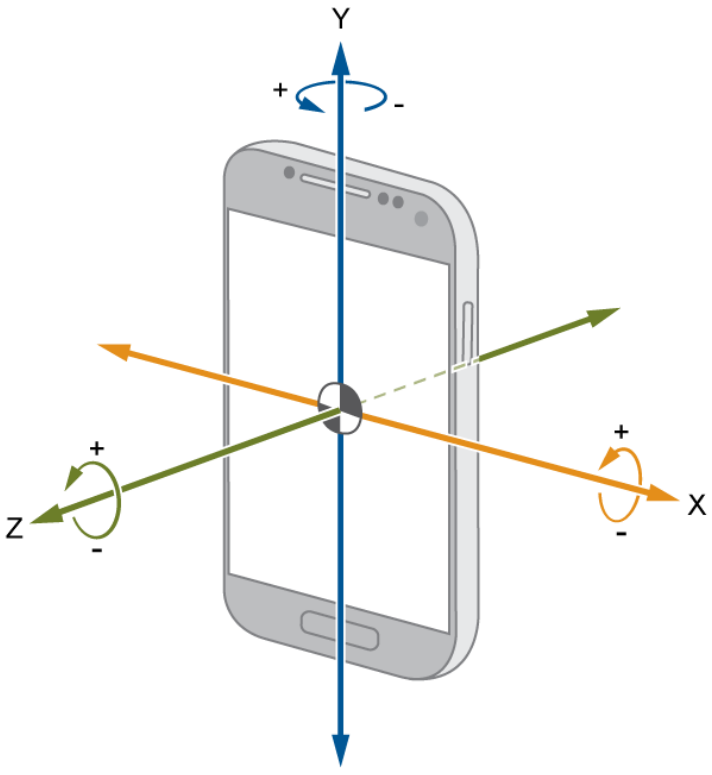
2.3 Adaptovaná data

Existuje velké množství případů, ve kterých se využívá adaptace pro co nejlepší dojem uživatele z rozhraní, v této práci se budu zabývat jen pár případy, které jsem si vybral.

2.3.1 Gyroskop

Gyroskop je zařízení sloužící k měření úhlové rychlosti a používá se k orientaci zařízení v prostoru. Senzor zaznamenává změny úhlu zařízení ve třech osách x, y a z, na základě změřených údajů vyhodnocuje úhel náklonu zařízení vzhledem k zemi. Jednotkou je rad/s [BAR].

Gyroskop se využívá zejména v navigaci, pak také u letadel, vrtulníků, řízených raket a v dalších případech. K navigaci je gyroskop určen také v mobilních telefonech a tabletech, ale má zde i další význam. Data z gyroskopu se zde využívají k adaptaci uživatelského rozhraní, konkrétně k otočení displeje v případě otočení zařízení. Pokud otočíme zařízení, které je displejem k nám o 90 stupňů doprava, tak i displej se otočí o 90 stupňů doprava (respektive o 270 stupňů doleva), což nenutí uživatele nepřirozeně točit hlavou, aby měl obraz rovně tak, aby to bylo pohodlné. Dále je gyroskop v mobilech využíván například ke hrám. Počáteční polohu mobilního zařízení, tedy otočení o 0 stupňů ve všech osách, budeme brát případ, když je displej mobilního telefonu před hlavou vzpřímeného člověka.



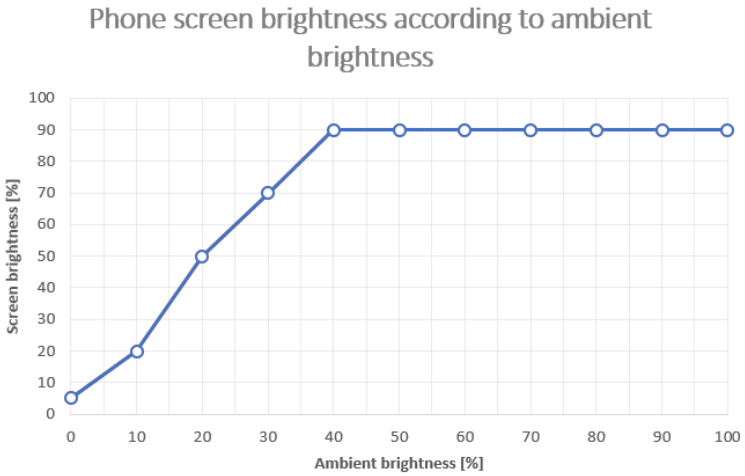
Obrázek 2.3: Osy natáčení mobilního telefonu [Gyr].

Data rotací telefonu okolo os pro naučení neuronové sítě byla pořízena sledováním chování mého osobního mobilního telefonu. Souřadnicový systém během rotací telefonu byl pevně spojený se zařízením, takže například osa y při jakémkoliv otočení vždy procházela středem telefonu přes celou jeho výšku, jak je vyobrazeno na obrázku 2.3.

2.3.2 Senzor okolního osvětlení

Senzor okolního osvětlení (anglicky Light sensor) slouží k získání úrovně osvětlení v okolí zařízení. Na základě dat ze senzoru se u mobilních telefonů mění intenzita osvětlení a teplota barev displeje, čímž se displej adaptuje na lepší viditelnost při jakémkoliv osvětlení. Díky údajům ze senzoru okolního osvětlení se také šetří baterie, protože zařízení podle okolního osvětlení vyhodnotí, že už není potřeba tak vysoká intenzita podsvícení displeje, a tak ji sníží, čímž se sníží i energetická náročnost displeje.

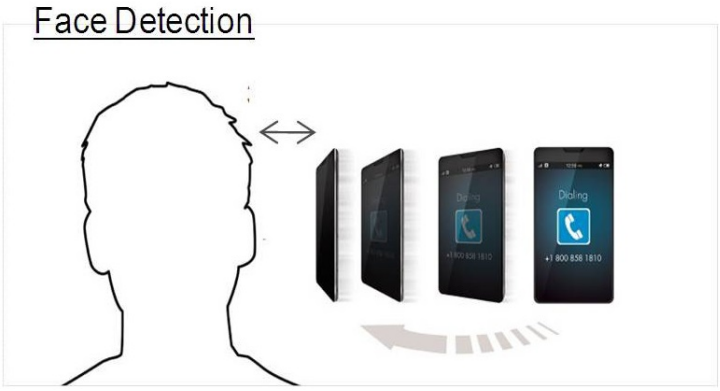
Obecně platí, že čím menší je osvětlení okolní prostředí, tím nižší je intenzita podsvícení displeje, ale nikdy neklesne na 0 %. A naopak platí, že čím větší je osvětlení okolního prostředí, tím vyšší je intenzita podsvícení displeje, ale v rámci šetření baterie nedosahuje 100 %.



Obrázek 2.4: Graf intenzity podsvícení displeje mobilního telefonu v závislosti na okolním osvětlení [ftiAS].

2.3.3 Senzor přiblížení

Senzor přiblížení (anglicky Proximity senzor) slouží k detekci objektu v okolí zařízení, aniž by byly se zařízením v kontaktu. Nejznámější využití tohoto senzoru pro běžné uživatele je v mobilních telefonech. Pokud se objekt přiblíží k displeji mobilního telefonu během hovoru na určitou vzdálenost, většinou několik centimetrů (5-7 cm), tak se displej vypne, aby případný dotyk displeje a ucha nespustil nežádoucí akci na zařízení.



Obrázek 2.5: Využití senzoru přiblížení v mobilních telefonech [iMP].

2.3.4 Emoce

Emoce jsou procesy, zahrnující subjektivní zážitky, pozitivní či negativní, provázené fyziologickými a motorickými změnami, způsobené okolními událostmi a výsledky činností podle subjektivního vztahu k hodnotiteli.

Podle knihy „Unmasking the face: A guide to recognize emotions from facial clues“ [1] existují následující základní emoce:

- Strach
- Překvapení
- Znechucení
- Vztek

[1] Unmasking the face: A guide to recognize emotions from facial clues je kniha napsána psychologem Paulem Ekmanem, který se zabýval studiem emocí ve vztahu k mimice. [EKM]

- Štěstí
- Smutek

V rámci této práce se budeme zabývat pouze 4-mi emocemi plus neutrální stav, u kterého neplatí ani jedna z následujících vybraných emocí:

- Smutek
- Štěstí
- Vztek
- Znechucení

Na základě emocí je možné přizpůsobit uživatelské rozhraní tak, aby odpovídalo aktuální emoci uživatele. V této práci se budu zabývat pouze spojení barev s emocemi neboli změna barvy rozhraní vzhledem k aktuální emoci uživatele.

Emoce a jejich barevná paleta

Existují studie, které zkoumají spojení barev s emocemi, vše je detailněji popsáno v bakalářské práci „Adaptace UI založena na emocích uživatele“ od A. Lunovové [LUN]. Tyto studie byly provedeny na náhodném vzorku lidí, kteří měli přiřazovat barvy k emocím. Ačkoliv emoce nejsou jednoduché na popsání a přiřazení barev jednotlivým emocím je velmi subjektivní, tak i přesto se podařilo vytvořit paletu barev spojených s konkrétními emocemi.

