VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

AUTOMATICKÁ KLASIFIKACE DIGITÁLNÍCH MODULACÍ POMOCÍ NEURONOVÝCH SÍTÍ
AUTOMATIC CLASSIFICATION OF DIGITAL MODULATIONS USING NEURAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Alexander Sinyanskiy**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Anna Kubánková, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Alexander Sinyanskiy ***ID:*** 183598
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Automatická klasifikace digitálních modulací pomocí neuronových sítí

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte digitální modulace ASK, FSK, PSK a QAM. Seznamte se s principy klasifikace digitálních modulací. Seznamte se s modulem neuronových sítí v prostředí Matlab. Vytvořte modulované signály a navrhněte příznaky, které se budou používat pro klasifikaci digitálních modulací. Vytvořte neuronovou síť pro klasifikaci. Vytvořte trénovací množinu a proveďte učení neuronové sítě. Analyzujte úspěšnost rozpoznávání signálů bez šumu a se šumem. Podle potřeby optimalizujte parametry neuronové sítě.

DOPORUČENÁ LITERATURA:

[1] ELSAYED AZZOUZ AND A.K. NANDI. Automatic modulation recognition of communication signals. New York: Springer, 2011. ISBN 1441951660.

[2] ELSAYED AZZOUZ AND A.K. NANDI. Algorithms for automatic modulation recognition of communication signals, IEEE Transactions on Communications, vol. 46, pp. 431-436, 1998.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: Ing. Anna Kubánková, Ph.D.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

Abstrakt

Tato diplomová práce se zabývá automatickou klasifikací digitálních modulací pomocí neuronových sítí. V práci je stručně popsána problematika a existující algoritmy řešení problému rozpoznávání modulace. Bylo zjištěno, že nejlepších výsledků bylo dosaženo při použití příznakové metody rozpoznávání umělé neuronové sítě. Takže byly teoreticky popsány digitální modulace, které byly vybrány pro rozpoznávání, a to jsou ASK, FSK, BPSK, QPSK a 16QAM. Tyto modulace jsou v dnešní době používány nejčastěji. Dále byla stručně popsán teorie neuronových sítí. V další části byla věnována pozornost charakteristickým příznakům modulací pro rozpoznávání modulací pomocí umělých neuronových sítí. V předposlední části jsou popsány parametry signálů pro simulace v prostředí Matlab, postup vytvoření klíčových příznaků v prostředí Matlab a analýza výsledků experimentální simulace. Poslední část obsahuje experimenty optimalizace neuronové sítě.

Klíčová slova

Digitální modulace, rozpoznávání modulací, klasifikace modulací, umělé neuronové sítí, klíčové příznaky.

Abstract

This master's thesis is about automatic digital modulation recognition using artificial neural networks. The paper briefly describes the issue and existing algorithms for solving the problem of modulation recognition. It was found that the best results are achieved when using the feature-recognition methods and artificial neural networks. The digital modulations that were chosen for recognition are described theoretically and they are ASK, FSK, BPSK, QPSK and 16QAM. These modulations are most commonly used today. Later was briefly described theory of neural networks. In another part was given to the characteristic features of modulation for modulation recognition using artificial neural networks. The penultimate part describes the parameters for signal simulation in Matlab, how to create the key features in Matlab and results after experimental simulation. The last part contains neural network optimization experiments.

Keywords

Digital modulation, modulation recognition, modulation classification, artificial neural networks, key features.

SINYANSKIY, A. Automatická klasifikace digitálních modulací pomocí neuronových sítí. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 73 s. Vedoucí diplomové práce Ing. Anna Kubánková, Ph.D..

Prohlášení

„Prohlašuji, že svou závěrečnou práci na téma „Automatická klasifikace digitálních modulací pomocí neuronových sítí“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ……

podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizovaný v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systémů (SIX); registrační číslo CZ.1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

Poděkování

Děkuji vedoucí diplomové práce paní Ing. Anně Kubánkové, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ……

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1. ÚVOD

Přechod na novou generaci počítačů vytvořených pomocí technologie neuronových sítí bude souviset s řadou změn, které umožní hovořit o zcela nové generaci počítačů. Pomocí neuronových sítí, nanotechnologie a další inovace budou vytvořené miniaturní technické zdroje, které budou mít až stokrát větší objem pamětí, s výpočetní rychlostí více než miliardkrát, s neuvěřitelnou řadou dalších funkcí a operací s minimální spotřebou energie a vysokou spolehlivostí. Jejich základem bude schopnost garantovat plynulý provoz a bezporuchovost.

Vývoj integrovaných telekomunikačních systémů, přechod k informacím vysokorychlostního toku, nové modulační a kódovací techniky zadaly řadu úkolů, jejichž úspěšné řešení volá po nových, netradičních metodách zpracování informací. Jde o návrh jednoduchých, efektivních a dostatečně spolehlivých metod zpracování signálu v telekomunikačních kanálech s poruchou. Důležitá role při řešení těchto problémů je dána použitím automatizovaných prostředků a komplexů příjmu a zpracování signálu.

Inteligentní přijímač musí umět automaticky rozpoznávat formát modulace detekovaného signálu. Z hlediska tohoto aspektu je věnována velká pozornost výzkumu a přípravě algoritmu pro rozpoznávání modulovaných signálů.

K nejznámějším digitálním modulacím patří ASK (Amplitude Shift Keying - amplitudové klíčování), BPSK (Binary Phase Shift Keying - dvoustavové fázové klíčování), QPSK (Quadrature Phase Shift Keying - čtyřstavové fázové klíčování), FSK (Frequency Shift Keying - frekvenční klíčování), QAM (Quadrature Amplitude Modulation – kvadraturní amplitudová modulace). K nejznámějšímu prostředí pro analýzy, modulaci signálu a využití umělých neuronových sítí patří Matlab (interaktivní programové prostředí a skriptovací programovací jazyk čtvrté generace).

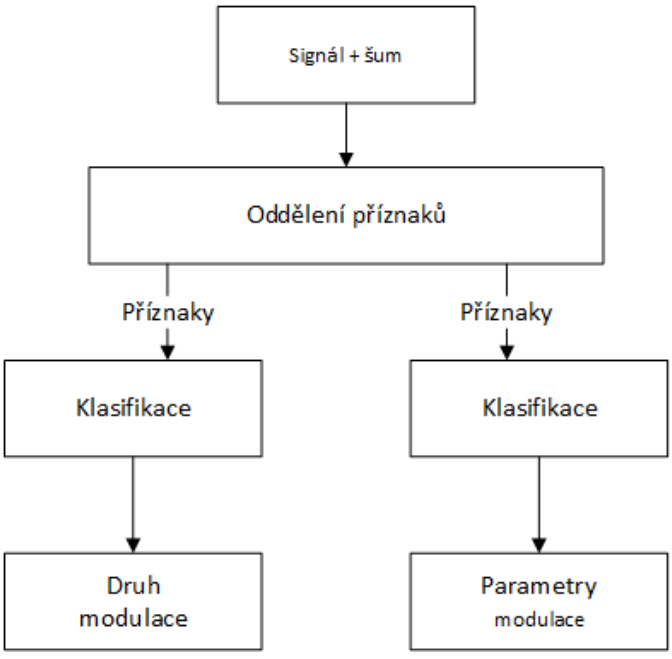
Ve své diplomové práci se tedy budu zabývat realizací metody a algoritmů pro rozpoznávání typů digitálních modulací pomocí neuronových sítí, které umožní klasifikovat modulace na přijímači při neznámých parametrech signálu. Pro rozpoznávání byly vybrány tyto typy digitálních modulací: BASK, BFSK, BPSK, QPSK a 16QAM. Uvedené typy jsou nejčastěji používány ve většině současných standardů komunikací. V případě potřeby realizace metody bude navržena optimalizace neuronové sítě.

2. SOUČASNÝ STAV PROBLEMATIKY

Charakteristickým příznakem moderních komunikačních sítí a systémů je použití různých metod pro adaptace, při jejichž použití se v komunikačních kanálech z různých důvodů můžou změnit typ modulace signálů a parametry, použité v rámci stejné relace. Nejjednodušší příklad je změna rychlosti provozu v závislosti na kvalitě komunikačních kanálů. Nicméně existují složitější metody adaptace. Kromě toho, jestliže předem není známa úplná informace o parametrech přijímaného signálu, vyřešení některých problémů jeho zpracování, jako je nastavení demodulátoru, rozhodnutí demodulátoru o správně přijatém modulačním parametru, vyžaduje co nejpřesnější znalost struktury modulace přicházející ze signálu vzduchem. Tyto faktory přispívají k tomu, aby při zprostředkování signálů se daly automaticky určit parametry a typy modulací přijímaných signálů[1].

Stanovení neznámých modulačních parametrů široké třídy přijímaných signálů je obecně složitý úkol, který vyžaduje vysoce složité výpočty. To vytváří obtížnost při provádění postupů automatického rozpoznávání typu a parametrů modulace, pracujících v reálném čase.

Řešení problému otevírání struktury modulačního signálu zahrnuje určení sady atributů, které popisují požadované typy modulace a jejich parametry, výběr kombinace vlastností přijímaného signálu, a směs hluku a vlastností s cílem rozhodnout parametry modulace analyzovaného signálu. Obecné schéma systémy analýzy je na obr. 2.1.[1]



Obr. 2.1 Obecné schéma systému analýzy

Vzhledem k současnému trendu používat digitální komunikaci místo analogových komunikací se většina publikací v oblasti rozpoznávaní analogových modulaci (AMR – analog modulation recognition) zabývá digitálními komunikačními signály. Tato publikace používá nejrozšířenější metody automatické klasifikace modulací, a to jsou statistické rozhodování (Decision Theory – DT) a příznakové metody rozpoznávání (Feature Based – FB).

Statistické rozhodování je oblast výzkumu, zahrnující pojmy a metody matematiky, statistiky, ekonomie, managementu a psychologie, s ohledem na výběr lidí při řešení problémů a úkolů, a je taky založena na způsobu, jak dosáhnout požadovaného výsledku.[2] Pro návrh DT klasifikátoru je nutné znát přesně parametry přijatého signálu, jako je přenosová rychlost, frekvence nosné a frekvenční zdvih. Metody statického rozhodovaní jsou optimální v tom smyslu, že jsou schopné minimalizovat pravděpodobnost chybné klasifikace. V ideálních podmínkách jsou schopny dosáhnout vysoké pravděpodobnosti rozpoznávaní typu modulace, ale musí mít relativně krátký rozsah přijatých dat. Kvůli těmto ideálním podmínkám mohou být použity jako měřítko pro hodnocení výkonu ostatních klasifikátorů.

Nicméně metody DT jsou v praxi omezené. Poprvé byly tyto metody použity jen pro určitý typ modulace. To znamená, že metoda určená pro PSK nemůže byt použita pro rozpoznávání modulace FSK. Za druhé, aby se snížila výpočetní složitost, používá se často přibližná realizace metod DT, což vede k neoptimálním řešením. Ale i když metoda byla zjednodušena, mnoho metod statického rozhodování je výpočetně složitější.

Pro příznakovou metodu musíme vyjádřit některé příznaky z přijatého signálu, jejichž analýzou můžeme definovat typ modulace. Příznakové metody rozpoznávání jsou obecně jednodušší než statistické rozhodováni z hlediska výpočetní složitosti, a jejich implementace je snadná. Často se jako klasifikátor používají umělé neuronové sítě (Artificial Neural Network – ANN)[3].

Ačkoli není optimální, metody obvykle potřebují méně nebo dokonce žádné předchozí znalosti o přijatých signálech, pokud budou dobře navrženy. Nicméně metody FB často vyžadují, aby přijatá sekvence obsahovala více symbolů. Za druhé, pro většinu FB je obtížné najít univerzální prahové hodnoty nebo rozhodovací oblasti, které se automaticky přizpůsobí signál šum (SNR - Signal to Noise Ratio) podmínky a parametry modulace - ve skutečnosti, v mnoha klasifikacích byly použity empirické nebo experimentálně stanovené prahové hodnoty nebo rozhodovací oblasti. Za čtvrté, některé příznakové metody rozpoznávání také závisí na přesných odhadech parametrů modulace.

Jak již bylo řečeno dříve, metody DT jsou omezené v praxi a protože to chceme rozpoznávat automaticky, budeme se zabývat jen příznakovými metodami. V tab. 2.1.

jsou shrnuty různé publikované přístupy řešení problému automatické klasifikace digitální modulace pomocí příznakové metody[3].

Tab. 2.1 Publikované přístupy (převzato z [3])

Autor přístupu	Typy charakteristických vlastností	Klasifikační metoda	Typy modulací	Podmínky
Liedtke [4]	Charakteristické vlastnosti v časové doméně	FB	ASK2, FSK2, PSK2, PSK4, PSK8	SNR>18 dB
Soliman-Hsue [5]	Charakteristické vlastnosti v časové doméně	FB	MPSK	
Whechel a kol. [6]	Charakteristické vlastnosti v časové doméně	FB (ANN)	QPSK, QASK	
Ghani-Lamontagne [7]	Spektrální charakteristické vlastnosti	FB (ANN)	ASK, BPSK, QPSK, NCFSK, CPFSK, FSK	
Azzouz-Nandi [8], [9]	Spektrální charakteristické vlastnosti a v časové doméně	FB	ASK2, ASK4, PSK2, PSK4, FSK2, FSK4	
Hero-Hadinejad Mahram [10]	Charakteristické vlastnosti v časové doméně (konstelační diagram)	FB	MPSK, QAM	
Lallo[11]	Spektrální charakteristické vlastnosti	FB	MPSK, QAM, MFSK	

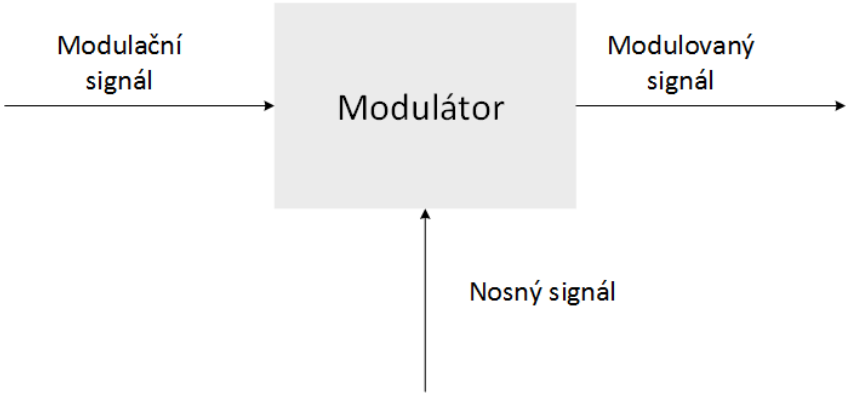
Mobasseri[12]	Charakteristické vlastnosti v časové doméně (konstelační diagram)	FB	QAM	
Boudreau[13]	Charakteristické vlastnosti Azzouz-Nandi	FB	FSK, PSK	
Lopatka-Pedzisz[14]	Charakteristické vlastnosti v časové doméně	FB	ASK, 4DPSK, 16QAM, FSK	Známa f_c, SNR>5dB
Nandi-Wong [15]	Spektrální charakteristické vlastnosti a v časové doméně	FB (ANN)	ASK2, ASK4, BPSK, QPSK, FSK2, FSK4 QAM16, QAM64	
Taira [16]	Charakteristické vlastnosti v časové doméně (konstelační diagram)	FB (ANN)	MQAM	
Kalinin-Kavalov [17], [18]	Charakteristické vlastnosti v časové doméně	FB (ANN)	BPSK, QPSK, QAM16	Známa f_c, délka symbolu, výkon signálu koherence
Delgosha-Menhaj [19]	Charakteristické vlastnosti v časové doméně	FB (ANN)	QPSK, SQPSK, MSK	SNR>8dB
Ramakonar [20]	Charakteristické vlastnosti v časové doméně	FB	FSK4, FSK8	

Spooner [21]	Charakteristické vlastnosti v časové doméně	FB	QAM, PSK	
Nikoofar [22]	Charakteristické vlastnosti v časové doméně (konstelační diagram)	FB	QAM, PSK	
Hsue-Soliman [23]	Charakteristické vlastnosti v časové doméně	FB	MPSK, MFSK	

Nejlepších výsledků bylo dosaženo při použití příznakové metody rozpoznávání a umělé neuronové sítě. Tato metoda bude podrobněji rozebrána dále.

3. DIGITÁLNÍ MODULACE

V oblasti telekomunikací je modulace proces předávání signálu zprávy, například digitální tok bitů, nebo analogový zvukový signál, uvnitř jiného signálu, který může být fyzicky předán. Toto je proces ovlivňování některého z parametrů nosného signálu signálem modulačním, který je zobrazen na obr. 2.1.



Obr. 3.1 Signály modulace s harmonickou nosnou vlnou (převzato z [24])

Díky tomu můžeme rozdělit digitální modulace na tyto typy:

- amplitudové klíčování – ASK (Amplitude Shift Keying),
- frekvenční klíčování – FSK (Frequency Shift Keying),
- fázové klíčování – PSK (Phase Shift Keying),
- složené digitální modulace – nejznámější je QAM (Quadrature Amplitude Modulation), která kombinuje amplitudové a fázové klíčování.

Všechny z výše uvedených typů klíčování taky můžeme rozdělit do různých typů vícestavové digitální modulace. Počet stavů modulace se označuje číslem před jejím názvem. Existuje vztah mezi počtem stavů modulace a počtem přenášených bitů, který lze popsat rovnicí[25]:

$$B = log_2 M \tag{3.1}$$

Kde:

B je počet bitů přenesených v jednom signálovém prvku,

M je celkový počet stavů signálu.

V digitální modulaci je analogový nosný signál modulován diskrétním signálem. Digitální modulační metody mohou být považovány za analog-to-digital převod a odpovídající demodulaci nebo detekci, jako konverzi analogového signálu na digitální.

V této kapitole budou popsány principy jednotlivých digitálních modulací, které byly vybrány pro rozpoznávání.

3.1 Modulace ASK

Klíčování amplitudovým posuvem (Amplitude-shift keying, ASK) je forma amplitudové modulace, která představuje digitální data jako změny amplitudy nosné vlny[24]. U modulace ASK je binární symbol 1 reprezentován přenosem s pevnou amplitudou nosné vlny a pevnou frekvencí. ASK je dobrá, pokud jde o využití šířky pásma, ale s výhradou zkreslení v přítomnosti šumu a není efektivní z hlediska spotřeby energie.

Amplitudové klíčování lze popsat rovnicí [24]:

$$S_{ASK}(t) = s_c(t) * g(t) \tag{3.2}$$

Kde:

$s_c(t) = S_c * cos\omega_c t$ je harmonický nosný signál,

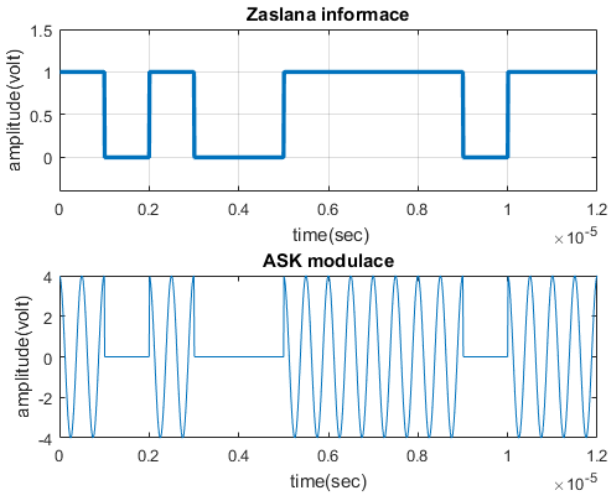
S_c je amplituda,

ω_c je úhlový kmitočet,

t je doba trvání signálového prvku,

$g(t) = \begin{matrix} \therefore 1 \\ \therefore 0 \end{matrix}$ je obdélníkový modulační signál.

Modulovaný signál BASK a zaslaná zpráva jsou zobrazené na obr. 2.2.



Obr. 3.2 ASK signál (viz níže) a zpráva (viz výše)

Morseova abeceda je často přenášena pomocí amplitudové modulace díky tomu, že modulační a demodulační procesy jsou poměrně levné. ASK technika je také běžně používaná pro přenos digitálních dat přes optické vlákno. Pro LED vysílače je binární 1 reprezentována krátkým pulsem světla a binární 0 je reprezentována nepřítomností světla. Pro rozpoznávání byla zvolena modulace BASK.

3.2 Modulace FSK

Klíčování frekvenčním posuvem (Frequency-shift keying, FSK) je metoda frekvenční modulace, u které se přenáší digitální informace pomocí diskrétních změn frekvence nosné vlny[26]. Nejjednodušší FSK je binární FSK (BFSK), který používá dvě frekvence pro přenos binární informace. Nula má signálový prvek s nižším kmitočtem, jednička má signálový prvek s vyšším kmitočtem. U modulace FSK má nosná vlna konstantní amplitudu. Pro rozpoznávání byla zvolena modulace BFSK.

U BFSK lze představení binárních 1 a 0 popsat rovnicemi[27]:

$$s_1(t) = S_c \cos(2\pi f_1 t + \Phi_1)\text{ , pro 0} \tag{3.3}$$

$$s_2(t) = S_c \cos(2\pi f_2 t + \Phi_2)\text{, pro 1} \tag{3.4}$$

Kde:

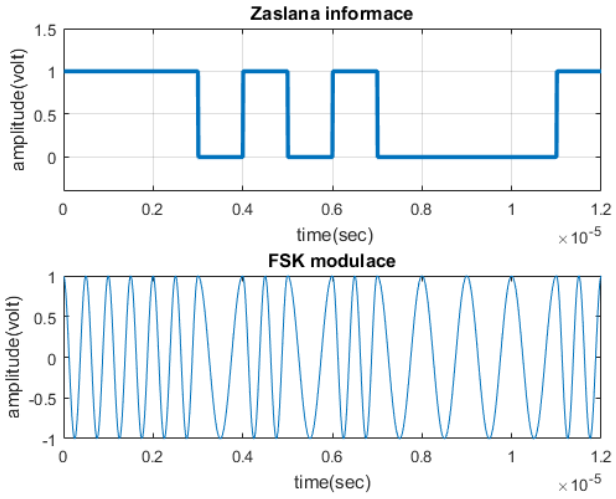
S_c je amplituda,

t je doba trvání signálového prvku,

f_1 a f_2 jsou různé frekvence,

Φ_1 a Φ_2 jsou počáteční fázi.