Neutrální stav

Základní stav, který je použit, pokud se neprojeví žádná z následujících emocí.

	R	G	B	Color
Background	238	201	177	
Font	71	87	111	

Tabulka 2.1: Barevná paleta pro neutrální stav

Smutek

	R	G	B	Color
Background	238	201	177	
Font	71	87	111	

Tabulka 2.2: Barevná paleta pro stav smutku

Štěstí

	R	G	B	Color
Background	238	201	177	
Font	71	87	111	

Tabulka 2.3: Barevná paleta pro stav štěstí

Vztek

	R	G	B	Color
Background	238	201	177	
Font	71	87	111	

Tabulka 2.4: Barevná paleta pro stav vzteku

Znechucení

	R	G	B	Color
Background	238	201	177	
Font	71	87	111	

Tabulka 2.5: Barevná paleta pro stav znechucení

Kapitola 3

Související práce

Tato bakalářská práce používá teoretické poznatky o senzorech popsané v bakalářské práci T. Titovové „Využití aspektově orientovaného přístupu a lokálních dat ve vývoji adaptivního uživatelského rozhraní pro Android“. [TIT] a údaje spojení barev s emocemi popsané v bakalářské práci A. Lunovové „Adaptace UI založena na emocích uživatele“ [LUN]. Obě práce se také zabývají adaptací uživatelského rozhraní, jenže adaptace v těchto pracích probíhá podle předem popsaných postupů a předpisů. V případě i menší změny v adaptaci je nutné přepisovat zdrojový kód. V případě neuronové sítě stačí pouze síť naučit na nové množině dat.

Kapitola 4

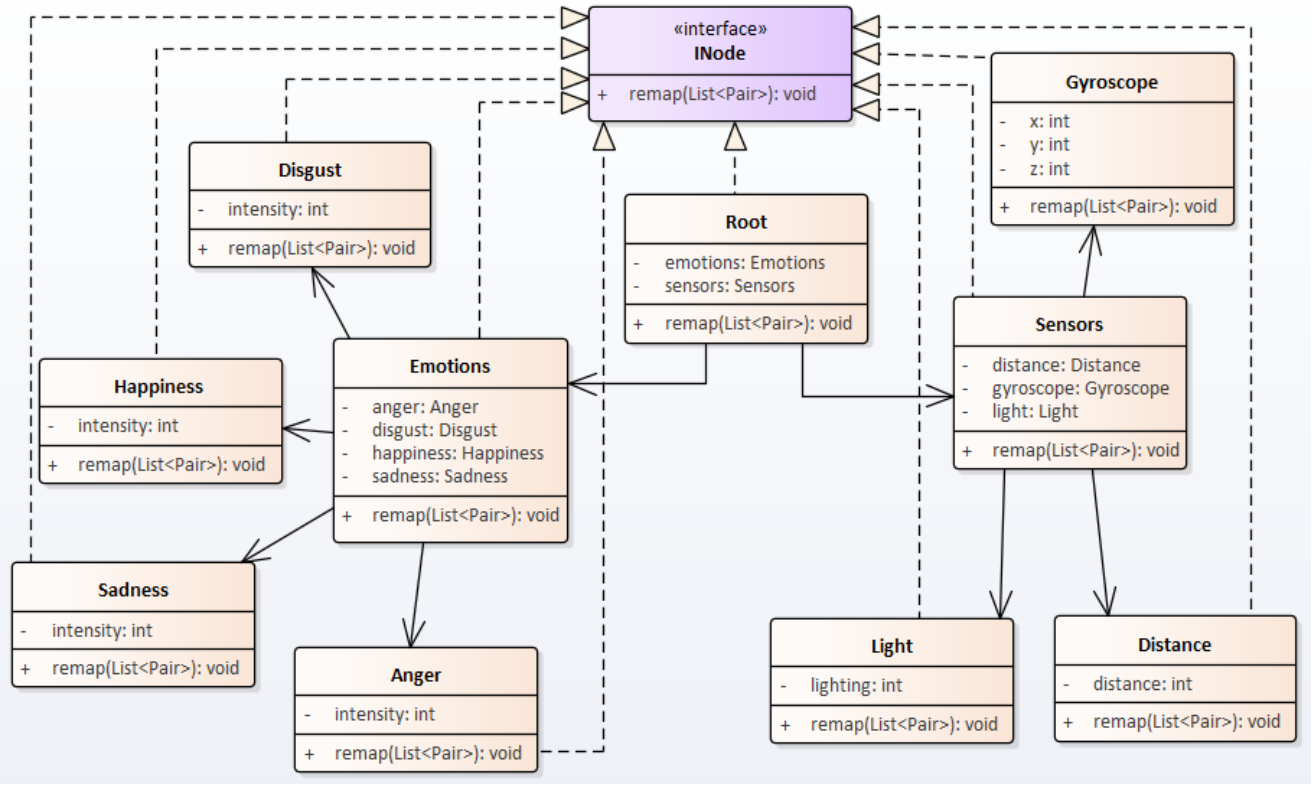
Analýza a návrh aplikace

Jak je již uvedeno v úvodu, cílem této práce je vytvořit framework využívající neuronovou síť, která bude schopna na základě vstupních dat, souvisejících s kapitolou Adaptovaná data 2.3, vrátit výstupní data, která budou reprezentovat adaptované rozhraní. Ale nejdříve je potřeba vyřešit následující problémy:

1. Model vstupních a výstupních dat neuronové sítě,
2. Návrh frameworku a testovací aplikace

4.1 Krok 1. Konceptuální datový model neuronové sítě

Neuronová síť dokáže pracovat pouze s daty ve formě pole čísel. Proto je potřeba vymyslet způsob, jak uchovávat data ve srozumitelné a přehledné formě jak v textovém souboru, tak v programu, a které bude možnost kdykoliv přeměnit na pole čísel a zpět. Proto musel být vymyšlen konceptuální model, který bude všechny předchozí podmínky splňovat.



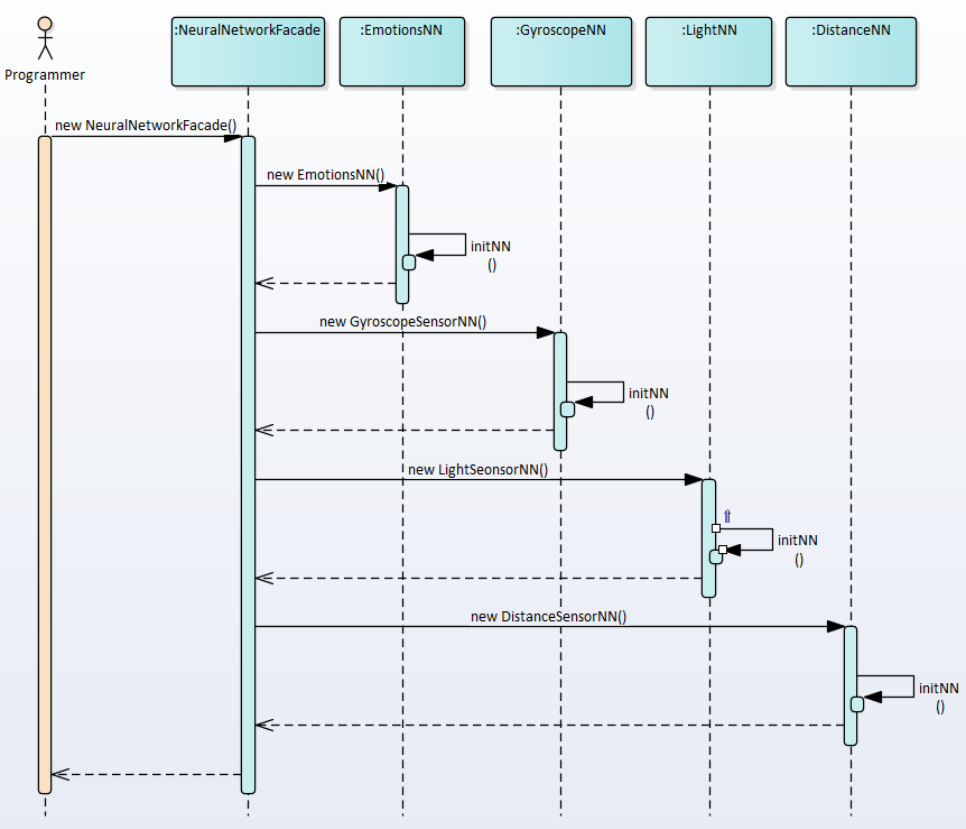
Obrázek 4.1: Konceptuální model vstupu neuronové sítě.

Jak je vidět na obrázku 4.1, tak třídy dohromady tvoří stromovou strukturu a v koncových třídách (listech stromu) jsou použity pouze číselné atributy, kvůli možnosti jednoduchého převodu modelu do číselného pole. Všechny třídy implementují rozhraní `INode`, reprezentující stromovou strukturu, s metodou remap, která funguje jako inorder prohledávání stromu do hloubky a sbírá data z listů stromu. Na stejném principu funguje i model pro výstupní data z neuronové sítě, ale třídy implementují navíc ještě rozhraní `IOutputNode`, které namapuje výstupní pole z neuronové sítě zpět do stromové struktury. Viz. obrázek 1 v příloze.

Největší výhoda tohoto návrhu je, že je možné jednoduše transformovat data z textové podoby do objektu, z objektu do pole a zase zpět. Další velká výhoda je, že je možnost transformovat jen část celého modelu, například, když nebude potřeba data o emocích, tak se zavolá metoda `remap()`, které se jako parametr dá prázdný list, do kterého sesbírá data z listů podstromu, pouze na podstrom s kořenem `SensorsOutput`. Nevýhoda tohoto návrhu je, že pokud budeme chtít rozšířit model o nové třídy, tak budeme muset přidat nové třídy do aktuálního modelu a implementovat jim metody všechny rozhraní ručně.

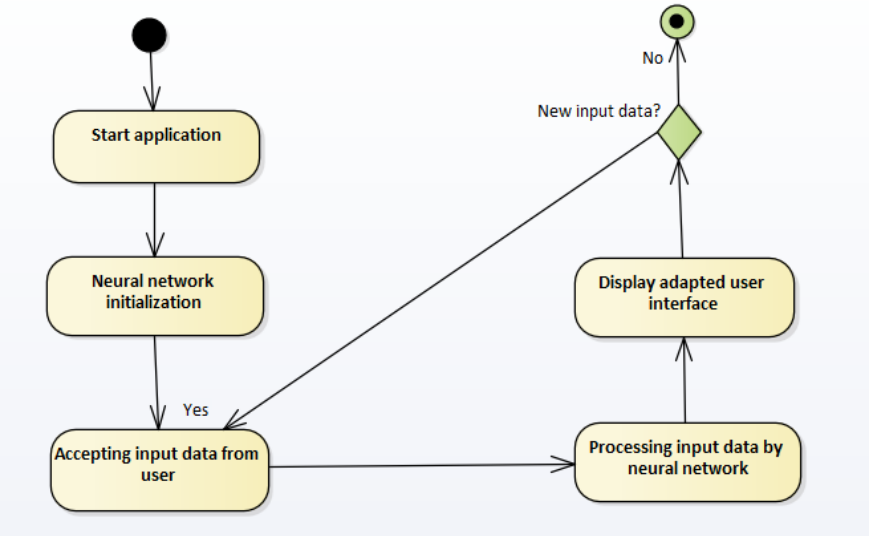
4.2 Krok 2. Návrh frameworku a testovací aplikace

Framework i aplikace budou napsány v jazyce Java 8. Framework se bude skládat z datového modelu a neuronových sítí, které budou tento datový model využívat jako svůj vstup a výstup. Postup inicializace neuronových sítí je zobrazen na sekvenčím diagramu na obrázku 4.2.



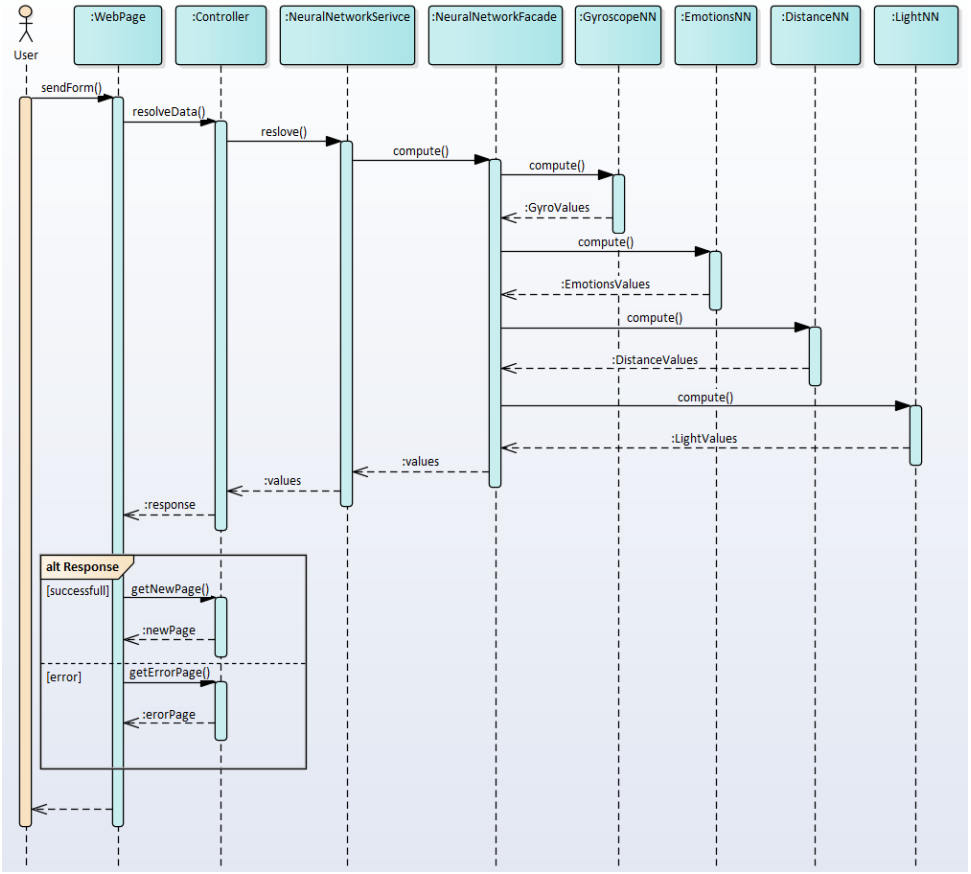
Obrázek 4.2: Sekvenční diagram znázorňující inicializaci neuronových sítí.

Pro otestování frameworku bude vytvořena jednoduchá Java EE aplikace, která z uživatelského vstupu, po zpracování frameworkem, zobrazí nově adaptované uživatelské rozhraní. Běh aplikace popisuje obrázek 4.3. Jde o diagram aktivit (anglicky activity diagram), který slouží k popisu chování aplikace. Diagram aktivit se používá pro modelování procesů a zachycuje po sobě jdoucí akce.



Obrázek 4.3: Diagram aktivit.

Interní fungování hlavní části aplikace zobrazuje sekvenční diagram na 4.4. Sekvenční diagram je nástroj pro přehled časově uspořádaných interakcí mezi objekty. Jsou zde zobrazeny pouze ty objekty, které se podílí na přímé interakci.



Obrázek 4.4: Sekvenční diagram popisující běh testovací apliakce s frameworkem.

Aplikace bude postavena na vzoru MVC (Model-View-Controller), jedná se o návrhový vzor užitečný pro stavbu interaktivních softwarových systémů. Klíčová myšlenka MVC je rozdělit od sebe uživatelské rozhraní (View) od dat (Model), které toto rozhraní mění. Prostředníkem, který reaguje na požadavky od uživatele je Controller. Na základě těchto požadavků zajistí změny v modelu a a základě změn v modelu přizpůsobí uživatelské rozhraní [LEF].

Část II

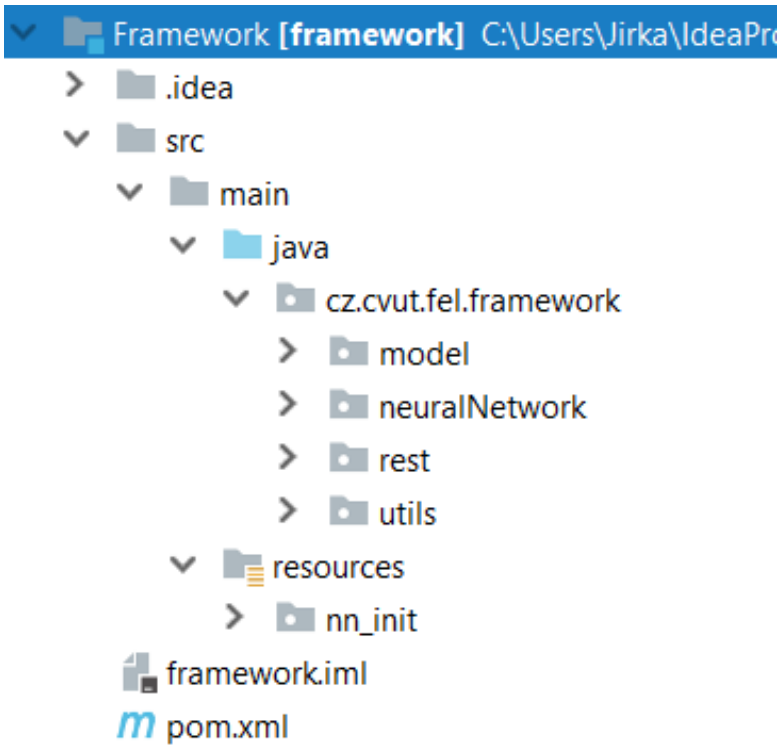
Praktická část

Kapitola 5

Implementace

Pro implementaci frameworku byl použit jazyk Java ve verzi 8. Serverová část testovací aplikace byla napsána za použití frameworku Java Spring [Frab] a na klientskou část aplikace byl použit šablonový framework FreeMarker [21]. Na neuronovou síť ve frameworku byla použita knihovna Encog [Fraa], což je knihova použitelná pro programovací jazyky Java a C#, která se zabývá strojovým učením a umělou inteligencí. Celá programovací část byla napsána ve vývojovém prostředí IntelliJ IDEA od firmy JetBrains.