Modulovaný signál BFSK a zaslaná zpráva jsou zobrazené na obr. 2.3.



Obr. 3.3 FSK signál (viz níže) a zpráva (viz výše)

FSK je nejčastější forma digitální modulace ve vysokofrekvenčním rádiovém spektru, a má důležité využití u telefonních okruhů. Na rozdíl od ASK je kmitočtové klíčování energeticky úsporné, ale nevyužívá efektivně šířku pásma.

3.3 Modulace PSK

Klíčování fázovým posuvem (Phase-shift keying, PSK) je metoda digitální modulace, která pro přenos informací používá změny fáze nosné vlny. U modulace PSK nemá nosná vlna konstantní amplitudu, protože pásmové omezení ukládá amplitudové informace na přechodech mezi po sobě následujícími symboly. PSK modulace je energeticky úsporná a efektivně využívá šířku pásma. Historické voice-band synchronní modemy používají PSK. Pro rozpoznávání byly zvoleny modulace BPSK a QPSK, které jsou dále stručně popsány.

3.3.1 Modulace BPSK

Založená na posunutí fáze harmonické nosné o 0° nebo 180° v závislosti na hodnotě binárního modulačního signálu. U modulace BPSK má nosná vlna konstantní amplitudu.

Dvoustavové fázové klíčování lze popsat rovnicí[24]:

$$S_{BPSK}(t) = s_c(t) * g(t) \tag{3.5}$$

Kde:

$s_c(t) = S_c * cos\omega_c t$ je harmonický nosný signál,

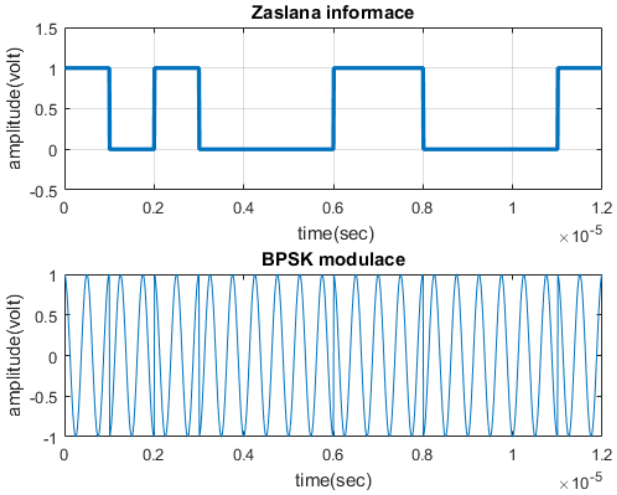
S_c je amplituda,

ω_c je uhlový kmitočet,

t je doba trvání signálového prvku,

$g(t) = \begin{matrix} \because 1 \\ \therefore -1 \end{matrix}$ je obdélníkový modulační signál (bipolární NRZ).

Modulovaný signál BPSK a zaslaná zpráva jsou zobrazené na obr. 2.4.



Obr. 3.4 BPSK signál (viz níže) a zpráva (viz výše)

Používá se například u Bluetooth, WiFi a RFID (identifikace na rádiové frekvenci).

3.3.2 Modulace QPSK

Kvadraturní fázové klíčování používá čtyři signálové prvky, vyjádřené nosnou vlnou s odlišnou počáteční fází. Pomocí čtyř fází, každému signálovému prvku odpovídá jedna bitová dvojice tj. dibit[24]. U modulace QPSK má nosná vlna konstantní amplitudu. Díky tomu je možné rychlost oproti BPSK zdvojnásobit nebo nechat stejnou rychlost a použít poloviční šířku pásma.

QPSK je možno popsat rovnicí[24]:

$$s(t) = \frac{S_c}{\sqrt{2}}[S_{bI}(t)\cos(\omega_c t) + S_{bQ}(t)\sin(\omega_c t)] \tag{3.6}$$

Kde:

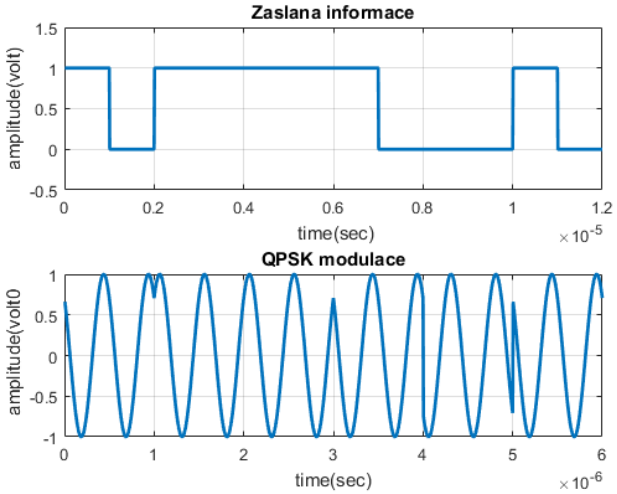
S_c je amplituda,

ω_c je uhlový kmitočet,

t je doba trvání signálového prvku,

S_{bI} a S_{bQ} jsou bipolární signály NRZ (Non Return To Zero).

Modulovaný signál QPSK a zaslaná zpráva jsou zobrazené na obr. 2.5.



Obr. 3.5 QPSK signál (viz níže) a zpráva (viz výše)

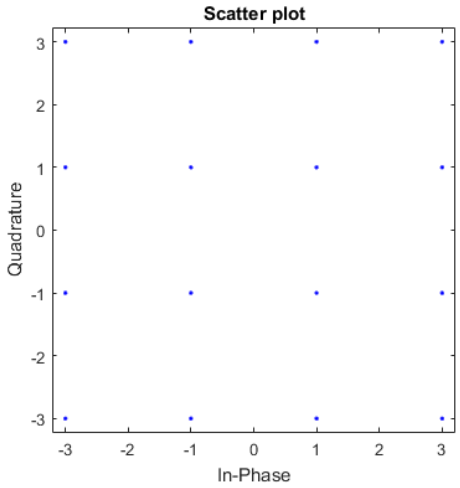
QPSK je široce používán v satelitním vysílání. QPSK je stále široce používán při streamování SD(Standard Definition) satelitním příjmem a některých HD (High Definition) kanálů.

3.4 Modulace 16QAM

Kvadraturní amplitudová modulace je klíčování, které se mění v závislosti na fázi a amplitudě signálu, čímž se zvyšuje množství informací, vysílané jedním stavem signálu. QAM využívá dvojici obvykle sinusových signálů s konstantním kmitočtem vzájemně fázově posunutých o 90°[24]. U modulace QAM nemá nosná vlna konstantní amplitudu. Pro rozpoznávání byla zvolena modulace 16QAM.

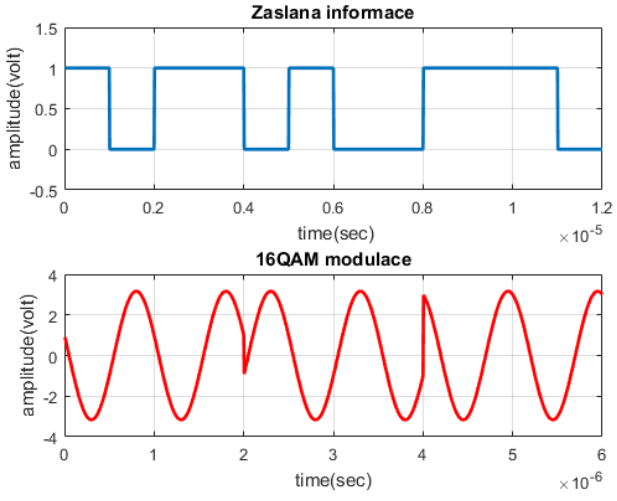
Hlavní důvod pro používání vícestavové modulace je skutečnost, že umožňuje šetřit šířku pásma nebo naopak se stejnou šířkou pásma zvýšit přenosovou rychlost.

Konstelační diagram modulace 16QAM je na obr. 2.6, pomocí kterého můžeme zjistit, jak při konkrétní amplitudě a fázovém posuvu rozlišit několik stavů, a tak přenášet několik bitů současně. Použitý konstelační diagram má 16 bodů. V našem případě může mít každá kvadraturní nosná 4 různé amplitudy. Pomocí jedné nosné lze rozlišit 2 bity informace, pomocí obou nosných 4 bity.



Obr. 3.6 Konstelační diagram modulace 16QAM

Výsledný signál má podobu signálu klíčovaného nebo modulovaného jak fázovým posuvem (PSK), tak amplitudovým posuvem (ASK). Modulovaný signál 16QAM a zaslaná zpráva jsou zobrazené na obr. 2.7.



Obr. 3.7 16QAM signál (viz níže) a zpráva (viz výše)

Čím je modulační schéma složitější, tím více škodlivé účinky zkreslují přenos, a tím kratší bude vzdálenost od základní stanice, ze které může být signál úspěšně přijat.

Kvadraturní amplitudová modulace se používá pro přenos barevných signálů v televizních normách PAL a NTSC, stereo vysílání a v optických systémech pro navýšení přenosové rychlosti.

4. UMĚLÉ NEURONOVÉ SÍTĚ

V poslední době byl velký zájem o neuronové systémy, které dnes již našly uplatnění v nejrůznějších oblastech lidské činnosti: lékařství, podnikání, informační technologie. Neuronové sítě jsou používány pro řešení problémů řízení, klasifikace, predikce, předpovídání, rozpoznávání vzorů. Jsou také používány v oblasti strojového učení a umělé inteligence, můžou být vloženy do her. Hlavním rysem neuronových sítí je schopnost učení.

Tento zájem je způsoben tím, že neuronové sítě:

- **mají bohaté možnosti.** Neuronové sítě jsou velmi silné modelovací nástroje, které umožní implementovat extrémně složité závislosti. Zejména neuronové sítě jsou nelineární povahované. Po mnoho let bylo lineární modelování primárním způsobem simulace ve většině oblastí, protože procedury optimalizace jsou pro ně dobře vypracovány. V aplikacích, kde lineární aproximace není vhodná, lineární modely nefungují dobře.
- **jsou snadné k použití.** Neuronové sítě se učí příkladem. Uživatel neuronové sítě vybere reprezentativní údaje, a pak spustí algoritmus učení, který automaticky pořídí datovou strukturu. V tomto případě od uživatele může být samozřejmě vyžadována určitá sada heuristické znalosti o tom, jak vybrat a připravit data, jak vybrat potřebnou síťovou architekturu a interpretovat výsledky, ale úroveň znalostí potřebných pro úspěšné uplatnění neuronových sítí je mnohem skromnější než, například, použití tradičních statistických metod.

Neuronové sítě jsou atraktivní z intuitivního hlediska, protože jsou založeny na biologickém modelu primitivního nervového systému. V budoucnu vývoj těchto neuro-biologických modelů může vést k vytvoření skutečně inteligentních počítačů.

4.1 Základní umělý model

Umělá neuronová síť je výpočetní model, a to softwarové nebo hardwarové provedení, založené na principu organizace a fungování biologických neuronových sítí. Tento koncept vznikl při studiu procesů probíhajících v mozku a při pokusu o simulaci těchto procesů. Neuronová síť se skládá z umělých neuronů, jejichž předobrazem je biologický neuron. Neuron má libovolný počet vstupů, ale pouze jeden výstup. Existuje hodně různých modelů neuronu, od úplně jednoduchých modelů až po ty nejsložitější, ve kterých je popsán každý detail chování neuronu živého organismu.

Nejpoužívanější model byl popsán McCullochem a Pittsem[28]:

$$y = f\left(\sum_{i=1}^{N}(I_i W_i)\right) \tag{4.1}$$

Kde:

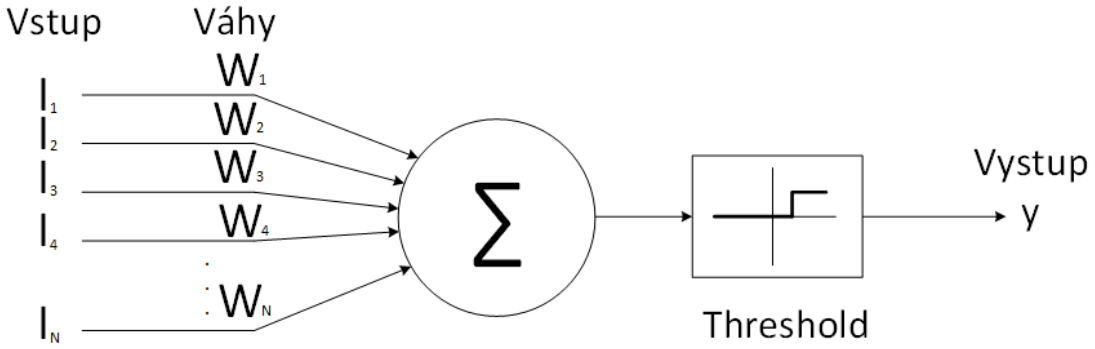
I_i jsou vstupy neuronu,

W_i jsou synaptické váhy,

$f(x)$ je přenosová funkce neuronu (někdy aktivační funkce),

y je výstup neuronu.

Perceptron je základní stavební prvek neuronové sítě. Perceptron je matematický model biologického neuronu. Jedná se o ústřední prvek neuronové sítě. Z historických důvodů se nazývá perceptron (označení matematického modelu biologického neuronu), z toho například označení vícevrstvé perceptronové sítě. V dnešní době však již dochází k záměně a používá se pro něj spíše označení "neuron". Na obrázku 4.1 je znázorněn model umělého neuronu.



Obr. 4.1 Model umělého neuronu (převzato z [28])

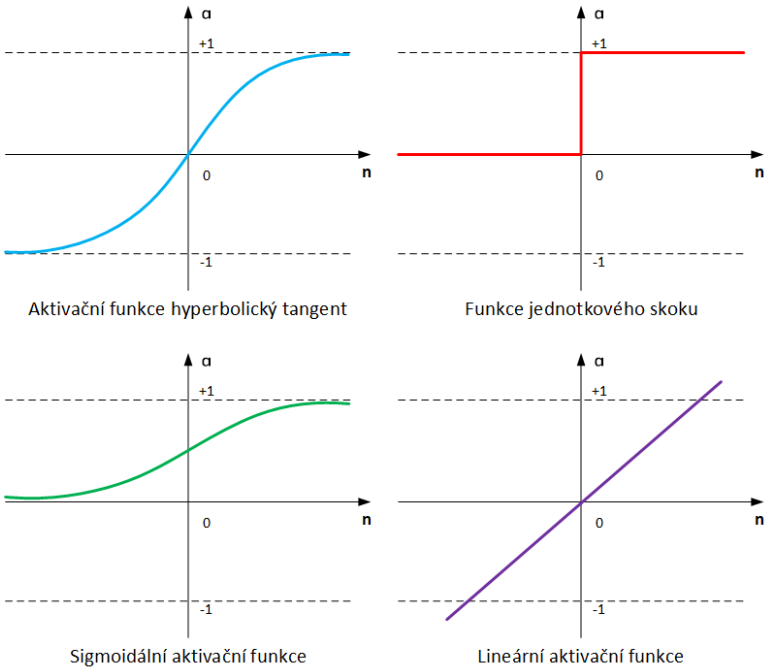
Hodnoty vah jsou normalizované v rozsahu (0;1) nebo (-1;1) a spojené s každou vstupní čárou. Velikost vah vyjadřuje uložení zkušeností do neuronu. Vyšší hodnota má vyšší prioritu[28].

Zjednodušeně řečeno, neurony jsou klasifikovány na základě jejich umístění v topologii sítě. Podíl:

- Vstupní neurony — užívají počáteční vektor a kódují vstupní signál.
- Výstupní neurony — představují v síti výstupy. jakékoliv výpočetní operace lze provádět ve výstupních neuronech.
- Skryté neurony — provádějí základní výpočetní operace.

V neuronových sítích a v jejich praktickém nasazení se používá velké množství aktivačních funkcí. V procesu testování se dospělo k využití zejména funkce hyperbolický tangent, skoková funkce, sigmoidální funkce a funkce lineární. Tyto funkce jsou na obrázku 4.2. Aktivační funkce u nabuzeného neuronu moduluje jeho výstupní signál, který je směrován do úrovně neuronové sítě. Výběr vhodné aktivační

funkce neuronu má vliv na konvergenci výpočtu naučení neuronové sítě. Ve většině případů je monotónně rostoucí a má rozsah hodnot [-1,1] nebo [0,1], ale existují výjimky.



Obr. 4.2 Aktivační funkce neuronů

4.2 Učení neuronových sítí

Neuronové sítě nejsou naprogramovány, jsou vyškoleny. Možnost vyškolení je jedna z hlavních výhod neuronových sítí oproti běžným algoritmům. Hlavním cílem vyškolení je najít koeficienty spojení mezi neurony. Během procesu trénování je neuronová síť schopna identifikovat komplexní vztahy mezi vstupy a výstupy, stejně jako provést zobecnění. To znamená, že v případě úspěšného učení bude síť schopna vrátit správný výsledek na základě údajů, které chyběly v trénovacích množinách, nebo když část dat je částečně porušena. Existují dva způsoby učení neuronových sítí: s učitelem a bez učitele.

Učení s učitelem [29] znamená, že pro každý vstupní vektor tréninku se nastaví požadovaný výstupní vektor hodnot, tzv. cíl. Tyto vektory vytvářejí učební pár. Váhu měníme tak dlouho, dokud se pro každý vstupní vektor a výstupní vektor nezíská přijatelná odchylka od cílového vektoru.

Učení bez učitele [29] je mnohem pravděpodobnější model učení z hlediska biologických kořenů umělých neuronových sítí. Tréninkové množství se skládá pouze ze vstupních vektorů. Algoritmus učení neuronové sítě přizpůsobí síťové váhy tak, aby bylo dosaženo souhlasnosti výstupních vektorů. To znamená, že stejné vstupní vektory mají stejný výstup.

Teorie učení má tři základní vlastnosti spojené s učením podle příkladu: kapacita, složitost modelů a výpočetní složitosti. Kapacitou je míněno to, jak si vzorky mohou vzpomenout na síť a jaké funkce a limity rozhodování mohou být v sii vytvořeny. Složitost vzorku určuje počet tréninkových příkladů potřebných k dosažení schopnosti sítě generalizovat. Příliš málo příkladů může způsobit „rekvalifikaci" sítě, pokud to funguje dobře na příkladech tréninkového vzorku, ale špatně na testovacích příkladech, podřízených stejnému statistickému rozdělení. Existují 4 typy pravidel učení: korekce omylem, strojem Boltzmanna, pravidlem Hebba a školení konkurencí.

- **Pravidlo korekce omylem.** Při učení s učitelem je pro každý vstupní příklad nastaven požadovaný výstupní *d*. Reálný výstup sítě *y* může být odlišný od požadovaného. Princip korekce chyb v tréninku je použití signálu (*d*-*y*) pro úpravu vah, čímž dochází k postupnému snižování chyby. Výcvik probíhá pouze tehdy, pokud perceptron dělá chyby. Jsou známé různé modifikace tohoto algoritmu učení[37].
- **Boltzmannův stroj.** Je stochastické pravidlo učení, které vyplývá z teorie informace a termodynamických principů. Účelem Boltzmannova tréninku je kalibrace váhových koeficientů, při kterých jsou viditelné neurony schopny splnit požadované rozdělení pravděpodobnosti. Boltzmannovo učení lze považovat za vlastní případ opravy chyb, u kterého chyba je srozumitelná jako divergence státních korelací ve dvou režimech[37].
- **Pravidlo Hebba.** Nejstarší učení je postulát učení Hebba. Hebb spoléhal na následující neurofyziologické sledování: v případě, že neurony na obou stranách synapse jsou aktivovány současně a v pravidelných intervalech, tak se pevnost synaptických spojení zvyšuje. Důležitým rysem tohoto pravidla je, že změna synaptických vah závisí pouze na aktivitě neuronů, které jsou spojeny touto synapsí. To výrazně zjednodušuje řetězce učení[37].
- **Soutěživé učení.** Na rozdíl od Hebbova učení, ve kterém se množina výstupních neuronů může aktivovat současně, při kompetitivním tréninku si výstupní neurony navzájem konkurují a tím dojde k jejich aktivaci. Tento jev je znám jako pravidlo „vítěz bere vše". Takový výcvik se koná v biologických neuronových sítích. Učení soutěže umožňuje seskupený vstup: takovými příklady jsou síťová seskupení podle korelací a jsou reprezentovány jediným prvkem[37].

4.3 Multi-Layered Perceptron

V roce 1958 Frank Rosenblatt vynalezl neuronovou síť, která se nazývá perceptron. Perceptron byl navržen tak, aby mohl klasifikovat objekty. Ve fázi učení „učitel" dává vědět perceptronu, do které třídy patří předložený objekt. Vyškolený

perceptron je schopen klasifikovat objekty, včetně těch, které nejsou použity v tréninku, ale zároveň dělá velmi málo chyb.

Nový vzestup teorie neuronových sítí začal v letech 1983-1986. Důležitou roli hraje skupina PDP (Parallel Distributed Processing), v níž byly diskutovány neuronové sítě, tzv. vícevrstvové perceptrony, které se ukázaly být velmi účinné pro to, aby řešily problémy rozpoznávání, řízení a predikce[47].

Nejjednodušší a funkční architekturou jsou vícevrstvové perceptrony. To jsou přímé distribuční sítě, které mají takový název, protože neurony jedné vrstvy mohou být připojeny pouze se sousedními vrstvami neuronů bez opakovaných a rekurentních vztahů. Typicky vícevrstvové perceptrony se skládají ze vstupní vrstvy, jedné nebo více skrytých vrstev (tzv. protože nemají přímé spojení s „vnějším světem“) a výstupní vrstvy. Pomoci takovéto sítě se data převádí ze vstupního n-rozměrného prostoru do m-rozměrného výstupu.

Tyto sítě musí být vyškolené pro vydávání požadovaných výsledků po předložení vzorků do vstupní vrstvy.