5.1 Struktura projektu



Obrázek 5.1: Struktura projektu frameworku.

Projekty v jazyce Java mají stromovou strukturu, jednotlivé třídy se nachází v balíčcích, které představují adresářovou strukturu cesty ke třídám. Balíčky slouží k lepšímu přehledu a k jednoznačnému rozpoznání třídy v rámci celého projektu. Není možné mít dvě stejně pojmenované třídy, název souboru nese název třídy, v jednom balíčku.

- **java**
  - **model**
    Balíček obsahující třídy odpovídající konceptuálnímu datovému modelu, které jsou použity pro vstup a výstup neuronových sítí.
  - **neuralNetwork**
    Balíček obsahující třídy zajišťující konfiguraci neuronových sítí a jejich následné použití. Je zde také hlavní rozhraní, které zajistí používání celého modulu.
  - **rest**

Balíček obsahující rozhraní, implementace jeho metod zajišťuje zasílání správných vstupních dat z klientu.

- **utils**

  Zde se nachází pomocné třídy celého projektu, jako je třeba třída pro transformaci dat.

- **resources**

  Zde se nachází adresář se vstupy a výstupy neuronových sítí, které slouží k jejich učení.

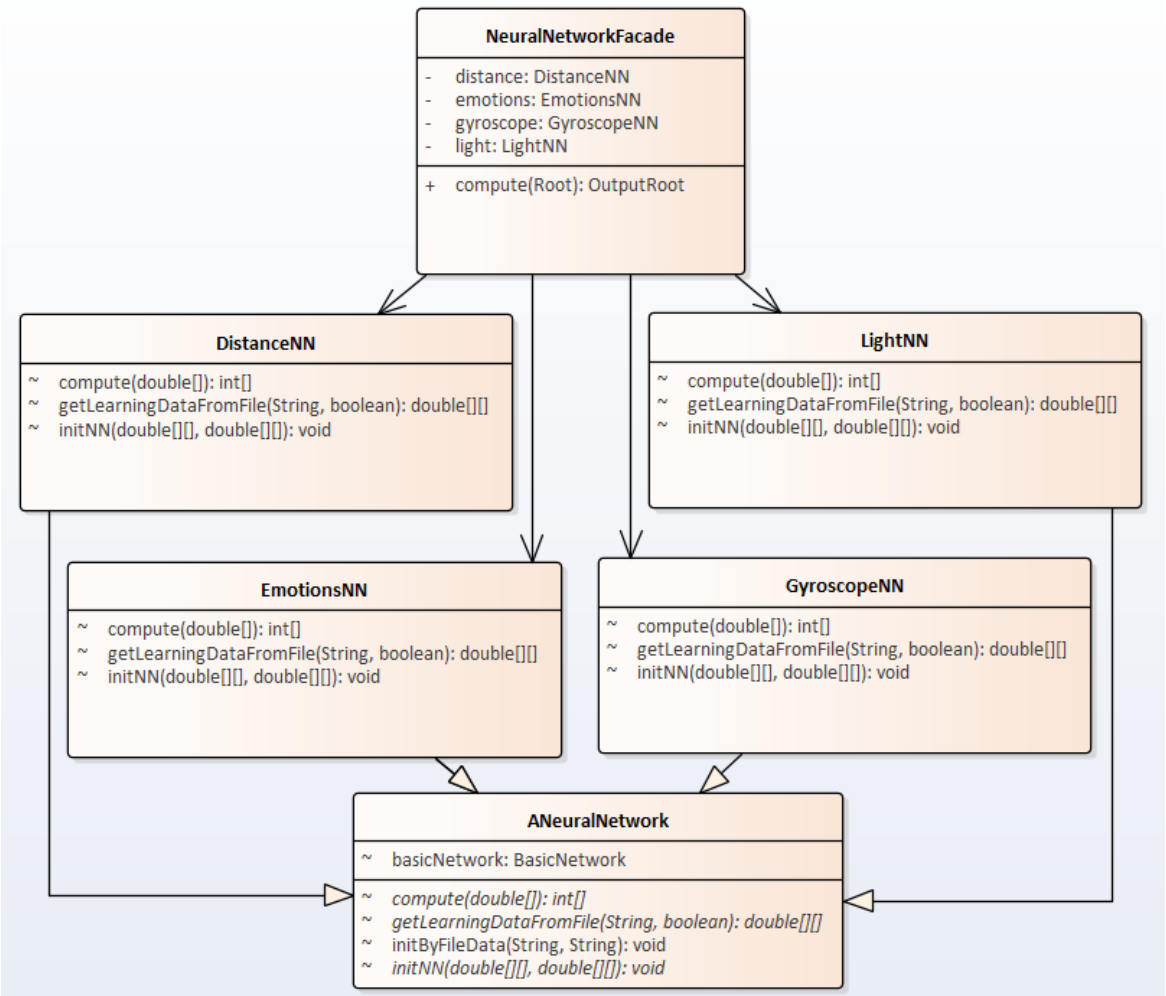
- **pom.xml**

  Základní soubor nástroje Maven, který řídí sestavování aplikace, jsou zde vyjmenovány všechny použité externí knihovny a informace důležité pro kompilaci projektu.

5.2 Neuronové sítě

Jak již bylo řečeno, k implementaci neuronových sítí byla použita externí knihovna Encog. Z důvodu snížení časové náročnosti výpočtů neuronové sítě, jsem se rozhodl rozdělit jedno velké pole vstupu na víc menších polí, které budou zpracovávat různé neuronové sítě, předem naučené na daný typ dat. Samozřejmě budou data rozdělena tak, aby údaje reprezentující data například z gyroskopu zpracovávala síť k tomu určená.

Z důvodu přehlednosti kódu využiji k implementaci různých neuronových sítí návrhový vzor Fasáda.



Obrázek 5.2: Návrhový vzor „Fasáda“ u neuronových sítí.

Jde o návrhový vzor, který slouží k snížení komplexnosti komunikace mezi uživatelem a rozhraním [SCH], v tomto případě mezi aplikací a frameworkem, třída `NeuralNetworkFacade` zabaluje jednotlivé neuronové sítě a směřuje na ně požadavky. Díky tomu si nemusí apliakce držet odkazy na všechny neuronové sítě, ale pouze na jeden objekt, který potřebná data posílá na konkrétní neuronové sítě. Třída `NeuralNetworkFacade` v testovací aplikaci figuruje jako Spring Bean [1]. To nám dává možnost předem vytvořený objekt této třídy vložit do jiné části aplikace.

Aby Spring framework věděl, že má spustit tuto metodu při spuštění

[1]Spring Bean je objekt, který je celý svůj životní cyklus spravován speciálním Spring kontejnerem a konfigurovány přes XML soubor nebo Java anotace. Tyto tyto objekty se inicializují se spuštěním serveru a je možno je pak použít v kódu aniž by programátor musel vytvářet nový objekt. [Def]

aplikace, tak se tato metoda musí nacházet v konfigurační třídě nebo v konfiguračním XML souboru. Já si zvolil první přístup. Tato metoda se nachází v třídě `NNConfig`, která je je anotací `@Configuration` přeměněna na konfigurační třídu aplikace, a Spring framework ví, že v této třídě se nachází metody, které musí být provedeny během startu celé aplikace.

Vytvoření samotné neuronové sítě za použití knihovny Encog se provádí vytvořením objektu třídy `BasicNetwork` a jeho následným nakonfigurováním, jako například výběr počtu vrstev, počtu neuronů a aktivačních funkcí u každé vrstvy. První vrstva je vstupní vrstva neuronové sítě a poslední je výstupní vrstva, všechny ostatní vrstvy mezi nimi jsou automaticky brány za skryté vrstvy neuronové sítě. Po vytvoření struktury neuronové sítě je potřeba dokončit vytváření neuronové sítě metodami `getStructure()` a následně ještě `finalizeStrucutre()`. Následně se musí ještě celá síť nainicializovat na počáteční hodnoty metodou `reset()`, čímž se nastaví hodnoty vah mezi neurony a prahové hodnoty neuronů na počáteční hodnotu, nyní je síť připravena na učení.

```
basicNetwork = new BasicNetwork();
basicNetwork.addLayer(new BasicLayer(new ActivationTANH(), true, 3));
basicNetwork.addLayer(new BasicLayer(new ActivationTANH(), true, 5));
basicNetwork.addLayer(new BasicLayer(new ActivationTANH(), true, 5));
basicNetwork.addLayer(new BasicLayer(new ActivationTANH(), true, 5));
basicNetwork.addLayer(new BasicLayer(new ActivationTANH(), true, 1));
basicNetwork.getStructure().finalizeStructure();
basicNetwork.reset();
```

Listing 5.1: Ukázka inicializace neuronové sítě knihovny Encog.

Učení neuronové sítě bude probíhat s učitelem. Kromě vstupních dat poskytneme síti ještě očekávaná výstupní data, na kterých se neuronová síť bude učit. Data pro naučení neuronové sítě jsou uloženy v projektu ve složce `src/main/resources/nn\_init`. Trénovací data jsou uložena ve formátu JSON (JavaScript Object Notation). Po načtení do aplikace knihovna Jackson namapuje data z JSON textové podoby na objekty z data modelu. Pomocí final třídy `ModelTransformUtils` se static metodami z balíčku `utils` transformujeme data z tříd modelu na pole, které slouží jako vstup a výstup neuronové sítě.

Samotné učení probíhá tak, že se prvně zvolí trénovací (učící) algoritmus, kterému se poskytnou trénovací data a neuronová síť určená k učení.

```
MLDataSet trainingSet = new BasicMLDataSet(input, output);
MLTrain train = new ResilientPropagation(basicNetwork, trainingSet);
long iterations = 0;
do {
```

```
   train.iteration();
   iterations++;
} while (iterations < 2000);
train.finishTraining();
```

Listing 5.2: Ukázka trénování neuronové sítě.

Vstupní a výstupní data jsou uloženy ve struktuře `MLDataSet`, která je spolu s vytvořenou neuronovou sítí poskytnuta třídě trénovacího algoritmu `ResilientProtpagation`, tento algoritmus iterativně prochází poskytnutá data a učí na nich neuronovou síť. Trénování sítě může být ukončeno buď předem daným počtem iterací a nebo můžeme nechat síť učit dokud nepřekročí předem stanovenou mez chyby výpočtu sítě. Trénovací algoritmus testuje sám sebe na poskytnutých datech a vyhodnocuje s jak moc velkou chybou proběhla trénovací iterace. Pro ukončení trénování je potřeba zavolat na trénovací algoritmus metodu `finishTraining()`.

Inicializace celého frameworku se provádí vytvoření objektu `NeuralNetworkFacade`, vytovřením objektu konstruktorem bez parametrů se použijí k trénování neuronových sítí data z frameworku, která jsou v adresáři `resources`. Aby měl programátor možnost použít svoje trénovací data k trénování neuronové sítě, tak je k dispozici konstruktor, do kterého lze jako parametry vložit cestu k souborům s trénovacími daty, ale tyto data musí být ve formátu JSON a mít stejnou strukturu, jako data na ukázce 5.3.

```
Input emotions:
{"anger":{"intensity":0},"disgust":{"intensity":0},
"happiness":{"intensity":0},"sadness":{"intensity":0}}

Output emotions:
{"backgroundColor":{"r":238,"g":201,
"b": 177},"fontColor":{"r":71,"g":87,"b":111}}

Input gyroscope:
{"x":0,"y":0,"z":0}

Output gyroscope:
{"turnLeftCnt":0}

Input distance sensor:
{"distance":0}

Output distance sensor:
{"displayOn":0}

Input ambient light sensor:
{"lighting":0}
```

```
Output ambient light sensor:
{"displayIntensity":5}
```

Listing 5.3: Ukázka vstupních a výstupních dat neuronových sítí ve formátu JSON.

Pokud by se někomu nechtělo vytvářet soubory, tak je k dispozici konstruktor, do kterého lze jako parametry vložit 2D double pole s trénovacími daty, v tomto případě si musí dát programátor pozor na rozměry pole, jeden rozměr značí počet vstupních neuronů sítě a druhý rozměr označuje počet trénovacích dat. K zajištění sběru správných dat pro neuronové sítě slouží rozhraní `IBaseController`, implementací jeho metod programátor zajistí, že aplikace bude přijímat od klienta data určená jako vstup do neuronových sítí.

5.3 Sběr dat od uživatele a zobrazení výsledku

Klientská část aplikace (UI) byla napsána ve frameworku FreeMarker. Ke komunikaci se serverem byl použit AJAX (Asynchronous JavaScript and XML) z knihovny jQuery pro JavaScript. FreeMarker je šablonový framework, to znamená, že programátor vytvoří šablonu, do které přidává proměnné a před odesláním stránky serverem na klienta se za proměnné dosadí hodnoty a vše se pošle jako HTML (HyperText Markup Language) stránka.

```
color: rgb(${fontR},${fontG},${fontB});
```

Listing 5.4: Ukázka z šablony pro HTML stránku.

```
model.addAttribute("fontR",
 out.getEmotionsOutput().getFontColor().getR());
model.addAttribute("fontG",
 out.getEmotionsOutput().getFontColor().getG());
model.addAttribute("fontB",
 out.getEmotionsOutput().getFontColor().getB());
```

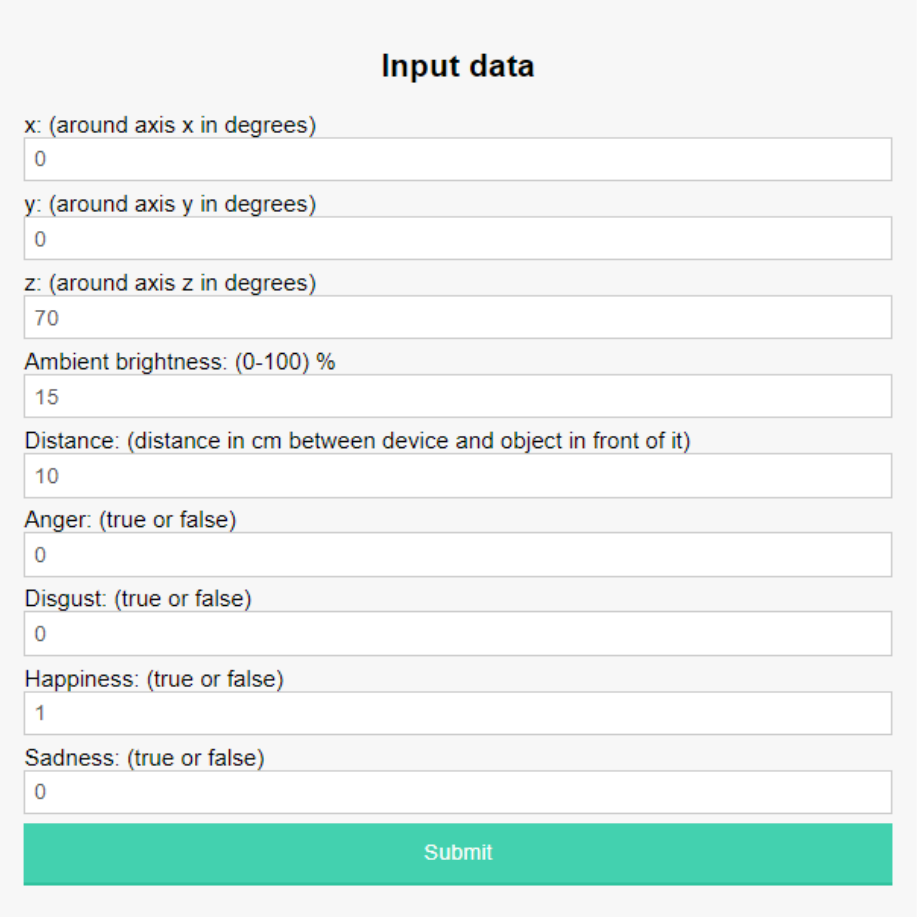
Listing 5.5: Ukázka naplnění šablony daty.

Uživatel vyplní HTML formulář, který je přes AJAX odeslán v JSON formátu na server ke zpracování neuronovou sítí, transformaci dat z formuláře na JSON objekt zajišťuje javascript funkce. Tuto funkci jsem nenapsal já, zdroj, ze kterého jsem funkci vzal, je uveden před zdrojovým kódem této funkce v souboru `script.js` . Požadavek přijme na serveru metoda controlleru, která je ve Springu schopna namapovat JSON objekt na objekt datového modelu,

využívá u toho Jackson knihovnu. Další průběh je znázorněn sekvenčním diagramem na obrázku 4.4.