Vzdělání sítě je tou pravou volbou vah spojů mezi prvky. Zvolí se taková váha, aby celková střední kvadratická chyba pro vzorky tréninkové množiny byla minimální. Toho lze dosáhnout různými způsoby. Po tréninku perceptronu se provádí testovací postup, který je proveden za účelem posouzení výsledků. Pro tento účel je obvykle potřeba rozdělit školicí sadu do dvou částí. Jedna část se používá pro výcvik, a druhá, pro kterou je výsledek znám, je zapojena do procesu testování. Procento správných výsledků síťového provozu ve fázi testování je indikátorem kvality práce perceptronu.

Vývoj backpropagation algoritmu k určení vah ve vícevrstvých perceptronových sítích učinil z této sítě nejpopulárnější síť mezi výzkumníky a uživateli neuronových sítí.

Existuje mnoho sporných otázek při návrhu přímých distribučních sítí, například, kolik vrstev je potřebných pro tento úkol, kolik byste měli zvolit prvků v každé vrstvě, jak síť bude reagovat na údaje, které nejsou zahrnuty v tréninkovém setu (což je schopnost sítě generalizovat), a jaká velikost připravených vzorků je potřebná k dosažení „dobré“ schopnosti sítě generalizovat.

Ačkoli dopředné sítě vícevrstvé jsou široce používány pro klasifikaci a aproximaci funkcí, mnohé parametry lze ještě třeba určit metodou pokus–omyl. Stávající teoretické výsledky poskytují pouze slabé vodítko pro výběr těchto parametrů v praktických aplikacích. [47]

4.4 Backpropagation algoritmus

Algoritmus zpětného šíření (backpropagation) se týká způsobu učení pro opravu chyb. Tento způsob výuky „s učitelem,“ ve kterém „učitel“ učí sítě, stejně jako dítě se

učí číst a psát. Při výcviku ve vstupní vrstvě se dodávají signálům obrázky opakovaně, jejichž klasifikaci se neuronová síť učí a váhy neuronů jsou upraveny tak, aby bylo dosaženo požadovaného výstupního signálu. Obrázky jsou dodávány do vstupní vrstvy pro zlepšení kvality rozpoznávání se může mírně lišit (přidané pomocí šumu a podobně)[46].

Podrobný postup pro školení je následující:

1. Vzorkování vstupních dat (množina snímků zařazených učitelem) je rozděleno na dvě části: to jsou sekvence učení a kontrolní sekvence. Typická posloupnost učeni obsahuje více obrázků než kontrolní.
2. Všechny váhy se inicializuji, včetně prahových hodnot, malými náhodnými hodnotami (obvykle v rozsahu [-1, 1]). To definuje výchozí bod na povrchu gradientu. Poloha může být rozhodující pro konvergenci sítě.
3. Provádí se přímý průchod sítí pro první snímek ze vzorku učení vstupní vrstvy přes skryté vrstvy k výstupní vrstvě: každý neuron sčítá součiny vstupu s vahami a předává výsledek aktivační funkce na neurony další vrstvy.
4. Vypočítává se rozdíl mezi skutečnou a požadovanou výstupní hodnotou každého neuronu výstupní vrstvy. Jestliže je nesoulad, znamená to, že došlo k chybě při rozpoznávání (klasifikace) obrázků.
5. Probíhá procedura backpropagation těchto chyb na komunikacích z výstupní vrstvy na vstupní vrstvu neuronů a jsou určeny chyby pro každý neuron.
6. Opravují se váhy neuronů.
7. Znovu probíhá také přímý průchod sítě již po následujícím ukázkovém snímku tréninkové množiny.

Kroky 3 až 7 se opakují, dokud není dosaženo určitých kritérií, jako je například nastavěný limit chyb.

Po dokončení učení se síť kontroluje pomocí kontrolní sekvence, která obsahuje obrázky neuvedené dříve. V tomto případě se spočítá jenom chyba sítě a neprovádí se žádná korekce vah. V případě, že kvalita práce je vyhovující, je síť považována za připravenou k provozu. V opačném případě bude síť podrobena re-vzdělávání, ve kterém můžeme měnit některé parametry (počáteční váhy, počet neuronů ve skrytých vrstvách, další školení obrázky atd.).

Je důležité, aby se síť „nepřeškolila“. V případě, že síť byla podrobena příliš velkému počtu tréninkových cyklů, začne „se učit“ tréninkové obrázky a již nemůže rozpoznat jiné, dokonce ani jim podobné.

5. ROZPOZNÁVÁNÍ DIGITÁLNÍCH MODULACÍ POMOCÍ UMĚLÝCH NEURONOVÝCH SÍTÍ

Klíčové příznaky signálu jsou používané při rozpoznávání modulace jako vstupy neuronové sítě. Nejdůležitějším cílem je zjištění těchto příznaků před rozpoznáváním modulace pomocí neuronových sítí. E.E. Azzouz a A.K. Nandi [8],[9] vypracovali několik funkcí založených na AMR algoritmech, které se týkají typických charakteristik z okamžité amplitudy, okamžité fáze a okamžitého kmitočtu přijatého signálu. Pomocí těchto příznaků je možné rozpoznávat analogové a digitální modulace. Při stanovení klíčových příznaků signálu je vhodné použít jeho analytické vyjádření.

Analytický signál je uměle vytvořený signál z výchozího signálu. Původ výchozího signálu může být například v měření nějaké fyzikální veličiny, a proto jej můžeme také nazvat reálný ve smyslu dalšího zobecnění na obor komplexních čísel. K tomuto reálnému signálu je v definici analytického signálu připojena imaginární část. Z reálné funkce se spojitým časem nebo z posloupnosti vzorkovaných hodnot vznikne komplexní funkce se spojitým časem nebo posloupnost komplexních čísel. Imaginární část je definována tak, aby umožnila snadno analyzovat modulační efekty nízkofrekvenčních signálů na nosné signály o vyšší frekvenci než je nejvyšší frekvence modulačního signálu. K vytvoření imaginární části analytického signálu je užita Hilbertova transformace[30]:

$$z(t) = x(t) + jy(t) = a(t)e^{j\phi(t)} \tag{5.1}$$

Kde:
$z(t)$ je analytický signál,
$x(t)$ je reálný signál,
$y(t)$ je Hilbertová transformace signálu $x(t)$,
j je imaginární jednotka,
$a(t)$ je okamžitá amplituda,
$\psi(t)$ je okamžitá fáze.
Okamžitou amplitudu a okamžitou fázi definujeme jako:

$$a(t) = |z(t)| \tag{5.2}$$

$$\phi(t) = arg(z(t)) \tag{5.3}$$

Se známým analytickým signálem $z(t)$ můžeme okamžitý kmitočet $f(t)$ jednoznačně definovat jako:

$$f(t) = \frac{d\phi(t)}{dt} \tag{5.4}$$

kde $\phi(t)$ je okamžitá fáze analytického signálu $z(t)$.

Azzouz a Nandi navrhli pro rozpoznávání digitálních modulací pět charakteristických příznaků[8]:

1. Maximální hodnota výkonové spektrální hustoty centrované normované okamžité amplitudy, γ_{max}

$$\gamma_{max} = \frac{max\left|DFT\left(a_{cn}(i)\right)\right|^2}{N_s} \tag{5.5}$$

kde N_s je počet vzorků v analyzovaném signálu, $a_{cn}(i)$ je hodnota centrované normované okamžité amplitudy $a(t)$ v časovém okamžiku $t = \frac{i}{f_s}$ (i = 1, 2, …, N_s), f_s je vzorkovací frekvence.

$$a_{cn}(i) = a_n - 1 \tag{5.6}$$

$$a_n(i) = \frac{a(i)}{m_a} \tag{5.7}$$

$$\mathrm{m}_a = \frac{1}{N_s}\sum_{i=1}^{N_s} a(i), \tag{5.8}$$

kde $a_n(i)$ je normovaná okamžitá amplituda signálu, $a(i)$ je jeho okamžitá amplituda a m_a je střední hodnota okamžité amplitudy.

Tento příznak slouží k odlišení modulovaných signálů, kde je informace nesena změnou amplitudy od modulovaných signálů, kde je informace nesena změnou kmitočtu.

2. Směrodatná odchylka absolutní hodnoty centrované nelineární složky fáze pro časový segment, kde je signál nad šumem, σ_{AP}

$$\sigma_{AP} = \sqrt{\frac{1}{C}\left(\sum_{a_{n(i)>a_t}} \phi_{NL}^2\,(i)\right) - \left(\frac{1}{C}\sum_{a_{n(i)>a_t}} |\phi_{NL}(i)|\right)^2} \tag{5.9}$$

kde ϕ_{NL} je hodnota centrované nelineární složky okamžité fáze v časovém okamžiku $t = \frac{i}{f_s}$ (i = 1, 2, …, N_s), C je počet vzorků ve vektoru $\{\phi_{NL}(i)\}$ pro $a_n(i) > a_t$, a a_t je práh. $\phi_{NL}(i)$ jsou hodnoty fázové charakteristiky bez podílu (účasti) nosné frekvence:

$$\phi_{NL}(i) = \phi_{UW}(i) - \frac{2\pi f_c}{f_s} \tag{5.10}$$

kde $\phi_{UW}(i)$ je rozbalená (unwrapped) fáze a f_c je nosná frekvence.

Tento příznak slouží k odlišení modulovaných signálů, které mají informace o absolutní fázi (4PSK) od modulovaných signálů, které nemají informace o absolutní fázi (2ASK, 2PSK).

3. Směrodatná odchylka centrované nelineární složky okamžité fáze pro časový segment, kde je signál nad šumem, σ_{DP}

$$\sigma_{DP} = \sqrt{\frac{1}{C}\left(\sum_{a_{n(i)>a_t}} \phi_{NL}^2(i)\right) - \left(\frac{1}{C}\sum_{a_{n(i)>a_t}} \phi_{NL}(i)\right)^2} \tag{5.11}$$

kde ϕ_{NL} je hodnota centrované nelineární složky okamžité fáze v časovém okamžiku $t = \frac{i}{f_s}$ (i = 1, 2, …, N_s), C je počet vzorků ve vektoru $\{\phi_{NL}(i)\}$ pro $a_n(i) > a_t$, a a_t je práh.

Tento příznak slouží k odlišení modulovaných signálů 2PSK od modulovaných signálů 2ASK a 4ASK.

4. Směrodatná odchylka absolutní hodnoty normované centrované okamžité amplitudy, σ_{AA}

$$\sigma_{AA} = \sqrt{\frac{1}{N_s}\left(\sum_{i=1}^{N_s} a_{cn}^2(i)\right) - \left(\frac{1}{N_s}\sum_{i=1}^{N_s} |a_{cn}(i)|\right)^2} \tag{5.12}$$

kde N_s je počet vzorků v analyzovaném signálu, $a_{cn}(i)$ je hodnota centrované normované okamžité amplitudy $a(t)$ v časovém okamžiku $t = \frac{i}{f_s}$ (i = 1, 2, …, N_s), f_s je vzorkovací frekvence.

Tento příznak slouží k odlišení modulovaných signálů 2ASK od modulovaných signálů 4ASK.

5. Směrodatná odchylka absolutní hodnoty normované okamžité frekvence pro časový segment, kde je signál nad šumem, σ_{AF}

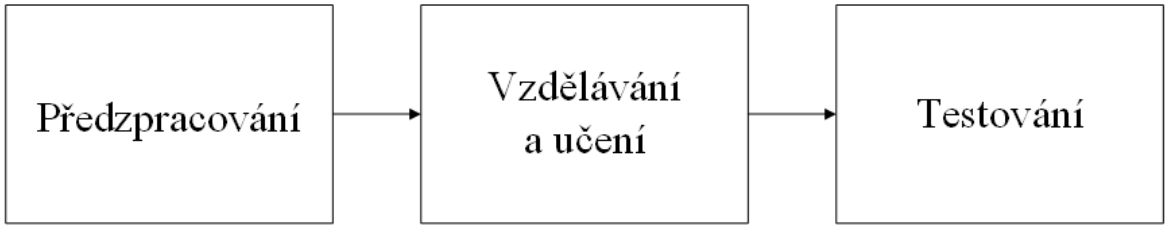
$$\sigma_{AF} = \sqrt{\frac{1}{C}\left(\sum_{a_{n(i)>a_t}} f_N^2(i)\right) - \left(\frac{1}{C}\sum_{a_{n(i)>a_t}} |f_N(i)|\right)^2} \tag{5.13}$$

kde $f_N(i) = \frac{f_m(i)}{R_s}$ je normovaná okamžitá frekvence, $f_m(i) = f_i - m_f$, $m_f = \frac{1}{N_s}\sum_{i=1}^{N_s} f(i)$ je střední hodnota okamžité frekvence, R_s je symbolová rychlost a $f(i)$ je okamžitá frekvence.

Tento příznak slouží k odlišení modulovaných signálů 2FSK od modulovaných signálů 4FSK.

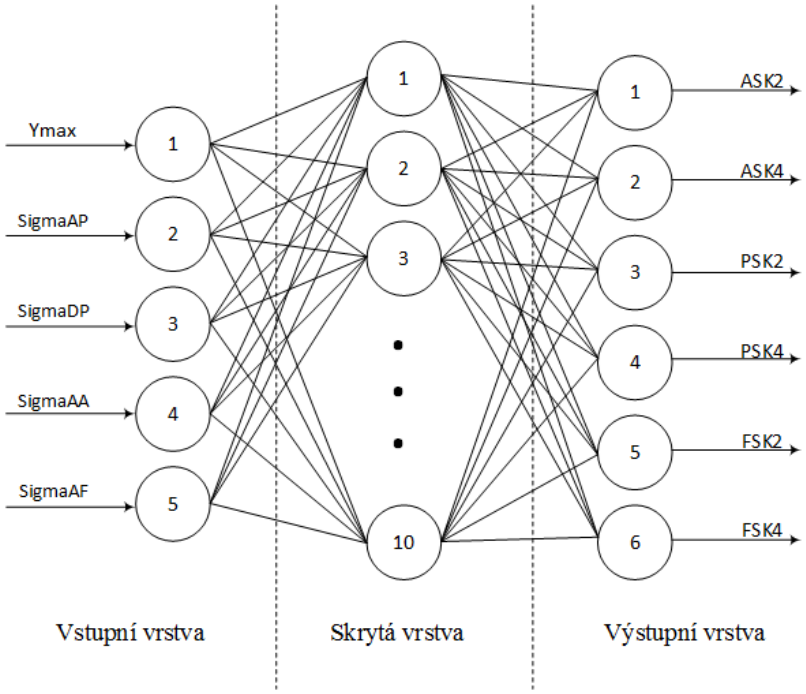
Podle Nandiho a Azzouza [8], [9] je možné pomocí těchto klíčových příznaků rozpoznávat modulace 2ASK, 4ASK, 2FSK, 4FSK, 2PSK, 4PSK. Pro rozpoznávání pomocí umělých neuronových sítí podle výše uvedených autorů byl zvolen následující postup, který se skládá ze tří částí: předzpracování, fáze vzdělávání a učení a fáze testování. V první části jde o získávání klíčových příznaků. Ve druhé fázi jde o nastavení

optimálních parametrů neuronové sítě při učení tak, aby bylo možné minimalizovat počet chyb při rozpoznávání. Pak následuje učení. Posledním krokem je otestování fungování umělé neuronové sítě. Tento postup je znázorněn na obrázku 5.1.



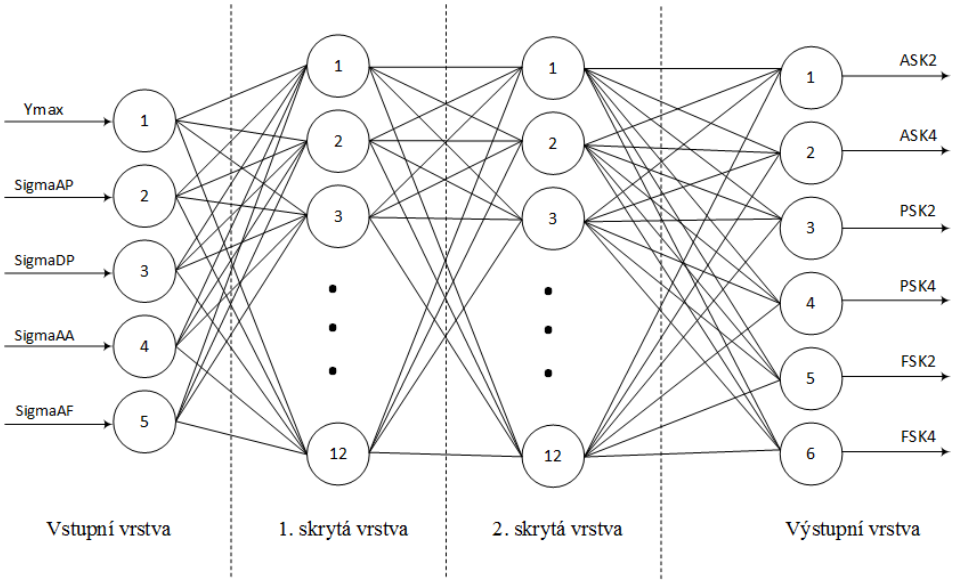
Obr. 5.1 Postup rozpoznávání modulace pomocí umělých neuronových sítí

Ve své práci Nandi a Azzouz [8] navrhli dvě základní neuronové sítě s rozdílem v počtu skrytých vrstev. Tyto sítě pak byly ostatními autory modifikovány. Tyto neuronové sítě jsou na obrázcích 5.2 a 5.3. Každá síť má ve vstupní vrstvě 5 neuronů, kde počet neuronů je roven počtu příznaků. Výstupní vrstva se skládá z 6 neuronů, kde počet neuronů je roven počtu množině rozpoznávaných modulovaných signálů. Počet neuronů ve skrytých vrstvách byl vybrán experimentálně a stanoven na 10 neuronů pro neuronovou síť s jednou skrytou vrstvou a 12 neuronů na každou skrytou vrstvu pro neuronovou síť s dvěma skrytými vrstvami.



Obr. 5.2 Neuronová síť s jednou skrytou vrstvou (převzato z [8])

V případě použití umělé neuronové sítě s jednou skrytou vrstvou neuronů se velmi obtížně nastavují prahy a váhy, kvůli tomu může být vyšší procento chybné klasifikace signálů. Při změně počtu neuronů ve skryté vrstvé se snižuje výkon neuronové sítě, zvyšuje se čas práce nad rozpoznáváním digitálních modulací a taky se zvyšuje procento chybného rozpoznávání.



Obr. 5.3 Neuronová síť s dvěma skrytými vrstvami (převzato z [8])

V případě použití umělé neuronové sítě s dvěma skrytými vrstvami neuronů oproti použití umělé neuronové sítě s jednou skrytou vrstvou neuronů se zvyšuje výkon a procento úspěšného rozpoznávání digitálních modulací.

6. SIMULACE V PROSTŘEDÍ MATLAB

Úkolem diplomové práce bylo vybrat některou metodu, která bude vhodná pro automatickou klasifikaci typu digitální modulace pomocí neuronových sítí. Takže úkolem bylo vytvořit modulované signály a navrhnout příznaky v prostředí Matlab, které se budou používat pro klasifikaci digitálních modulací. Dalším úkolem bylo vytvořit neuronovou síť pro klasifikaci v prostředí Matlab. Dále je potřeba vytvořit trénovací množinu, provést učení neuronové sítě a analyzovat úspěšnost rozpoznávání signálů bez šumu a se šumem. Podle potřeby musíme optimalizovat parametry neuronové sítě. Příznakovým metodám rozpoznávání byla věnovaná velká pozornost, protože jsou více používané v praxi a obvykle potřebují méně nebo dokonce žádné předchozí znalosti o přijatých signálech a dle doporučené literatury Automatic modulation recognition of communication signals od Azzouza i Nandiho [8], [9] ve které dosahují výborných výsledků.

Proces rozpoznávání modulovaných signálu pomocí prostředí Matlab musí obsahovat tyto základní kroky:

1. modulace signálu
2. načtení matice namodelovaného signálu
3. transformace vektoru reálného signálu na vektor analytického signálu
4. výpočet matice okamžité amplitudy, matice okamžité fáze a matice okamžitého kmitočtu
5. výpočet klíčových příznaků pro rozpoznání druhu modulace
6. návrh struktury umělé neuronové sítě
7. vytvoření trénovací množiny
8. proces učení umělé neuronové sítě s pomocí učitele
9. hotová metoda automatické klasifikace digitálních modulací pomocí neuronových sítí.

6.1 Modulace signálů v prostředí Matlab

Jak již bylo řečeno dříve, pro rozpoznávání byly vybrány tyto typy digitálních modulací: BASK, BFSK, BPSK, QPSK a 16QAM, které byly popsány výše. Pro všechny modulátory bylo použito stejné nastavení. To znamená, že pro všechny modulátory byl použit modulační signál se stejnou bitovou periodou. Dále všechny modulátory používaly stejný nosný kmitočet, v případě modulace 2FSK byly použity kmitočty $f1$ a $f2$ takové, aby střední nosný kmitočet byl roven kmitočtu nosného signálu u ostatních modulací. Vzorkovací kmitočet byl opět u všech modulací volen stejný.

Každá modulace byla vytvořena jako funkce pro využití v následujících aplikacích potřebných pro realizaci úkolů diplomové práce. Dále je uveden příklad využití funkce v prostředí Matlab a potřebné hodnoty pro modulace signálu.

```
function [ ask,t ] = ASK( T,fv,fc,SNR ),
```

kde `ASK` je název funkce vhodný pro spuštění, `T` je celková doba simulace, `fc` je kmitočet nosné, `fv` je vzorkovací kmitočet, `ask` je matice, která obsahuje namodelovaný signál, `t` je časový vektor, `SNR` je signal-to-noise ratio.

Takže každému typu modulace signálu byl přidán bílý Gaussovský šum. Aditivní bílý Gaussovský šum (AWGN) je typ rušivého vlivu na informační přenosový kanál, vyznačující se rovnoměrnou stejnou na všech frekvencích spektrální výkonovou hustotou, normálně distribuovaným časovým významem a aditivní metodou vlivu na signál[44]. Je nejběžnějším typem šumu, použitým k výpočtu a simulaci radiokomunikačních systémů. Pojem „aditivní“ znamená, že tento typ šumu je přidán do užitečného signálu a je statisticky nezávislý na signálu. V přírodě a technice „čistý“ bílý šum není, nicméně do kategorie bílého šumu vstupuje libovolný šum, jehož spektrální hustota je stejná na všech frekvencích. V prostředí Matlab na to existuje funkce `wgn`, jejíž pomocí můžeme vygenerovat šum.