5.4 Ukázka testovací aplikace

Ukázka funkcionality aplikace testující vytvořený framework z pohledu uživatele.



Obrázek 5.3: Ukázka formuláře vstupních dat.

Na 5.3 je webová stránka na které je formulář, který slouží jako vstupní data pro neuronovou síť. Na základě těchto dat probíhá adaptace uživatelského rozhraní. Tento formulář se skládá z následujících devíti vstupů.

- **Gyroskop**
  - **x**
    Otočení zařízení okolo osy x. Min 0, max 360.
  - **y**
    Otočení zařízení okolo osy y. Min 0, max 360.
  - **z**
    Otočení zařízení okolo osy z. Min 0, max 360.

Podle těchto tří vstupů framework rozhodne, zda se má displej otočit pro větší pohodlí uživatele.

- **Senzor osvětlení**
  - **Ambient brightness**
    Osvětlení okolo zařízení. Min 0 (naprostá tma), max 100 (přímý sluneční svit).

Podle údaje o okolním osvětlení framework rozhodne, jak moc intenzivní má být úroveň podsvícení displeje telefonu.

- **Senzor vzdálenosti**
  - **Distance**
    Vzdálenost objektu od přístroje. Min 0, max 20.

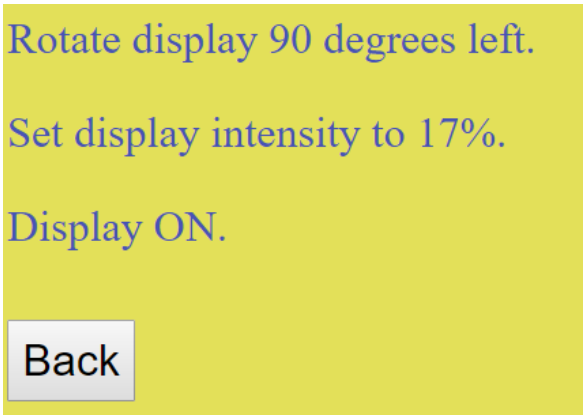
Na základě údaje o vzdálenosti objektu od zařízení framework rozhodne, zda má být displej vypnut nebo zapnut.

- **Emoce**
  - **Anger**
    Emoce popisující vztek. Min 0 (uživatel nepociťuje vztek), max 1 (uživatel pociťuje vztek).
  - **Disgust**
    Emoce popisující znechucení. Min 0 (uživatel nepociťuje znechucení), max 1 (uživatel pociťuje znechucení).

- **Happiness**
  Emoce popisující štěstí. Min 0 (uživatel nepociťuje štěstí), max 1 (uživatel pociťuje štěstí).
- **Sadness**
  Emoce popisující smutek. Min 0 (uživatel nepociťuje smutek), max 1 (uživatel pociťuje smutek).

Podle výběru emoce uživatele framework rozhodne o barvě pozadí a textu uživatelského rozhraní.

Na obrázku 5.4 je zobrazeno již adaptované uživatelské rozhraní na základě vstupu z obrázku 5.3.



Obrázek 5.4: Ukázka adaptovaného uživatelského rozhraní neuronovou sítí.

Kapitola 6

Testování a porovnávání

V této kapitole se budeme věnovat testování a vzájemné porovnávání různých konfigurací neuronových sítí, abychom zjistili, jaká konfigurace z testovaných je nejlepší pro konkrétní neuronovou síť. Měnit se bude jejich aktivační funkce a počet skrytých vrstev a neuronů v těchto vrstvách. U různých konfigurací neuronových sítí budu porovnávat chybu, s jakou se síť naučí na trénovacích datech, rychlost výpočtu neuronové sítě a nakonec každá konfigurace podstoupí zkoušku na několika testovacích vstupech, aby bylo vidět, jak dobře jaká konfigurace funguje.

Testované aktivační funkce jsou:

- lineární
- sigmoida
- hyperbolický tangens

Co se skrytých vrstev a počtu neuronů těchto vrstev týče, tak budou otestovány tyto případy:

- bez skryté vrstvy
- 1 skrytá vrstva s 5-ti neurony
- 3 skryté vrstvy s 5-ti neurony

- 1 skrytá vrstva s 20-ti neurony
- 3 skryté vrstvy s 20-ti neurony

Následující testování všech konfigurací bude probíhat na čtyřech různých vstupech. Tabulku těchto vstupů s očekávanými výstupy lze vidět na následující tabulce 6.1.

Input/Output data		Input				Expected output			
Test No.		1	2	3	4	1	2	3	4
Gyroscope	x	15	0	0	60	1	0	0	1
	y	0	0	60	32				
	z	50	0	0	200				
Distance	distance	6	10	34	50	14	20	78	90
Ambient light	intensity	1	20	15	12	0	1	1	1
Emotions	anger	0	1	0	0	[238, 201, 177, 71, 87, 111]	[114, 189, 232, 71, 83, 181]	[255, 199, 201, 122, 40, 52]	[242, 124, 56, 248, 248, 248]
	disgust	0	0	1	0				
	happiness	0	0	0	0				
	sadness	0	0	0	1				

Tabulka 6.1: Tabulka testovaných vstupů a výstupů.

Výsledky testování jsou zapsány v následujících tabulkách.

Neural network		Gyroscope		Distance		Light		Emotions	
		Avg. compute time [ns]	Train error	Avg. compute time [ns]	Train error	Avg. compute time [ns]	Train error	Avg. compute time [ns]	Train error
Linear	0 Hidden layers	34738	1.409	4670	0.1	5570	253.566	5286	0
	1 Hidden Layer, 5 neurons	9156	1.409	2832	0.1	3303	253.566	3398	0
	1 Hidden Layer, 20 neurons	9911	1.409	3115	0.1	4248	253.566	3682	0
	3 Hidden Layers, 5 neurons	10572	1.409	4342	0.1	4626	253.566	5.286	unstable
	3 Hidden Layers, 20 neurons	19729	1.41	9001	0.1	10685	253.566	10005	unstable
Sigmoid	0 Hidden layers	13027	1.486	4248	0	6419	5675.073	5192	29045.23
	1 Hidden Layer, 5 neurons	13121	1.639	6797	0.138	6042	5674.935	5475	29045.23
	1 Hidden Layer, 20 neurons	18879	1.639	6419	0	7363	5674.935	7552	29045.23
	3 Hidden Layers, 5 neurons	14065	1.571	8212	0.138	6986	5674.935	6703	29045.23
	3 Hidden Layers, 20 neurons	25770	1.639	27092	0.138	20295	5674.935	14915	29045.23
Tanh	0 Hidden layers	16048	1.513	4909	0.029	6419	5674.935	5286	29045.23
	1 Hidden Layer, 5 neurons	16425	1.429	5664	0	6702	5674.935	7174	29045.23
	1 Hidden Layer, 20 neurons	30207	1.417	9440	0	11045	5674.935	15198	29045.23
	3 Hidden Layers, 5 neurons	15670	1.432	7243	0	8968	5674.935	8779	29045.23
	3 Hidden Layers, 20 neurons	24260	1.64	14726	0	15103	5674.935	19257	29045.23

Tabulka 6.2: Tabulka průměrné doby výpočtu a trénovací chyby různých konfigurací neuronových sítí.

Modře obarvené buňky v tabulce 6.2 označují lepší výsledky, tyto konfigurace neuronových sítí jsou adepty na volbu výsledné nejlepší konfigurace. 0 ve sloupci Train error neznamená, že by se síť natrénovala na nulovou chybu, ale označuje velice malé číslo, kde se nenulová číslice nacházela až kolem sedmého

desetinného místa. Rozdíl v řádu stovek nanosekund trvání výpočtu považuji za minimální, proto beru výsledky s tímto rozdílem za stejné.

V následující tabulkách 6.3 a 6.4 jsou zaznamenány výstupy neuronových sítí ve všech předchozích konfiguracích v závislosti na testovacích vstupech z tabulky 6.1.

Neural network		Gyroscope				Distance				Light			
Test No.		1	2	3	4	1	2	3	4	1	2	3	4
Linear	0 Hidden layers	1	1	1	1	0	1	1	1	37	40	58	70
	1 Hidden Layer, 5 neurons	1	1	1	1	0	1	1	1	37	40	58	70
	1 Hidden Layer, 20 neurons	1	1	1	1	0	1	1	1	37	40	58	70
	3 Hidden Layers, 5 neurons	1	1	1	1	0	1	1	1	37	40	58	70
	3 Hidden Layers, 20 neurons	1	1	1	1	0	1	1	1	37	40	58	70
Sigmoid	0 Hidden layers	0	0	0	0	0	1	1	1	1	1	1	1
	1 Hidden Layer, 5 neurons	1	1	1	1	1	1	1	1	1	1	1	1
	1 Hidden Layer, 20 neurons	1	0	1	1	0	1	1	1	1	1	1	1
	3 Hidden Layers, 5 neurons	1	1	0	1	1	1	1	1	1	1	1	1
	3 Hidden Layers, 20 neurons	1	1	1	1	0	1	1	1	1	1	1	1
Tanh	0 Hidden layers	1	0	0	1	0	1	1	1	1	1	1	1
	1 Hidden Layer, 5 neurons	1	0	0	1	0	1	1	1	1	1	1	1
	1 Hidden Layer, 20 neurons	1	0	0	1	0	1	1	1	1	1	1	1
	3 Hidden Layers, 5 neurons	1	0	0	1	0	1	1	1	1	1	1	1
	3 Hidden Layers, 20 neurons	1	0	1	1	0	1	1	1	1	1	1	1
	Expected output	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**14**	**20**	**78**	**90**

Tabulka 6.3: Tabulka výstupů pro testovací vstupy různých konfigurací neuronových sítí.