Signál-to-noise ratio (též SNR) je poměr signálu vůči šumu. V praxi se obvykle jedná o obecné označení pro odstup signálu od šumu, udávané v dB (samotný poměr je ale číslo bezrozměrné — bez jednotky). Často se s tímto údajem setkáváme u zvukových karet, ozvučovacích aparatur, hudebních přehrávačů/přístrojů apod. Obecně samozřejmě platí, že čím je hodnota SNR vyšší, tím je výsledný zvukový výstup kvalitnější (tím nižší je hladina šumu)[45].

$$SNR = \frac{P_{signal}}{P_{šum}} \tag{6.1}$$

$$SNR(dB) = 10log_{10}\frac{P_{signal}}{P_{šum}} \tag{6.2}$$

Kde:

P_{signal} je výkon signálu,

$P_{šum}$ je výkon šumu.

Dále je ukázáno, jak byl vytvořen bílý Gaussovský šum v diplomové práci:

```
Ps=sum(st.^2)/length(st); % výkon signálu
noise=wgn(length(st),1,(10*log10(Ps)-SNR)); % vygenerovaní šumu
ask=st+noise; % součet signálu a šumu
```

kde `Ps` je výkon signálu, `st` je vytvořený signál, `noise` je šum, `SNR` je signal-to-noise ratio a `ask` je signál se šumem.

Pro ostatní typy modulace byly vytvořeny stejné funkce se stejnými parametry, jen každá má odlišný název funkce a název matice se signálem. Parametry modulací mohou být klidně změněny kdykoliv.

Tab. 6.1 Parametry modulací

Kmitočet nosného signálu	10 MHz
Vzorkovací kmitočet, fv	$j \cdot fc$, kde fc je kmitočet nosně a $j \in [2; 10]$
Počet bitů	10000
Délka analyzovaného signálu	20001 vzorků

Vektor zprávy obsahuje náhodné hodnoty 1 a 0, pomocí dvou funkcí `round` a `rand`. Pro modulaci 16QAM byla použita funkce `qammod`, což je standardní funkce z prostředí Matlab pro návrh konstelačního diagramu modulace s libovolným počtem stavů. Kódy všech modulovaných signálů jsou v příloze.

6.2 Návrh příznaků v prostředí Matlab

Na začátku byla vytvořena aplikace, která umožňuje spustit různé druhy modulace pro následující analýzu a výpočet klíčových příznaků. Cílem je dostat po spuštění aplikace z modulace vektor modulovaného signálu a uznat kmitočet pro následující výpočet příznaků.

Před výpočtem klíčových příznaků musíme z reálného signálu vytvořit analytický signál. V prostředí Matlab na to existuje funkce Hilbertova transformace:

```
z = hilbert(Mod_sig); % Vytvoříme analytický signál
```

kde *Mod_sig* je reálný signál pro zvolené modulace a *z* je získaný analytický signál.

Následující krok je výpočet vektoru okamžitě amplitudy, vektoru okamžité fáze a vektoru okamžitého kmitočtu. Podle rovnic uvedených dříve vypadá implementace okamžité amplitudy v prostředí Matlab takto:

```
inst_ampl = abs(z);  % okamžitá amplituda
```

Okamžitá fáze byla definována jako nelineární složka rozvinutého úhlu analytického signálu. Nelineární složka okamžité fáze je získána odečtením složky lineární fázové nosné frekvence z rozvinutého úhlu analytického signálu:

```
inst_phas = unwrap(angle(z));  % okamžitá fáze
```

Okamžitá frekvence je jednoduše definována jako časová rychlost změny okamžité fáze. Použil jsem funkci *diff* v prostředí Matlab, abych mohl spočítat rozdíl mezi po sobě jdoucími intervaly v *PhiNL*. Okamžitá frekvence byla získána takto:

```
inst_freq = diff(unwrap(angle(z))); % okamžitý kmitočet
```

kde *z* je analytický signál.

Další krok je implementovat příznaky:

1. Maximální hodnota výkonové spektrální hustoty centrované normované okamžité amplitudy byla získána takto:

GammaMax = max((abs(fft(Acn))).^2)/Ns;
kde *Ns* je počet vzorků a *Acn* je normalizovaný a centralizovaný signál.

2. Směrodatná odchylka absolutní hodnoty normované centrované okamžité amplitudy byla získána takto:

SigmaAA = sqrt((1/Ns*sum(Acn.^2))-(1/Ns*sum(abs(Acn))).^2);
kde *Ns* je počet vzorků a *Acn* je normalizovaný a centralizovaný signál.

3. Směrodatná odchylka absolutní hodnoty centrované nelineární složky fáze pro časový segment byla získána takto:

SigmaAP = sqrt((1/C*sum(PhiNLC.^2))-(1/C*sum(abs(PhiNLC))).^2);
kde *C* je počet vzorků a *PhiNLC* je normalizovaná a centralizovaná fáze.

4. Směrodatná odchylka centrované nelineární složky okamžité fáze pro časový segment byla získána takto:

SigmaDP = sqrt((1/C*sum(PhiNLC.^2))-(1/C*sum(PhiNLC)).^2);
kde *C* je počet vzorků a *PhiNLC* je normalizovaná a centralizovaná fáze.

5. Směrodatná odchylka absolutní hodnoty normované okamžité frekvence pro časový segment byla získána takto:

SigmaAF = sqrt((1/length(freqn)*sum(freqn.^2))-
-(1/length(freqn)*sum(abs(freqn))).^2);
kde *freqn* je normalizovaný a centralizovaný kmitočet.
Výsledky experimentu jsou uvedeny v následující tabulce 6.2.

Tab. 6.2 Výsledek návrhu příznaků

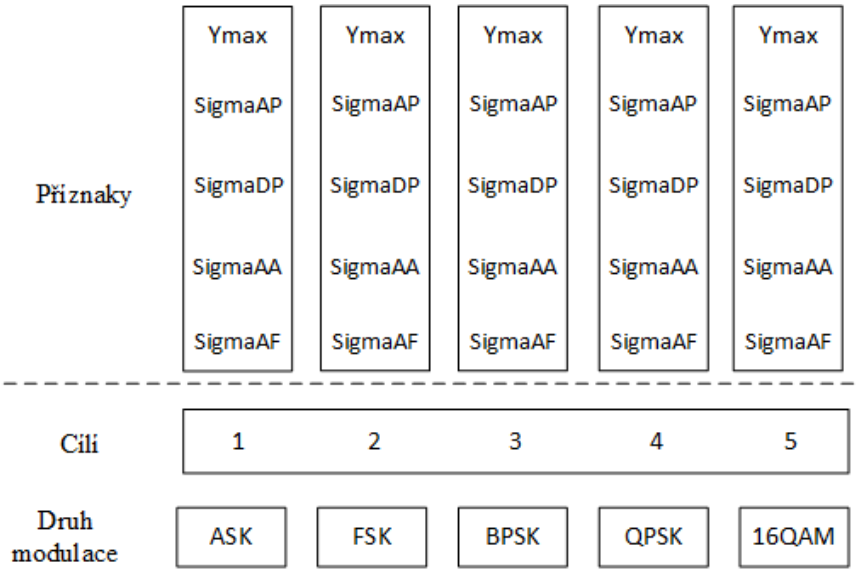
Název modulace	σ_{AA}	σ_{AF}	σ_{AP}	σ_{DP}	γ_{max}
ASK	0,0594	1,2379	0,1069	0,1120	504,2392
FSK	0,0131	0,0595	0,9248	1,8042	0,0213
BPSK	0,0515	3,1386	0,1452	1,5605	3,9581
QPSK	0,0278	6,5294	0,9214	1,5783	0,2390
16QAM	0,2192	8,4614	0,8996	1,3442	130,9958

6.3 Vytvoření trénovací množiny

Dalším krokem je vytvoření aplikace pro automatické výpočty příznaků pro různé typy modulace s různými parametry signálu, a to jsou vzorkovací kmitočet a SNR.

Před vlastním učením sítě je nutno mít vhodně vybraná učební data dostatečně reprezentující jednotlivé stavy úlohy, které by ve výpočetním procesu měla síť umět

rozhodovat, pro toto je nutné vytvořit matice. První část matice je tvořena vektory s charakteristickými příznaky daných modulací, druhá část matice se skládá z cílů, a to jsou hodnoty, odpovídající zvoleným druhům modulací. Tato matice byla importována do souboru „*xls“ pomocí prostředí MATLAB. Matice s cíli může mít obecně až n vzorů, ale minimální počet vzorů musí odpovídat počtu zkoumaných druhů modulovaných signálů (v našem případě je to 5 typů modulovaných signálů, tedy je zapotřebí minimálně 5 vzorů). Pro názornost jsou tyto matice zobrazeny na obrázku 6.1.



Obr. 6.1 Matice cílů a příznaků

Dále je uvedeno, jak byla provedena implementace uložení příznaků a cílů modulace ASK do trénovací matice, pro následující trénování neuronové sítě. Pro každou hodnotu SNR a hodnotu vzorkovacího kmitočtu vypočítá aplikace příznaky, navržené Azzouzem a Nandim[8], a ukládá je do pomocné matice. Potom hned přidává vektor cíle, odpovídající druhu modulace. Posledním krokem je uložit získané hodnoty do trénovací množiny.

```
for i= SNRValueMin:SNRpres:SNRValueMax
    for j=2:1:10
        fv=j*fc;            % vzorkovaci kmitocet
        for k=1:1:Pocet_iteraci
            disp('Spusteni modulace ASK...');
            [ModSig1,t1] = ASK(T,R,fv,fc,i);
            VektorASK = VypocetPriznaku(ModSig1,t1);
            %VektorASK = VektorASK/max(max(VektorASK)); % normalizace
            ModVektorASK = [VektorASK, 1, 0, 0, 0, 0];
    ...
            Mnozina = [Mnozina; ModVektorASK; ModVektorFSK; ...
                ModVektorBPSK; ModVektorQPSK; ModVektor16QAM];
        end
    end
end
```

kde `SNRValueMin` je minimální hodnota SNR, `SNRpres` je krok SNR, `SNRValueMax` je maximální hodnota SNR, `fv` je vzorkovací kmitočet, `fc` je kmitočet nosné, `ModSig1`

je modulovaný signál, `t1` je časový vektor, `T` je celková doba simulace, `VypocetPriznaku` je aplikace výpočtu příznaků `Mnozina` je matice s příznaky `ModVektorASK` je matice obsahujíce příznaky modulace ASK, `ModVektorFSK` je matice obsahujíce příznaky modulace FSK, `ModVektorBPSK` je matice obsahujíce příznaky modulace BPSK, `ModVektorQPSK` je matice obsahujíce příznaky modulace QPSK, `ModVektor16QAM` je matice obsahujíce příznaky modulace 16QAM.

V případě potřeby normalizace hodnot příznaků pro učení neuronové sítě stačí odkomentovat řádek normalizace, který provede dělení příznaků na maximální hodnotu těchto příznaků, aby výsledky byly v rozsahu od 0 až 1 Tuto normalizaci příznaků používají Azzouz a Nandi [8] pro zrychlení učení neuronové sítě.

Pro zlepšení kvality učení umělé neuronové sítě v této diplomové práci byly zvoleny následující parametry v aplikaci vytvoření trénovací množiny, které jsou uvedeny v tabulce 6.3.

Tab. 6.3 Zvolené parametry pro vytvořeni trénovací množiny

Signal-to-noise ratio, SNR	[-10; 50]
Vzorkovací kmitočet, fv	$j \cdot fc$, kde fc je kmitočet nosně a $j \in [2; 10]$
Počet opakování pro každý SNR a vzorkovací kmitočet, Počet_iterací	100

Výsledkem jsou dvě trénovací množiny, které obsahují výsledky výpočtu příznaků podle metody, kterou nabídli Azzouz a Nandi[8]. Stručný příklad těchto příznaků je v tabulkách 6.4 a 6.5.

Tab. 6.4 Příklad trénovací množiny bez normalizace příznaků

σ_{AA}	σ_{AF}	σ_{AP}	σ_{DP}	γ_{max}	Cíl					Typ modulace
0,1078	1,4825	0,1449	0,1669	299,8015	1	0	0	0	0	ASK
0,0494	0,1675	0,9292	1,8321	0,0917	0	1	0	0	0	FSK
0,0669	1,4601	0,1564	1,5556	1,9993	0	0	1	0	0	BPSK
0,0540	1,2426	0,9578	1,6513	0,3144	0	0	0	1	0	QPSK
0,1891	1,5044	0,9035	1,7072	289,1561	0	0	0	0	1	16QAM

Tab. 6.5 Příklad trénovací množiny s normalizací příznaků

σ_{AA}	σ_{AF}	σ_{AP}	σ_{DP}	γ_{max}	Cíl					Typ modulace
2,5833e-04	0,0015	2,2978e-05	2,4650e-05	1	1	0	0	0	0	ASK
0,0065	0,0325	0,5109	1	0,0077	0	1	0	0	0	FSK
0,0149	0,8697	0,0259	0,4754	1	0	0	1	0	0	BPSK
0,0065	1	0,1763	0,3534	0,0599	0	0	0	1	0	QPSK
0,0013	0,0478	0,0056	0,0081	1	0	0	0	0	1	16QAM

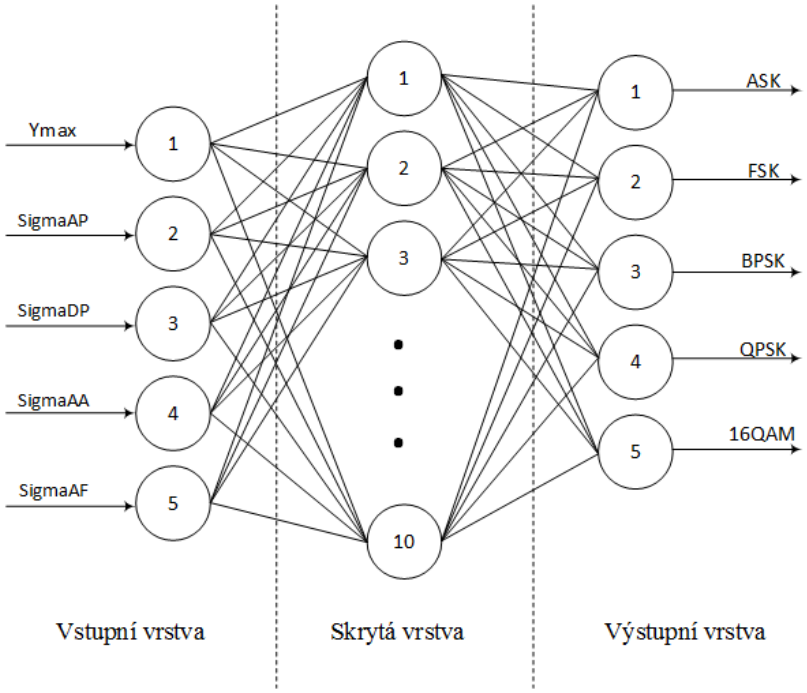
Konečné trénovací množiny mají 274500 vzorů, to je 54900 vzorů pro každý druh modulace, což bude stačit pro úspěšné trénování umělé neuronové sítě.

6.4 Vytvořeni neuronové sítě v prostředí Matlab

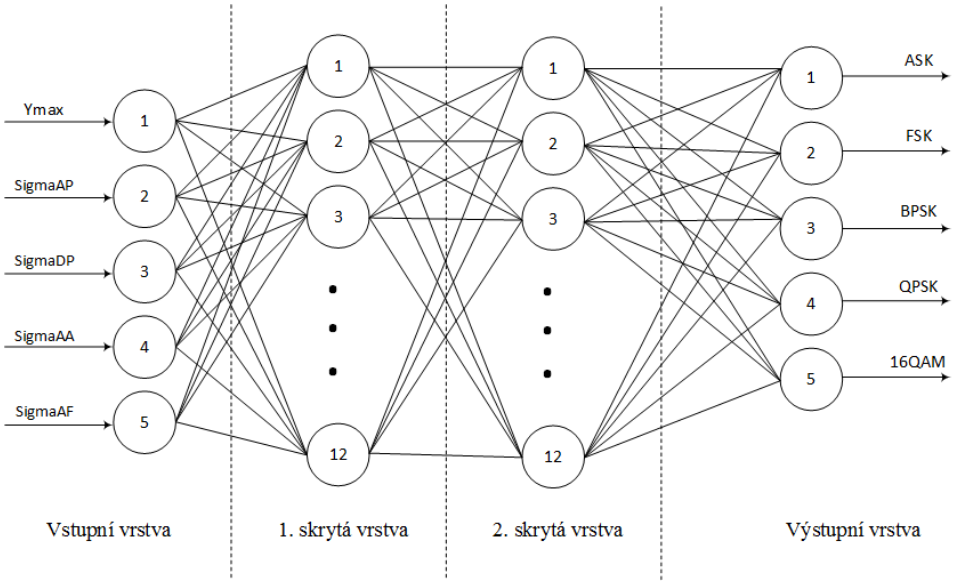
Pro potřeby rozpoznávání digitálních modulací pomocí umělých neuronových sítí byla využita knihovna Neural Network Toolbox. Neural Network Toolbox je sada aplikací nabízející funkce pro modelování komplexních nelineárních systémů, které jsou obtížně analyticky modelovatelné. Neural Network Toolbox umožňuje design, trénování, vizualizaci a simulaci neuronových sítí a obsahuje nástroje pro vytváření neuronových sítí určených pro klasifikaci dat, předvídání časové řady, modelování dynamických systémů, řízení a další aplikace[31].

Postup návrhu struktury neuronové sítě je založen na testování různých struktur za účelem dosažení přijatelné přesnosti. Z hlediska řešení problému rozpoznávání diskrétně modulovaných signálů je velmi obtížné vybrat optimální strukturu umělé neuronové sítě[32].

Z důvodů uvedených v minulých kapitolách pro praktickou realizaci umělé neuronové sítě, která bude použita pro následující klasifikaci problému, byly zvolené sítě se dvěma skrytými vrstvami a s jednou skrytou vrstvou, ale s modifikovaným počtem neuronů na výstupní vrstvě tak, aby počet neuronů odpovídal počtu druhů vybraných modulací. Tyto modifikované neuronové sítě jsou na obrázcích 6.2 a 6.3.



Obr. 6.2 Modifikovaná neuronová síť s jednou skrytou vrstvou

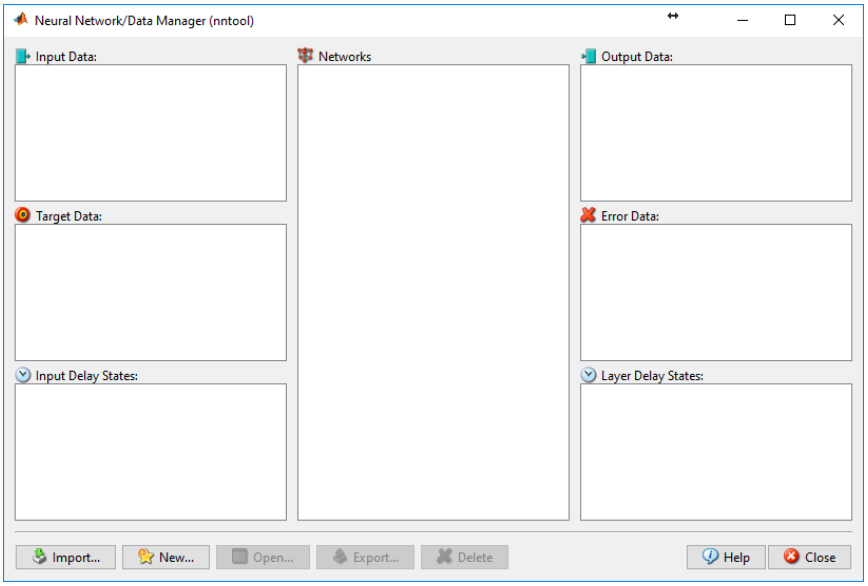


Obr. 6.3 Modifikovaná neuronová síť se dvěma skrytými vrstvami

Nandimu i Azzouzovi [8] [9], Richterové [32] i Iversnovi [33] se jako nejvhodnější pro řešení problému rozpoznávání diskrétních modulovaných signálů jeví vícevrstvé umělé neuronové sítě. Umělá neuronová síť, která bude použita pro tento problém klasifikace, je vícevrstvý perceptron (Multi-Layered Perceptron - MLP). V Matlabu je pojmenována jako feed-forward back-propagation network.

Tato síť může být realizována pomocí grafického správce sítě *nntool* (obr. 6.4). MLP se musí skládat ze vstupní vrstvy, jedné skryté vrstvy nebo dvou skrytých vrstev a výstupní vrstvy. Vstupní vrstva musí mít pět uzlů, z nichž každý bude představovat

jeden z pěti klíčových příznaků uvedených výše. Výstupní vrstva musí mít také pět uzlů, z nichž každý představuje jedno z pěti modulačních schémat. Počet uzlů ve skrytých vrstvách je zvolen podle základní neuronové sítě Azzouza i Nandiho [8].

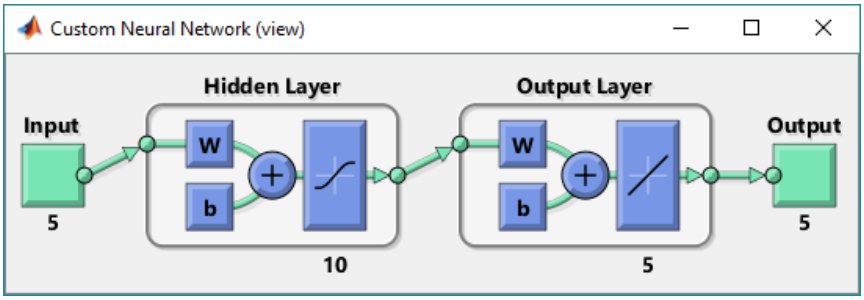


Obr. 6.4 NNTOOL v Matlab

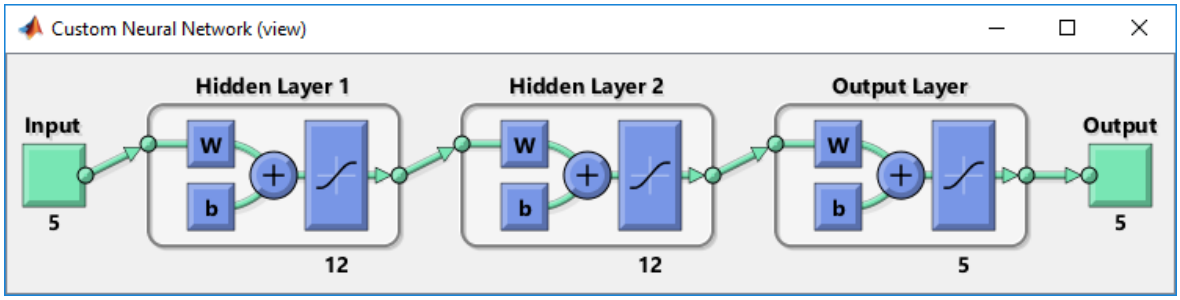
Nntool nabízí celou řadu školení a učení neuronových sítí. Zde bude zvolena cvičná metoda Levenberg-Marquardt (LM), protože prokázala dobré výsledky v podobném problému klasifikace [8], [33], [34], ale dále může být zaměněna za výhodnější metodu. Metoda LM je iterační metoda, která řeší problém minimalizace sumy kvadrátu odchylek obecné nelineární funkce [35]. Algoritmus LM je považován za jeden z nejrychlejších tréninkových algoritmů, ale jeho hlavní nevýhodou je, že má požadavek na velikost paměti [36]. Existují způsoby, jak snížit využití paměti, ale tím je zvýšena doba provádění.

Uzly v neuronové síti mohou mít různé aktivační funkce. Mezi nejčastější aktivační funkce ve vícevrstvém perceptrone patří esovitý log-sigmoid, který se pohybuje v rozmezí od 0 do 1 a tan-sigmoid, který se pohybuje od -1 do 1. Podle Haykina [37] se síť obecně učí rychleji při použití funkce aktivace tan-sigmoid. Přístup, který zde bude použit, je identický s [37], který používal tan-sigmoid pro skryté vrstvy a log-sigmoid pro výstupní vrstvu.

Vytvořené umělé neuronové sítě pro následující učení a klasifikaci v diplomové práci jsou na obrázcích 6.5 a 6.6.



Obr. 6.5 Vytvořená neuronová síť pro klasifikaci s jednou skrytou vrstvou



Obr. 6.6 Vytvořená neuronová síť pro klasifikaci s dvěma skrytými vrstvami

6.5 Učení umělé neuronové sítě

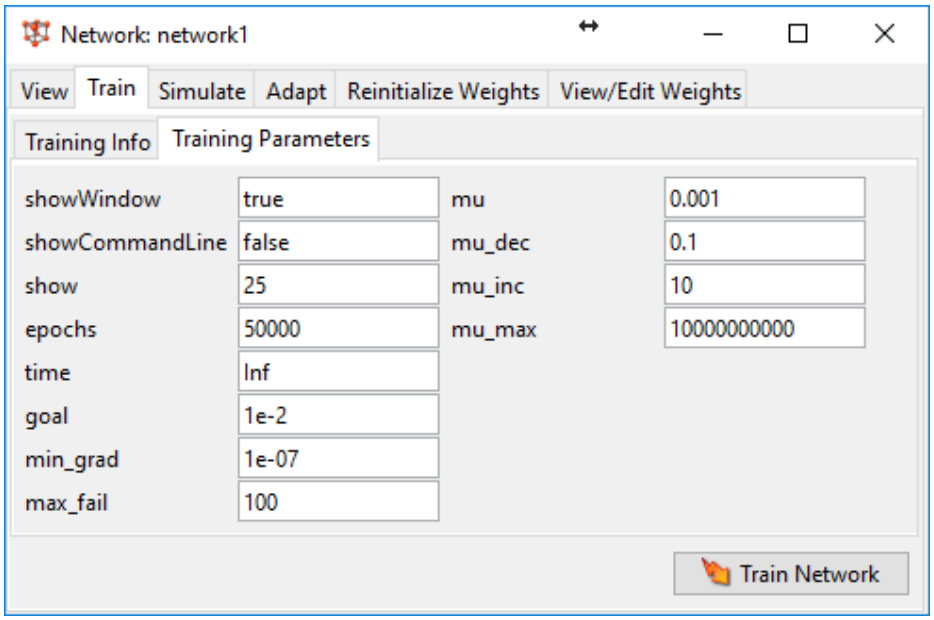
Dosud jsme se zabývali pouze přípravou koeficientů, které budou vstupovat do neuronové sítě. Samotný proces rozpoznávání zabezpečuje již natrénovaná (adaptovaná) síť.

Jak bylo již zmíněno, vytvořené neuronové sítě využívají učeni s učitelem. Po předchozích krocích se program dostal do fáze, kdy jsou vypočítány příznaky různých druhů digitálních modulací, a je vytvořena trénovací množina, ze které jsou jako vektor předány příznaky do vstupu neuronové sítě, a jako vektor jsou předány cíle na výstup neuronové sítě.

Pro klasifikaci je použit vícevrstvý perceptron s trénovacím algoritmem zpětného šíření chyby (backpropagation). Jak již bylo v teorii uvedeno, jedná se o neuronovou síť, jejíž úlohou je klasifikovat vzory do předem známých tříd. V našem případě bude neuronová síť klasifikovat příznaky do pěti tříd.

Neuronová síť se snaží srovnáváním aktuálního výstupu s požadovaným výstupem (učitelem) přenastavit váhy sítě tak, aby se při daném konkrétním vstupu snížil rozdíl mezi skutečným a požadovaným výstupem.

Pro učeni neuronové sítě stačí v Nntool v prostředí Matlab zvolit parametry učeni a stisknout tlačítko „Train Network“(obr. 6.7). Zvolené parametry jsou vybrány podle těch parametrů, které uvedli ve své prací Azzouz a Nandi[8]. Při dosažení nějakého kritéria stop funkce se síť přestává učit a můžeme následně ověřit její chybovost.



Obr. 6.7 Parametry učení neuronové sítě

Všechny neuronové sítě dosáhly stop kritéria v 50000 epochách a přestaly se učit. Neuronová síť s jednou skrytou vrstvou bez normalizace příznaků se učila skoro 20 hodin a s normalizací příznaků se učila skoro 12 hodin. Neuronová síť se dvěma skrytými vrstvami bez normalizace příznaků se učila skoro 30 hodin a s normalizací příznaků skoro 16 hodin. To je docela dlouhá doba a neexistuje dostatečná jistota, že použité parametry neuronové sítě přivedou ke kvalitnímu výsledku rozpoznávaní druhů modulace. Je potřeba optimalizovat proces učení.

6.6 Vytvořeni aplikace výpočtu chybovosti neuronové sítě

Následujícím krokem je implementace aplikace, která vytvoří nové odlišné modulace různých typu signálů, spočítá nové příznaky a provede rozpoznávaní pomocí simulace neuronové sítě a porovná výsledek s cíli.

Dále je ukázáno na příklade modulace ASK, jak byla provedena implementace. Simulace se provádí pro každou hodnotu SNR a hodnotu vzorkovacího kmitočtu. Celkový počet těchto simulací odpovídá zvolenému počtu pokusů.

```
for i= SNRValueMin:SNRpres:SNRValueMax
      for j=2:1:10
        fv=j*fc;              % vzorkovaci kmitocet
           for j=1:1:PocetPokusu
            [Mod_sig1,t1] = ASK(T,R,fv,fc,i);
            forNetASK = VypocetPriznaku(Mod_sig1,t1);
            %forNetASK = forNetASK/max(max(forNetASK)); % normalizace
            AnsASK = round(sim(net,forNetASK'));
            MaticeASK = [MaticeASK; AnsASK'];
       end
    end
end
```

kde `SNRValueMin` je minimální hodnota SNR, `SNRpres` je krok SNR, `SNRValueMax` je maximální hodnota SNR, `fv` je vzorkovací kmitočet, `fc` je kmitočet nosné, `ModSig1` je modulovaný signál, `t1` je časový vektor, `T` je celková doba simulace, `VypocetPriznaku` je aplikace výpočtu příznaků, `net` je naučená neuronová síť, `AnsASK` je získaná odpověď, `MaticeASK` je matice osahujíce získané odpovědi.

Simulace neuronové sítě se provádí pomocí příkazů „sim“ v prostředí Matlab. Výsledek je potřeba zaokrouhlit, aby bylo možné klidně srovnávat s hodnotami cílů. Pro následující výpočty uložíme výsledky do pomocné matice, kde každá provedená simulace bude mít svůj řádek.

Tady byla také realizována možnost využití normalizace příznaků. V případě potřeby normalizace hodnot příznaků pro ověření chybovosti neuronové sítě stačí odkomentovat řádek normalizace.

Dále je potřeba srovnat zapsaný výsledek s cíli, kterých muselo být dosaženo. V případě úspěšného rozpoznávání přidáváme 1 pro zvolený typ modulace a až cyklus přestane počítat, spočítáme procento úspěchu a zapíšeme toto procento úspěšného rozpoznávání do pomocné matice pro každou hodnotu SNR zvlášť.

```
   for o=1:1:PocetPokusu
    indASK = (i-SNRValueMin)*PocetPokusu/SNRpres+o;
        if MaticeASK(indASK,1) == 1 && ...
           MaticeASK(indASK,2) == 0 && ...
           MaticeASK(indASK,3) == 0 && ...
           MaticeASK(indASK,4) == 0 && ...
           MaticeASK(indASK,5) == 0

           UspechASK =UspechASK+1;

        end
   end
ProcUspechASK = UspechASK*100/PocetPokusu;
MaticeProcUspechASK = [MaticeProcUspechASK; ProcUspechASK];
```

kde `SNRValueMin` je minimální hodnota SNR, `SNRpres` je krok SNR, `MaticeASK` je matice osahujíce získané odpovědi, `indASK` je index v matice, `UspechASK` je hodnota úspěšného rozpoznávaní, `ProcUspechASK` je hodnota úspěšného rozpoznávaní v procentech, `MaticeProcUspechASK` je matice obsahujíce získaná procenta.

Pro jednodušší analýzu výsledků nakreslíme grafy se všemi hodnotami úspěšného rozpoznávání pro každý typ modulace. Hodnoty úspěšného rozpoznávání jsou závislé na hodnotě SNR.

```
figure('Name','All-in-One','NumberTitle','off')
plot(SNR, MaticeProcUspechASK,SNR,MaticeProcUspechFSK,SNR,...
MaticeProcUspechBPSK,SNR,MaticeProcUspechQPSK,...
SNR,MaticeProcUspech16QAM);
legend('ASK','FSK','BPSK','QPSK','16QAM');
axis([ SNRValueMin SNRValueMax 0 100]);
grid on;
xlabel('SNR');
ylabel('Procent uspechu');
title('All in One');
```

kde `SNR` je matice hodnot SNR, `SNRValueMin` je minimální hodnota SNR, `SNRValueMax` je maximální hodnota SNR, `MaticeProcUspechASK` je matice obsahujíce získaná procenta úspěšného rozpoznávání modulace ASK, `MaticeProcUspechFSK` je matice obsahujíce získaná procenta úspěšného rozpoznávání modulace FSK, `MaticeProcUspechBPSK` je matice obsahujíce získaná procenta úspěšného rozpoznávání modulace BPSK, `MaticeProcUspechQPSK` je matice obsahujíce získaná procenta úspěšného rozpoznávání modulace QPSK, `MaticeProcUspech16QAM` je matice obsahujíce získaná procenta úspěšného rozpoznávání modulace 16QAM.

Autorem této diplomové práce byly zvoleny následující parametry testování umělé neuronové sítě, které jsou uvedeny v tabulce 6.6. Čím vyšší bude počet pokusů na testování umělé neuronové sítě pro každou hodnotu SNR, tím bude vyšší přesnost výsledku chybovosti.

Tab. 6.6 Parametry testování umělé neuronové sítě

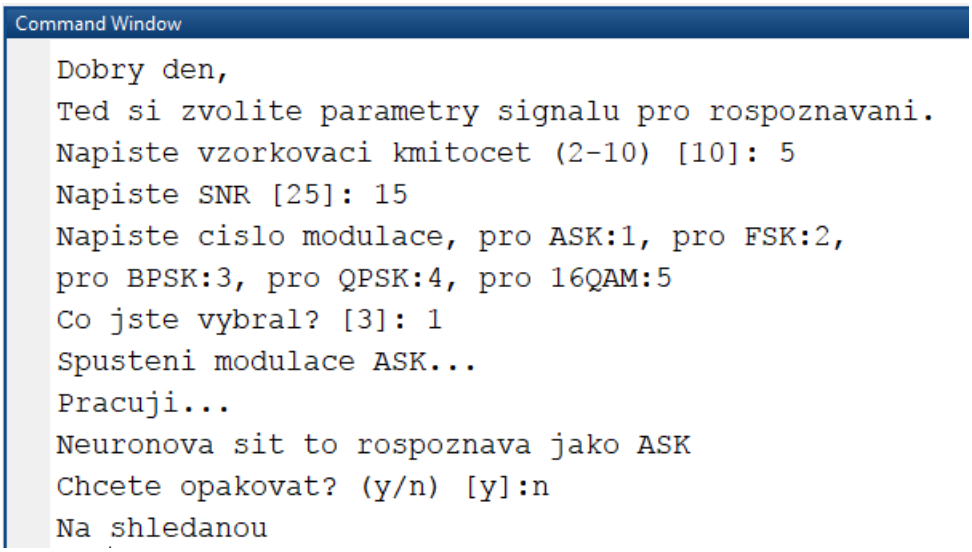
Signal-to-noise ratio, SNR	[-10; 50]
Vzorkovací kmitočet, fv	$j \cdot fc$, kde fc je nosný kmitočet a $j \in [2; 10]$
Počet opakování pro každý SNR a vzorkovací kmitočet, PocetPokusu	100
Kmitočet nosného signálu	10 MHz
Počet bitů	10000
Délka analyzovaného signálu	20001 vzorků

6.7 Vytvořeni aplikace pro využití neuronové sítě

Posledním krokem je implementování aplikace, která umožní využití vytvořené a naučené umělé neuronové sítě. Základem je možnost použít libovolné nastavení parametrů signálu, včetně možnosti zvolit druh modulace pro následující rozpoznávaní, protože v práci se vždy setkáme s náhodnými signály, o kterých nic nevíme.

Aplikace byla vytvořena formou dialogu s uživatelem v příkazovém řádku (CLI — Command-line interface) v prostředí Matlab. Příkazový řádek představuje uživatelské rozhraní, ve kterém uživatel s programy nebo operačním systémem komunikuje zapisováním příkazů do příkazového řádku. Na rozdíl od textového rozhraní a grafického uživatelského rozhraní nevyužívá myš ani menu a nedovede pracovat s celou plochou obrazovky (terminálu)[38]. Uživatel vkládá příkazy napsáním jejich názvu a stiskem klávesy „Enter“. Interpret příkazů pak vložený text přijme, analyzuje a spustí příslušný program. Uživatel může zvolit libovolné hodnoty SNR, vzorkovacího

kmitočtu a vybrat typ modulace, který je potřebný. Na obrázku 6.8 je ukázáno, jak vypadá implementována aplikace a jak probíhá dialog s uživatelem.



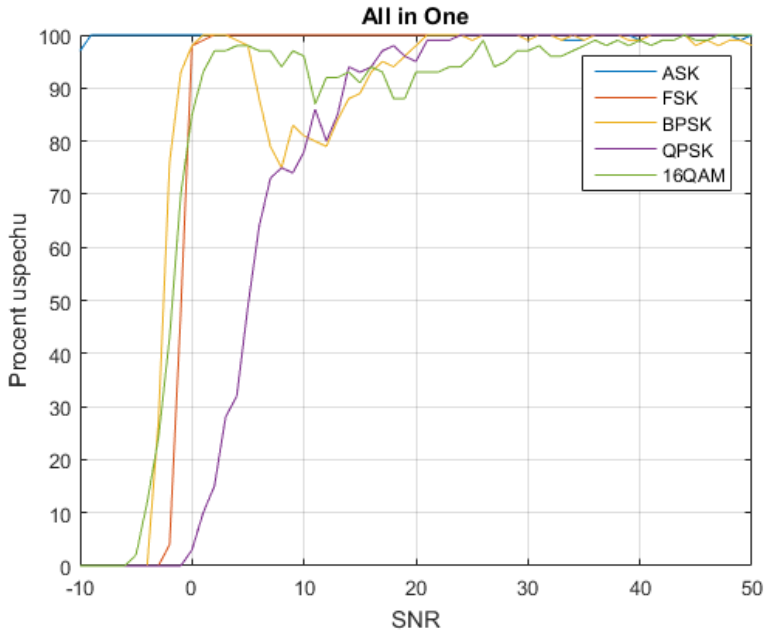
Obr. 6.8 Příklad dialogu implementované aplikace pro využití neuronové sítě

6.8 První experimenty

Naše první experimenty s umělou neuronovou sítí se budou skládat ze dvou částí. V první části porovnáme výsledky našich vytvořených a naučených neuronových sítí s jednou skrytou vrstvou bez a s normalizací příznaků pomocí taky vytvořené aplikace výpočtu chyb. Cílem druhé části je porovnat výsledky našich vytvořených a naučených neuronových sítí se dvěma skrytými vrstvami bez a s normalizací příznaků. Tímto ověříme existující metodu rozpoznávaní digitálních modulací pomocí neuronových sítí.

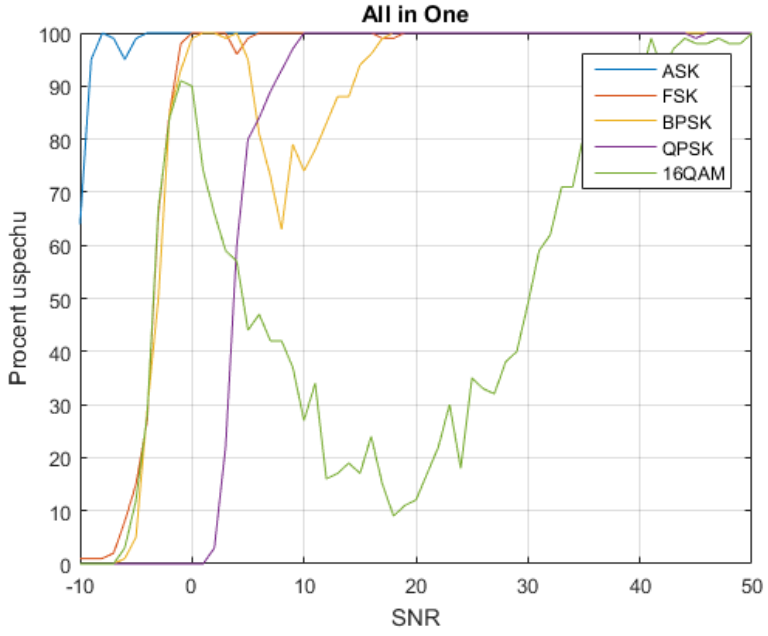
Postupně spustíme vytvořené aplikace výpočtu úspěchu rozpoznávaní, pro každou naučenou neuronovou síť zvlášť.

První test uděláme s neuronovou sítí s jednou skrytou vrstvou bez normalizace příznaků. Graf s výsledkem je uveden na obrázku 6.9. Podle grafu je vidět, že celkový výsledek je docela dobrý, ale existuje nějaký problém rozpoznávaní druhů modulace BPSK a QPSK při hodnotách SNR nižších než 15dB, což může ovlivnit střední úspěch rozpoznávaní. Je patrné, že rozložení úspěšnosti je rovnoměrné.



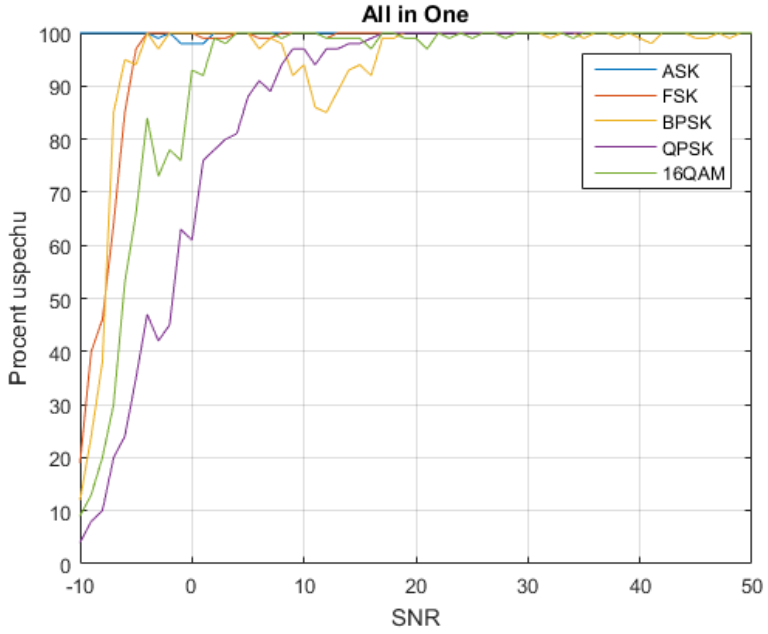
Obr. 6.9 Graf úspěchů rozpoznávání pomocí neuronové sítě s jednou skrytou vrstvou

Druhy test uděláme s neuronovou sítí s jednou skrytou vrstvou s normalizací příznaků. Graf s výsledkem je uveden na obrázku 6.10. Podle grafu je vidět, že neuronová síť se naučila dobře určit typy modulace ASK, FSK a QPSK, ale má velký problém s odlišením modulace 16QAM. Je patrné, že rozložení úspěšnosti není rovnoměrné.



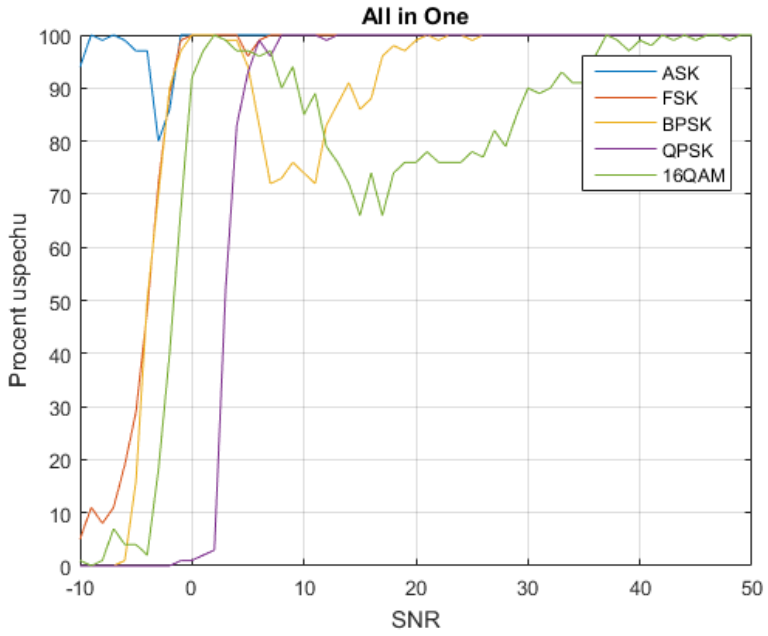
Obr. 6.10 Graf úspěchů rozpoznávání pomocí neuronové sítě s jednou skrytou vrstvou s normalizací příznaků

Třetí test uděláme s neuronovou sítí se dvěma skrytými vrstvami bez normalizace příznaků. Graf s výsledkem je uveden na obrázku 6.11. Podle grafu je vidět, že máme malý problém s rozpoznáváním druhů modulace BPSK a QPSK při velmi nízkých hodnotách SNR, ale celkem je výsledek ideální, a můžeme hned začít využívat neuronovou síť. Je patrné, že rozložení úspěšnosti je rovnoměrné.



Obr. 6.11 Graf úspěchů rozpoznávání pomocí neuronové sítě se dvěma skrytými vrstvami

Poslední test uděláme s neuronovou sítí se dvěma skrytými vrstvami s normalizací příznaků. Graf s výsledkem je uveden na obrázku 6.12. Podle grafu je vidět, že normalizace příznaků znova zhoršila odlišení druhu modulace 16QAM, jako je ve druhém testu s neuronovou sítí s jednou skrytou vrstvou. Rozpoznávání modulace BPSK se také zhoršilo při hodnotách SNR do 20dB. Je patrné, že rozložení úspěšnosti není rovnoměrné.



Obr. 6.12 Graf úspěchů rozpoznávání pomocí neuronové sítě se dvěma skrytými vrstvami s normalizací příznaků

Azzouz a Nandi ve své práci[8], spočítávají střední hodnotu úspěšného rozpoznávaní podle SNR, který má hodnoty 10dB a 20dB, protože přibližně s těmito hodnotami se můžeme nejčastěji setkat v praxi, a při SNR 30dB a vyšší hodnotě dochází téměř k nedostatku šumu. Při nižších hodnotách SNR je porucha signálu docela velká a výsledek rozpoznávání může být nekorektní. V této diplomové práci použijeme stejný způsob výpočtu střední hodnoty úspěchu. Výsledky úspěšného rozpoznávaní vytvořených neuronových sítí jsou uvedeny v tabulce 6.7.

Tab. 6.7 Výsledek testování naučených neuronových sítí

Parametry sítě a příznaků	**Procento správného rozpoznávání**			
	SNR 10dB	**SNR 20dB**	**Střední úspěch se šumem**	**Bez šumu**
1 skrytá vrstva bez normalizace příznaků	91%	97,2%	91,6%	99,6%
1 skrytá vrstva s normalizací příznaků	80,2%	82,4%	81,3%	100%
2 skryté vrstvy bez normalizace příznaků	98,2%	99,8%	99%	100%
2 skryté vrstvy s normalizací příznaků	91,8%	95%	93,4%	100%

Jak je vidět z tabulky 6.7 a provedených testů, nejhorších výsledků se šumem bylo dosaženo při použití normalizace příznaků. To je možné kvůli tomu, že využíváme v této diplomové práci jiné typy modulace, než jsou uvedeny v práci Azzouze a Nandiho [8]. Nejlepšího výsledku bylo dosaženo při využití neuronové sítě se dvěma skrytými vrstvami bez normalizace příznaků, který je skoro stejný jako v doporučené literatuře i když využíváme jiné typy modulací. V ideálním případě, když není žádný šum, každá neuronová síť se naučila dobře rozpoznávat různé typy modulace skoro se 100% úspěchem, ale v praxi se s těmito případy nesetkáme.

Podle provedených testů můžeme říct, že vybraná metoda rozpoznávaní druhů modulace pomocí umělé neuronové sítě dokázala dosáhnout dobrých výsledků, a že v našem případě je lepší používat trénovací množinu bez normalizace příznaků, a pro zrychlení učení neuronové sítě je lepší vybrat novou metodu. V další kapitole se budeme zabývat optimalizací parametrů neuronových sítí.

7. OPTIMALIZACE PARAMETRŮ NEURONOVÉ SÍTĚ

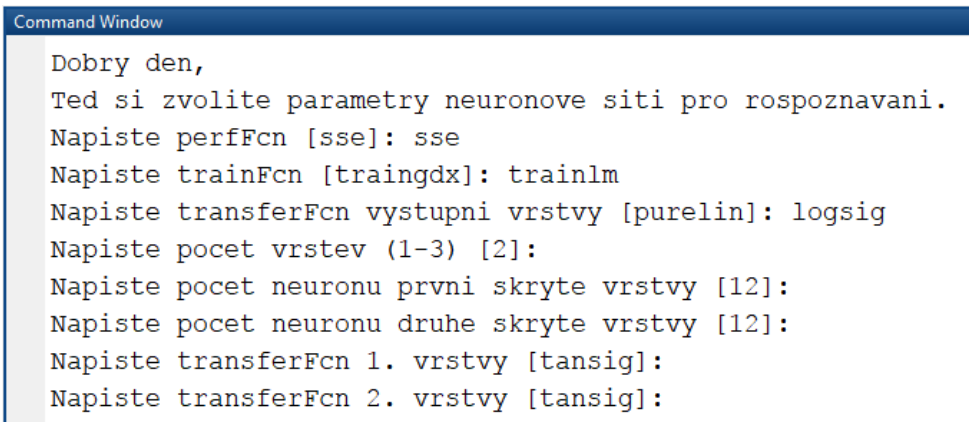
7.1 Příprava pro optimalizaci

Jak již bylo zmíněno v minulých kapitolách, mnohé parametry lze ještě třeba určit metodou pokus–omyl a stávající teoretické výsledky poskytují pouze slabé vodítko pro výběr těchto parametrů v praktických aplikacích. Tyto parametry jsou například:

- kolik vrstev je potřebných pro tento úkol,
- kolik byste měli zvolit prvků v každé vrstvě,
- jaké aktivační funkce použít,
- jaká velikost připravených vzorků je potřebná k dosažení „dobré“ schopnosti sítě generalizovat.

V minulé kapitole bylo dosaženo dobrých výsledků a díky tomu není potřeba zvětšovat počet vzorků v trénovací množině. Ostatní uvedené parametry je možné změnit při vytvoření umělé neuronové sítě. Pro snadnou změnu těchto parametrů vytvoříme aplikaci, která představuje dialog s uživatelem (na obr. 7.1) a povoluje zvolit následující parametry:

- funkce měření výkonnosti (`perfFcn` v Matlab)
- trénovací funkce (`trainFcn` v Matlab)
- funkce aktivace ve výstupní vrstvě (`transferFcn` v Matlab)
- počet vrstev neuronové sítě
- počet neuronů ve skryté vrstvě
- funkci aktivace ve skrytých vrstvách (`transferFcn` v Matlab)



Obr. 7.1 Příklad dialogu s uživatelem pro výběr parametrů neuronové sítě

Tato aplikace předává zvolené parametry také do vytvořených funkcí v prostředí Matlab, které umožní vytvořit a naučit neuronové sítě s jednou skrytou vrstvou, se

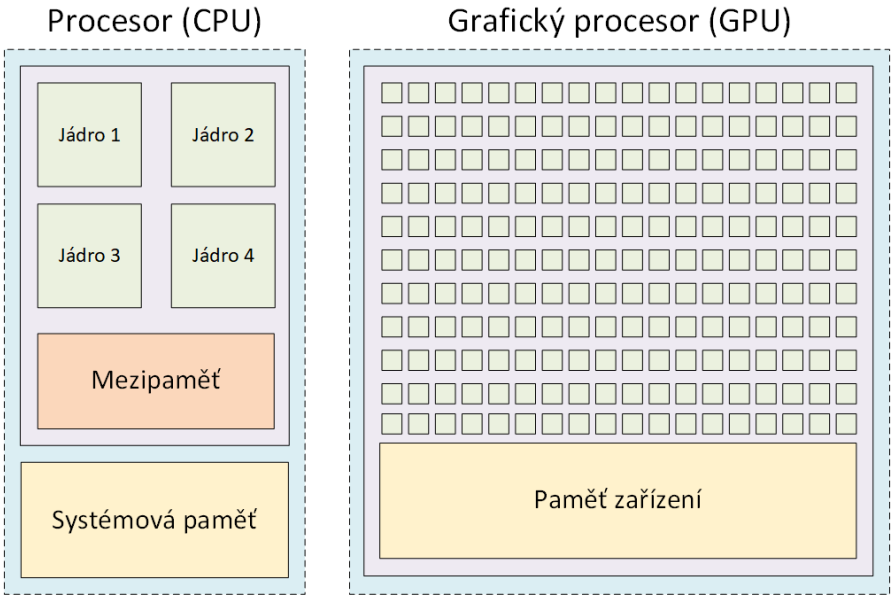
dvěma skrytými vrstvami nebo se třemi skrytými vrstvami. Tyto funkce byly automaticky vytvořeny pomocí grafického správce sítě *nntool.* Po předání těchto parametrů najdou tyto funkce vytvořenou trénovací množinu, a předávají z ní na vstup neuronové sítě příznaky modulace a na výstup neuronové sítě cíle, kterých má být dosaženo. Potom tyto funkce spustí proces učení neuronové sítě pomocí příkazu `train` v prostředí Matlab. Po skončení tréninku sítě lze uložit do souboru s názvem „*.mat“ pomoci příkazu `save`. Až proběhne proces učení, funkce spustí vytvořenou aplikaci výpočtu chybovosti neuronové sítě.

7.2 Výběr parametrů pro optimalizaci

Pro zrychlení učení neuronové sítě byla Azzouzem a Nandim [8] vybrána metoda normalizace příznaků, ale podle prvních testů nedokázala mít dobré výsledky pro zvolené druhy modulace a celková doba učení trvá poměrně dlouho. Naším úkolem je vybrat novější a vhodnější metodu pro zrychlení učení neuronové sítě.

Neuronové sítě jsou neodmyslitelně paralelní algoritmy. Více paralelních procesorů, grafických procesorů (GPU — graphic processing unit) a clusterů počítačů s více procesory a grafickými procesory mohou využít této paralelnosti.

Vícejádrové stroje a vícevláknová technologie umožňují vědcům, inženýrům a finančním analytikům urychlit výpočetně intenzivní aplikace v různých oborech. Další typ hardwaru dnes slibuje ještě vyšší výpočetní výkon: grafická jednotka.



Obr. 7.2 Porovnání CPU[1] a GPU

[1] CPU (central processing unit) — centrální procesorová jednotka je v informatice označení základní elektronické součásti v počítači, která umí vykonávat strojové instrukce, ze kterých je tvořen počítačový program a obsluhovat jeho vstupy a výstupy[39].

Původně použité k urychlení vykreslování grafiky, GPU mají stále větší uplatnění ve vědeckých výpočtech. Na rozdíl od tradičního procesoru, který obsahuje jenom pár jader, GPU mají masivně paralelní řadu celočíselných a s plovoucí desetinnou čárkou procesorů a vysokorychlostní paměť. Typický GPU obsahuje stovky těchto menších procesorů (obr. 7.2) [43].

Souprava Parallel Computing Toolbox, je-li použita ve spojení s Neural Network Toolbox, umožňuje učení a simulaci neuronových sítí s využitím každého z paralelních režimů. Pro využití paralelních výpočtů pomocí CPU je nutné funkce učení neuronové sítě doplnit příkazem `'useParallel','yes'`. Aby bylo možné snadno využívat grafický procesor, stačí funkci učení neuronové sítě doplnit příkazem 'useGPU','yes' a prostředí Matlab začne využívat existující grafickou kartu na počítači. Dále jsou uvedeny příklady těchto navržených příkazů.

```
[network]=train(network,P,T,'useParallel','yes'); % Využití CPU
[network]=train(network,P,T,'useGPU','yes'); % Využití GPU
```

kde `network` je vytvořená síť, `P` je matice příznaků, `T` je matice cílů.

V této diplomové práci pro následující testy budeme používat jenom metodu využívající grafickou jednotku, protože čas učení neuronové sítě bude několikrát nižší než při využití CPU.

Značně zvýšená propustnost umožněna pomocí GPU však něco stojí. Podle dokumentace Matlab není možné využívat trénovací metodu Levenberga-Marquardta (LM), je potřeba ji vyměnit například za algoritmus Variable Learning Rate (`traingd` nebo `traingdx` v Matlab). *traingd* je funkce školení sítě, která aktualizuje hodnoty vah a předsudku podle gradientu sestupu chybnosti [41]. *traingdx* je funkce školení sítě, která aktualizuje hodnoty vah a předsudku podle gradientu sestupu chybnosti a adaptivní rychlosti učení[42].

7.3 Provedení testů

Ted je potřeba otestovat různé parametry neuronové sítě¨. Zde budou uvedeny pouze nejlepší výsledky. Postup hledání optimální konfigurace již byl uveden dříve a obsahuje jenom metodu pokus–omyl. Také budeme postupně přidávat a odebírat neurony a sledovat průběh trénování a porovnáme výsledky nad testovacími daty.

Autorem této diplomové práce pro následující testy byla využita jedna grafická karta nVIDIA GIGABYTE GTX 970, která má 1664 jader CUDA (Compute Unified Device Architecture). CUDA je hardwarová a softwarová architektura, která umožňuje na GPU spouštět programy napsané v jazycích C/C++, FORTRAN nebo programy postavené na technologiích OpenCL, DirectCompute a jiných. Použití této architektury je omezeno pouze na grafické akcelerátory společnosti nVIDIA, která ji vyvinula[40]. Jak již bylo uvedeno v minulé podkapitole, pomocí známého jazyka MATLAB můžeme využívat

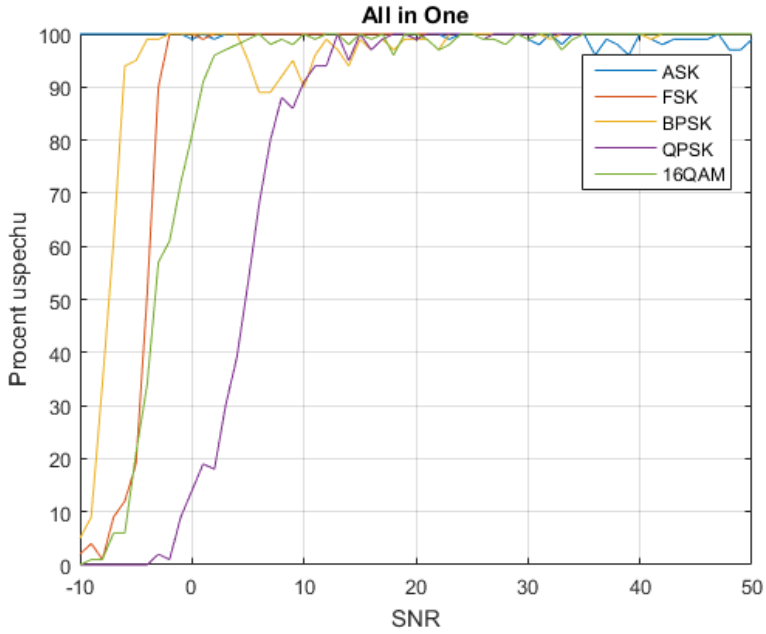
výpočetní technologii CUDA GPU, a nemusíme se naučit složitosti architektur GPU nebo použití nízkoúrovňových výpočetních knihoven GPU.

Celkem bylo provedeno 60 různých testů, a nejvíce testů bylo provedeno s neuronovou sítí se dvěma skrytými vrstvami, protože dokázala v předchozí kapitole dosáhnout velmi výborných výsledků.

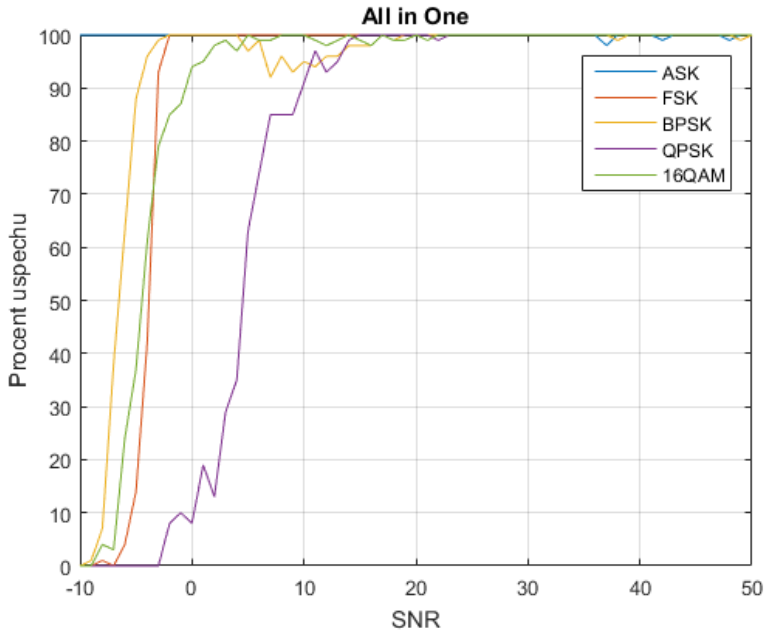
Bylo zjištěno, že zvyšováním počtu neuronů ve skrytých vrstvách ztrácí neuronová síť schopnost generalizace a dochází k přetrénování. Naopak snížení počtu vykazuje lepší výsledky klasifikace druhů modulace.

Při klasifikaci druhů modulace bylo autorem této diplomové práce dosaženo zajímavých výsledků s jednovrstvými sítěmi, kde ve skryté vrstvě byl trojnásobek počtu výstupních neuronů.

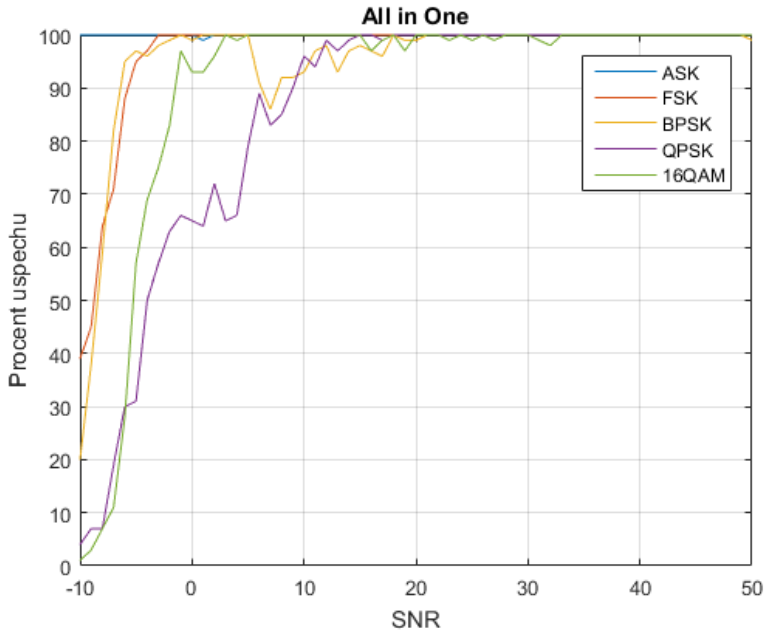
Dále na obrázcích 7.3,7.4 a 7.5 jsou uvedeny grafy úspěšného rozpoznávání pro neuronové sítě s jednou, se dvěma a se třemi skrytými vrstvami, se kterými byly získány nejlepší výsledky, a rozložení úspěšnosti je rovnoměrné.



Obr. 7.3 Graf nejlepších úspěchů rozpoznávání pomocí neuronové sítě s jednou skrytou vrstvou



Obr. 7.4 Graf nejlepších úspěchů rozpoznávání pomocí neuronové sítě se dvěma skrytými vrstvami



Obr. 7.5 Graf nejlepších úspěchů rozpoznávání pomocí neuronové sítě se třemi skrytými vrstvami

Jak je vidět z tabulky 7.1, díky novým funkcím učení a jiným parametrům učení neuronových sítí se podařilo získat lepší výsledky pro neuronovou síť‘ s jednou skrytou vrstvou a velmi podobné výsledky pro neuronovou síť‘ se dvěma skrytými vrstvami. Neuronová síť‘ se třemi skrytými vrstvami také dostala dobrých výsledků.

Tab. 7.1 Nejlepší výsledky testováni naučených neuronových sítí s využitím GPU

Parametry sítě	Procento správného rozpoznávání			
	SNR 10dB	SNR 20dB	Střední úspěch se šumem	Bez šumu
1 skrytá vrstva, sigmoidální přenosová funkce ve skryté vrstvě, funkce hyperbolické tangenty ve výstupní vrstvě	96,2%	99,6%	97,9%	99,8%
2 skryté vrstvy, přenosová funkce hyperbolické tangenty ve skrytých vrstvách. lineární funkce ve výstupní vrstvě	98,2%	100%	99,1%	100%
3 skryté vrstvy přenosová funkce hyperbolické tangenty ve skrytých vrstvách. lineární funkce ve výstupní vrstvě	97,8%	99,2%	98,5%	99,8%

Doba tréninku neuronové sítě, díky zvolené metodě učení pomocí grafické karty, se značně snížila a teď je v rozsahu od 20 minut do 1 hodiny pro stejné parametry učení s výjimkou na trénovací funkce.

7.4 Shrnutí

- Nejlepších výsledků bylo dosaženo kombinací přenosové funkce hyperbolické tangenty ve skrytých vrstvách s lineární funkcí ve výstupní vrstvě.
- Lepších výsledků bylo dosaženo s neuronovou sítí se dvěma skrytými vrstvami.
- Nejhůře rozpoznatelným typem modulace je QPSK.
- Doba učení neuronové sítě při využití grafické karty se snížila někdy 90x a minimálně trvá kolem 20 minut.

8 ZÁVĚR

V rámci diplomové práce byly popsány některé metody klasifikace digitálních signálů. Z nich byla vybrána příznaková metoda, protože je jednodušší z hlediska výpočetní složitosti a je nejméně náročná na parametry signálu, a protože se často jako klasifikátor používají umělé neuronové sítě. Vzhledem k existujícím publikacím na základě klíčových příznaků jsem uvedl, že nejlepších výsledků bylo dosaženo s využitím neuronových sítí.

Dále byly popsány druhy modulací, které mají být použity pro rozpoznávání. Nejvíce pozornosti bylo věnováno charakteristickým vlastnostem signálů.

V práci byly popsány teoretické základy nutné pro dobré pochopení principů nejpoužívanějších neuronových sítí. Bylo zde popsáno pozadí algoritmu zpětného šíření chyby – backpropagation.

V samostatné kapitole byla uvedena metoda rozpoznávání digitálních modulací pomocí umělých neuronových sítí, která tam byla podrobně popsána, stejně jako byly popsány vybrané klíčové příznaky pro následující klasifikace. Byly charakterizovány schémata neuronových sítí, které Azzouz a Nandi [8] rozpracovali ve své práci.

Jedním z hlavních cílů této diplomové práce bylo vytvoření modulovaných signálů a navržení příznaků, které budou použity pro klasifikaci digitálních modulací. Parametry modulovaných signálů byly odebrány v práci, rovněž byly zobrazeny získané příznaky vytvořených modulovaných signálů různých typů modulací.

Součástí diplomové práce bylo vytvoření neuronové sítě, kterou je nutné naučit. Z knihovny Neural Network toolbox byla vybrána vrstvená perceptronová síť, která je pro klasifikaci nejčastěji používána jinými autory. Parametry neuronové sítě byly také vybrány podle prací jiných autorů. Také byly vytvořeny trénovací množiny a následně provedena učení neuronových sítí.

Existující metoda dokázala dosáhnout dobrých výsledků klasifikace druhů digitálních modulací, ale vykazovala velmi dlouhou dobu trvání procesu učení neuronové sítě. Nejlepších výsledků bylo dosaženo s dvouvrstvou neuronovou sítí s 12 neurony ve skryté vrstvě. Celková úspěšnost klasifikátoru je 99%. Je možné pokusit se tyto klasifikace použít pro rozpoznávání reálných vzorků modulovaných signálů.

Poslední kapitola popisuje přípravu pro optimalizaci parametrů neuronových sítí a zvolenou metodu pro zkrácení času tréninku neuronových sítí. Realizované experimenty poskytují přehled o úspěšnosti klasifikace a ukazuji zkrácení doby tréninku někde i 90x. Celková úspěšnost klasifikátoru po optimalizaci je 99,1%. Je pravděpodobné, že existuje výhodnější kombinace parametrů, se kterými bude síť schopná dosáhnout lepších výsledků. Proto by při dalším pokračování bylo vhodné otestovat i jiné metody učení, kterých je v prostředí Matlab hodně.

LITERATURA

[1] STEPANOV, A. a S. MATVEEV. Metody automatického rozpoznávání druhů a parametrů modulace [online]. [cit. 2016-10-12]. Dostupné z: goo.gl/iX2RTW

[2] Decision theory. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-11-12]. Dostupné z: https://en.wikipedia.org/wiki/Decision_theory

[3] SHKLYAEVA, ANNA. Automatická klasifikace digitálních modulací. Brno, 2008. doktorská práce. Vysoké učení technické v Brně. Vedoucí práce Doc. Ing. Vít Novotný, Ph.D.

[4] LIEDTKE, F. Computer simulation of an automatic classification procedure for digitally modulated communication signals with unknown parameters. Signal Processing, vol. 6, no. 4, pp. 311–323, Aug. 1984.

[5] SOLIMAN, S. S., HSUE, S. Signal classification using statistical moments. IEEE Trans. On Communication, COM 40(5), May, 908-916. 1992.

[6] WHELCHEL, J. E., MCNEILL, D. L., HUGHES, R. D., LOOS, M. M. Signal understanding: An artificial intelligence approach to modulation classification. Tools for Artificial Intelligence: Architectures, Languages and Tools, IEEE international Conf., Fairfax, VA, Oct. 231-236. 1989.

[7] GHANI, N., LAMONTAGNE, R. Neural networks applied to the classification of spectral features for automatic modulation recognition. MILCOM,Conf., 1, Boston, MA, Oct., 111-115. 1993.

[8] ELSAYED AZZOUZ AND A.K. NANDI. Automatic modulation recognition of communication signals. Boston: Kluwer Academic Publishers, c1996. ISBN 0792397967

[9] ELSAYED AZZOUZ AND A.K. NANDI. Automatic modulation recognition of communication signals. New York: Springer, 2011. ISBN 1441951660.

[10] HERO, A. O., HADINEJAD-MAHRAM, H. Digital modulation classification using power moment matrices. Acoustics, Speech and Signal Processing, IEEE International Conf. on, ICASSP 98, Vol. 6, Seattle, Washington, USA, 12-15 May, 3285-3288. 1998.

[11] LALLO, P. Signal classification by discrete Fourier transform. MILCOM Conf. Proceedings, 1, Atlantic City, NJ, 31Oct.-3Nov., 197-201. 1999.

[12] MOBASSERI, B. Digital modulation classification using constellation shape. Signal Processing, Jan., 80(2), 251-277. 2000.

[13] BOUDREAU, D., DUBUC, C., PATENAUDE, F., DUFOUR, M., LODGE, J., INKOL, R. A fast automatic modulation recognition algorithm and its implementation in a spectrum monitoring application. In Proceedings of the IEEE Military Communications Conference MILCOM 2000. October 2000, p. 732 – 736.

[14] LOPATKA, J., PEDZISZ, M. Automatic modulation classification using statistical moments and a fuzzy classifier. Signal Processing Proceedings,WCCC- ICSP 2000, 5th International Conf., Beijing, China, Aug. 3, 1500-1506. (2000).

[15] WONG, M. L. D., NANDI, A. K. Automatic modulation recognition using spectral and statistical features with multi layer perceptrons. Sixth International Symposium on Signal processing and its Application, Kuala Lumpur, 2, Aug., 390-393. 2001.

[16] TAIRA, S. Automatic classification of QAM signals by neural networks. in Proc. ICASSP 2, Salt Lake City, Utah, May, 1309–1312. 2001.

[17] KALININ, V., KAVALOV, D. Application of SAW artificial neural network processor to digital modulations. Ultrasonics Symposium, 1, San Juan, Puerto Rico, Oct., 51-54. 2000.

[18] KAVALOV, D., KALININ, V. Improved noise characteristics of SAW artificial neural network RF signal processor for modulation recognition. Ultrasonics Symposium, 1, Oct., Atlanta, USA, 19-21. 2001.

[19] DELGOSHA, F., MENHAJ, M. B. Amplitude-based neuro-classifier for classification of digital quadrature and staggered modulations. Neural Networks, Proceedings, IJCNN '01, International Joint Conf., 1, Washington DC, USA, July, 721-725. 2001.

[20] RAMAKONAR, V., HABIBI, D., BOUZERDOUM, A. Classification of bandlimited FSK4 and FSK8 signals. Signal Processing and its Applications, Sixth International symposium, 2, Kuala Lumpur, Malaysia, Aug., 398-401. 2001.

[21] SPOONER, C. M. On the utility of sixth-order cyclic cumulants for RF signal classification. Conf. Record of Thirty-Fifth Asilomar Conference, on Signals, Systems and Computers, 1, Pacific Grove, CA, Nov., 890-897. 2001.

[22] NIKOOFAR, H. R., SHERAFAT, A. R., SHAHMOHAMMADI, M. Modulation recognition for PSK/QAM signals using constellation features and soft clustering. ICEE, Tabriz, IRAN, 338-344. 2002.

[23] HSUE, S. Z., SOLIMAN, S. S. Automatic Modulation Recognition of Digitally Modulated Signals. In Proceedings of the IEEE Military Communications Conference MILCOM 1989. 1989, p.0645-0649.

[24] ČÍŽ R. Principy modulací a přenosu sdělovacích signálů pro integrovanou výuku VUT a VŠB-TUO. 1. vyd. Brno: Vysoké učení technické v Brně, 2014. 140 s. ISBN 978-80-214-5117-9.

[25] BUNIN, S. a L. JAJNENKO. Příručka radioamatéra. Tehnika. Kiev, 1984

[26] KENNEDY a GEORGE. Electronic communication systems. 4. ed. Lake Forest, Ill.: MacMillan/McGraw-Hill, 1993. ISBN 0071126724.

[27] XIONG, F. Digital Modulation Techniques. London: ARTECH HOUSE, INC., 2000. ISBN 0-89006-970-0

[28] The McCulloch-Pitts Model of Neuron [online]. [cit. 2016-12-12]. Dostupné z: goo.gl/CifuXg
[29] VESELOVSKÝ, M. Neuronové sítě [online]. Dostupné z: http://www.avari.cz/uir/index.php?pg=uceni
[30] TŮMA, Jiří. Zpracování signálů získaných z mechanických systémů užitím FFT. Praha: Sdělovací technika, 1997. ISBN 80-901936-1-7.
[31] Neural Network Toolbox [online]. MathWorks [cit. 2016-09-20]. Dostupné z: https://www.mathworks.com/products/neural-network.html
[32] RICHTEROVÁ, Marie a David JURÁCEK. Klasifikátor modulací s využitím umělé neuronové sítě. Univerzita obrany, Katedra KIS. PČR MŘ Brno.
[33] IVERSEN, Alexander. Classification of digital modulation schemes using multi-layered perceptrons. Heriot-Watt University, 2004.
[34] ARULAMPALAM G., RAMAKONAR V., BOUZERDOUM A., and D. HABIBI. Classification of digital modulation schemes using neural networks. In Proceedings of the International Symposium on Signal Processing and itsApplications, ISSPA99, pages 649-652, 1999.
[35] VELEBA, V., PIVOŇKA, P.: Adaptive Controller with Identification Based on Neural Network for Systems with Rapid Sampling Rates. WSEAS Transactions on Systems, 2005, vol. 4, issue 4, pp. 385-388. ISSN: 1109-2777
[36] HUDSON BEALE M., M.T. HAGAN a H.B. DEMUTH. Neural Network Toolbox User's Guide [online]. The MathWorks,Inc.2016 [cit. 2016-11-09]. Dostupné z: https://www.mathworks.com/help/pdf_doc/nnet/nnet_ug.pdf
[37] HAYKIN, Simon. Neural networks: a comprehensive foundation. 2nd ed. Delhi: Pearson Education, 1999. ISBN 8178083000.
[38] Příkazový řádek. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-04-16]. Dostupné z: https://cs.wikipedia.org/wiki/P%C5%99%C3%ADkazov%C3%BD_%C5%99%C3%A1dek
[39] Centrální procesorová jednotka. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-05-10]. Dostupné z:https://cs.wikipedia.org/wiki/Centr%C3%A1ln%C3%AD_procesorov%C3%A1_jednotka
[40] CUDA. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-05-10]. Dostupné z: https://cs.wikipedia.org/wiki/CUDA
[41] Gradient descent backpropagation. MATLAB: Documentation [online]. Massachusetts: MathWorks, 2017 [cit. 2017-05-3]. Dostupné z: https://www.mathworks.com/help/nnet/ref/traingd.html

[42] Gradient descent with momentum and adaptive learning rate backpropagation. MATLAB: Documentation [online]. Massachusetts: MathWorks, 2017 [cit. 2017-05-3]. Dostupné z: https://www.mathworks.com/help/nnet/ref/traingdx.html

[43] MATLAB GPU GPU Programming in MATLAB. MATLAB: Technical Articles and Newsletters [online]. Massachusetts: MathWorks, 2017 [cit. 2017-05-3]. Dostupné z: https://www.mathworks.com/company/newsletters/articles/gpu-programming-in-matlab.html

[44] Aditivní bílý Gaussovský šum. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-04-5]. Dostupné z: https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B4%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%B1%D0%B5%D0%BB%D1%8B%D0%B9_%D0%B3%D0%B0%D1%83%D1%81%D1%81%D0%BE%D0%B2%D1%81%D0%BA%D0%B8%D0%B9_%D1%88%D1%83%D0%BC

[45] Signal-to-noise ratio. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-04-5]. Dostupné z: https://en.wikipedia.org/wiki/Signal-to-noise_ratio

[46] ROJAS, Raúl. Neural networks: a systematic introduction. New York: Springer-Verlag, c1996. ISBN 978-3-540-60505-8.

[47] KLIMENKO, A.V. a V.F. URAZMETOV. The habitat of the information society. Internet [online]. Grant. Moskva: RCFTI, 1995 [cit. 2017-05-23]. ISBN 5-88835-001-X. Dostupné z: http://www.rfbr.ru/rffi/ru/books/o_39449

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ASK	-	Amplitude Shift Keying
BPSK	-	Binary Phase Shift Keying
QPSK	-	Quadrature Phase Shift Keying
FSK	-	Frequency Shift Keying
QAM	-	Quadrature Amplitude Modulation
AMR	-	Analog modulation recognition
DT	-	Decision Theory
FB	-	Feature Based
ANN	-	Artificial Neural Network
SNR	-	Signal to Noise Ratio
CLI	-	Command-line interface
AWGN	-	Additive white Gaussian noise
CUDA	-	Compute Unified Device Architecture
LM	-	trénovací metoda Levenberga-Marquardta
MLP	-	Multi-Layered Perceptron
CLI	-	Command-line interface
PDP	-	Parallel Distributed Processing
SD	-	Standard Definition
HD	-	High Definition
CPU	-	Central Processing Unit
GPU	-	Graphic Processing Unit
LED	-	Light-Emitting Diode
NRZ	-	Non Return To Zero
WiFi	-	Wireless Fidelity
RFID	-	Radio Frequency Identification
PAL	-	Phase Alternating Line
NTSC	-	National Television System(s) Committee
OpenCL	-	Open Computing Language
NNTOOL	-	Neural Network Toolbox
B		je počet bitů přenesených v jednom signálovém prvku
M		je celkový počet stavů signálu
$s_c(t)$		je harmonický nosný signál
S_c		je amplituda
ω_c		je úhlový kmitočet
t		je doba trvání signálového prvku
$g(t)$		je obdélníkový modulační signál
f_1 a f_2		jsou různé frekvence

Φ_1 a Φ_2 jsou počáteční fázi

S_{bI} a S_{bQ} jsou bipolární signály NRZ (Non Return To Zero)

I_i jsou vstupy neuronu

W_i jsou synaptické váhy

$f(x)$ je přenosová funkce neuronu (aktivační funkce)

y je výstup neuronu

$z(t)$ je analytický signál

$x(t)$ je reálný signál

$y(t)$ je Hilbertová transformace signálu $x(t)$

j je imaginární jednotka

$a(t)$ je okamžitá amplituda

$\psi(t)$ je okamžitá fáze

$\phi(t)$ je okamžitá fáze analytického signálu $z(t)$

N_s je počet vzorků v analyzovaném signálu

$a_{cn}(i)$ je hodnota centrované normované okamžité amplitudy

f_s je vzorkovací frekvence

$a_n(i)$ je normovaná okamžitá amplituda signálu

$a(i)$ je okamžitá amplituda

m_a je střední hodnota okamžité amplitudy

ϕ_{NL} je hodnota centrované nelineární složky okamžité fáze

C je počet vzorků

a_t je práh

$\phi_{NL}(i)$ jsou hodnoty fázové charakteristiky bez podílu (účasti) nosné frekvence

$\phi_{UW}(i)$ je rozbalená (unwrapped) fáze

f_c je nosná frekvence

$f_N(i)$ je normovaná okamžitá frekvence

m_f je střední hodnota okamžité frekvence

R_s je symbolová rychlost

$f(i)$ je okamžitá frekvence.

σ_{AF} je směrodatná odchylka absolutní hodnoty normované okamžité frekvence pro časový segment, kde je signál nad šumem

σ_{AA} je směrodatná odchylka absolutní hodnoty normované centrované okamžité amplitudy

σ_{DP} je směrodatná odchylka centrované nelineární složky okamžité fáze pro časový segment, kde je signál nad šumem

σ_{AP} je směrodatná odchylka absolutní hodnoty centrované nelineární složky fáze pro časový segment, kde je signál nad šumem

γ_{max} je maximální hodnota výkonové spektrální hustoty centrované normované okamžité amplitudy

`ASK`	je název funkce vhodný pro spuštění
`T`	je celková doba simulace
`fc`	je nosn kmitočet
`fv`	je vzorkovací kmitočet
`t`	je časový vektor
`SNR`	je signal-to-noise ratio
P_{signal}	je výkon signálu
$P_{šum}$	je výkon šumu
`Ps`	je výkon signálu
`st`	je vytvořený signál
`noise`	je šum
`ask`	je signál se šumem.
Mod_sig	je reálný signál pro zvolené modulace
z	je získaný analytický signál
Ns	je počet vzorků
Acn	je normalizovaný a centralizovaný signál
C	je počet vzorků
PhiNLC	je normalizovaná a centralizovaná fáze
freqn	je normalizovaný a centralizovaný kmitočet
`SNRValueMin`	je minimální hodnota SNR
`SNRpres`	je krok SNR
`SNRValueMax`	je maximální hodnota SNR
`ModSig1`	je modulovaný signál
`t1`	je časový vektor
`network`	je vytvořená síť
`P`	je matice příznaků
`T`	je matice cílů
`Mnozina`	je matice s příznaky
`net`	je naučená neuronová síť
`AnsASK`	je získaná odpověď neuronové sítě
`MaticeASK`	je matice osahujíce získané odpovědi
`indASK`	je index v matice
`UspechASK`	je hodnota úspěšného rozpoznávaní
`ProcUspechASK`	je hodnota úspěšného rozpoznávaní v procentech
`VypocetPriznaku`	je aplikace výpočtu příznaků
`ModVektorASK`	je matice obsahujíce příznaky modulace ASK
`ModVektorFSK`	je matice obsahujíce příznaky modulace FSK
`ModVektorBPSK`	je matice obsahujíce příznaky modulace BPSK
`ModVektorQPSK`	je matice obsahujíce příznaky modulace QPSK

`ModVektor16QAM` je matice obsahujíce příznaky modulace 16QAM

`MaticeProcUspechASK` je matice obsahujíce získaná procenta

`SNR` je matice hodnot SNR

`MaticeProcUspechFSK` je matice obsahujíce získaná procenta úspěšného rozpoznávání modulace FSK

`MaticeProcUspechBPSK` je matice obsahujíce získaná procenta úspěšného rozpoznávání modulace BPSK

`MaticeProcUspechQPSK` je matice obsahujíce získaná procenta úspěšného rozpoznávání modulace QPSK

`MaticeProcUspech16QAM` je matice obsahujíce získaná procenta úspěšného rozpoznávání modulace 16QAM.

SEZNAM PŘÍLOH

PŘÍLOHA 1

Zdrojový kód: Aplikace vypočtu klíčových příznaků

```
function [ Priznaky ] = VypocetPriznaku( Mod_sig,t )

Z=hilbert(Mod_sig);
a=abs(Z);
ma=mean(a);
an=a/ma;
acn=an-1;
Ns=length(acn);

GammaMax=(max((abs(fft(acn))).^2))/length(Mod_sig);
SigmaAA=sqrt(1/Ns*(sum(acn.^2))-(1/Ns*sum(abs(acn))).^2);

O=angle(Z);
Onw=unwrap(O);
Onl=Onw'-2*pi*10000000*t(1:length(Z));

j=1;
for i=1:length(Onl)
    if an(i)>1
        Onl1(j)=Onl(i);
        j=j+1;
    end
end
C=length(Onl1);

Onl2 = [];
k=0;
for i=1:length(Onl1)              %normovani faze od -pi do pi
    if Onl1(i) > pi
       k=round(Onl1(i)/(2*pi));
       Onl2(i)=Onl1(i)- k*2*pi;
    end;
    if Onl1(i) < -pi
       k=round(Onl1(i)/(2*pi));
       Onl2(i)=Onl1(i)- k*2*pi;
    end;
end;

Onlc=Onl2-mean(Onl2);

SigmaAP=sqrt(1/C*(sum(Onlc.^2))-(1/C*sum(abs(Onlc))).^2);
SigmaDP=sqrt(1/C*(sum(Onlc.^2))-(1/C*sum(Onlc)).^2);

freq=diff(Onl);
freq=freq((floor(0.01*length(freq))):(length(freq)-
floor(0.01*length(freq))));

j=1;
for i=1:length(freq)
    if an(i)>1
        freq1(j)=freq(i);
        j=j+1;
    end
```

```
end

freqa=abs(freq1);
mfa=mean(freqa);
freqan=freqa/mfa;

SigmaAF=sqrt(1/length(freqan)*(sum(freqan.^2))-(1/length(freqan)*...
sum(abs(freqan))).^2);

Priznaky = [SigmaAA SigmaAF SigmaAP SigmaDP GammaMax];

end
```

PŘÍLOHA 2

Zdrojový kód: Aplikace vytvoření trénovací množiny

```
%========================== Uklid ==========================%

clearvars;
clc;
close all;
delete Mnozina.xlsx Mnozina;

%====================Dialog s uzivatelem====================%

disp('Dobry den,');
disp('Ted se provede vytvoreni trenovaci mnoziny');

%========================= Aplikace =========================%

Pocet_iteraci = 100;
SNRpres = 1;
SNRValueMin = -10;
SNRValueMax = 50;

Mnozina = [];
MnozinaASK = [];
MnozinaFSK = [];
MnozinaBPSK = [];
MnozinaQPSK = [];
Mnozina16QAM = [];

for i= SNRValueMin:SNRpres:SNRValueMax

    R=500000;
    T=0.0002;               % celkova doba simulace
    fc=10000000;            % kmitoèet nosne
    %fv=100000000;           % vzorkovaci kmitocet

    for j=2:1:10

        fv=j*fc;            % vzorkovaci kmitocet

        for k=1:1:Pocet_iteraci

            clearvars ModSig1 ModSig2 ModSig3 ModSig4 ModSig5;

            disp('Spusteni modulace ASK...');
            [ModSig1,t1] = ASK(T,R,fv,fc,i);
            VektorASK = VypocetPriznaku(ModSig1,t1);
            %VektorASK = VektorASK/max(max(VektorASK)); % normalizace
            ModVektorASK = [VektorASK, 1, 0, 0, 0, 0];

            disp('Spusteni modulace FSK...');
            [ModSig2,t2] = FSK(T,R,fv,fc,i);
            VektorFSK = VypocetPriznaku(ModSig2,t2);
            %VektorFSK = VektorFSK/max(max(VektorFSK)); % normalizace
            ModVektorFSK = [VektorFSK, 0, 1, 0, 0, 0];
```

```
            disp('Spusteni modulace BPSK...');
            [ModSig3,t3] = BPSK(T,R,fv,fc,i);
            VektorBPSK = VypocetPriznaku(ModSig3,t3);
            %VektorBPSK = VektorBPSK/max(max(VektorBPSK)); % normalizace
            ModVektorBPSK = [VektorBPSK, 0, 0, 1, 0, 0];

            disp('Spusteni modulace QPSK...');
            [ModSig4,t4] = QPSK(T,R,fv,fc,i);
            VektorQPSK = VypocetPriznaku(ModSig4,t4);
            %VektorQPSK = VektorQPSK/max(max(VektorQPSK)); % normalizace
            ModVektorQPSK = [VektorQPSK, 0, 0, 0, 1, 0];

            disp('Spusteni modulace 16QAM...');
            [ModSig5,t5] = QAM16(T,R,fv,fc,i);
            Vektor16QAM = VypocetPriznaku(ModSig5,t5);
          %Vektor16QAM = Vektor16QAM/max(max(Vektor16QAM)); % normalizace
            ModVektor16QAM = [Vektor16QAM, 0, 0, 0, 0, 1];

            Mnozina = [Mnozina; ModVektorASK; ModVektorFSK; ...
                ModVektorBPSK; ModVektorQPSK; ModVektor16QAM];
        end
    end
end

%============================ Export ============================%

xlswrite('Mnozina.xlsx', Mnozina);

%============================ Network ============================%

SecondAplikaceVytvoreniNeuronoveSite;

%============================ Uspech ============================%

ThirdAplikaceVypocetProcent;

%============================ Aplikace ============================%

FINISHwithAplikaceDialogProRozpoznavani;
```

PŘÍLOHA 3

Zdrojový kód: Aplikace vytvoření neuronové sítě

```
%========================== Uklid ==========================%

delete network3layers network2layers network1layer
clearvars;
clc;
close all;

%====================Dialog s uzivatelem====================%

disp('Dobry den,');
disp('Ted si zvolite parametry neuronove siti pro rospoznavani.');

FcnPerf = input('Napiste perfFcn [sse]: ', 's');
    if isempty(FcnPerf)
    FcnPerf = 'sse'; %mse
    end
FcnTrain = input('Napiste trainFcn [traingdx]: ', 's');
    if isempty(FcnTrain)
    FcnTrain = 'traingdx';%traingdx
    end
FcnTransferOutput = input('Napiste transferFcn vystupni vrstvy [purelin]:
', 's');
    if isempty(FcnTransferOutput)
    FcnTransferOutput = 'purelin';
    end

networklayers = input('Napiste pocet vrstev (1-3) [2]: ');
if isempty(networklayers)
    pocetneuronu1vrstva = input('Napiste pocet neuronu prvni skryte
vrstvy [12]: ');
        if isempty(pocetneuronu1vrstva)
        pocetneuronu1vrstva = 12;
        end
    pocetneuronu2vrstva = input('Napiste pocet neuronu druhe skryte
vrstvy [12]: ');
        if isempty(pocetneuronu2vrstva)
        pocetneuronu2vrstva = 12;
        end
    FcnTransfer1Layer = input('Napiste transferFcn 1. vrstvy [tansig]: ',
's');
        if isempty(FcnTransfer1Layer)
        FcnTransfer1Layer = 'tansig';
        end
    FcnTransfer2Layer = input('Napiste transferFcn 2. vrstvy [tansig]: ',
's');
        if isempty(FcnTransfer2Layer)
        FcnTransfer2Layer = 'tansig';
        end

   net = DoubleLayeredNetwork( pocetneuronu1vrstva,
pocetneuronu2vrstva,...
       FcnPerf, FcnTrain, FcnTransfer1Layer, FcnTransfer2Layer,
FcnTransferOutput);
```

```
   save('network2layers', 'net');

elseif networklayers == 1
    pocetneuronu = input('Napiste pocet neuronu [12]: ');
       if isempty(pocetneuronu)
        pocetneuronu = 12;
       end
    FcnTransfer = input('Napiste transferFcn [tansig]: ', 's');
        if isempty(FcnTransfer)
        FcnTransfer = 'tansig';
        end

   net = OneLayeredNetwork( pocetneuronu, FcnPerf, FcnTrain,...
       FcnTransfer, FcnTransferOutput);
   save('network1layer', 'net');

elseif networklayers == 2
    pocetneuronu1vrstva = input('Napiste pocet neuronu prvni skryte
vrstvy [12]: ');
        if isempty(pocetneuronu1vrstva)
        pocetneuronu1vrstva = 12;
        end
    pocetneuronu2vrstva = input('Napiste pocet neuronu druhe skryte
vrstvy [12]: ');
        if isempty(pocetneuronu2vrstva)
        pocetneuronu2vrstva = 12;
        end
    FcnTransfer1Layer = input('Napiste transferFcn 1. vrstvy [tansig]: ',
's');
        if isempty(FcnTransfer1Layer)
        FcnTransfer1Layer = 'tansig';
        end
    FcnTransfer2Layer = input('Napiste transferFcn 2. vrstvy [tansig]: ',
's');
        if isempty(FcnTransfer2Layer)
        FcnTransfer2Layer = 'tansig';
        end

   net = DoubleLayeredNetwork( pocetneuronu1vrstva,
pocetneuronu2vrstva,...
       FcnPerf, FcnTrain, FcnTransfer1Layer, FcnTransfer2Layer,
FcnTransferOutput);
   save('network2layers', 'net');

elseif networklayers == 3
    pocetneuronu1vrstva = input('Napiste pocet neuronu prvni skryte
vrstvy [12]: ');
        if isempty(pocetneuronu1vrstva)
        pocetneuronu1vrstva = 12;
        end
    pocetneuronu2vrstva = input('Napiste pocet neuronu druhe skryte
vrstvy [12]: ');
        if isempty(pocetneuronu2vrstva)
        pocetneuronu2vrstva = 12;
        end
    pocetneuronu3vrstva = input('Napiste pocet neuronu treti skryte
vrstvy [12]: ');
        if isempty(pocetneuronu3vrstva)
```

```
        pocetneuronu3vrstva = 12;
        end
    FcnTransfer1Layer = input('Napiste transferFcn 1. vrstvy [tansig]: ',
's');
        if isempty(FcnTransfer1Layer)
        FcnTransfer1Layer = 'tansig';
        end
    FcnTransfer2Layer = input('Napiste transferFcn 2. vrstvy [tansig]: ',
's');
        if isempty(FcnTransfer2Layer)
        FcnTransfer2Layer = 'tansig';
        end
    FcnTransfer3Layer = input('Napiste transferFcn 3. vrstvy [tansig]: ',
's');
        if isempty(FcnTransfer3Layer)
        FcnTransfer3Layer = 'tansig';
        end
   net = TripleLayeredNetwork( pocetneuronu1vrstva,
pocetneuronu2vrstva,...
       pocetneuronu3vrstva, FcnPerf, FcnTrain, FcnTransfer1Layer,...
       FcnTransfer2Layer, FcnTransfer3Layer, FcnTransferOutput);
   save('network3layers', 'net');

else
    error('Chyba...');
end

ThirdAplikaceVypocetProcent;
```

PŘÍLOHA 4

Zdrojový kód: Aplikace výpočtu chybovosti neuronové sítě

```
%========================== Uklid ==========================%

clearvars -except net;
clc;
close all;

%====================Dialog s uzivatelem====================%

disp('Dobry den,');
disp('Tento program provede vypocet uspesneho rozpoznavani a nakresli
graphy');
PocetPokusu = input('Kolik pokusu chcete udelat pro kazdy typ modulace?:
');

%PocetPokusu = 100;
SNRpres = 1;
SNRValueMin = -10;
SNRValueMax = 50;

MaticeASK = [];
MaticeFSK = [];
MaticeBPSK = [];
MaticeQPSK = [];
Matice16QAM = [];
MaticeProcUspechASK = [];
MaticeProcUspechFSK = [];
MaticeProcUspechBPSK = [];
MaticeProcUspechQPSK = [];
MaticeProcUspech16QAM = [];
SNR = [];

for i= SNRValueMin:SNRpres:SNRValueMax

    fprintf('SNR value je %d. \n', i );

    T = 0.0002;
    R=500000;
    fc = 10000000;            % kmitocet nosne
    fv=100000000;           % vzorkovaci kmitocet

    SNR = [SNR; i];
    for j=2:1:10

        fv=j*fc;            % vzorkovaci kmitocet

       for j=1:1:PocetPokusu

            %disp('rospoznavani ASK');
            [Mod_sig1,t1] = ASK(T,R,fv,fc,i);
            forNetASK = VypocetPriznaku(Mod_sig1,t1);
            %forNetASK = forNetASK/max(max(forNetASK)); % normalizace
            AnsASK = round(sim(net,forNetASK'));
```

```
            MaticeASK = [MaticeASK; AnsASK'];
            %disp('zapis vysledku');

            %disp('rospoznavani FSK');
            [Mod_sig2,t2] = FSK(T,R,fv,fc,i);
            forNetFSK = VypocetPriznaku(Mod_sig2,t2);
            %forNetFSK = forNetFSK/max(max(forNetFSK)); % normalizace
            AnsFSK = round(sim(net,forNetFSK'));
            MaticeFSK = [MaticeFSK; AnsFSK'];
            %disp('zapis vysledku');

            %disp('rospoznavani BPSK');
            [Mod_sig3,t3] = BPSK(T,R,fv,fc,i);
            forNetBPSK = VypocetPriznaku(Mod_sig3,t3);
            %forNetBPSK = forNetBPSK/max(max(forNetBPSK)); % normalizace
            AnsBPSK = round(sim(net, forNetBPSK'));
            MaticeBPSK = [MaticeBPSK; AnsBPSK'];
            %disp('zapis vysledku');

            %disp('rospoznavani QPSK');
            [Mod_sig4,t4] = QPSK(T,R,fv,fc,i);
            forNetQPSK = VypocetPriznaku(Mod_sig4,t4);
            %forNetQPSK = forNetQPSK/max(max(forNetQPSK)); % normalizace
            AnsQPSK = round(sim(net, forNetQPSK'));
            MaticeQPSK = [MaticeQPSK; AnsQPSK'];
            %disp('zapis vysledku');

            %disp('rospoznavani 16QAM');
            [Mod_sig5,t5] = QAM16(T,R,fv,fc,i);
            forNet16QAM = VypocetPriznaku(Mod_sig5,t5);
          %forNet16QAM = forNet16QAM/max(max(forNet16QAM)); % normalizace
            Ans16QAM = round(sim(net, forNet16QAM'));
            Matice16QAM = [Matice16QAM; Ans16QAM'];
            %disp('zapis vysledku');

    end
 end

UspechASK = 0;
for o=1:1:PocetPokusu
 indASK = (i-SNRValueMin)*PocetPokusu/SNRpres+o;
     if MaticeASK(indASK,1) == 1 && ...
        MaticeASK(indASK,2) == 0 && ...
        MaticeASK(indASK,3) == 0 && ...
        MaticeASK(indASK,4) == 0 && ...
        MaticeASK(indASK,5) == 0

        UspechASK =UspechASK+1;

     end
end

UspechFSK = 0;
for k=1:1:PocetPokusu
 indFSK = (i-SNRValueMin)*PocetPokusu/SNRpres+k;
     if MaticeFSK(indFSK,1) == 0 && ...
        MaticeFSK(indFSK,2) == 1 && ...
```

```
        MaticeFSK(indFSK,3) == 0 && ...
        MaticeFSK(indFSK,4) == 0 && ...
        MaticeFSK(indFSK,5) == 0

        UspechFSK =UspechFSK+1;

    end
end

UspechBPSK = 0;
for l=1:1:PocetPokusu
 indBPSK = (i-SNRValueMin)*PocetPokusu/SNRpres+l;
    if MaticeBPSK(indBPSK,1) == 0 && ...
        MaticeBPSK(indBPSK,2) == 0 && ...
        MaticeBPSK(indBPSK,3) == 1 && ...
        MaticeBPSK(indBPSK,4) == 0 && ...
        MaticeBPSK(indBPSK,5) == 0

        UspechBPSK =UspechBPSK+1;

    end
end

UspechQPSK = 0;
for m=1:1:PocetPokusu
 indQPSK = (i-SNRValueMin)*PocetPokusu/SNRpres+m;
    if MaticeQPSK(indQPSK,1) == 0 && ...
        MaticeQPSK(indQPSK,2) == 0 && ...
        MaticeQPSK(indQPSK,3) == 0 && ...
        MaticeQPSK(indQPSK,4) == 1 && ...
        MaticeQPSK(indQPSK,5) == 0

        UspechQPSK =UspechQPSK+1;

    end
end

Uspech16QAM = 0;
for n=1:1:PocetPokusu
 ind16QAM = (i-SNRValueMin)*PocetPokusu/SNRpres+n;
    if Matice16QAM(ind16QAM,1) == 0 && ...
        Matice16QAM(ind16QAM,2) == 0 && ...
        Matice16QAM(ind16QAM,3) == 0 && ...
        Matice16QAM(ind16QAM,4) == 0 && ...
        Matice16QAM(ind16QAM,5) == 1

        Uspech16QAM =Uspech16QAM+1;

    end
end

 ProcUspechASK = UspechASK*100/PocetPokusu;
 ProcUspechFSK = UspechFSK*100/PocetPokusu;
 ProcUspechBPSK = UspechBPSK*100/PocetPokusu;
 ProcUspechQPSK = UspechQPSK*100/PocetPokusu;
 ProcUspech16QAM = Uspech16QAM*100/PocetPokusu;
```

```
    MaticeProcUspechASK = [MaticeProcUspechASK; ProcUspechASK];
    MaticeProcUspechFSK = [MaticeProcUspechFSK; ProcUspechFSK];
    MaticeProcUspechBPSK = [MaticeProcUspechBPSK; ProcUspechBPSK];
    MaticeProcUspechQPSK = [MaticeProcUspechQPSK; ProcUspechQPSK];
    MaticeProcUspech16QAM = [MaticeProcUspech16QAM; ProcUspech16QAM];

end

clearvars -except SNRValueMin SNRValueMax MaticeProcUspechASK ...
MaticeProcUspechFSK MaticeProcUspechBPSK MaticeProcUspechQPSK ...
MaticeProcUspech16QAM SNR net;

save('MaticeProcUspechASK.mat','MaticeProcUspechASK');
save('MaticeProcUspechFSK.mat','MaticeProcUspechFSK');
save('MaticeProcUspechBPSK.mat','MaticeProcUspechBPSK');
save('MaticeProcUspechQPSK.mat','MaticeProcUspechQPSK');
save('MaticeProcUspech16QAM.mat','MaticeProcUspech16QAM');

figure('Name','ASK','NumberTitle','off')
plot(SNR,MaticeProcUspechASK,'r','linewidth',2);
axis([ SNRValueMin SNRValueMax 0 100]);
grid on;
xlabel('SNR');
ylabel('Procent uspechu');
title('Zavislost rospoznavani ASK od SNR');

figure('Name','FSK','NumberTitle','off')
plot(SNR,MaticeProcUspechFSK,'r','linewidth',2);
axis([ SNRValueMin SNRValueMax 0 100]);
grid on;
xlabel('SNR');
ylabel('Procent uspechu');
title('Zavislost rospoznavani FSK od SNR');

figure('Name','BPSK','NumberTitle','off')
plot(SNR,MaticeProcUspechBPSK,'r','linewidth',2);
axis([ SNRValueMin SNRValueMax 0 100]);
grid on;
xlabel('SNR');
ylabel('Procent uspechu');
title('Zavislost rospoznavani BPSK od SNR');

figure('Name','QPSK','NumberTitle','off')
plot(SNR,MaticeProcUspechQPSK,'r','linewidth',2);
axis([ SNRValueMin SNRValueMax 0 100]);
grid on;
xlabel('SNR');
ylabel('Procent uspechu');
title('Zavislost rospoznavani QPSK od SNR');

figure('Name','16QAM','NumberTitle','off')
plot(SNR,MaticeProcUspech16QAM,'r','linewidth',2);
axis([ SNRValueMin SNRValueMax 0 100]);
grid on;
xlabel('SNR');
ylabel('Procent uspechu');
```

```
title('Zavislost rospoznavani 16QAM od SNR');

figure('Name','All-in-One','NumberTitle','off')
plot(SNR, MaticeProcUspechASK,SNR,MaticeProcUspechFSK,SNR,...
MaticeProcUspechBPSK,SNR,MaticeProcUspechQPSK,...
SNR,MaticeProcUspech16QAM);
legend('ASK','FSK','BPSK','QPSK','16QAM');
axis([ SNRValueMin SNRValueMax 0 100]);
grid on;
xlabel('SNR');
ylabel('Procent uspechu');
title('All in One');
```

PŘÍLOHA 5

Zdrojový kód: Aplikace pro využití neuronové sítě

```
%========================== Uklid ==========================%

clearvars -except net
clc;
close all;

%====================Dialog s uzivatelem====================%

disp('Dobry den,');
disp('Ted si zvolite parametry signalu pro rospoznavani.');

T = 0.0002;
R= 500000;
fc = 10000000;

fv = input('Napiste vzorkovaci kmitocet (2-10) [10]: ');
if isempty(fv)
    fv = 100000000;
elseif (2<=fv)&&(fv<=10)
   fv = fv*fc;
else
    error('Chyba...');
end

SNRvalue = input('Napiste SNR [25]: ');
if isempty(SNRvalue)
    SNRvalue = 25;
end

disp('Napiste cislo modulace, pro ASK:1, pro FSK:2, ');
disp('pro BPSK:3, pro QPSK:4, pro 16QAM:5');

Cislo_Modulace = input('Co jste vybral? [3]: ');
    if isempty(Cislo_Modulace)
        Cislo_Modulace = 3;
    end

    if (Cislo_Modulace == 1)
        disp('Spusteni modulace ASK...');
        disp('Pracuji...');
        %==================================
        [Mod_sig1,t1] = ASK(T,R,fv,fc,SNRvalue);
        forNetASK = VypocetPriznaku(Mod_sig1,t1);
        answer = round(sim(net,forNetASK'));

    elseif (Cislo_Modulace == 2)
        disp('Spusteni modulace FSK...');
        disp('Pracuji...');
        %==================================
        [Mod_sig2,t2] = FSK(T,R,fv,fc,SNRvalue);
        forNetFSK = VypocetPriznaku(Mod_sig2,t2);
        answer = round(sim(net,forNetFSK'));
```

```
    elseif (Cislo_Modulace == 3)
        disp('Spusteni modulace BPSK...');
        disp('Pracuji...');
        %==================================
        [Mod_sig3,t3] = BPSK(T,R,fv,fc,SNRvalue);
        forNetBPSK = VypocetPriznaku(Mod_sig3,t3);
        answer = round(sim(net, forNetBPSK'));

    elseif (Cislo_Modulace == 4);
        disp('Spusteni modulace QPSK...');
        disp('Pracuji...');
        %==================================
        [Mod_sig4,t4] = QPSK(T,R,fv,fc,SNRvalue);
        forNetQPSK = VypocetPriznaku(Mod_sig4,t4);
        answer = round(sim(net, forNetQPSK'));

    elseif (Cislo_Modulace == 5);
        disp('Spusteni modulace 16QAM...');
        disp('Pracuji...');
        %==================================
        [Mod_sig5,t5] = QAM16(T,R,fv,fc,SNRvalue);
        forNetSixteenQAM = VypocetPriznaku(Mod_sig5,t5);
        answer = round(sim(net, forNetSixteenQAM'));
    end;

if (answer == [1,0,0,0,0]')
    disp('Neuronova sit to rospoznava jako ASK');
elseif (answer == [0,1,0,0,0]')
    disp('Neuronova sit to rospoznava jako FSK');
elseif (answer == [0,0,1,0,0]')
    disp('Neuronova sit to rospoznava jako BPSK');
elseif (answer == [0,0,0,1,0]')
    disp('Neuronova sit to rospoznava jako QPSK');
elseif (answer == [0,0,0,0,1]')
    disp('Neuronova sit to rospoznava jako 16QAM');
else
    disp('Nepodarilo zjistit typ modulace :-(');
end

question = input('Chcete opakovat? (y/n) [y]:', 's');
if isempty(question)
    FINISHwithAplikaceDialogProRozpoznavani;
elseif question == 'y'
    FINISHwithAplikaceDialogProRozpoznavani;
elseif question == 'Y'
    FINISHwithAplikaceDialogProRozpoznavani;
elseif question == 'n'
    disp('Na shledanou');
elseif question == 'N'
    disp('Na shledanou');
else
    disp('Chyba vstupu...');
    disp('Na shledanou');
end
```

PŘÍLOHA 5
CD

Na přiloženém CD je uložena elektronická podoba tohoto dokumentu a zdrojové soubory vytvořených aplikací.