Neural network		Emotions			
Test No.		1	2	3	4
Linear	0 Hidden layers	[238, 201, 177, 71, 87, 111]	[114, 189, 232, 71, 83, 181]	[255, 199, 201, 122, 40, 52]	[242, 124, 56, 248, 248, 248]
	1 Hidden Layer, 5 neurons	[238, 201, 177, 71, 87, 111]	[114, 189, 232, 71, 83, 181]	[255, 199, 201, 122, 40, 52]	[242, 124, 56, 248, 248, 248]
	1 Hidden Layer, 20 neurons	[238, 201, 177, 71, 87, 111]	[114, 189, 232, 71, 83, 181]	[255, 199, 201, 122, 40, 52]	[242, 124, 56, 248, 248, 248]
	3 Hidden Layers, 5 neurons	[240, 211, 181, 79, 81, 118]	[114, 187, 231, 69, 84, 180]	[254, 193, 199, 118, 43, 48]	[242, 124, 56, 248, 248, 248]
	3 Hidden Layers, 20 neurons	[238, 201, 177, 71, 87, 111]	[114, 189, 232, 71, 83, 181]	[255, 199, 201, 122, 40, 52]	[242, 124, 56, 248, 248, 248]
Sigmoid	0 Hidden layers	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
	1 Hidden Layer, 5 neurons	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
	1 Hidden Layer, 20 neurons	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
	3 Hidden Layers, 5 neurons	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
	3 Hidden Layers, 20 neurons	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
Tanh	0 Hidden layers	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
	1 Hidden Layer, 5 neurons	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
	1 Hidden Layer, 20 neurons	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
	3 Hidden Layers, 5 neurons	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
	3 Hidden Layers, 20 neurons	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 1, 1, 1]
	Expected output	**[238, 201, 177, 71, 87, 111]**	**[114, 189, 232, 71, 83, 181]**	**[255, 199, 201, 122, 40, 52]**	**[242, 124, 56, 248, 248, 248]**

Tabulka 6.4: Tabulka výstupů pro testovací vstupy různých konfigurací neuronové sítě zabývající se emocemi uživatele.

V tabulkách 6.3 a 6.4 je vidět, jak správně dokázaly různé konfigurace

neuronových sítí vypočítat výsledek, zeleně obarvené buňky znamenají očekávaný výsledek a červeně obarvené buňky znamenají naprosto špatný výsledek. Konfigurace s červeně zbarvenými buňkami nemají pro danou neuronovou síť smysl, zato konfigurace se zeleně zbarvenými buňkami s výsledky jsou pro danou síť nejlepší možnou volbou konfigurace z testovaných.

6.1 Zhodnocení testování

Spojením tabulek 6.2,6.3 a 6.4 dostáváme odpověď na otázku, jaké konfigurace neuronových sítí se nejlépe hodí pro tuto aplikaci. Udělejme si přehled pro každou neuronovou síť podle jejího zaměření.

Gyroskop - očekávaných výstupů dosáhla síť pouze v konfiguraci za použití aktivační funkce tanh (hyperbolický tangens). Z tabulky 6.2 vyplývá, že nejlepší možností je vybrat konfiguraci s 1 skrytou vrstvou s 5-ti neurony nebo 3 skryté vrstvy s 5-ti neurony.

Senzor přiblížení - tato síť dosahovala výborné výsledky s hyperbolickým tangens a lineární aktivační funkcí, ale lineární funkce poráží hyperbolický tangens v rychlosti výpočtu, i když s malou trénovací chybou. Proto zde mohu doporučit použít obě aktivační funkce a to v konfiguraci s 1 skrytou vrstvou s 5-ti neurony.

Senzor okolního osvětlení - tato neuronová síť dosahovala vyloženě špatných výsledků v konfiguraci s aktivačními funkcemi sigmoid a tanh, proto pro tento problém doporučuji z testovaných zvolit lineární aktivační funkci s jednou skrytou vrstvou s 5-ti neurony. V této konfiguraci síť sice také nepodávala očekávané výsledky, ale byly alespoň reálnější.

Emoce - stejně jako předchozí neuronová síť i tato naprosto selhala v kombinaci s aktivačními funkcemi sigmoid a tanh, ale narozdíl od předchozí sítě, tato síť s lineární aktivační funkcí vracela očekávané výsledky ve 4 z 5 konfiguracích. Vzhledem k tabulce 6.2, doporučuji pro tuto síť použít konfiguraci s jednou nebo třemi skrytými vrstvami s 5-ti neurony.

Kapitola 7

Instalace

Framework potřebuje ke zkompilování následující programy: Java verze 8 a vyšší, Maven verze 3 a vyšší.

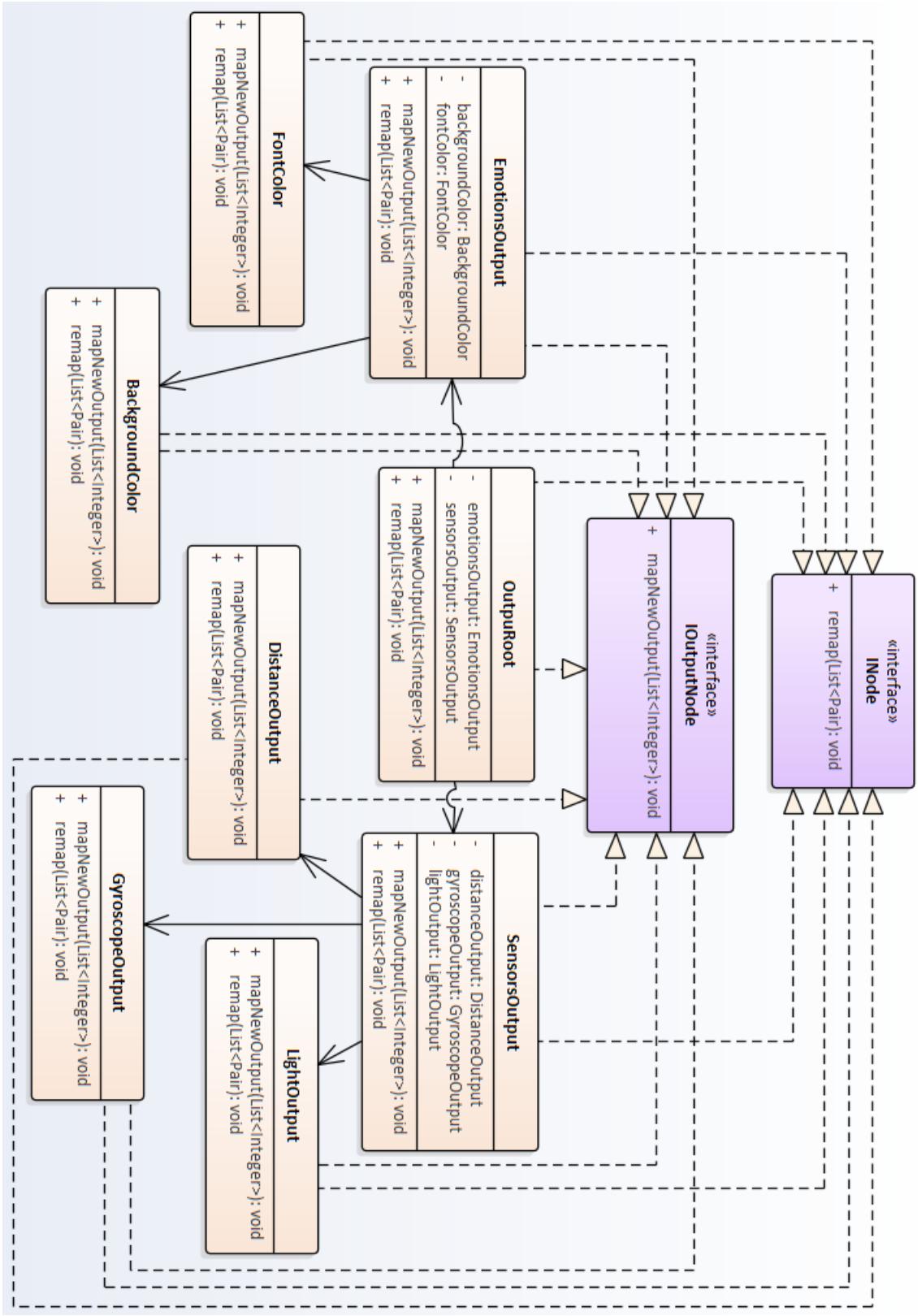
- Vložte CD do mechaniky.
- Zkopírujte soubor AUINeuralNetwork.zip na svůj disk a zde soubor rozbalte.
- Pomocí vývojového prostředí otevřete projekt “testApp” a k němu naimportujte projekt “Framework” jako modul z existujícího zdroje (Maven).
- Nyní můžete spustit testovací aplikaci přes vývojové prostředí.

Kapitola 8

Závěr

Cílem práce bylo nastudovat některé z typů adaptace uživatelského rozhraní a jejich následné propojení s neuronovými sítěmi. Práce se zabývá čtyřmi oblastmi adaptace uživatelského rozhraní, jsou to senzory gyroskop, přiblížení a osvětlení a změna uživatelského rozhraní podle emocí uživatele. Na tyto oblasti byl vymyšlen datový model, který umožňuje snadné převádění dat mezi textem čitelným pro člověka, objektové reprezentace v aplikaci a vstupem/výstupem neuronových sítí. Následně byl implementován framework využívající vytvořeného datového modelu a neuronových sítí k adaptaci uživatelského rozhraní. Tento framework byl otestován Java Spring aplikací s různými konfiguracemi neuronových sítí. Testování odhalilo, že některé konfigurace mají lepší výsledky než jiné pro určitý typ vstupu a výstupu a jiné konfigurace mají výsledky velmi špatné. Výsledky jsou shrnuty v kapitole Zhodnocení testování 6.1. Za některými špatnými výsledky může být zvolená množina trénovacích dat, která v některých případech nebyla dostatečně obsáhlá, aby se neuronová síť dokázala co nejlépe naučit. Ale i přesto byla pro každý typ adaptace uživatelského rozhraní nalezena konfigurace neuronové sítě, která podávala výsledky, které se dají označit za velmi dobré. Do budoucna by se dala práce rozšířit o více typů neuronových sítí (například rekurzivní neuronové sítě), tato práce testovala pouze MLP (Multi-Layered Perceptron) v několika konfiguracích. Dále by se daly otestovat další aktivační funkce a více kombinací vnitřních vrstev a počtu neuronů v nich. Práce by se také dala rozšířit o další případy adaptace uživatelského rozhraní.

Přílohy



Obrázek 1: Konceptuální model výstupu neuronové sítě.

Příloha A

Literatura

[21] *Apache freemarker [online].*, Dostupné z: https://freemarker.apache.org/.

[BAJ] Kreeti; JAIN Neeti. BAJPAI, Saumya; JAIN, *Artificial neural networks.*, International Journal of Soft Computing and Engineering (IJSCE), 2011,1.NCAI2011.

[BAR] Kalyan Pathapati; DANTU Ram. BARTHOLD, Christopher; SUBBU, *Evaluation of gyroscope-embedded mobile phones.*, In: Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on. IEEE, 2011. p. 1632-1638.

[BAS] M. BASHEER, Imad A.; HAJMEER, *Artificial neural networks: fundamentals, computing, design, and application.*, Journal of microbiological methods, 2000, 43.1: 3-31.

[Def] Spring Bean Definition., *Tutorials point [online].*, Dostupné z: https://www.tutorialspoint.com/spring/spring_bean_definition.htm.

[EKM] Wallace V. EKMAN, Paul; FRIESEN, *Unmasking the face: A guide to recognizing emotions from facial clues.*, Ishk, 2003.

[Fraa] Encog Machine Learning Framework., *Heaton research [online].*, Dostupné z: http://www.heatonresearch.com/encog/.

[Frab] Spring Framework., *Pivotal software [online].*, Dostupné z: https://spring.io/.

[ftiAS] BrightnessGate for the iPhone & Android Smartphones., *Display mate [online].*, Dostupné z: http://www.displaymate.com/AutoBrightness_Controls_2.htm.

[GAR] S. R. GARDNER, Matt W.; DORLING, *Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences.*, Atmospheric environment, 1998, 32.14-15: 2627-2636.

[GOL] John H. GOLDBERG, David E.; HOLLAND, *Genetic algorithms and machine learning.*, Machine learning, 1988, 3.2: 95-99.

[Gyr] Gyroscope., *Z mathworks [online].*, Dostupné z: https://www.mathworks.com/help/supportpkg/android/ref/gyroscope.html.

[iMP] Exclusive: Proximity Sensing in Mobile Phones., *Z wireless design mag [online].*, Dostupné z: https://www.wirelessdesignmag.com/article/2014/01/exclusive-proximity-sensing-mobile-phones.

[JAI] Jianchang; MOHIUDDIN K. Moidin. JAIN, Anil K.; MAO, *Artificial neural networks: A tutorial.*, Computer, 1996, 29.3: 31-44.

[KAR] A. Vehbi. KARLIK, Bekir; OLGAC, *Performance analysis of various activation functions in generalized mlp architectures of neural networks.*, International Journal of Artificial Intelligence and Expert Systems, 2011, 1.4: 111-122.

[LEF] James T. LEFF, Avraham; RAYFIELD, *Web-application development using the model/view/controller design pattern.*, In: Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International. IEEE, 2001. p. 118-127.

[LUN] Anastasiia. LUNOVA, *Adaptace ui založena na emocích uživatele.*, Prague 2017. Bachelor thesis. Departmen of Computer Graphics and Interaction, CTU FEE. Leader J. Šebek.

[Net] Applied Deep Learning Part 1: Artificial Neural Networks., *Z towards data science [online].*, Dostupné z: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6.

[ns] Umělá neuronová síť., *Wikipedia [online].*, Dostupné z: https://cs.wikipedia.org/wiki/Um

[OPP] Reinhard. OPPERMANN, *User-interface design.*, Handbook on information technologies for education and training. Springer, Berlin, Heidelberg, 2002. p. 233-248.

[oui] Types of user interface., *w3computing [online]*, Dostupné z: http://www.w3computing.com/systemsanalysis/types-user-interface/.

[SCH] et al. SCHMIDT, Douglas C., *Pattern-oriented software architecture, patterns for concurrent and networked objects.*, John Wiley & Sons, 2013.

[TIT] Tamara. TITOVA, *Využití aspektově orientovaného přístupu a lokálních dat ve vývoji adaptivního uživatelského rozhraní pro android.*, Prague 2016. Bachelor thesis. Department of Economics, Management and Humanities. Leader J. Šebek.

[tNN] Introduction to Neural Networks., *In: learnartificialneuralnetworks [online].*, Dostupné z: http://www.learnartificialneuralnetworks.com/introduction-to-neural-networks.html.

Příloha B

Přiložené CD

- AUINeuralNetwork.zip - implementace frameworku a testovací apliakce
- enterprise_architect_diagrams - adresář s Enterprise architect projektem, ve kterém jsou použité UML diagramy
- LaTeX_project - zdrojové kódy textové části práce, použité obrázky
- bakalarskaPraceJiriFrantisek.pdf - elektronická verze bakalářské práce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **František** Jméno: **Jiří** Osobní číslo: **406892**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Využití neuronové sítě při adaptaci uživatelského rozhraní

Název bakalářské práce anglicky:

Use of Artificial Neural Network for adaptive user interface

Pokyny pro vypracování:

1. Prostudujte existující nástroje pro podporu aspektové orientovaného programování (AOP) a neuronové sítě. [1][2][3]
2. Seznamte se s možnostmi adaptivního uživatelského rozhraní. [4]
3. Zpracujte rešerši ohledně vývoje aplikace, která využívá neuronové sítě a existující práce na téma adaptivní uživatelské rozhraní.
4. Navrhněte a implementujte framework, který využije tyto informace při generování uživatelského rozhraní.
5. Otestujte aplikaci a testy vyhodnoťte

Seznam doporučené literatury:

1. A. Abraham, Artificial Neural Networks. In: Handbook of Measuring System Design, 2005
2. ScienceDirect. Deep learning in neural networks: An overview. Sciencedirect.com [online]. Dostupné z URL: http://www.sciencedirect.com/science/article/pii/S0893608014002135
3. M. Kubat, An Introduction to Machine Learning, Springer International Publishing AG 2015, 2017
4. Šebek, J., Trnka, M., Černý, T. On Aspect-oriented Programming in Adaptive User Interfaces. In: Proceedings of the 2nd International Conference on Information Science and Security. The 2nd International Conference on Information Science and Security, Seoul, 2015-12-14/2015-12-16. IEEE, 2015, pp. 147-151.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jiří Šebek, laboratoř inteligentního testování softwaru FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **19.01.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **30.09.2019**

Ing. Jiří Šebek	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce		podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta