VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Inteligentní metody na bázi strojového učení pro klasifikace a identifikaci objektů zájmu: laboratorní úloha

Intelligent Methods based on Machine Learning for Classification and Identification of Objects of Interest: A Laboratory Task

Bc. Terezie Kubošková

Diplomová práce

Vedoucí práce: Ing. Jan Kubíček, Ph.D.

Ostrava, 2022

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Terezie Kubošková**

Studijní program: N0988A060001 Biomedicínské inženýrství

Téma: Inteligentní metody na bázi strojového učení pro klasifikaci a identifikaci objektů zájmu: laboratorní úloha
Intelligent Methods based on Machine Learning for Classification and Identification of Objects of Interest: A Laboratory Task

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Nastudování základních principů klasifikace 1D a 2D dat na bázi umělé inteligence.
2. Nastudování základních principů segmentace a identifikace událostí na bázi metod umělé inteligence.
3. Rešerše aplikací klasifikace dat v rámci zpracování biomedicínských signálů a obrazů.
4. Tvorba datové báze 1D a 2D signálů pro potřeby strojového učení.
5. Design a realizace vybraných metod strojového učení pro úkoly klasifikace 1D a 2D dat.
6. Implementace a objektivní evaluace metod strojového učení pro řešené aplikační případy.
7. Vytvoření laboratorní úlohy pro edukaci metod strojového učení.
8. Zhodnocení výsledků práce.

Seznam doporučené odborné literatury:

[1] HUANG, Sunan, Kok Kiong TAN a Kok Zuea TANG. Neural network control: theory and applications. Baldock: Research Studies Press, c2004. Control and signal/image processing series, 3. ISBN 0-86380-285-0.

[2] MITCHELL, Tom Michael. Machine learning. Boston: WCB/McGraw-Hill, c1997. McGraw-Hill series in computer science. ISBN 0-07-042807-7.

[3] DEISENROTH, Marc Peter, A. Aldo FAISAL a Cheng Soon ONG. Mathematics for machine learning. Cambridge: Cambridge University Press, 2020. ISBN 978-1-108-47004-9.

[4] ZAKI, Mohammed J. a Wagner MEIRA. Data mining and machine learning: fundamental concepts and algorithms. Second edition. Cambridge: Cambridge University Press, 2020. ISBN 978-1-108-47398-9.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Kubíček, Ph.D.**

Datum zadání: 01.09.2021

Datum odevzdání: 30.04.2022

prof. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Jan Platoš, Ph.D.
děkan fakulty

Abstrakt

Podstatou diplomové práce je návrh exemplárních laboratorních úloh, jejichž cílem je seznámit studující s klasifikací dat za pomocí neuronových sítí. Jednotlivé úlohy se zabývají klasifikací dat. První úkol se věnuje základní metodě klasifikace pomocí perceptronu. Další úkoly se věnují jednak metodě optimalizace neuronové sítě pomocí genetických algoritmů a jednak využití konvolučních neuronových sítí pro klasifikaci jednorozměrných akustických signálů a dvourozměrných obrazů. V jednotlivých úlohách jsou data, pro natrénovaní neuronových sítí, buď dynamicky vytvořena nebo načtena při startu. Následně úlohy demonstrují vytvoření jednotlivých sítí včetně způsobu jejich natrénovaní dle různých vstupních parametrů. V posledních krocích laboratorních úloh se algoritmy validují a výsledky analyzují. Všechny algoritmy dílčích částí byly naprogramovány v prostředí MATLAB s využitím technologie Live Script.

Klíčová slova

Klasifikace, preceptron, audio signály, obrazy, optimalizace, genetický algoritmus, konvoluční neuronová síť, GoogLeNet, MATLAB

Abstract

The aim of the master thesis is to design exemplary laboratory tasks to introduce students to data classification with neural networks. The individual tasks deal with data classification. The first assignment is about the basic method of classification using perceptron. Other tasks focus on the neural network optimization method using genetic algorithms and the next assignment deals with the use of convolutional neural networks for classification of one-dimensional acoustic signals and two-dimensional images. For each task, data is created or loaded to train the neural networks, then create the network and train it under different settings. In the last step of the lab tasks, the algorithms are validated, and the results are analysed. All the algorithms of the subsections were programmed in MATLAB using Live Script technology.

Key Words

Classification, preceptron, audio signals, images, optimization, genetic algorithm, convolutional neural network, GoogLeNet, MATLAB

Poděkování

Ráda bych touto cestou poděkovala Ing. Janu Kubíčkovi, Ph.D. za jeho za jeho odborné konzultace, připomínky a vstřícnost při konzultacích této diplomové práce. V neposlední řadě tímto děkuji své rodině za podporu při vypracovávání této práce.

Obsah

Seznam použitých zkratek a symbolů

1D	jednodimenzionální signál
2D	dvoudimenzionální signál
CFR	podmíněné náhodné pole (conditional random field)
CNN	konvoluční neuronová síť (convolutional neural networks)
DCNN	hluboká konvoluční neuronová síť (deep convolutional neural networks)
EKG	elektrokardiografický signál
FCN	plně propojená síť (fully connecterd network)
JPU	joint pyramid upsampling
KNN	algoritmus k-nejbližších sousedů (K-nearest neighbors)
LoG	Laplacian of Gaussian
MLP	vícevrstvý perceptron (multilayerd perceptron)
NN	algoritmus nejbližší sousedi (nearest neighbors)
NN	neuronová síť (neural networks)
ReLu	vrstva rektifikované lineární jednotky (Rectified Linear Unit)
RGB	barevný model červená-zelená-modrá (red-green-blue)

Seznam obrázků

Seznam tabulek

Úvod

Strojové učení vzniklo na základě rozpoznávání vzorů a teorie, že se počítače mohou učit, tudíž mohou vykonat činnosti bez toho, aby k nim byly přímo naprogramovány. Přestože se v moderním světě používá ve velmi velké míře, tak díky novým výpočetním technologiím se dnešní strojové učení nepodobá strojovému učení v minulosti. Do strojového učení patří nespočet metod, kdy každá slouží jiným, byť podobným účelům nebo jsou jednotlivé metody stále vylepšovány a vznikají nové metody se stejným cílem, ale lepším provedením a výsledkem. Mnoho metod strojového učení existuje již dlouho, avšak schopnost automaticky aplikovat složité matematické výpočty na velká data, stále znovu a rychleji, se vyvinula teprve nedávno. Strojového učení se využívá pro klasifikaci jednoduchých věcí jako je např. třídění hodnot do skupin, přes rozpoznání spamu v elektronické poště, až po odhad velmi komplikovaných věci, které ani člověk nemusí znát, jako je odhad cen na burze.

Ve zdravotnictví má strojové učení rychle rostoucí oblibu, a to především díky nástupu nositelných zařízení a senzorů životních funkcí. Strojové učení využívá získaná data k vyhodnocování zdravotního stavu pacienta v reálném čase. Tato technologie může také pomoci lékařským odborníkům analyzovat údaje a identifikovat trendy, které mohou vést ke zlepšení diagnózy a léčby.

Tato diplomová práce je zaměřena na studenty biomedicínských oborů a má sloužit jako pomůcka pro výuku základní práce se strojovým učením. Ze široké oblasti strojového učení je zacíleno na práci s neuronovými sítěmi a jejich schopnost klasifikovat data.

Práce je rozdělena na teoretickou a praktickou část. V části teoretické se práce zabývá hlavními typy klasifikací a segmentací dat. Kapitola o klasifikaci obsahuje základní klasifikační metody se zaměřením na teorii o neuronových sítích. V segmentaci dat jsou popsány nejdůležitější segmentační metody a sémantická segmentace. Teoretická část je zakončena rešerší, která se zabývá použitím metod klasifikace a segmentace dat v medicínské praxi.

Praktická část popisuje zpracování laboratorních úloh v prostředí MATLAB. Jednotlivé práce se zaměřují na témata perceptronu, optimalizace neuronových sítí a klasifikaci jednodimenzionálních a dvoudimenzionálních dat pomocí konvolučních neuronových sítí. Samostatná kapitola se věnuje vytváření databází pro potřeby klasifikace v jednotlivých laboratorních úlohách. A dále se zabývá návrhem algoritmů pro natrénovaní sítí a postupem validací výsledných sítí. Kromě úlohy s perceptronem, se všechny úlohy zabývají vlivem nastavení vstupních parametrů sítě pro natrénovaní na její výslednou přesnost na klasifikaci dat.

Výsledné validační hodnoty (fitness funkce a přesnost klasifikace třídy) jsou objektivně analyzovány. Výsledky přesnosti klasifikace třídy jsou zjištěny z matic záměn, a zároveň je popsáno, jak matice záměn interpretovat. Výsledky všech úloh jsou zhodnoceny z hlediska nastavení kritérií neuronových sítí a genetického algoritmu a je zjištěno nejvhodnější nastavení sítě pro natrénovaní, při němž se dosáhne nejlepší přesnosti nebo časové efektivnosti při velmi dobré přesnosti.

1 Klasifikace dat na bázi strojového učení

Klasifikace dat je definována jako proces rozpoznávání, chápání a seskupování objektů do předem stanovených kategorií, tzv. „tříd“. S pomocí předem kategorizovaných trénovacích souborů dat lze vytvořit strojové učení pro klasifikaci budoucích dat do příslušných a relevantních kategorií. Za tímto účelem se používá široká škála algoritmů, které budou popsány dále. Jiné klasifikační algoritmy používané ve strojovém učení využívají vstupní trénovací data za účelem předpovědi pravděpodobnosti, že následující data budou spadat do jedné z předem stanovených kategorií. Stručně řečeno, klasifikace je formou „rozpoznávání vzorů“. Za úlohy klasifikace lze považovat [1; 2]:

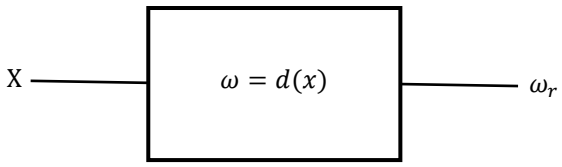
- Rozpoznávací úlohy – rozhodnutí o charakteru objektu a posouzení kvality stavu analyzovaného objektu.
- Predikční úlohy – rozhodnutí o budoucnosti objektu.

Klasifikace dat s úlohou rozpoznat se například používá pro identifikaci osob z kamerového záznamu nebo stav pacienta ležícím na posteli. U úlohy predikční je možno jako příklad uvést jaký bude průběh nemoci pacienta.

Klasifikace dat se realizuje prostřednictvím klasifikátoru (Obr. 1), což je algoritmus popisující analyzovaný objekt, jenž na vstupu obdrží data popisující analyzovaný objekt a jehož výstupem je identifikátor klasifikační třídy do niž klasifikátor zařadil vstupní data. Platí dle vzorce (1) [3]:

$$\omega_r = d(X) \tag{1}$$

Kde $d(X)$ je rozhodovací pravidlo klasifikátoru a je funkcí argumentu *X*, který reprezentuje vstupní data a ω_r (kde $r = 1, \dots, \mathrm{R}$) je identifikátor klasifikační třídy.



Obr. 1 Schéma klasifikátoru [3]

Klasifikátory lze rozčlenit do několika kategorií, jak podle výstupních dat, způsobu učení či metody klasifikace. Jednotlivé kategorie klasifikátorů jsou popsány v následujících podkapitolách.

1.1 Identifikátor klasifikační třídy

Podle jednoznačnosti identifikátoru klasifikační třídy lze rozdělit klasifikátory na deterministické a pravděpodobnostní. Deterministické klasifikátory mají jasně definované, do jaké třídy naleží a jeden objekt nemůže patřit do více tříd najednou [1].

V případě pravděpodobnostních klasifikátorů přiřazují se objekty do tříd buďto dle pravděpodobnosti anebo pomocí tzv. fuzzy klasifikátoru, který klasifikuje na základě síly příslušnosti.

Není pak tedy zcela jednoznačné, do jaké třídy přísluší a díky tomu data v tomto typu klasifikátoru mohou náležet do více tříd současně [1].

1.2 Způsob učení

Klasifikátory lze rozdělit dle způsobu učení na učení s učitelem (supervised) anebo učení bez učitele (unsupervised). Základním rozdílem mezi těmito dvěma typy spočívá v tom, že u učení s učitelem jsou apriorní znalosti o tom, jaké by měly být výstupní hodnoty. Cílem učení s učitelem je naučit se funkci, která při dané trénovací množině co nejlépe aproximuje vztah mezi získaným výstupem a daným výstupem. Na druhou stranu učení bez učitele nemá popsané výstupy, takže jeho cílem je odvodit přirozenou strukturu přítomnou v datech [4; 5].

1.2.1 Učení bez učitele

Učení bez učitele nevyžaduje školení modelu. Funguje na principu nalezení skrytých vzorů a příznaků přímo z dat. Lze jej přirovnat k učení, které probíhá v lidském mozku při učení se novým věcem [4; 6].

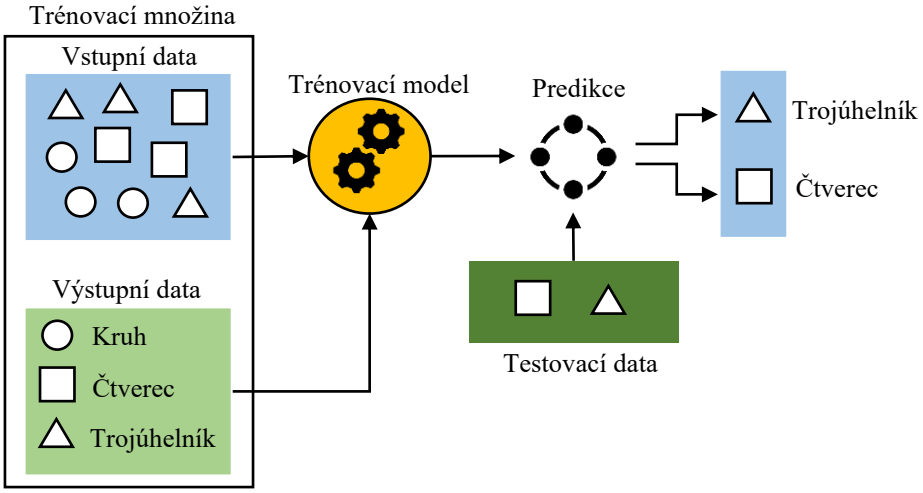
Učení bez učitele nelze přímo použít na klasifikační problém, protože na rozdíl od učení s učitelem jsou sice k dispozici vstupní data, ale bez odpovídajících výstupních dat. Cílem učení bez učitele je najít základní strukturu v souboru dat, seskupit tato data podle podobnosti a reprezentovat tento soubor dat v komprimovaném formátu [4; 6].

Výhodou učení bez učitele je, že je použitelné pro složitější úlohy oproti učení s učitelem, protože nepotřebuje trénovací množinu s označenými výstupy, která se mohou těžce získávat. Avšak výsledek algoritmu učení bez učitele může být méně přesný, data nejsou označeny a dopředu se neví, jak data algoritmus roztřídí [4; 6].

1.2.2 Učení s učitelem

Učení s učitelem disponuje trénovací množinou, pro kterou jsou známé rozdělení jednotlivých objektů do individuálních klasifikačních tříd. Vstupní data jsou označena odpovídajícím výstupem. Lze to přirovnat k učení, které probíhá za přítomnosti učitele. Dále lze ještě rozdělit na učení s dokonalým učitelem, u kterého předpokládáme správnost přiřazení objektů do klasifikačních tříd, a učení s nedokonalým učitelem, při kterém se se můžou v klasifikačních třídách nacházet nesprávně přirazené objekty [1; 7].

Jako příklad učení s učitelem lze uvést diagram (Obr. 2), kde jsou data různých tvarů (kruh, čtverec a trojúhelník) včetně jejich označení. V prvním kroku naučíme model rozpoznávat jednotlivé tvary a to tak, že např. jestli bude mít objekt čtyři strany jedná se o čtverec, jestli bude mít tři strany jedná se o trojúhelník atd. Po natrénovaní modelu jej lze otestovat tím způsobem, že se zadá úloha identifikovat objekty na datech pro testování. Pokud by model našel nové tvary, kromě již naučených, dokáže je klasifikovat na základě počtu stran, a předpovědět tak výsledek [8].



Obr. 2 Diagram ukázky fungování učení s učitelem [8]

Mezi výhody učení s učitelem patří, že lze mít přesnou představu o klasifikačních třídách, kam se data přiřazují a model dokáže identifikovat výsledek a optimalizovat průběh na základě předchozích zkušeností [4; 8].

Oproti tomu nevýhodou učení s učitelem je velká výpočetní a časová náročnost. Modely nejsou vhodné pro vyhodnocování příliš komplexních úkolů. Také metoda nedokáže predikovat výsledek, jestliže se testovací data liší vůči trénujícím datům [4; 8].

1.2.3 Augmentace dat

Technika augmentace dat se používá pro datasety s malým množstvím signálů či snímku pro trénování anebo pro rozšíření a zlepšení rozpoznávacích schopností algoritmů učení s učitelem. Podstatou augmentace je generování nových testovacích dat ze stávajících, tím že je různými způsoby modifikujeme.

Na 1D vstupní data se aplikují techniky augmentace dat jako např. přidání náhodného množství Gaussova šumu, kombinace sinusového signálu s náhodnou počáteční fází a amplitudou, náhodný posun základní linie nebo převrácení signálu [9].

A na 2D vstupní data tedy práce se snímky lze použít techniku natočení obrazu, horizontální či vertikální převrácení, změnu měřítka, náhodné výřezy ze snímku, změnu jasu anebo, pokud je na snímku jednolité pozadí, objekt se může přesouvat po osách x a y. Takto se modely během tréninku sotva setkají se dvěma identickými vstupy. Například pro modely hlubokého učení je to velmi užitečné pro zlepšení jejich výkonu a robustnosti [9].

1.3 Metody klasifikace

Klasifikační metody se dělí do několika kategorií podle principu klasifikace: klasifikace založená na diskriminačních funkcích, klasifikace založená na minimální vzdálenosti, klasifikace pomocí hranic anebo za použití neuronové sítě [1; 2].

První metoda klasifikace využívá výpočet tzv. diskriminačních funkcí, které indikují stupeň příslušnosti objektu nebo subjektu ke stanovené kategorizační třídě. Objektu je přiřazena klasifikační

třída, pro kterou má diskriminační funkce nejvyšší hodnotu. Bayesův klasifikátor je příkladem tohoto typu klasifikační metody [1; 2].

Pro hraniční klasifikaci je nezbytné stanovit hranice mezi jednotlivými klasifikačními třídami. Dvěma příklady jsou metoda podpůrných vektorů a Fisherova lineární diskriminace [1; 2].

Klasifikace minimální vzdálenosti je založena na výpočtu vzdáleností objektů na základě požadavků na klasifikační třídu. Příkladem je metoda nejbližšího souseda nebo metoda průměrné vazby [1; 2].

K určení vzdálenosti se využívá několik metrik a použití zvolené metriky záleží na typu úlohy a požadavcích na výsledek. Mezi základní metriky patří [1; 2]:

- **Euklidova metrika** je nejpoužívanější metrika s názornou geometrickou interpretací vypočtená dle vzorce (2) [1; 2].

$$\mathrm{D_E}(\mathrm{x_1}, \mathrm{x_2}) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2} \tag{2}$$

- **Hammingova metrika** má oproti Euklidově metrice menší výpočetní náročnost, metrika je definována vztahem (3) [1; 2].

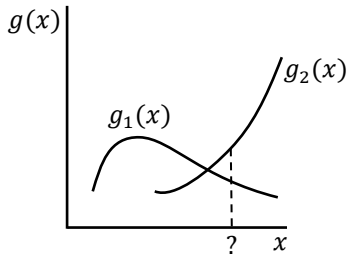
$$\mathrm{D_H}(\mathrm{x_1}, \mathrm{x_2}) = \sum_{i=1}^{n} |x_{1i} - x_{2i}| \tag{3}$$

- **Minkovského metrika** je zobecněním Euklidovy i Hammingovy metriky, jak vyplývá ze vzorce (4). Konstanta m ovlivňuje důležitost rozdílů mezi proměnnými, protože čím větší je m tím větší mají váhu velké rozdíly mezi proměnnými. (Při $m = 2$ dostaneme vztah pro Euklidovu metriku a při $m = 1$ Hammingovu) [1; 2].

$$\mathrm{D_M}(\mathrm{x_1}, \mathrm{x_2}) = \left(\sum_{i=1}^{n} |x_{1i} - x_{2i}|^m \right)^{\frac{1}{m}} \tag{4}$$

1.3.1 Naivní Bayesův klasifikátor

Základním principem u klasifikace dle diskriminačních funkcí je to, že míru příslušnosti ke každé klasifikační třídě objektu x do tříd $\omega_1, \omega_2, \dots, \omega_K$ vyjadřuje diskriminační funkce $g_1(x)$, $g_2(x), \dots, g_K(x)$. Objekt x se přiřadí do třídy, pro niž je $\omega_K(x)$ maximální. Ukázka klasifikátoru na Obr. 3 zobrazuje, že neznámý objekt (otazník) se zařadí do třídy ω_2, neboť $g_2(x) > g_1(x)$ [1; 10].



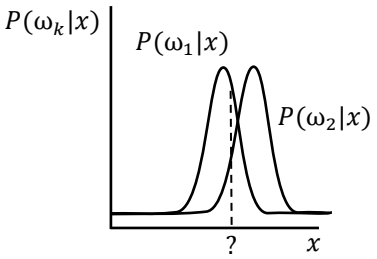
Obr. 3 Ukázka klasifikace diskriminačních funkcí [1]

Příkladem diskriminační funkce je naivní Bayesův klasifikátor, který přiřazuje objekty do klasifikačních tříd podle podmíněné pravděpodobnosti zatřízení objektu do daných tříd, kde součet těchto pravděpodobností se pro každý objekt rovná jedné. Při výpočtu se vychází z Bayesova vzorce (5) [1; 10].

$$P(\omega_k|x) = \frac{p(x|\omega_k)P(\omega_k)}{p(x)} \tag{5}$$

Kde $P(\omega_k|x)$ je podmíněná pravděpodobnost zatřízení objetu x do třídy ω_k ; $p(x|\omega_k)$ je podmíněná hustota pravděpodobnosti výskytu objektu x ve třídě ω_k . $P(\omega_k)$ je apriorní pravděpodobnost třídy ω_k a $p(x)$ je celková hustota pravděpodobnosti rozložení objektu x v celém prostoru. Naivní Bayesův klasifikátor předpokládá, podmíněnou nezávislost proměnných. V důsledku toho, pokud již byla značka zavedena, předpokládá, že pravděpodobnost jednotlivých proměnných spolu nesouvisí [1; 10].

Na Obr. 4 je znázorněná klasifikace naivním Bayesův klasifikátorem na základě jedné proměnné. Testovací objekt x se zatřídí do třídy ω_1, protože aposteriorní pravděpodobnost zařazení objektu x do třídy ω_1 je větší než aposteriorní pravděpodobnost zařazení objektu x do třídy ω_2 [1].



Obr. 4 Klasifikace naivním Bayesův klasifikátorem jedné proměnné [1]

Naivní Bayesův klasifikátor se často používá jako spamový filtr, tedy rozlišuje příchozí emaily na spam a ham (chtěný email) výběrem slov. Jedná o naivní Bayesův klasifikátor, kde se předpokládá, že mezi sousedními slovy není žádná závislost a pořadí slov ve větě nemá žádný význam [1].

1.3.2 Metoda nejbližšího souseda

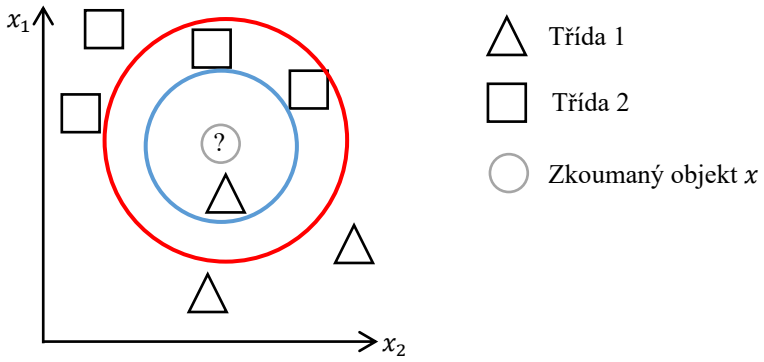
U metody nejbližšího souseda (nearest neighbour, NN) se roztřízení do klasifikačních tříd řeší způsobem přiřazení objektu do třídy dle vzdálenosti objektů. Metoda hledá nejmenší vzdálenost objektu x od již klasifikovaných objektů x_j z celé množiny všech objektů C dle vztahu (6) [1].

$$D_{NN}(x, C) = \min_{x \in C} D(x, x_j) \tag{6}$$

Uvedená metoda má několik nevýhod, např. pokud se zkoumaný objekt nachází v řidší oblasti může jej být obtížné správně klasifikovat anebo pokud se více objektů různých tříd nachází ve stejné vzdálenosti dochází pak k vzájemnému překryvu. Z těchto důvodů vznikla pokročilejší metoda K nejbližších sousedů (K-nearest neighbour, KNN), která zkoumá K množství sousedů okolo zkoumaného objektu. Zatřízení do klasifikační třídy tedy funguje na principu, že když se většina objektů x_j nachází v jedné třídě, tak je do ní pak přiřazen i zkoumaný objekt x [10; 11].

U metody KNN se většinou za K dosazuje liché číslo, tak aby se zabránilo shodě počtu položek z různých tříd, což znemožňuje určení, do které třídy objekt zařadit. Pokud nastane podobná situace, je objekt obvykle náhodně zařazen do jedné z klasifikačních tříd anebo případně do rizikovější třídy. Klasifikaci se zpravidla provádí s použitím více hodnot K a poté se vybere taková hodnota K, pro kterou se dosáhne nejlepších výsledků, protože dopředu nemusí být jasné, které K je pro konkrétní data nejvhodnější [1; 10; 11].

Příklad principu a rozdílu obou metod NN a KNN jsou zobrazeny na Obr. 5, kde metoda NN je zobrazena jako modrá kružnice a dle čeho lze usoudit, že objekt x je klasifikovaný do Třídy 1. Na rozdíl od toho je metoda KNN, která je vyobrazena červenou kružnicí je objekt x klasifikovaný do Třídy 2, za použití $K = 3$ [1].



Obr. 5 Ukázka principu a rozdílu obou metod
NN (modrá kružnice) a KNN (červená kružnice) [1]

1.3.3 Metoda průměrné vazby

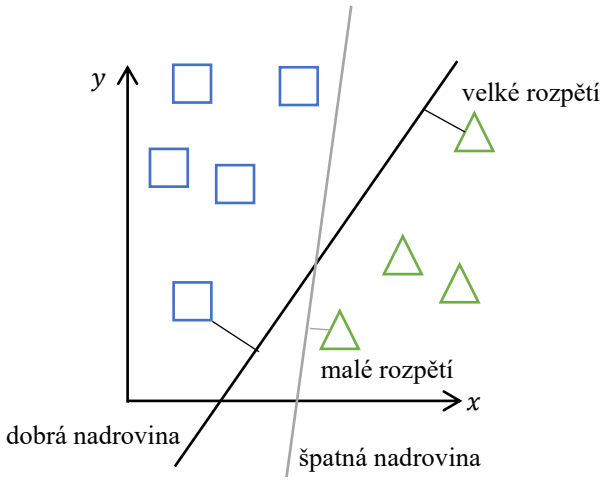
Metody NN a KNN klasifikují objekty dle minimální vzdálenosti od jednotlivých již zatřízených objektů oproti tomu metoda průměrné vazby zkoumá nejmenší průměrnou vzdálenost. Podstatou této metody je vypočet průměrné vzdálenosti od zkoumaného objetu x ke všem objektů jednotlivých tříd.

Výsledné průměrné vzdálenosti vztahující se k jednotlivých klasifikačním třídám se porovnají a zjistí se, která třída má nejmenší průměrnou vzdálenost od objektu x a do té třídy se zatřídí [1; 10].

1.3.4 Metoda podpůrných vektorů

Metoda podpůrných vektorů (support vector machines, SVM) je jednou z nejpopulárnějších algoritmu učení s učitelem, která se používá pro klasifikační problémy ve strojovém učení [1; 12].

Cílem algoritmu podpůrných vektorů je najít nadrovinu v N-rozměrném prostoru (N – počet příznaků), která jednoznačně klasifikuje datové body do tříd. SVM vybírá extrémní body / vektory, které pomáhají vytvořit nadrovinu. Tyto se pak nazývají podpůrnými vektory. Níže uvedený diagram Obr. 6, ve kterém jsou znázorněny dvě různé kategorie, které jsou klasifikovány pomocí rozhodovací hranice nebo nadroviny. Metoda maximalizuje rozpětí z obou bodů. Tedy nadrovina (v tomto případě přímka), jejíž vzdálenost k nejbližšímu prvku každého bodu je největší [1; 12].



Obr. 6 Diagram lineární SVM [12]

Lineární SVM se používá pro lineárně oddělitelná data, což znamená, že pokud lze soubor dat klasifikovat do dvou tříd pomocí jediné přímky, pak se taková data označují jako lineárně oddělitelná data a klasifikátor se používá jako lineární SVM klasifikátor [1; 12].

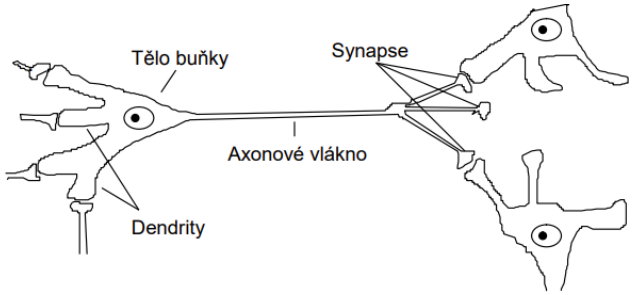
Nelineární SVM se použije v případě nelze-li soubor dat klasifikovat pomocí jedné přímky, pak se taková data označují jako nelineární data a použitý klasifikátor je nelineární [1; 12].

1.4 Neuronové sítě

Neuronové sítě (neural network, NN) jsou souborem algoritmů volně modelovaných podle fungování lidského mozku a jsou určeny k rozpoznávání vzorů. Interpretují smyslová data pomocí strojového vnímání a označují nebo shlukují surové vstupní údaje. Všechna data z reálného světa, ať už jde o obrázky, zvuky, texty nebo časové řady, musí být převedena do vzorů, které lze rozpoznat, přičemž jsou číselné a zakódované ve vektorech [13; 14; 15; 16].

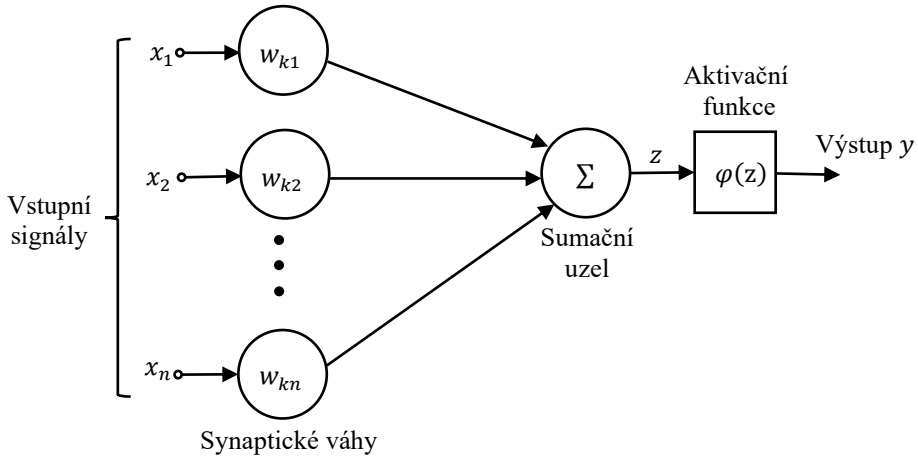
Pro ilustraci struktury umělé neuronové sítě je třeba se nejprve podívat na lidský mozek z anatomického a funkčního hlediska. Lidský mozek se skládá z přibližně 10^{11} výpočetních jednotek „neuronů“, které pracují paralelně a vyměňují si informace prostřednictvím svých spojnic „synapsí“.

Tyto neurony sečtou všechny informace, které do nich přicházejí, a pokud je výsledek vyšší než daný akční potenciál, vyšlou axonem impuls do další fáze. Anatomie lidského neuronu je znázorněna na Obr. 7 [13; 14; 15; 16].



Obr. 7 Biologický vzor [13]

Stejným způsobem se umělá neuronová síť skládá z jednoduchých výpočetních jednotek „umělých neuronů“ a každá jednotka je propojena s ostatními jednotkami pomocí váhových spojek (Obr. 8). Tyto jednotky pak vypočítají vážený součet přicházejících vstupů a zjistí výstup pomocí aktivační funkce [13; 14; 15; 16].



Obr. 8 Blokové schéma umělého neuronu (perceptron) [16]

Na základě blokového schématu a funkce neuronové sítě jsou stanoveny tři základní prvky modelu neuronové sítě [16]:

1. Synapse neboli spojovací články mající váhu, při níž vstupní signál x_i připojený k neuronu se násobí synaptickou váhou w_i.
2. Sumační uzel pro sčítání vážených vstupů.
3. Aktivační funkce pro vytvoření výstupu neuronu.

1.4.1 Perceptron

Perceptron je, obdobně jako neuron v mozku, základním stavebním blokem modelů neuronových sítí, které se dnes používají. Jedná se o lineární a binární klasifikátor a jeho potenciál je definován jako vážený součet příchozích signálů. Schéma perceptronu je zobrazeno na Obr. 8 a matematicky to lze

vyjádřit vztahem (7). Kde x_i je vstupní signál, w_i je synaptická váha a φ je aktivační funkce [13; 14; 15].

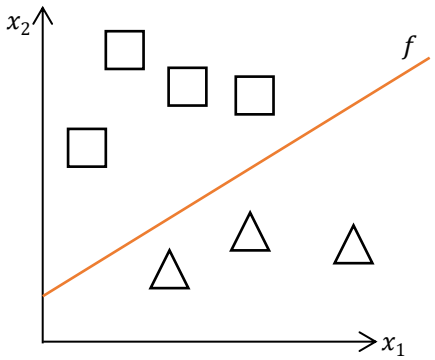
$$y = \varphi(z) = \varphi\left(\sum_{i=1}^{n} x_i \cdot w_i\right) \tag{7}$$

Aktivační funkce slouží k udržení výsledného signálu mezi danými limity, jako v případě biologického neuronu. K tomuto účelu se používají různé aktivační funkce, ale nejběžnější je signum funkce, což je spojitě diferencovatelná funkce dle vzorce (8) [13; 14; 15].

$$y = \frac{1}{1 + \exp(-z)} = \varphi(z) \tag{8}$$

Excitace neuronu se pohybuje v rozmezí 0 a 1, kde hodnota 1 je úplnou excitací neuronu a oproti tomu hodnota 0 odpovídá stavu inhibice. Přestože je perceptron pouze jeden neuron, dokáže řešit jednoduché lineární identifikační problémy, viz Obr. 9, kde je zobrazena binární klasifikace a klasifikační přímka f získaná ze vztahu (9) [13; 14].

$$f = w_1 x_1 + w_2 x_2 \tag{9}$$



Obr. 9 Jednovrstvý perceptron, dvourozměrné rozpoznávání [zdroj vlastní]

1.4.2 Adaptace perceptronu

Aby dokázal perceptron korektně roztřiďovat vstupní signály do tříd, musí se vhodně nastavit jeho hodnoty vah. Tedy dle vhodného algoritmu je zapotřebí perceptron adaptovat na základě trénovací množiny. Existuje několik algoritmů a jedním z nestarších a nejzákladnějších je adaptace dle Hebbova učení [13; 14; 15].

Hebbovo učení je založeno na neurofyziologickém srovnání. To lze chápat tak, že pokud je neuron svými vstupy správně excitován, dochází k zesílení excitačních vstupů, zatímco nesprávná excitace oslabuje excitační spojení. Pro neuron s binárním vstupem x, vahami w, výstupem y a předpokládaným výstupem $\hat{y}$ vypadá Hebbovo pravidlo následovně [13; 14; 15]:

1. Je-li neuron excitován **korektně** tj. $(y = 1; \hat{y} = 1)$, tak se v příštím diskrétním časovém kroku $n + 1$ posilují o Δ váhy w_i, které tuto excitaci vyvolaly podle vztahu (10).

$$({}_{n+1}w_i = w_i + \Delta, \forall i: x_i) \tag{10}$$

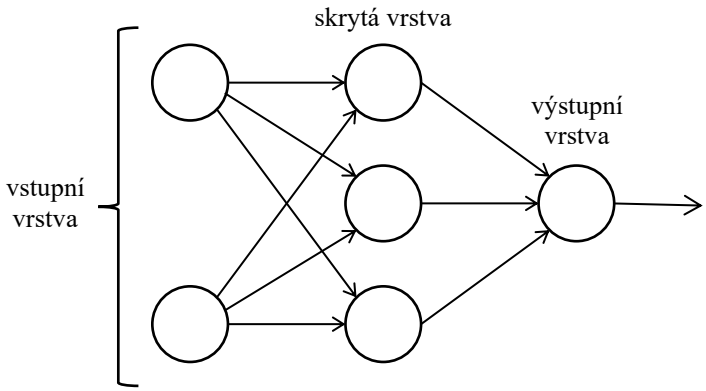
2. Je-li neuron excitován **nekorektně** tj. $(y = 1; \hat{y} = 0)$, tak se oslabují o Δ váhy w_i, které tuto excitaci vyvolaly dle vzorce (11).

$$({}_{n+1}w_i = w_i - \Delta, \forall i: x_i) \tag{11}$$

3. Není-li neuron excitován $(y = 0)$, nic se neděje, tedy váhy se nijak nemění.

1.4.3 Vícevrstvé perceptrony

Ve vícevrstvé síti (multilayer perceptron, MLP) jsou perceptrony uspořádány do vzájemně propojených vrstev viz Obr. 10.



Obr. 10 Schéma vícevrstvé sítě [zdroj vlastní]

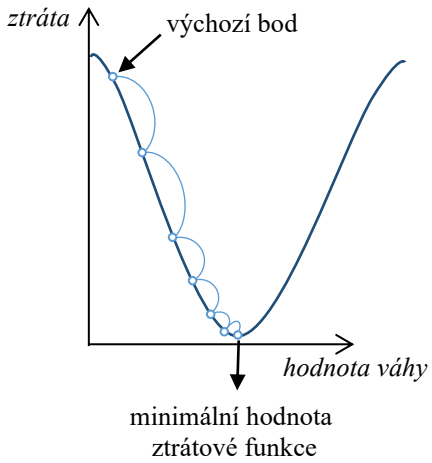
Vrstva při levém okraji této sítě se nazývá vstupní vrstvou a neurony v ní se nazývají vstupní neurony. Výstupní vrstva, která je poslední vrstvou schématu obsahuje výstupní neurony nebo, jako v tomto případě, jeden výstupní neuron. Prostřední vrstva se nazývá skrytá vrstva, protože neurony v této vrstvě nejsou ani vstupy, ani výstupy. Výše uvedená síť má pouze jednu skrytou vrstvu, ale některé sítě mohou obsahovat mnohem více skrytých vrstev [13; 14; 15].

Princip vícevrstvé sítě spočívá v tom, že nejdříve jsou neurony na vstupní vrstvě excitovány na odpovídající úroveň. Pomocí vazeb jsou pak tyto excitace přivedeny k následující vrstvě a buďto jsou zesíleny nebo zeslabeny pomocí synaptických vah. Jednotlivé neurony vyšších vrstev provedou sumaci upravených signálů od neuronů nižších vrstev. Tento proces se opakuje skrz všechny vrstvy od vnitřní vrstvy až po výstupní vrstvu, kde se nakonec získají excitační stavy všech jejích neuronů. Touto metodou dopředného šíření (feedforward) je získána odezva neuronové sítě na vstupní podnět daný excitací neuronů vstupní vrstvy. Pro správnou odezvu na vstupní signál jsou důležité synaptické váhy a jejich vhodné stanovení je úkolem adaptace neuronové sítě neboli jejím správným naučením z trénovací množiny [13; 14; 15].

1.4.4 Adaptace vah – Gradientní sestup

Gradientní sestup (gradient descent) je optimalizační algoritmus, který se běžně používá k trénování modelů strojového učení a neuronových sítí. Je to optimalizační algoritmus používaný k minimalizaci funkce iterativním pohybem ve směru nejstrmějšího sestupu, který je definován zápornou hodnotou gradientu. Ve strojovém učení se používá gradientní sestup k aktualizaci vah modelů [14; 17].

Cílem gradientního sestupu je minimalizovat ztrátovou funkci (cost function). K tomu jsou zapotřebí dva datové body – směr a rychlost učení. Tyto faktory se určují pomocí parciálních derivačních výpočtů budoucích iterací, což jim umožňuje postupně dospět k lokálnímu nebo globálnímu minimu, tj. bodu konvergence (Obr. 11) [17].



Obr. 11 Gradientní sestup [17]

Výchozí bod je náhodně určený bod, v němž je zjištěna derivace (neboli sklon) a odtud je možno pomocí tečny pozorovat strmost sklonu. Sklon informuje o změnách parametrů, tj. vah a zkreslení. Sklon ve výchozím bodě bude strmější, ale s generováním nových parametrů by se strmost měla postupně snižovat, dokud nedosáhne nejnižšího bodu na křivce, známého jako bod konvergence [17].

Velikost kroků je označována jako **míra učení**. Je důležité nastavit správnou míru učení, tak aby nebyla ani příliš vysoká ani příliš nízká. U vysoké míry učení jsou kroky velké a je možnost nenalezení minima, za to při nízké míře učení nakonec dosáhne gradientní sestup lokálního minima, ale při větší výpočetní náročnosti [10].

Ztrátová funkce měří rozdíl mezi skutečnou a předpokládanou hodnotou v její aktuální pozici podle vztahu (12).

$$f(w) = \frac{1}{N}\sum_{i=1}^{n}(y_i - w_i x_i)^2 \tag{12}$$

Kde w je váha a N je počet prvků. Ze ztrátové funkce se vypočítá gradient (13).

$$f'(w) = \frac{df}{dw} = \frac{1}{N}\sum -2x_i(y_i - wx_i) \tag{13}$$

Při řešení gradientu se iteruje přes datové body pomocí nových hodnot w a vypočítá se parciální derivace. Tento nový gradient udává sklon ztrátové funkce v aktuální poloze (aktuální hodnoty parametrů) a směr, kterým se pohybovat, pro změnu parametru [10; 14; 17].

1.4.5 Adaptace metodou zpětného šíření

Metoda zpětného šíření (backpropagation) umožňuje adaptaci neuronové sítě dané trénovací množině. Je to algoritmus pro učení neuronových sítí s učitelem pomocí gradientního sestupu. Při zadání neuronové sítě a ztrátové funkce metoda vypočítá gradient chybové funkce vzhledem k vahám neuronové sítě [13; 14].

„Zpětná“ část názvu vychází z toho, že výpočet gradientu probíhá zpětně skrz síť, přičemž gradient poslední vrstvy vah se počítá jako první a gradient první vrstvy vah se počítá jako poslední. Částečné výpočty gradientu z jedné vrstvy se znovu použijí při výpočtu gradientu pro předchozí vrstvu. Tento zpětný tok ztrátových informací umožňuje efektivní výpočet gradientu v každé vrstvě oproti předchozímu přístupu, kdy se gradient každé vrstvy počítá zvlášť [13; 14].

Princip metody je zpočátku stejný jako u dopředné metody, až do bodu, kdy se signál dostane do výstupní vrstvy. Zde se následně vypočítá chyba podle vzorce (14) [13; 14].

$$chyba = výstup_{skutečný} - výstup_{požadovaný} \tag{14}$$

Kde $výstup_{skutečný}$ je získán z průchodu vrstvami neuronové sítě a $výstup_{požadovaný}$ je hodnota z trénovací množiny [13; 14].

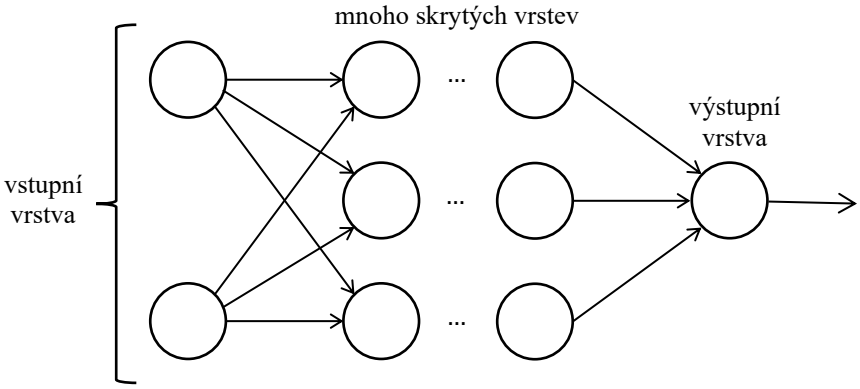
Následně se přejde zpět z výstupní vrstvy do skryté vrstvy a upraví se hodnoty synaptických vah, tak aby se chyba snížila. Tento postup je pak opakován, dokud není dosaženo požadovaného výstupu [13; 14].

1.4.6 Hluboké učení

Hluboké učení (deep learning) je specifická metoda strojového učení, která zahrnuje neuronové sítě v po sobě jdoucích vrstvách s cílem učit se z dat iterativním způsobem. Hluboké učení je užitečné zejména tehdy, když se snažíte naučit vzory z nestrukturovaných dat [17; 18; 19]. Schéma hlubokého učení je na Obr. 12.

Hluboké učení komplexní neuronové sítě je navrženo, tak aby napodobovalo fungování lidského mozku, takže počítače lze vycvičit k řešení abstrakcí a problémů, které jsou špatně definované. Průměrné pětileté dítě dokáže snadno rozpoznat rozdíl mezi tváří svého učitele a tváří hlídače na přechodu. Naproti tomu počítač musí vynaložit spoustu práce, aby zjistil, kdo je kdo. Neuronové sítě i hluboké učení se často používají v aplikacích pro rozpoznávání obrazu, řeči a počítačového vidění [17; 18; 19].

Hluboké učení je sice velmi podobné tradiční neuronové síti, ale má mnohem více skrytých vrstev. Čím složitější je problém, tím více skrytých vrstev bude v modelu. Hluboké učení se obvykle učí z neoznačených a nestrukturovaných dat [17; 18; 19].



Obr. 12 Schéma hlubokého učení [zdroj vlastní]

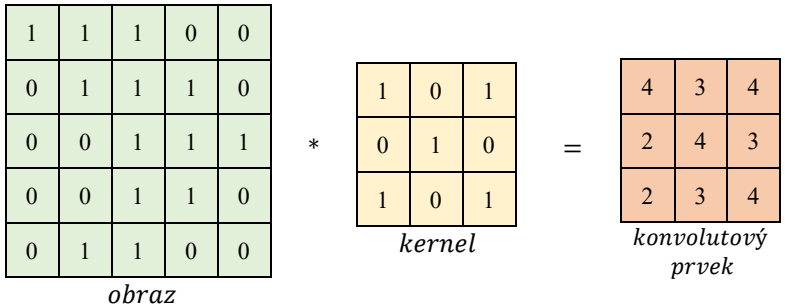
1.4.7 Konvoluční neuronové sítě

Konvoluční neuronová síť (convolution neuron network, CNN) je druh hlubokých neuronových sítí. CNN se nejčastěji používá v odvětví výpočetní techniky nazvané počítačové vidění. Při zadání série obrázků nebo signálů se s využitím CNN systém umělé inteligence naučí automaticky extrahovat vlastnosti těchto vstupů pro splnění konkrétní úlohy, např. rozpoznání arytmie z EKG, ověřování pravosti obličejů nebo sémantická segmentace obrázků [20; 21].

V modelech CNN jedna nebo více konvolučních vrstev extrahuje jednoduché příznaky ze vstupního signálu prováděním konvolučních operací. Úkolem sítě CNN je redukovat vstupní signál do podoby, která je snadněji zpracovatelná, aniž by se ztratily vlastnosti, které jsou rozhodující pro získání dobré predikce klasifikace. To je důležité, když se má navrhnout architektura, která se nejen dobře učí příznaky, ale je také škálovatelná na enormní soubory dat [20; 21].

Jádro konvoluce CNN

Prvek, který se podílí na provádění konvoluční operace, se nazývá jádrem K (kernel nebo filtr), znázorněný žlutou barvou na Obr. 13. K je zvolena jako matice 3x3. Zelená část představuje vstupní obraz 5x5 O. Délka kroku (stride) je 1, neboli Non-Strided. Jádro se tedy posune 9krát, přičemž pokaždé provede operaci násobení matice K s částí obrazu, nad kterou se jádro pohybuje D. Filtr se pohybuje doprava s určitou hodnotou kroku, dokud nezpracuje celou šířku obrazu O. Při dalším pohybu se přesune na začátek (vlevo) obrázku se stejným krokem se posune dolů a proces opakuje, dokud neprojde celou matici zleva doprava odshora dolů [20; 21].



Obr. 13 Konvoluce obrazu 5x5 s jádrem 3x3 pro získání konvoluční funkce 3x3 [21]

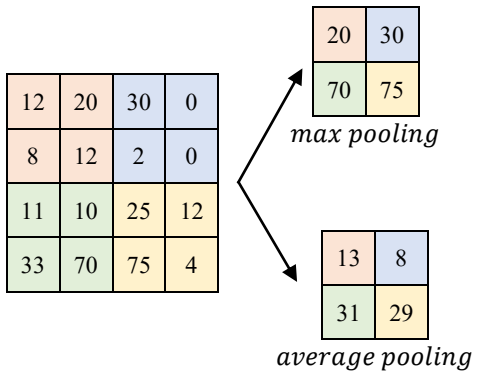
V případě obrazů se může pracovat s více vstupními kanály. Tyto kanály obsahují barevnou hloubku, pro snímek v odstínech šedi je pouze jeden. Naopak pokud má snímek více kanálů (nejčastěji 3, kde každý označuje jednu matici z RGB modelu) má jádro stejnou hloubku jako vstupní obraz. Mezi maticemi K_n a O_n se provede násobení matic ($[K_1, O_1]$; $[K_2, O_2]$; $[K_3, O_3]$), všechny výsledky se sečtou, čímž získáme zhuštěný výstup s jednou hloubkou kanálu konvoluční funkce [21].

Jedna konvoluční vrstva může obsahovat více konvolučních filtrů a dle počtu těchto filtrů se vytváří nové kanály. Pro ukázku je na vstupu konvoluční vrstvy obraz 5x5x3, který obsahuje tři barevné složky, a šest korelačních filtrů 3x3x1. Po konvoluci obrazu s prvním filtrem 3x3x1 má obraz mapu příznaků velikost 3x3x1. Výsledná mapa příznaků po všech konvolucích je velká 3x3x6. Konvoluční vrstva získá příznaky z obrazu, zmenší jeho rozlišení a zvětší počet kanálů.

Vrstva sdružení (pooling) CNN

Podobně jako konvoluční vrstva je vrstva sdružení významná při zmenšení prostorových informací obrazu. Důvodem pro zmenšování je snížení výpočetního výkonu potřebného ke zpracování dat prostřednictvím redukce rozlišení. Dále je užitečná pro extrakci dominantních příznaků, které jsou rotačně a polohově invariantní, čímž se zachovává proces efektivního trénování modelu. Vyskytují se hlavní dva typy sdružování: Max Pooling a Average Pooling, viz Obr. 14 [20; 21]:

- **Max Pooling** – vrací maximální hodnotu z části obrazu pokryté jádrem.
- **Average Pooling** – vrací průměr všech hodnot z části obrazu pokryté jádrem.



Obr. 14 Ukázka Max Pooling a Average Pooling [21]

Padding CNN

Jak je ukázáno na Obr. 13, při konvoluci dochází k zmenšení výstupního obrazu. Někdy nemusí být cílem zmenšit výstupný obraz, a tedy je použita výplň (padding). Ta „obalí“ obraz nulovými hodnotami, tím pádem proběhne více výpočtů konvolucí a mapa příznaků bude stejně velká jako vstupní signál. Výplň patří mezi vlastnosti konvoluční i sdružovací vrstvy [20; 21].

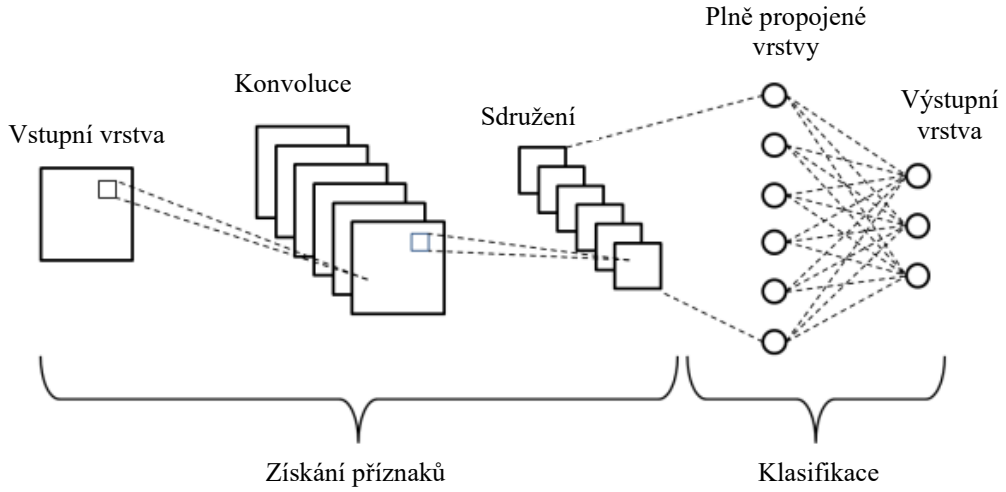
Nastavení výplně CNN se rozlišuje na dva typy dle výsledného rozlišení, a to na jeden, při kterém se zmenší rozlišení výsledné mapy příznaků ve srovnání se vstupem, a druhý, při kterém rozlišení zůstane stejné [20; 21]:

- **Same Padding** – označuje takovou výplň, při které se zachová velikost výsledné mapy příznaků vložením výplně do vstupního snímku. Např. obraz 5x5 se rozšíří na obraz 7x7. Po aplikaci konvoluce s jádrem 3x3 bude výsledná mapa příznaků mít velikost 5x5.
- **Valid Padding** – naopak pokud se zachová velikost obrazu před konvolucí, nepřidá se žádná výplň, výsledná mapa příznaků bude menší než původní obraz.

Plně propojené vrstvy

Plně propojené vrstvy (fully connected layers) jsou takové vrstvy, kde jsou všechny vstupy z jedné vrstvy propojeny s každou aktivační jednotkou následující vrstvy. Ve většině populárních modelů strojového učení je několik posledních vrstev plně propojenými vrstvami, které sestavují data získaná předchozími vrstvami a tvoří konečný výstup. Do plně propojené vrstvy vstupuje signál o velikosti 1x1 a počet kanálů odpovídá počtu klasifikačních tříd. Z toho vyplývá, že obrazy jsou maximálně zmenšeny a z toho se vyhodnotí, klasifikuje, do jaké třídy patří [20; 21].

V jednoduchosti je konvoluční neuronová síť posloupnost vrstev a každá vrstva CNN převádí mapu příznaků do další vrstvy prostřednictvím vlastní funkce. K sestavení základní architektury CNN se používají tři hlavní typy vrstev: konvoluční vrstva, sdružovací vrstva a plně propojená vrstva, viz Obr. 15 [20; 21].



Obr. 15 Jednoduchá architektura konvoluční neuronové sítě [22]

1.4.8 Předučené sítě CNN

Doladění předučené sítě pomocí přenosového učení je obvykle mnohem rychlejší a jednodušší než trénování od nuly. Vyžaduje nejmenší množství dat a výpočetních zdrojů. Přenosové učení využívá znalosti z jednoho typu problému k řešení podobných problémů. Začne se s předučenou sítí a použije se k naučení nové úlohy. Jednou z výhod přenosového učení je, že předučená síť se již naučila bohatou sadu příznaků, které lze použít na celou řadu dalších podobných úloh.

K dispozici jsou různé předučené architektury sítí CNN, které jsou klíčové při vytváření algoritmů, jež pohánějí a v dohledné budoucnosti budou pohánět téměř veškerou umělou inteligenci. Některé z nich jsou uvedeny níže:

- **LeNet-5** – síť má 5 vrstev s naučitelnými parametry. Tři sady konvolučních vrstev s kombinací average pooling. Po konvolučních vrstvách a vrstvách s average pooling jsou dvě plně propojené vrstvy [23].
- **AlexNet** – síť, která v roce 2012 zpopularizovala CNN v oblasti počítačového vidění. Architektura se skládá z osmi vrstev: pěti konvolučních vrstev a tří plně propojených vrstev [24].
- **GoogLeNet** – konvoluční neuronová síť z roku 2014, která má 22 vrstev. Výrazně snížila počet parametrů (na 4 miliony oproti 60 milionům z AlexNetu) [25].
- **ResNet** – obsahuje speciální přeskakovací spojení. Nemá plně propojenou výstupní vrstvu [26].
- **VGGNet** – VGG je zkratka pro Visual Geometry Group a jedná se o standardní hlubokou architekturu CNN. Používá masku konvoluce 3x3 a pro sdružování masku 2x2. Existují dva typy, přičemž VGG-16 nebo VGG-19 a ty se skládají z 16 a 19 konvolučních vrstev [27].

2 Segmentace dat

Segmentace obrazu je technika, při níž se počítačový snímek rozdělí na různé podskupiny zvané segmenty, které pomáhají snížit složitost snímku, tak aby bylo další zpracování nebo zkoumání snímku méně obtížné. Segmentace je zjednodušeně řečeno přiřazování názvů pixelům. Všem složkám obrazu nebo pixelům, které mají místo s podobnou třídou, je přiděleno typické označení. Segmentace obrazu může sloužit, jako krok předzpracování dat před použitím algoritmu strojového učení, ke snížení časové náročnosti, kterou algoritmus strojového učení potřebuje ke zpracování obrazu [28; 29].

Segmentační algoritmy pro obrazy jsou obecně založeny na nespojitosti nebo podobnosti hodnot intenzity jasu obrazu. Přístup založený na diskontinuitě rozděluje obraz na základě náhlých změn intenzity např. hranová segmentace. Další přístup založený na podobnosti rozděluje obraz na oblasti, které jsou si podobné podle souboru předem definovaných kritérií např. regionální segmentace. Výběr techniky segmentace obrazu tedy závisí na uvažovaném problému. Výsledkem segmentace obrazu je sada oblastí, které společně pokrývají celý obraz a sada obrysů extrahovaných z obrazu. Všechny pixely v oblasti jsou si podobné s ohledem na některé charakteristiky, jako je barva, intenzita nebo textura. Sousední regiony se výrazně liší s ohledem na stejnou individualitu [28; 29].

2.1 Hranová segmentace

Proces klasifikace a umístění ostrých nespojitostí v obraze se nazývá detekce hran. Hrany jsou lokální změny v intenzitě obrazu a obvykle se vyskytují na hranici mezi dvěma oblastmi. Nespojitosti jsou okamžité změny koncentrace pixelů, které rozlišují hranice objektů ve scéně. Klasické metody detekce hran zahrnují konvoluci obrazu s operátorem, který je konstruován, tak aby byl vnímavý k velkým gradientům v obraze, ačkoli v jednotných oblastech vrací nulové hodnoty.

K dispozici je velmi velké množství technik detekce hran, přičemž každá z nich je navržena, tak aby vnímala určité typy hran. Proměnné, které se týkají výběru operátoru detekce hran, se skládají z orientace hran, struktury hran a šumového prostředí. Geometrie operátoru určuje charakteristický směr, ve kterém je nejlépe vnímavý k hranám. Operátory lze optimalizovat, tak aby vyhledávaly svislé, vodorovné nebo diagonální hrany [28; 30; 31].

V literatuře existuje mnoho technik detekce hran pro segmentaci obrazu. Těmito technikami jsou Robertsova detekce hran, Sobelova detekce hran, Prewittova detekce hran, Kirshova detekce hran, Robinsonova detekce hran, Marr-Hildrethova detekce hran, LoG detekce hran a Cannyho detekce hran [31].

2.1.1 Sobelova detekce hran

Sobelova metoda detekce hran pro segmentaci obrazu vyhledává hrany pomocí Sobelovy aproximace k derivaci. Hrany se hledají v těch bodech, kde je gradient nejvyšší. Sobelova technika provádí na obraze 2-D prostorovou gradientní veličinu a tak zvýrazňuje oblasti s vysokou prostorovou frekvencí, které odpovídají hranám. Obecně se používá k nalezení odhadované absolutní velikosti gradientu v každém bodě n vstupního obrazu ve stupních šedi. Operátor se skládá alespoň z kombinace 3x3 komplikačních jader, jak je uvedeno v Obr. 16. Jedno jádro je jednoduše druhé otočené o 90° [31].

G_x

-1	-2	-1
0	0	0
$+1$	$+2$	$+1$

G_y

-1	0	$+1$
-2	0	$+2$
-1	0	$+1$

Obr. 16 Schéma Sobelova operátoru [31]

2.1.2 Robertsova detekce hran

Provádí jednoduché, rychle vypočitatelné měření 2-D prostorového gradientu na obrázku. Tato metoda zdůrazňuje oblasti s vysokou prostorovou frekvencí, které často odpovídají hranám. Vstupem pro operátor je obraz ve stupních šedi, stejný jako na výstupu, což je nejběžnější použití této techniky. Hodnoty pixelů v každém bodě výstupu představují odhadovanou úplnou velikost prostorového gradientu vstupního obrazu v daném bodě. Schéma operátoru je na Obr. 17 [31].

G_x

-1	0
0	$+1$

G_y

0	-1
$+1$	0

Obr. 17 Schéma Robertsova operátoru [31]

2.1.3 LoG detekce hran

LoG (Laplacian of Gaussian) obrazu $f(x, y)$ je derivace druhého řádu definovaná podle (15).

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{15}$$

Metoda má dva efekty, vyhlazuje obraz a počítá Laplacián, čímž vzniká obraz s dvojitým okrajem. Lokalizace hran pak spočívá v nalezení nulových průsečíků mezi dvojitými hranami. Digitální implementace Laplaciánovy funkce se obvykle provádí pomocí masky v Obr. 18 [31; 30].

G_x

0	-1	0
-1	4	-1
0	-1	0

G_y

-1	-1	-1
-1	8	-1
-1	-1	-1

Obr. 18 Schéma LoG operátoru [31]

2.2 Regionální segmentace

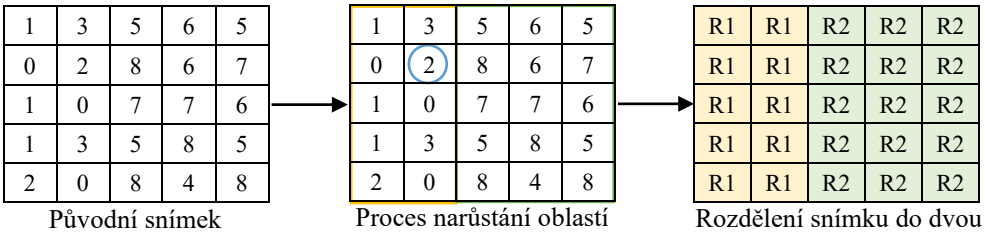
Oblast lze klasifikovat jako skupinu propojených pixelů, které vykazují podobné vlastnosti. Podobnost mezi pixely se může týkat intenzity, barvy atd. V tomto typu segmentace jsou přítomna určitá předem definovaná pravidla, která musí pixel dodržet, aby mohl být zařazen do určitých oblastí pixelů. Metody regionální segmentace mají v případě zašuměného obrazu přednost před metodami

segmentace detekce hran. Regionální segmentace se dále dělí na dva typy podle přístupů, kterými se řídí [30; 32].

2.2.1 Metoda narůstání oblastí

V případě metody narůstání oblastí se začne s jedním pixelem označovaném jako počáteční (seed) pixel. Poté se kontroluje jasová složka sousedních pixelů. Pokud sousední pixely vyhovují předem definovaným pravidlům, je tento pixel přidán do oblasti počátečního pixelu a následující proces se opakuje, dokud nezůstane žádná podobnost. V případě zvětšování regionu lze preferované pravidlo nastavit jako prahovou hodnotu [30; 32].

Příklad na Obr. 19: ve snímku o velikosti 5x5 je označen počáteční pixel (modrý kroužek) a prahová hodnota je nastavena na 3. Pokud je pixel ve snímku podobný dle prahové hodnoty počátečnímu pixelu, bude považován za oblast uvnitř počátečního pixelu (žlutá). V opačném případě bude považován za oblast jinou (zelená) [30; 32].



Obr. 19 Ukázka metody narůstání oblastí [32]

2.3 Aktivní kontury

Aktivní kontury jsou křivky, které se deformují v digitálních obrazech a slouží k získání tvaru objektů. Lze jej definovat jako proces získávání deformovatelných modelů nebo struktur s omezujícími silami v obraze. Obrysové modely popisují hranice objektu nebo jiné prvky obrazu, tak aby tvořily parametrickou křivku nebo obrys. Zakřivení modelů se určuje pomocí různých obrysových algoritmů s využitím vnějších a vnitřních sil, které se na ně aplikují. Funkční energie je vždy spojena s křivkou definovanou v obraze. Vnější energie je definována jako kombinace sil působících na obraz, která se konkrétně používá k řízení umístění obrysu na obraz, a vnitřní energie, která řídí deformovatelné změny [30; 33].

2.4 Metoda Level-set

Obdobný přístup jako aktivní kontury využívá level-set segmentace. Křivka je reprezentována tzv. nulovou hladinou což je řez v rovině *xy* nějakou vícerozměrnou funkcí. Tato funkce se nazývá level-set funkce a každému bodu roviny *xy* přiřazuje jeho výšku *u* nad nulovou hladinou. Povrch funkce se postupně adaptuje vzhledem k zadaným metrikám křivosti a obrazovým gradientům [30].

Aktivní kontury explicitně přesouvají předem definované body na základě schématu minimalizace energie, zatímco přístupy založené na level-set přesouvají kontury implicitně prostřednictvím level-set funkce [30].

3 Segmentace dat na bázi strojového učení

Při segmentaci obrazu patří každý anotovaný pixel v obraze do jedné třídy. Často se používá k označování snímků pro aplikace, které vyžadují vysokou přesnost a je náročná na ruční práci, protože vyžaduje přesnost a znalosti problému. Zpracování jednoho obrázku může trvat delší dobu. Při segmentaci s pomocí umělé inteligence je výstupem maska, která obkresluje tvar objektu na obrázku s popisem. Segmentační anotace existují v mnoha různých typech jako je sémantická segmentace, segmentace instancí, panoptikální segmentace [34; 35].

3.1 Sémantická segmentace

Sémantická segmentace je proces klasifikace každého pixelu, který patří k určité značce. Neliší se v různých instancích téhož objektu. Například pokud jsou na rentgenovém snímku dvě kosti, sémantická segmentace přidělí všem pixelům obou kostí stejnou značku [34; 35].

3.2 Segmentace instancí

Segmentace instancí se od sémantické segmentace liší v tom, že každé instanci určitého objektu v obraze přiřazuje jedinečnou značku. Příkladem může být rentgenový snímek s třemi kostmi, kdy všem třem kostem jsou přiřazeny různé barvy, tj. různé značky. Při sémantické segmentaci by všem byla přiřazena stejná barva označující, že se jedná o kost [34; 35].

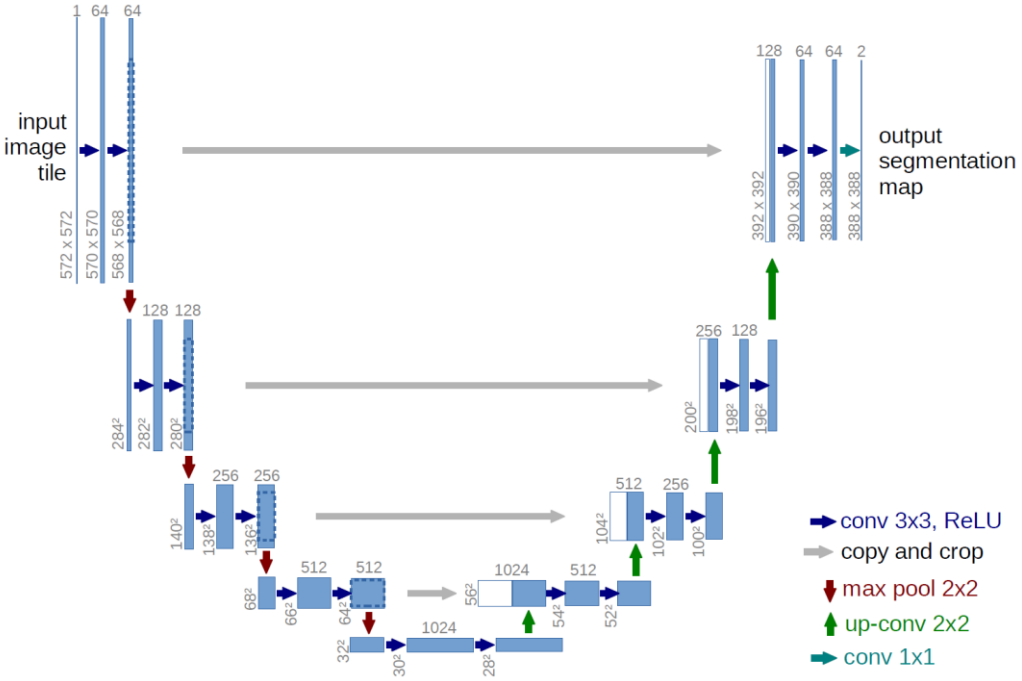
3.3 Architektury obrazové segmentace

Základní architektura segmentace obrazu se skládá z kodéru a dekodéru. Kodér získává z obrazu vlastnosti pomocí předem natrénované klasifikační sítě. Dekodér je zodpovědný za generování konečného výstupu, tedy sémantické promítnutí diskriminačních příznaků (nižší rozlišení) naučené kodérem do pixelového prostoru (vyšší rozlišení) a získat tak hustou klasifikaci. Většina architektur má tuto strukturu nebo její variantu [34; 35].

3.3.1 U-net

U-net je konvoluční neuronová síť vyvinutá pro segmentaci biomedicínských obrazů. Při vizualizaci vypadá její architektura jako písmeno U, a proto se jmenuje U-net. Její architektura se skládá ze dvou částí, levé části – **uzavírací cesty** a pravé části – **rozšiřovací cesty**, viz Obr. 20. Účelem uzavírací cesty je zachytit kontext, zatímco úlohou rozšiřovací cesty je pomoci při přesné lokalizaci [36]. Na Obr. 20 každý modrý rámeček odpovídá vícekanálové mapě prvků. Počet kanálů je označen v horní části rámečku. Velikost x-y je uvedena na levém dolním okraji rámečku. Bílé boxy představují zkopírované mapy prvků. Šipky označují různé operace [36].

Uzavírací cesta se řídí typickou architekturou konvoluční sítě. Skládá se z opakované aplikace dvou konvolucí 3x3, z nichž každá je následována rektifikovanou lineární jednotkou (ReLU) a operací 2x2 max pooling s krokem 2 pro downsampling. V každém kroku převzorkování zdvojnásobíme počet kanálů funkce [36].

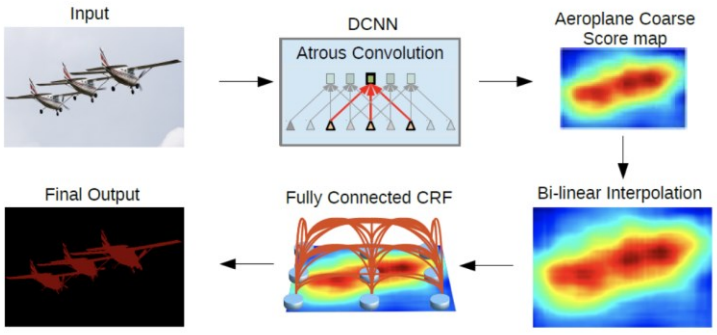


Obr. 20 Architektura U-net, příklad pro 32x32 pixelů v nejnižším rozlišení [36]

Rozšiřovací cesta se skládá z upsampling mapy příznaků konvolucí 2x2, která sníží počet kanálů příznaků na polovinu. Mapa příznaků se spojí s příslušnou mapou příznaků z uzavírací cesty a dvou konvolucí 3x3, po nichž následuje ReLU. V poslední vrstvě se použije konvoluce 1x1 k namapování každého 64složkového vektoru příznaků na požadovaný počet tříd. Celkem má síť 23 konvolučních vrstev [36].

3.3.2 DeepLab

V této architektuře (Obr. 21) se pro úlohy, které zahrnují hustou predikci, používají konvoluce s filtry se zvýšeným vzorkováním. Segmentace objektů ve více měřítcích se provádí pomocí atrousového prostorového pyramidového sdružování. Nakonec jsou DCNN použity ke zlepšení lokalizace hranic objektů. Atrous konvoluce se dosahuje upsamplingem filtrů pomocí vkládání nul nebo řídkého vzorkování vstupních map příznaků [37].

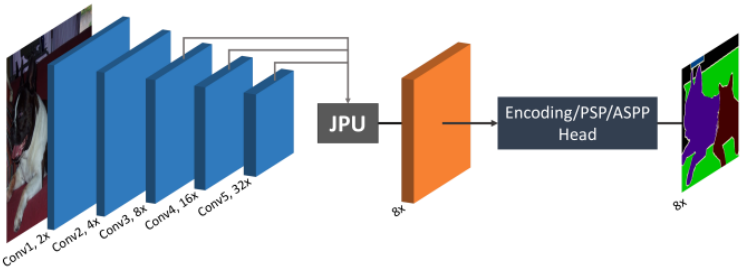


Obr. 21 Ilustrace modelu DeepLab [37]

Hluboká konvoluční neuronová síť je použita plně konvolučním způsobem, přičemž se používá atrous konvoluce ke snížení stupně převzorkování signálu (z 32x na 8x). Stupeň bilineární interpolace zvětšuje mapy prvků na původní rozlišení obrazu. Poté se použije plně propojený CRF pro zpřesnění výsledku segmentace a lepší zachycení hranic objektu. [37]

3.3.3 FastFCN —Fast Fully Convolutional Network

V této architektuře (Obr. 22) se používá modul Joint Pyramid Upsampling (JPU), který nahrazuje dilatační konvoluce, protože ty vyžadují mnoho operační paměti a výpočetního času. Ve svém jádru používá plně propojenou síť, přičemž pro převzorkování používá JPU. JPU převzorkuje mapy prvků s nízkým rozlišením na mapy prvků s vysokým rozlišením [38].



Obr. 22 Ilustrace modelu FastFCN [38]

Tato metoda využívá stejnou základnu jako původní FCN. Po základní kostře je navržen nový modul převzorkování s názvem Joint Pyramid Upsampling (JPU), který bere poslední tři mapy příznaků jako vstupy a generuje mapu příznaků s vysokým rozlišením. K vytvoření finální mapy značek je pak použit modul s více měřítky / globálním kontextem. [38]

3.4 Porovnání manuální segmentace a sémantické segmentace

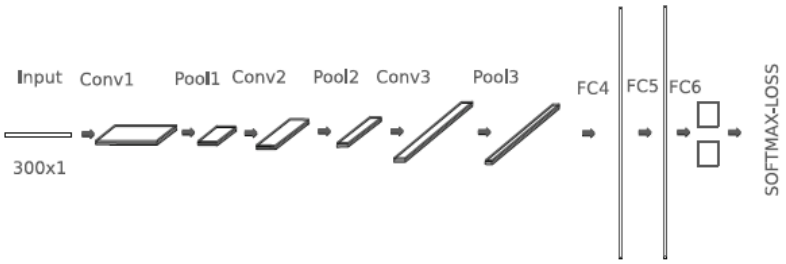
U segmentací manuálních, tedy regionální segmentaci nebo použitím aktivních kontur, je nutné mít dohled zkušeného pracovníka, který na celou segmentaci nejen dohlíží, ale i zahajuje, třídí, označuje. Manuální segmentace vytvoří požadované regiony, ty je však nutné označit manuálně, oproti sémantické segmentaci, kde již jsou vzniklé segmenty označeny. Přesto však i při sémantické segmentaci je nutné mít dozor nad správností označení. Ačkoliv manuální segmentace potřebuje k dosažení dobrých výsledků hodně času, hlavně z důvodu ručního označení, je stále více rozšířená než sémantická segmentace, která se poslední dobou stává stále více populární [28; 29; 34; 35].

4 Rešeršní studie

Tato rešeršní práce je zaměřena na použití metod hlubokého učení v praxi na medicínských datech. V první části rešerše jsou zkoumány práce zabývající se mimo jiné metodami detekce QRS komplexu z EKG signálu pomocí neuronových sítí. Následně pak v druhé části studie se práce zaobírá sémantickou segmentací a jejím použitím pro rozpoznávání objektů na různých medicínských snímcích.

4.1 Analýza EKG signálu

Studie pro rozbor EKG signálu nejčastěji používaly konvoluční nervové sítě (CNN) s vlastními návrhy architektur co se týče počtu konvolučních vrstev nebo velikosti konvolučního jádra. Autoři článku [39] používají dvě jednoduché CNN, kde první je velmi hluboká a má malá receptivní pole (Obr. 23), tj. malá filtrační jádra na první vrstvě, zatímco druhá vrstva používá větší filtrační jádra. Naproti tomu autoři dalšího článku [40] navrhli pro klasifikaci EKG signálů 16vrstvou hlubokou konvoluční síť. Následující studie navrhují komplikovanější architektury sítí.



Obr. 23 Architektura CNN s velkými receptivními poli
se třemi konvolučními, sdruženými a plně propojenými vrstvami [39]

4.1.1 CNN s dilatovanými bloky

Ve článku [41] autoři navrhují CNN model, který se skládá ze třech paralelních dilatovaných bloků CNN a každý konvoluční blok obsahuje šest 1D konvolučních vrstev. Velikost jejich konvolučních jader je v zobrazena v Tab. 1 spolu s dilatační mírou pro jednotlivé bloky.

Tab. 1 Detaily dilatovaných bloků [41]

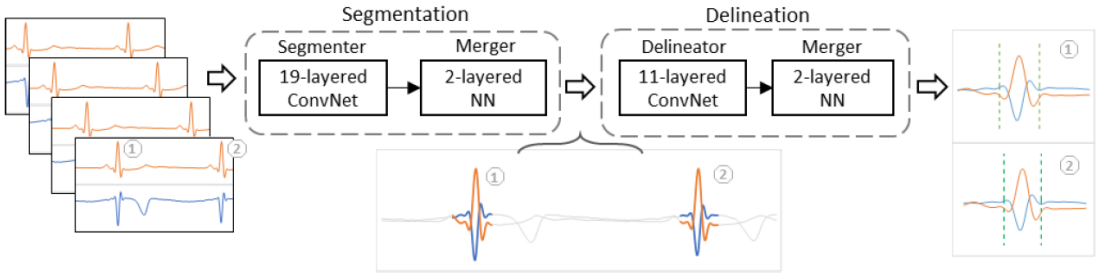
Vrstva	**Velikost jádra**	**Míra dilatace**		
		Blok 1	Blok 2	Blok 3
Conv1	11	1	2	4
Conv2	7	1	2	4
Conv3	7	1	4	8
Conv4	5	1	8	16
Conv5	5	1	8	32
Conv6	5	1	8	64

Míra dilatace 2 znamená, že konvoluce bere každý druhý bod jako vstup. Kombinací různých rychlostí dilatace se získají různé receptivní pole pro výstupní neurony. Následně jsou příznaky extrahované konvolučními bloky přivedeny do squeeze-and-excitation networks (SENet), po nichž

následují tři plně propojené vrstvy. Poslední vrstva používá sigmoidní aktivaci k predikci QRS komplexů [41].

4.1.2 Vymezení EKG signálu

Další studie [42] se zabývá vymezením EKG signálu. Navrhují systém vymezení tvořený dvěma moduly, z nichž každý se skládá ze dvou modelů hlubokého učení, viz Obr. 24. Obr. 24 znázorňuje dva moduly a ukazuje fungování systému. Nejdříve segmentační modul tvořený segmentátorem a segmentačním slučovačem, lokalizuje QRS vlny v EKG signálu. Poté jsou data opět rozdělena do oken s použitím pouze oblastí obsahujících komplex QRS. Nakonec vymezovací modul, který je tvořen vymezovačem a vymezovacím slučovačem, definuje značky začátku a posunu QRS.



Obr. 24 Schéma vymezovacího systému [42]

V segmentačním modulu se segmentátor skládá z 19vrstvé 1D konvoluční sítě složené ze 16 konvolučních a tří plně propojených vrstev. Každá konvoluční vrstva je tvořena 16 jádry o délce 3. Segmentační slučovač je dvouvrstvý NN složený ze dvou plně propojených vrstev s 512 a jedním neuronem v první a druhé vrstvě [42].

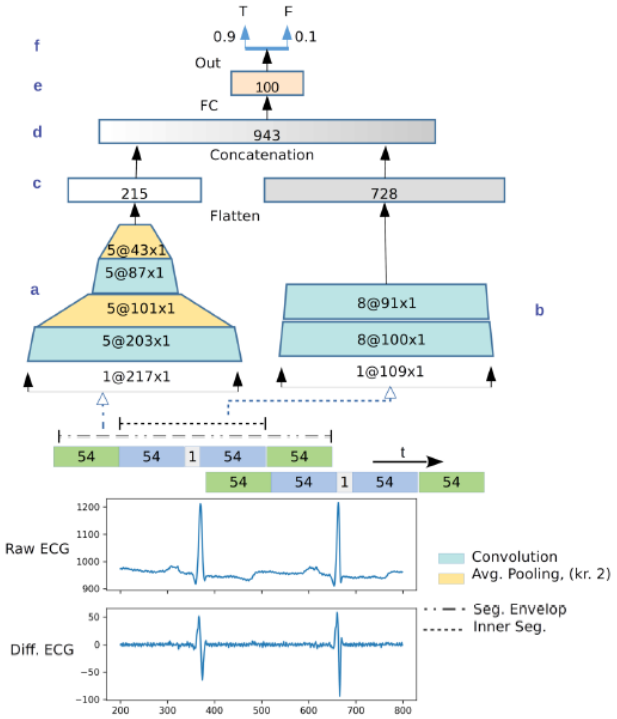
Analogicky k segmentačnímu modulu jsou i vymezovač a vymezovací slučovač 1D CNN a NN. Kromě toho, že má 8 konvolučních vrstev s jádry o velikosti 5, je vymezovač složen ze stejných prvků jako segmentátor. Na druhou stranu je vymezovací slučovač totožný se segmentačním [42].

4.1.3 Two-level CNN model

Autoři článků [43] i [44] navrhují na základě 1-D konvoluční neuronové sítě (CNN) přesnou metodu detekce QRS komplexu. Navržená CNN se skládá z object-level CNN a part-level CNN pro automatickou extrakci různě hrubých morfologických rysů EKG. Všechny extrahované morfologické rysy jsou následně zpracovány buďto plně propojenými vrstvami [43] nebo vícevrstvým perceptronem (MLP) [44] pro detekci QRS komplexu. Kromě toho je použita jednoduchá technika předzpracování EKG signálu, která obsahuje pouze diferenční operace v časové oblasti.

Pro první úroveň CNN (object-level CNN) je provedena operace průměrování několika vzorků EKG, po níž následovala operace rozdílu každého zprůměrovaného výsledku, tak aby se jako vstup vytvořil segment průměrných rozdílných signálů, který poté prošel dvěma sadami vrstev konvoluce a sdružování, tímto vzniknou hrubé příznaky. Vstupní segment pro druhou vrstvu CNN (part-level CNN) je vytvořen jednoduchou operací rozdílu mezi po sobě jdoucími vzorky, které pak prošly jednou vrstvou konvolučně-sdružených vrstev pro extrakci jemných příznaků. Oba směry příznaků jsou poté spojeny a následně vloženy do klasifikátoru (tj. plně propojené vrstvy či MLP) pro klasifikaci [43; 44].

Schéma architektury je v Obr. 25, kde je (a) CNN první úrovně, která se skládá ze dvou sad střídavých konvolučních (azurová) a průměrných sdružovacích (žlutá) vrstev, (b) je CNN druhé úrovně skládající se ze dvou po sobě jdoucích konvolučních vrstev bez sdružovací vrstvy, (c) představuje zploštělé funkce CNN každé úrovně CNN, (d) je spojení zploštělých funkcí dvou úrovní, která je přiváděna do plně propojené vrstvy (100 neuronů) (e) a následně do výstupní vrstvy (f), která se skládá ze dvou neuronů. Spodní část obrázku ukazuje vytvoření segmentu z nezpracovaného a diferencovaného EKG signálu. Čárkovaná čára představuje celý segment, a tečkovaná čára představuje vnitřní segment se středovým bodem jako bodem detekce R-peak [43].



Obr. 25 Architektura modelu two-level attention-based CNN [43]

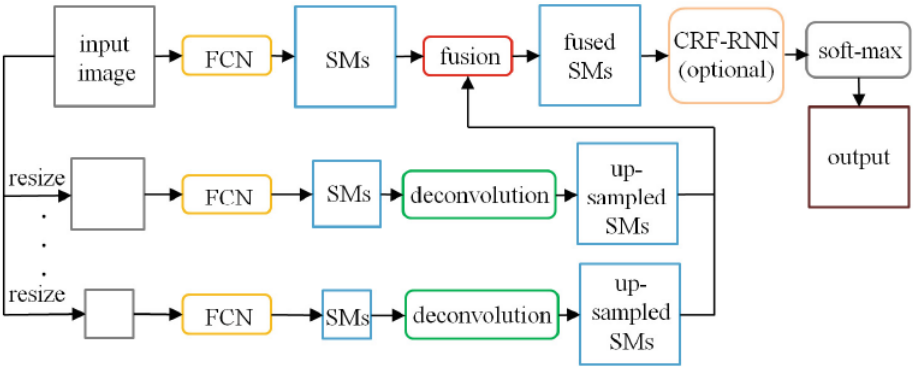
4.2 CNN v sémantické segmentaci pro analýzu lékařských snímků

S rozvojem hlubokého učení, zejména s nástupem CNN, počítačové algoritmy s hlubokým učením nejen přesně klasifikují obrazy, ale také lépe pracují na segmentaci. V úloze sémantické segmentace obrazu počítačové algoritmy segmentují obrazy na základě sémantiky a pixelů prezentovaných v obrazech. Vstupem je tříkanálový obraz RGB o rozměrech H × W × 3 a výstupem je odpovídající matice H × W, jejíž prvek označil sémantickou značku v příslušném pixelu. Analytické výsledky sémantické segmentace nejen identifikují objekty, ale také označují hranice jednotlivých objektů. Mezi současné populární algoritmy pro sémantickou segmentaci patří FCN, SegNet, DeepLab a UNet [45].

4.2.1 Multi-scale FNC

Autoři článku [46] navrhují víceúrovňovou (muli-scale) síť (Obr. 26) založenou na plně konvoluční síti (fully convolutional network, FCN). FCN je lepší než konvoluční neuronové sítě (CNN) při klasifikaci na úrovni pixelů. FCN převádí plně propojené vrstvy CNN na konvoluční vrstvy,

aby se snížil nadbytečný výpočet způsobený překrývajícími se posuvnými okny. Velikost výsledné mapy je však stále menší, než velikost vstupního obrazu. FCN dále spojuje dekonvoluční vrstvy, aby se mapa výsledků zvětšila na velikost vstupního obrazu.



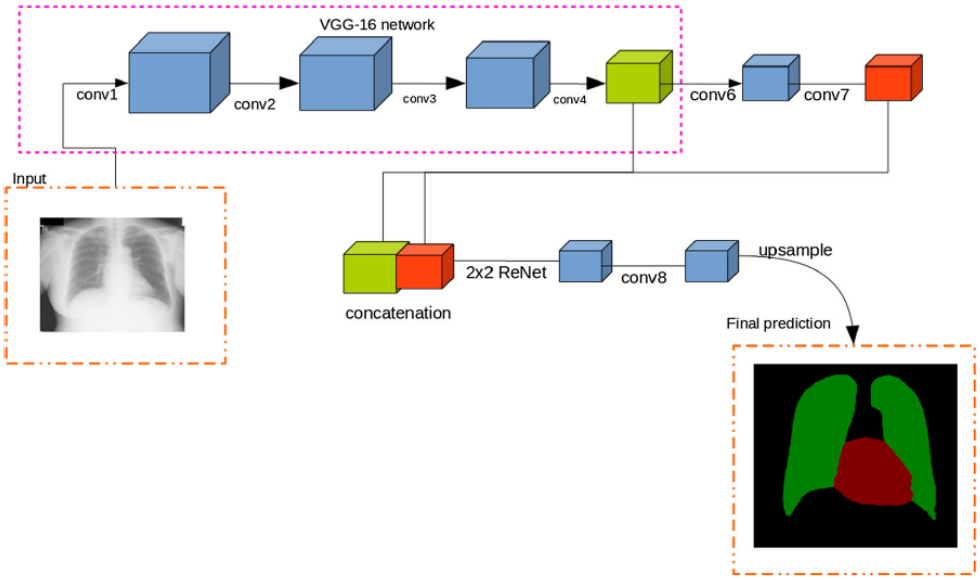
Obr. 26 Architektura víceúrovňové sítě FCN [46]

Jejich hlavní myšlenkou je nejprve použít různé FCN, z nichž každá se stará o jiný rozměr vstupního obrazu. Poté sloučí výsledné mapy (SM) vytvořené těmito FCN. Nakonec sloučené výsledné mapy projdou funkcí soft-max, aby se vypočítala klasifikační ztráta křížové entropie [46].

4.2.2 Model ReNet

Následující model ReNet vyskytující se ve článku [47] navazuje na model ReSeg: model rekurentní neuronové sítě pro sémantickou segmentaci, který předpovídá architekturu struktury pomocí lokálního obecného příznaku extrahovaného CNN. Model ReNet zpracovává vstupní obraz v první vrstvě předtrénovaného modelu VGG-16 na předtrénovaném obrazu Net a tomuto obrazu se sníží rozlišení. Poté je výstup nebo mapa příznaků získaná z této vrstvy přivedena do vrstvy ReNet, která prochází napříč obrazem a na konci je použita vrstva up-sampling, která mění velikost poslední mapy příznaků, aby měla stejné rozlišení jako vstupní obraz.

Na Obr. 27 lze vidět, že nejprve byla základní architektura plně konvoluční sítě upravena ze sítě VGG-16. Použili pouze některé z prvních vrstev sítě VGG-16 (čtyři konvoluční vrstvy a čtyři sdružovací vrstvy max pooling). Protože kombinací příznaků z více vrstev se získá více diskriminačních příznaků pro problémy sémantické segmentace a detekce objektů, spojili mapu příznaků z různých vybraných vrstev VGG-16, přičemž výběr příznaků z každé vrstvy byl proveden systematicky. Za vrstvu plně konvoluční sítě vybraných vrstev VGG-16 je vložena skupina vrstev ReNet, za kterou následuje jedna konvoluční vrstva a převzorkování pomocí transponované konvoluce nebo konvoluce s částečným krokem [47].



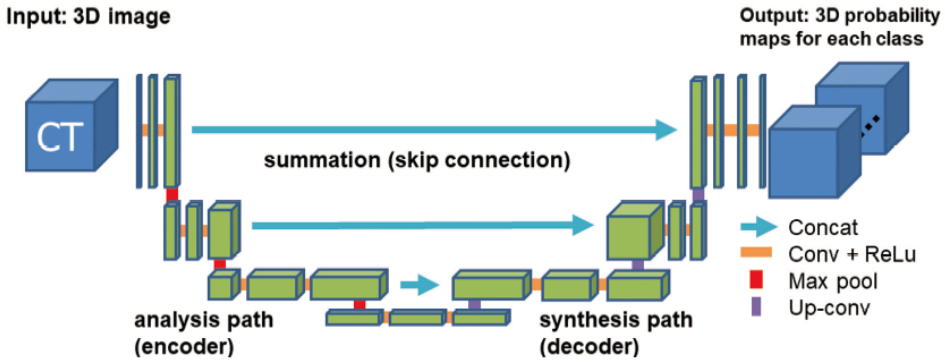
Obr. 27 ReNet s plně konvoluční strukturou sítě [47]

4.2.3 Fully Convolutional dense Dilated Net

Navrhovaná síť autory v článku [48] je Fully Convolutional dense Dilated Net (FCdDN). Aby bylo možné zachovat podrobnější informace v obrazech s nízkým rozlišením a kontrastem, používá tato síť strukturu podobnou U-net pro vstup mapy prvků generované v procesu od fáze downsampling do fáze upsampling pomocí přeskokových spojení. Navrhovaná síť integruje výhody husté konektivity, rozšířené konvoluce a faktorizovaných filtrů a navrhuje novou „vrstvu“ jako základní strukturu sítě. Pro další vylepšení jejich sítě navrhují optimalizační metodu založenou na ztrátové funkci křížové entropie.

4.2.4 3D FCN

Studie [49] se zabývá jak zpracováním 2D lékařských snímků, tak i 3D snímků. Navrhují pro to architekturu založenou na klasické síti U-Net, a to 3D U-Net neboli 3D Fully convolutional networks.



Obr. 28 Architektura 3D U-Net [49]

Architektura (Obr. 28) se skládá ze dvou symetrických cest, analýzy a syntézy, se čtyřmi úrovněmi rozlišení. Každá úroveň rozlišení v rámci cesty analýzy obsahuje dvě konvoluční vrstvy s jádry 3x3x3, za nimiž následuje aktivace ReLU a 2x2x2 max pooling s kroky po dvou v každé

dimenzi. Při syntéze se transponované konvoluce používají k přemapování map prvků s nižším rozlišením na vstupní obrazy s vyšším rozlišením. Po nich opět následují dvě 3x3x3 konvoluce, z nichž každá využívá aktivaci ReLU. Dále se využívá přeskočení (skip) spojení vrstev se stejným rozlišením, tedy z analýzy do syntézy z důvodu přenesení příznaků s vyšším rozlišením. Poslední konvoluční vrstva využívá voxelovou aktivační funkci softmax k výpočtu 3D mapy pravděpodobnosti pro každý z cílových orgánů jako výstup sítě [49].

5 Úvod do praktické části

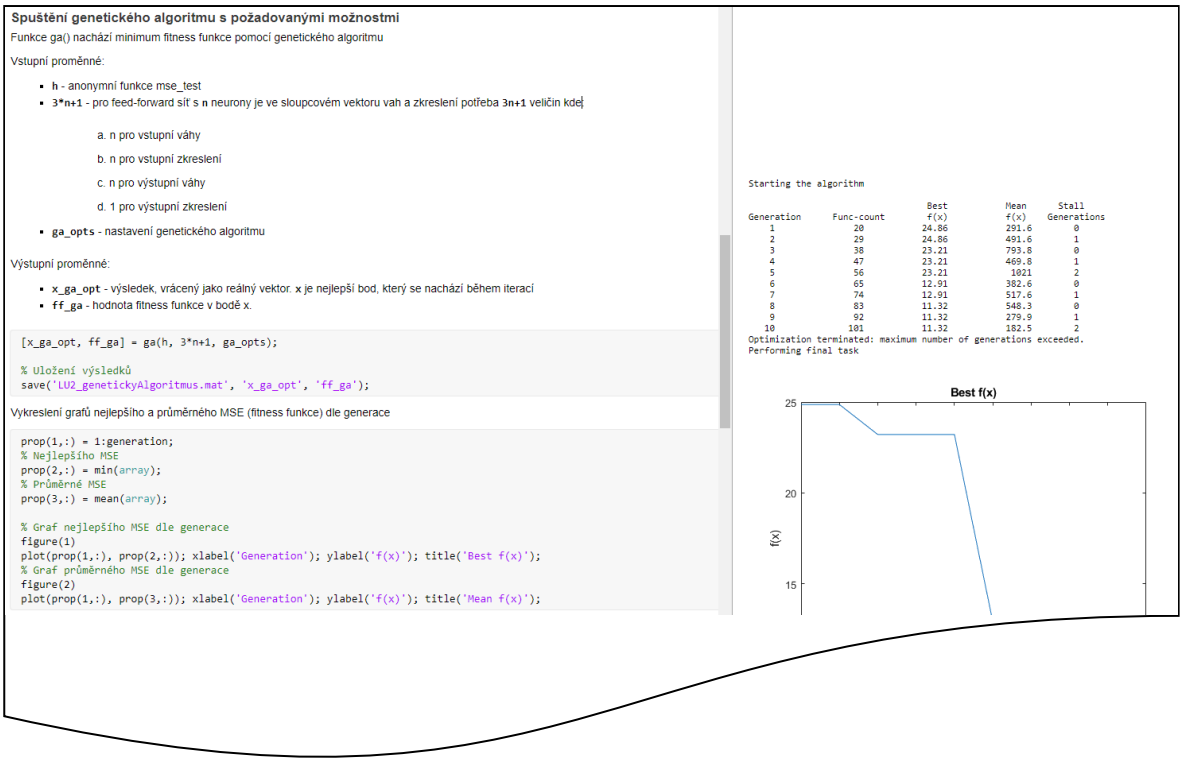
V praktické části diplomové práce se řeší čtyři hlavní klasifikační úkoly, ze kterých následně vznikly laboratorní úlohy pro studenty předmětu Aplikovaná umělá inteligence. První úlohou je klasifikace dat za pomocí základního perceptronu. Pro seznámení studentů s principem učení percepronu jsou vytvořeny dvě metody učení. První, výchozí, pomocí implementovaných funkcí a druhá na základě vytvoření vlastní funkce pro naučení perceptronu.

V další úloze se studenti obeznámí s optimalizací neuronových sítí genetickým algoritmem. Prostřednictvím několika různých nastavení genetického algoritmu jako je počet generací nebo velikost populace, se dozví, jaký mají vliv na výslednou optimalizaci.

Třetí laboratorní úloha se zabývá klasifikací audio nahrávek, tedy jednodimenzionálních signálů konvoluční neuronovou sítí. Studenti se seznámí s tím, jak vytvořit v prostředí MATLAB konvoluční neuronovou síť dle zadaní. Dále budou upravovat jednotlivé nastavení jako je velikost trénovací množiny a počet epoch a jak to ovlivňuje výslednou přesnost klasifikace neuronové sítě.

Poslední laboratorní úloha je zaměřena na klasifikaci obrazových dat, přesněji obličejů s rouškou a bez roušky, za pomoci předučené hluboké konvoluční neuronové síti GoogLeNet. Cílem je obeznámit se s prací s předučenými sítěmi a pokud se změní nastavení jako je počet epoch, jaký to bude mít vliv na míru přesnosti.

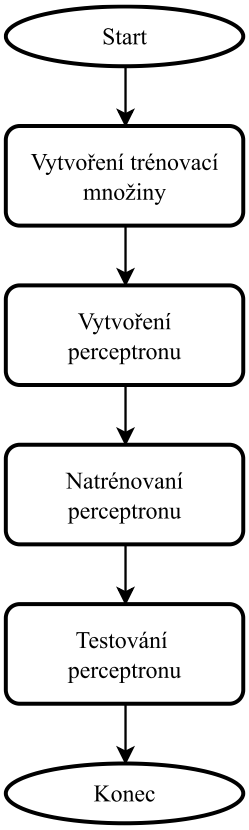
Ke všem laboratorním úlohám byl vytvořen návod v prostředí MATLAB formou Live Scriptu, kde jsou krok po kroku popsané postupy a způsob získání výsledků. Na Obr. 29 je ukázka z Live Scriptu, ve které je popsaná funkce, dále je funkce volaná v kódu, přičemž se ve vedlejším sloupci zobrazují výsledky.



Obr. 29 Ukázka Live Scriptu z druhé laboratorní úlohy [zdroj vlastní]

6 Klasifikace lineárně oddělitelných dat Perceptronem

Tato část diplomové práce se zabývá klasifikací lineárně oddělitelných dat perceptronem, tento postup výpočtu je znázorněn na Obr. 30. Pro učení percepronem jsou vytvořeny dvě metody. První metoda využívá implementovaných funkcí v prostředí MATLAB. Druhá metoda učení je založena na vytvoření vlastní funkce pro výpočet vah neboli natrénovaní percepronu. Po natrénovaní, je preceptron testován klasifikací náhodně vygenerovaných bodů.



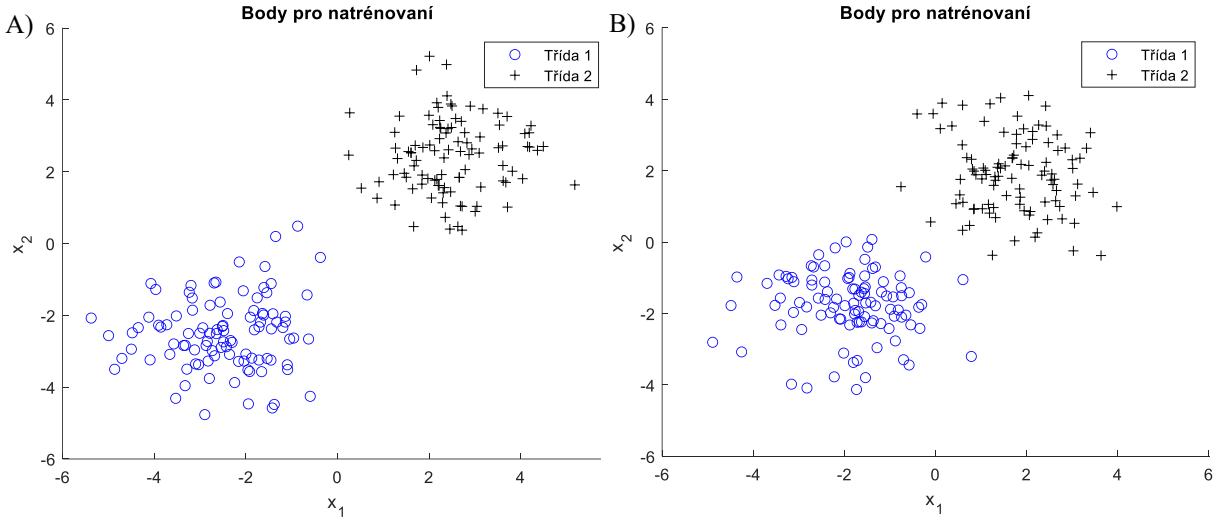
Obr. 30 Vývojový diagram klasifikace perceptronu [zdroj vlastní]

6.1 Trénovací množina perceptronu

Vstup trénovací množiny pro natrénovaní perceptronu je složen z 200 bodů x. Vstupní body x se vytvořili vygenerováním souřadnic (x_1, x_2) v normálním rozdělení. Aby byly body x lineárně oddělitelné, je přidán offset $\mathrm{OFS} = 1{,}8$. Dle offsetu jsou body rozděleny do dvou shluků po 100 bodech x podle vztahu (16). Vstupním bodům jsou přiřazeny výstupy y, které je rozdělují do třídy 1 a třídy 2.

$$\begin{aligned} \text{třída } 1 &= (x_1 - OFS, x_2 - OFS) \\ \text{třída } 2 &= (x_1 + OFS, x_2 + OFS) \end{aligned} \tag{16}$$

Trénovací množiny pro obě metody trénovaní percepronu jsou vygenerované stejným postupem, avšak každá obsahuje různá data viz Obr. 31.



Obr. 31 Trénovací množiny, A) pro první metodu, B) pro druhou metodu [zdroj vlastní]

6.2 Implementace perceptronu MATLAB

Na začátku této metody se vytvoří neuronová síť typu preceptron implementovanými funkcemi v MATLAB [50]. Dále se síť natrénuje zadáním neuronové sítě a trénovací množiny. Funkce pro trénování v MATLAB se klasifikuje pomocí tvrdého prahování se výstupem $y = \{0, 1\}$, kde hodnota 0 je třída 1 a hodnota 1 je třída 2.

Funkce prvně inicializuje váhy $w = [0, 0]$ a zkreslení $b = 0$. Vynásobením vstupních hodnot x s váhami a přičtením zkreslení zjistí výstupní hodnotu α dle vzorce (17).

$$\alpha = hardlim(w \cdot x + b) = hardlim\left([w_1 \ w_2] \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b\right) \tag{17}$$

Pokud se hodnota α nerovná výstupu y, použije se pravidlo perceptronu k nalezení postupných změn vah a zkreslení na základě chyby podle vztahů (18).

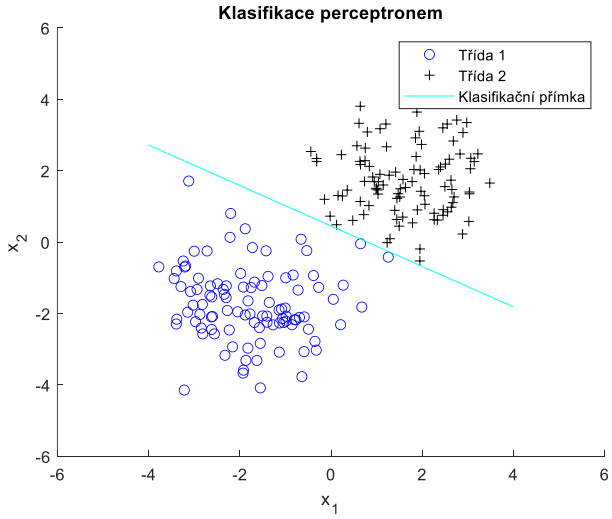
$$\begin{aligned} e &= y - \alpha \\ \Delta w &= e \cdot x \\ \Delta b &= e \end{aligned} \tag{18}$$

Kde e je rozdíl výstupů, Δw je velikost změny vah, Δb je velikost změny zkreslení. Tyto hodnoty se použijí k výpočtu nových vah a zkreslení pomocí pravidel (19) aktualizace perceptronu.

$$\begin{aligned} w_{new} &= w + \Delta w \\ b_{new} &= b + \Delta b \end{aligned} \tag{19}$$

Z výsledných vah se nakonec vypočte klasifikační přímka f (Obr. 35) dle vzorce (20).

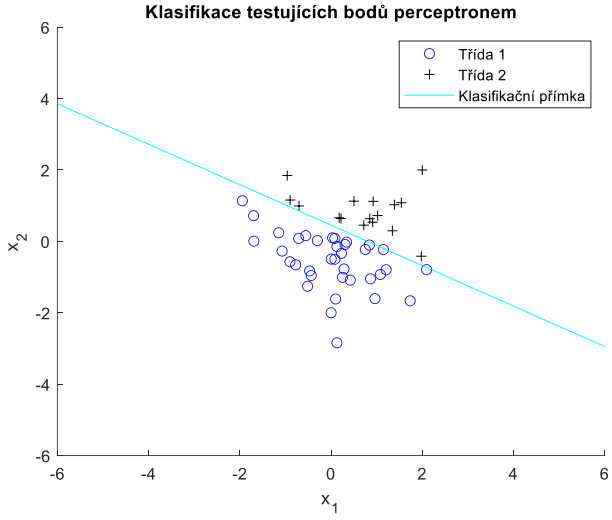
$$f = w_1 x_1 + w_2 x_2 + b \tag{20}$$



Obr. 32 Trénovací množiny s klasifikační přímkou, první metoda [zdroj vlastní]

6.2.1 Testování klasifikace

Pro testování implementované klasifikace perceptronem je použito 50 vstupních bodů. Body jsou vygenerovány z normálního rozdělení. Testovací body jsou spolu s výslednou klasifikační přímkou graficky zobrazeny na Obr. 33, kde lze vidět, jak klasifikátor jednotlivé body zatřídil.

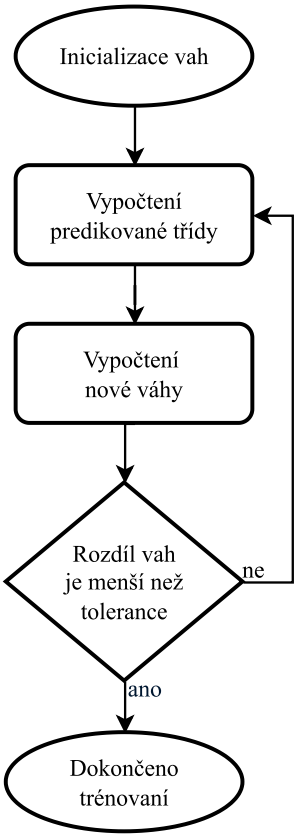


Obr. 33 Testovací body s klasifikační přímkou, první metoda [zdroj vlastní]

6.3 Vlastní funkce učení perceptronu

V této metodě klasifikace dat je vytvořena vlastní funkce, která vypočítává hodnoty vah při učení perceptronu (Obr. 34). Na začátku se vytvoří počáteční váhy, s nimi se vypočte predikovaný výstup. Pomocí predikovaného výstupu se vypočtou nové hodnoty vah.

Pro binární klasifikace je použita aktivační funkce signum, která vrací hodnoty $\{-1, 1\}$. Výstupem trénovací množiny jsou hodnoty $y = \{-1,1\}$, kde -1 je třída 1 a 1 je třída 2.



Obr. 34 Vývojový diagram vlastní funkce pro natrénování perceptronu [zdroj vlastní]

Prvně jsou inicializovány váhy na hodnoty $w = \{-1, 1\}$. Cyklus trénovaní započne výpočtem predikovaného výstupu $\hat{y}$ vynásobením matice vah w a vstupního bodu x dle vzorce (21).

$$\hat{y} = sign(w \cdot x) = sign\left([w_1 \ w_2] \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) \tag{21}$$

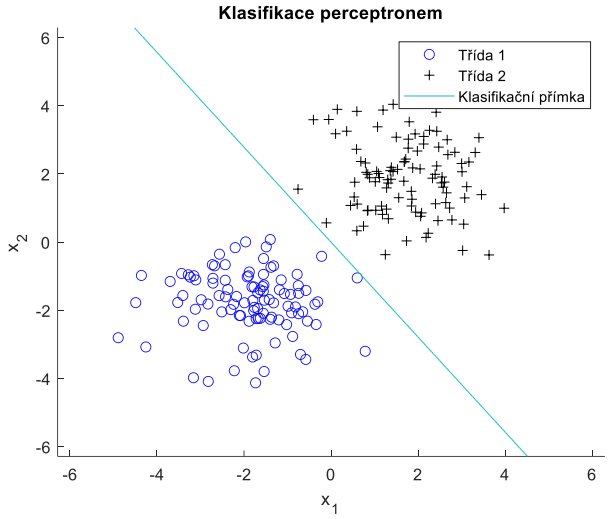
V dalším bodě procesu se vypočtou nové váhy. Ke stávajícím vahám se připočte míra učení η, která se pohybuje v rozsahu $(0, 1)$ (vyšší hodnoty způsobují větší proměnlivost změn vah). Míra je pak vynásobená rozdílem pravé třídy a predikované třídy a vstupem x podle vzorce (22). V práci je použita míra učení s hodnotou $\eta = 0{,}7$.

$$w_{new} = w + \eta \cdot (y - \hat{y}) \cdot x \tag{22}$$

Cyklus trénování se zakončí je-li rozdíl vah z nynějšího a předcházejícího cyklu menší než požadovaná tolerance. Pokud je podmínka splněna vrátí hodnoty vah $w = [w_1, w_2]$. Tolerance ovlivňuje výslednou přesnost klasifikace, kdy při nižší hodnota tolerance bude mít lepší výsledky klasifikace, ale prodlužuje délku trvání výpočtů, v práci je tolerance nastavena na $tol = 0{,}5$.

Z výsledných vah se nakonec vypočte klasifikační přímka f (Obr. 35) dle vzorce (23).

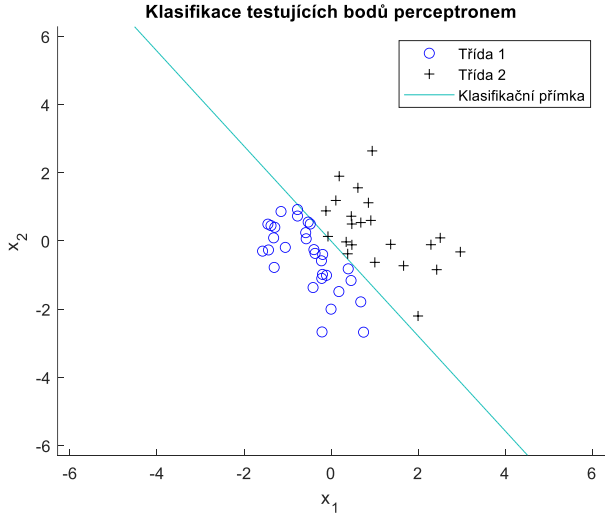
$$f = w_1 x_1 + w_2 x_2 \tag{23}$$



Obr. 35 Trénovací množiny s klasifikační přímkou, druhá metoda [zdroj vlastní]

6.3.1 Testování klasifikace perceptronem

Pro testování klasifikace perceptronem je vygenerováno 50 vstupních body z normálního rozdělení. Vypočet výstupní hodnoty y je proveden pro každý bod x a to vynásobením vah w s bodem x viz rovnice (21). Testovací body jsou spolu s klasifikační přímkou graficky zobrazeny na Obr. 36, lze vidět, jak jsou body zatřízeny pomocí naučeného klasifikátoru.



Obr. 36 Testovací body s klasifikační přímkou, druhá metoda [zdroj vlastní]

6.4 Vytvoření laboratorní úlohy klasifikace perceptronem

Dle popisu v předchozích kapitolách byla k této úloze vytvořena laboratorní práce pro klasifikaci lineárně oddělitelných bodů perceptronem. Zadání laboratorní úlohy obsahuje definici obou metod pro vytvoření perceptronu s následným teoretickým rozborem. Prostřednictvím zadání a návodu ve formě Live Scriptu se studenti detailně seznámí se základní klasifikací perceptronem v prostředí MATLAB. Laboratorní práce se nachází v elektronické příloze.

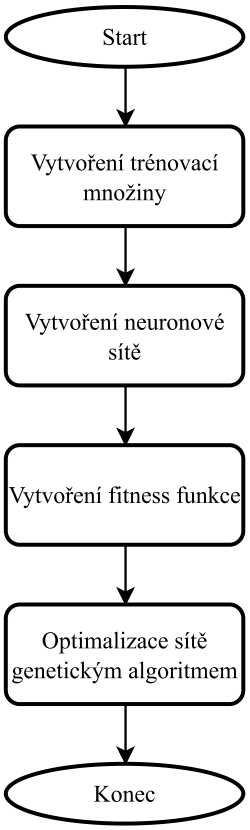
7 Optimalizace neuronové sítě genetickým algoritmem

V této kapitole práce zkoumá genetický algoritmus za účelem optimalizace vah dopředné (feedforward net) neuronové sítě. Jako fitness funkce je aplikován výpočet střední kvadratické chyby. Prvně se vytvoří trénovací množina a neuronová síť, dále se připraví fitness funkce a nastaví se genetický algoritmus. Nakonec se síť zoptimalizuje a urči se její úspěšnost (Obr. 37).

Genetický algoritmus funguje na principu evolučního algoritmu. Obsahuje chromozomy, které nesou informaci, v tomto případě o nastavení vah v neuronové síti. Pomocí těchto vah se zjistí nová predikovaná třída a porovná se s pravou třídou. V průběhu optimalizace se místo ztrátové funkce počítá s tzv. fitness funkcí. Ta ukazuje úspešnost optimalizace neboli přesnost klasifikace při tréninku. Dle funkce pro výpočet fitness funkce se hledá buďto minimální nebo maximální hodnota.

Princip genetických algoritmů spočívá ve vypočtení mnohá fitness funkcí. Ta se vypočte pro každého jedince v generaci, kde jedinec se skládá z chromozomů (nastavení vah) a generace se skládá z oněch jedinců (populace).

Hlavním cílem je seznámení se s možnostmi nastavení genetického algoritmu jako je množství generací, velikost populace nebo počet neuronů v neuronové síti a vliv těchto změn na výslednou fitness funkci genetického algoritmu. Pro tento účel je vytvořeno několik konfigurací s různými parametry.



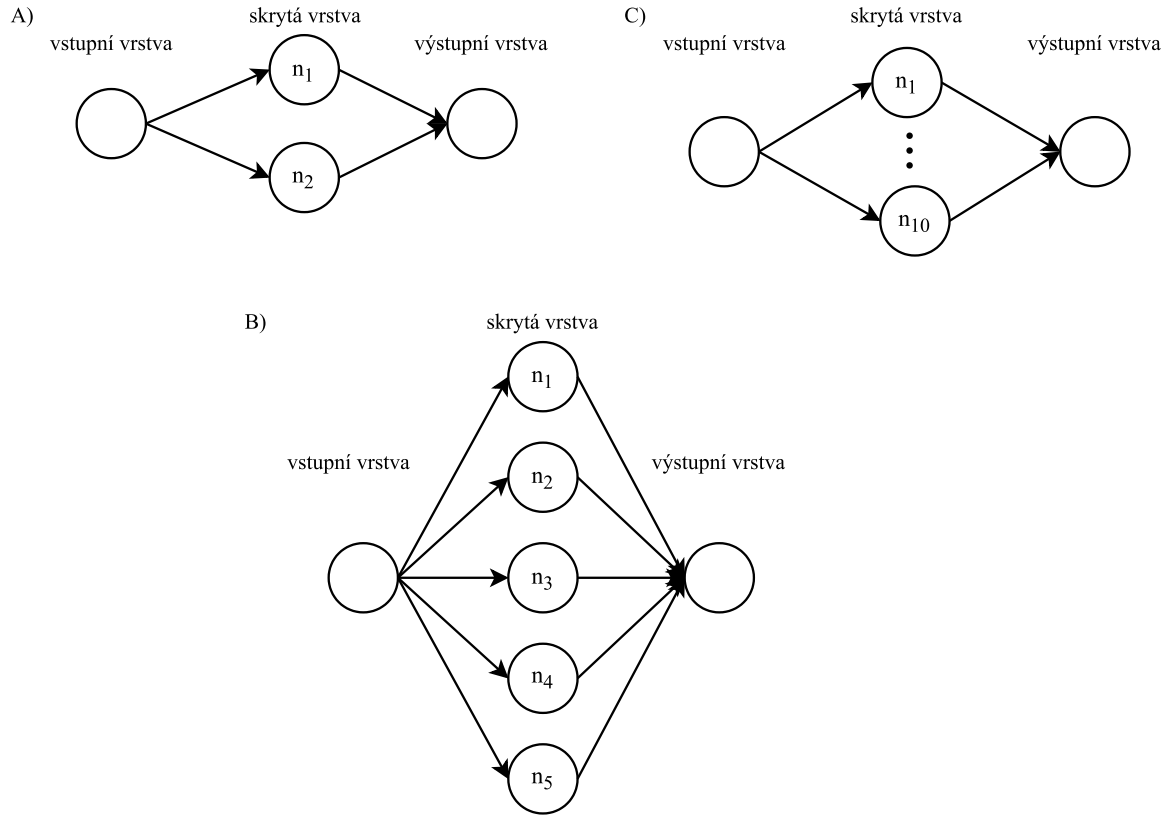
Obr. 37 Diagram vývoje optimalizace vah genetickým algoritmem [zdroj vlastní]

7.1 Trénovací množina neuronové sítě

Vstupní trénovací množina se skládá z množiny hodnot $x = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Výsledné třídy do kterých neuronová síť klasifikuje se odvozují z funkce $y = \cos(x^2)$. Cílem optimalizace je nalézt takové hodnoty vah neuronové sítě, jenž se nejvíce podobají právě této funkci.

7.2 Vytvoření neuronové sítě

Pro klasifikaci je použita základní jednoduchá dopředná neuronová síť. Obsahuje tři vrstvy, vstupní vrstvu, skrytou vrstvu a výstupní vrstvu. Ve skryté vrstvě se pro různé konfigurace mění počet neuronů a to na 2, 5 a 10.



Obr. 38 Struktura dopředné neuronové sítě
pro A) 2 neurony, B) 5 neuronů, C) 10 neuronů [zdroj vlastní]

7.3 Nastavení genetických algoritmů

U genetických algoritmů je potřeba nastavit počet generací, tedy kolikrát se vypočte fitness funkce a jakmile se dosáhne maximálního počtu algoritmus se dokončí. Zkoumaný počet generací je 50, 100 a 200.

Dalším důležitým nastavením je různá velikost populace neboli kolik chromozomů, což je nosič informací, se nachází v jedné generaci, a tedy kolikrát se vypočte fitness funkce v průběhu jedné generace. Sledované velikosti populace jsou 100, 500 a 1000.

Dále je funkční tolerance nastavena na hodnotu $tol = 10^{-8}$. Tolerance patří mezi parametry, které ovlivňují ukončení algoritmu. Pro možnost výpočtu fitness funkce u všech generací je tolerance

nastavena velmi nízkou hodnotu. K přehlednému zobrazení všech konfigurací a jejich parametrů slouží Tab. 2.

Tab. 2 Paramenty nastavení pro všechny konfigurace optimalizace vah [zdroj vlastní]

Pořadí konfigurací	Počet generací	Velikost populace	Počet neuronů
Konf 1	50	100	2
Konf 2	50	100	5
Konf 3	50	100	10
Konf 4	50	500	2
Konf 5	50	500	5
Konf 6	50	500	10
Konf 7	50	1000	2
Konf 8	50	1000	5
Konf 9	50	1000	10
Konf 10	100	100	2
Konf 11	100	100	5
Konf 12	100	100	10
Konf 13	100	500	2
Konf 14	100	500	5
Konf 15	100	500	10
Konf 16	100	1000	2
Konf 17	100	1000	5
Konf 18	100	1000	10
Konf 19	200	100	2
Konf 20	200	100	5
Konf 21	200	100	10
Konf 22	200	500	2
Konf 23	200	500	5
Konf 24	200	500	10
Konf 25	200	1000	2
Konf 26	200	1000	5
Konf 27	200	1000	10

7.3.1 Fitness funkce

Fitness funkce je funkce, která je slouží k optimalizaci neuronové sítě a úkolem je najít její minimum. Je to nejdůležitější parametr genetického algoritmu a pro účely práce je zvolena fitness funkce na základě výpočtu střední kvadratické chyby (mean square error, MSE).

MSE se vypočítá mezi predikovanou třídou a pravou třídou podle vztahu (24), kde n je počet hodnot, y_i je predikovaná třída a $\hat{y}_i$ je pravá třída. Čím blíže je hodnota MSE nule, tím je menší neshoda mezi třídami a fitness funkce je lepší.

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{24}$$

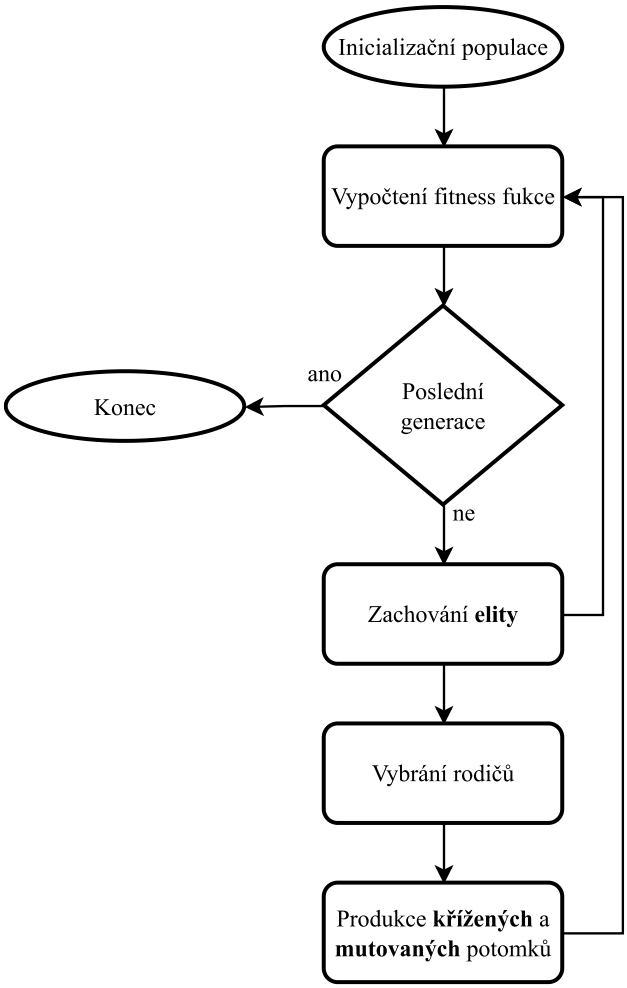
7.4 Průběh optimalizace genetickým algoritmem

Každá optimalizace genetickým algoritmem začne inicializační generací, která obsahuje náhodné hodnoty parametrů v chromozomech. Každému chromozomu odpovídá určité nastavení váhy v neuronové síti. Podle nastavení vah je zjištěna predikovaná třída a dle toho vypočtena hodnota MSE neboli hodnota fitness funkce [51].

V každém kroku genetický algoritmus použije aktuální populaci k vytvoření potomků, kteří tvoří další generaci. Algoritmus (Obr. 39) vybere skupinu jedinců (chromozomů) v aktuální populaci, tzv. rodiče, kteří přispívají svými geny (hodnotami svých vektorů) svým dětem. Algoritmus obvykle vybírá jako rodiče jedince, kteří mají lepší hodnoty fitness funkce [51].

Genetický algoritmus vytváří tři základní typy potomků pro další generaci [51]:

- **Elitní jedinci:** chromozomy v aktuální generaci s nejlepšími hodnotami fitness funkce. Tito jedinci automaticky přežívají do další generace.
- **Křížení jedinci:** vznikají kombinací vektorů dvojice rodičů.
- **Mutační potomci:** vznikají zaváděním náhodných změn neboli mutací do jednoho rodiče.



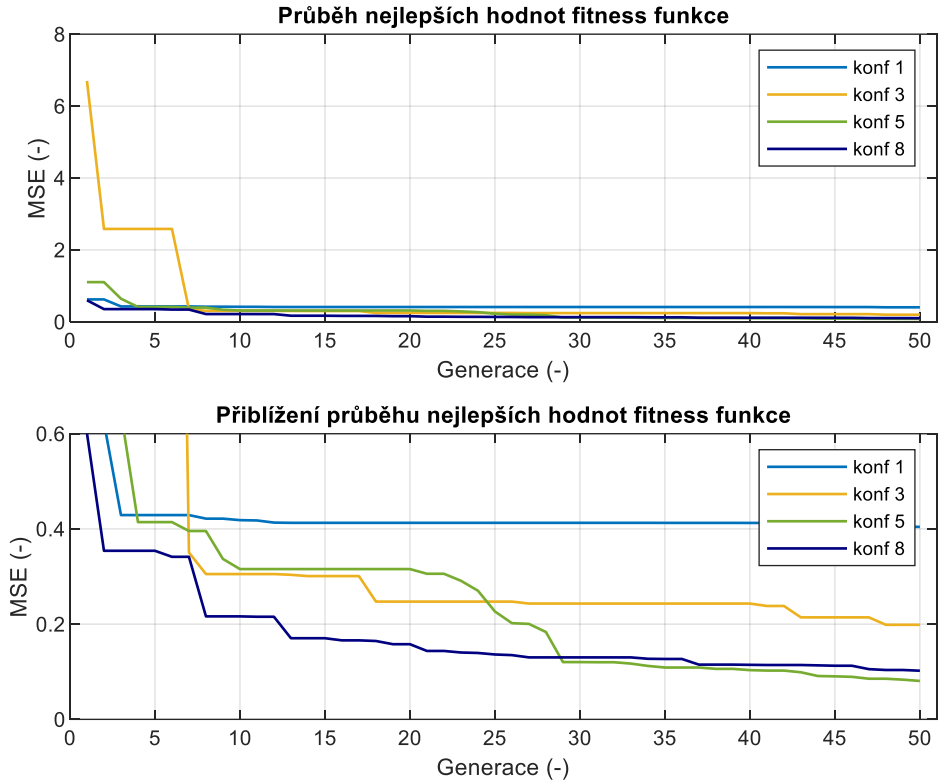
Obr. 39 Vývojový diagram genetického algoritmu [zdroj vlastní]

Na grafech níže je zobrazen průběh nejlepších hodnot fitness funkce vzhledem ke generaci. Přičemž vždy horní graf obsahuje celý průběh konfigurací a dolní graf pro lepší viditelnost jednotlivých průběhů má na ose *y* nastaven limit na hodnoty 0 až 0,6.

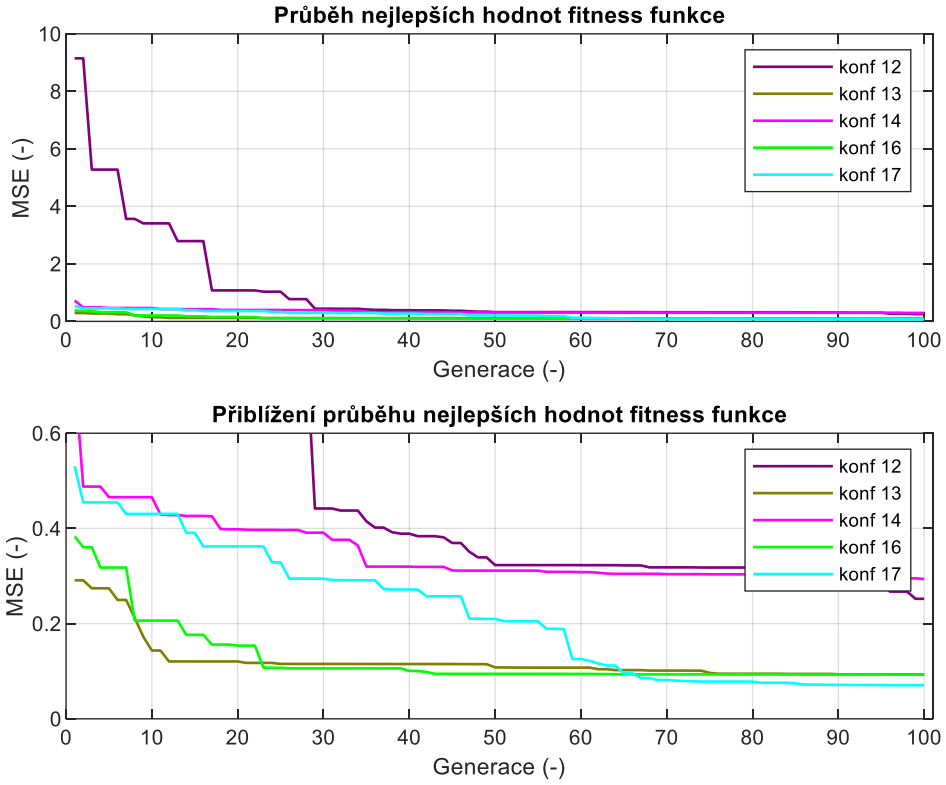
Na Obr. 40 jsou konfigurace s 50 generacemi, vybrány jsou dva nejlepší (5 a 8) a dva nejhorší (1 a 3) průběhy. Všechny průběhy jsou pro jednotlivé konfigurace s 50 generacemi jsou pak zobrazeny v grafech pod označením Příloha 1.

Na Obr. 41 jsou konfigurace se 100 generacemi a vybrány jsou tři nejlepší konfigurace (13, 16 a 17) a dva nejhorší (12 a 14) průběhy. Všechny průběhy jsou pro jednotlivé konfigurace se 100 generacemi jsou pak zobrazeny v grafech pod označením Příloha 2.

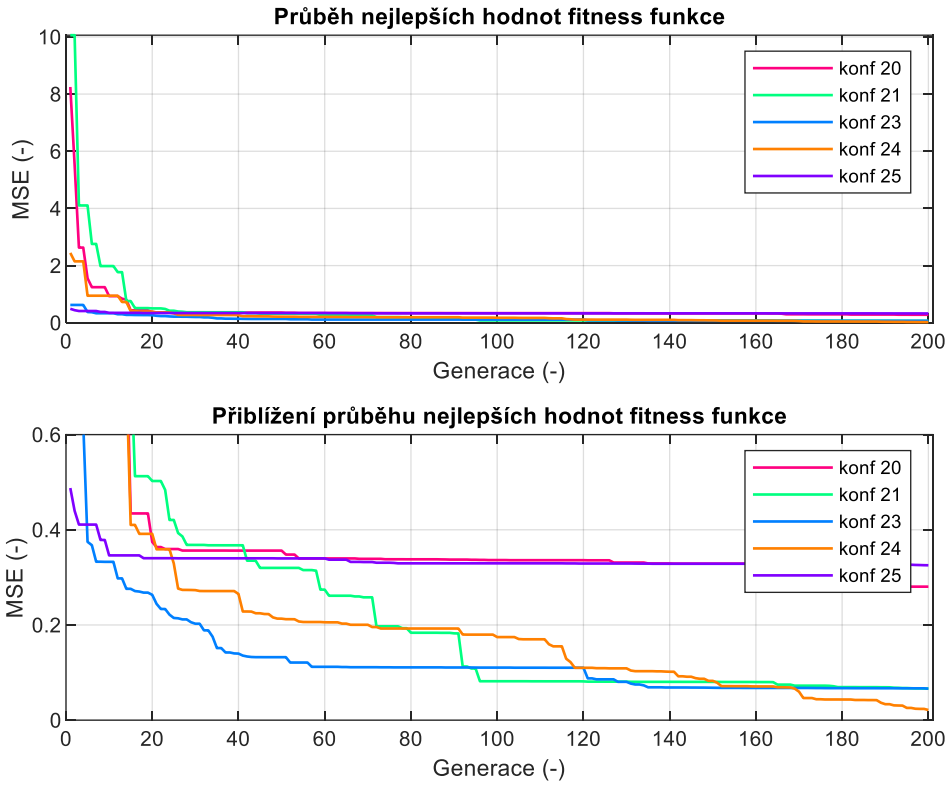
Na Obr. 42 jsou konfigurace s 200 generacemi a vybraný jsou tři nejlepší konfigurace (21, 23 a 24) a dva nejhorší (20 a 25) průběhy. Všechny průběhy jsou pro jednotlivé konfigurace s 200 generacemi jsou pak zobrazeny v grafech pod označením Příloha 3.



Obr. 40 Průběhy nejlepších hodnot fitness funkce
pro vybrané konfigurace s 50 generacemi [zdroj vlastní]



Obr. 41 Průběhy nejlepších hodnot fitness funkce
pro vybrané konfigurace se 100 generacemi [zdroj vlastní]



Obr. 42 Průběhy nejlepších hodnot fitness funkce
pro vybrané konfigurace s 200 generacemi [zdroj vlastní]

7.5 Vyhodnocení optimalizace genetickým algoritmem

Vyhodnocení optimalizace klasifikace neuronovou sítí za použití genetického algoritmu je zhotoveno pro každou konfiguraci. V Tab. 3 jsou nejlepší a průměrné hodnoty fitness funkcí jednotlivých konfigurací a dále je vypsána časová náročnost výpočtu. Pro skupiny rozdělené dle počtu generací, jsou vybrány nejlepší a nejhorší výsledky.

Ve skupině s 50 generacemi (konf 1 až konf 9) má nejnižší nejlepší hodnotu fitness funkce, a tedy dosáhla nejlepší klasifikace, konfigurace 5. Naopak nejvyšší hodnoty dosáhla konfigurace 1. Nejnižšího průměru nabyla konfigurace 7, přesto má vyšší hodnotu fitness funkce než konfigurace 3, která má největší průměr.

Ve skupině se 100 generacemi (konf 10 až konf 18) má nejnižší nejlepší hodnotu fitness funkce, a tedy dosáhla nejlepší klasifikace, konfigurace 17. Naopak nejvyšší hodnoty dosáhla konfigurace 14. Nejnižší (nejlepší) průměr má konfigurace 16, která skončila jako druhá nejlepší ve skupině a nejhorší průměrnou hodnotu fitness funkce má konfigurace 10.

Ve skupině s 200 generacemi (konf 19 až konf 27) má nejnižší nejlepší hodnotu fitness funkce, a tedy dosáhla nejlepší klasifikace, konfigurace 24. Naopak nejvyšší hodnotu dosáhla konfigurace 25. Největší průměr má konfigurace 20, přičemž zároveň její hodnota nejlepší fitness funkce dopadla jako druhá nejhorší. Nejnižší průměr má konfigurace 23, která skončila jako druhá nejlepší konfigurace ve skupině i celkově.

Z celkového hlediska skončila konfigurace 24 nejlépe s nejlepší klasifikaci a nejhůře dopadla konfigurace 1 s nejvyšší hodnotou fitness funkce. Lze zpozorovat trend, že s vyšším počtem generací se celkově snižují hodnoty nejlepší fitness funkce i průměrné hodnoty fitness funkce.

Pokud by se konfigurace hodnotily z hlediska výpočetní náročnosti, lze vidět, že čím větší obsahuje konfigurace populaci a čím více obsahuje generací tím déle trvá vypočítat výsledek. Naproti tomu počet neuronů má nepatrný vliv na dobu trvání výpočtu.

Jako nejoptimálnější z hlediska nastavení genetických algoritmů je konfigurace 24, ovšem pokud se vezme v úvahu i výpočetní náročnost, tak je nejlepším nastavením konfigurace 5, která má sice se trochu horší klasifikace, ale doba trvání je výrazně nižší.

Tab. 3 Výsledné hodnoty fitness funkce [zdroj vlastní]

Pořadí konfigurací	Nejlepší hodnota fitness funkce	Průměrná hodnota fitness funkce	Výpočetní náročnost (min)
Konf 1	**0,404**	4,198	00:44
Konf 2	0,344	2,870	00:43
Konf 3	0,199	**8,231**	00:43
Konf 4	0,312	1,437	03:33
Konf 5	**0,081**	2,728	03:33
Konf 6	0,207	6,141	03:33
Konf 7	0,237	**0,844**	07:24
Konf 8	0,102	1,795	07:06
Konf 9	0,092	4,381	07:07
Konf 10	0,199	**1,836**	01:27
Konf 11	0,149	0,513	01:27
Konf 12	0,252	1,768	01:27
Konf 13	0,093	0,345	07:12
Konf 14	**0,294**	1,154	07:13
Konf 15	0,210	1,468	07:12
Konf 16	0,093	**0,255**	14:23
Konf 17	**0,071**	0,581	14:22
Konf 18	0,111	1,762	14:25
Konf 19	0,090	0,142	02:53
Konf 20	0,281	**0,597**	02:54
Konf 21	0,066	0,274	02:54
Konf 22	0,094	0,154	14:28
Konf 23	0,066	**0,149**	14:27
Konf 24	**0,021**	0,309	14:29
Konf 25	**0,325**	0,462	28:54
Konf 26	0,074	0,217	28:58
Konf 27	0,203	0,500	29:01

7.6 Vytvoření laboratorní úlohy optimalizace genetickým algoritmem

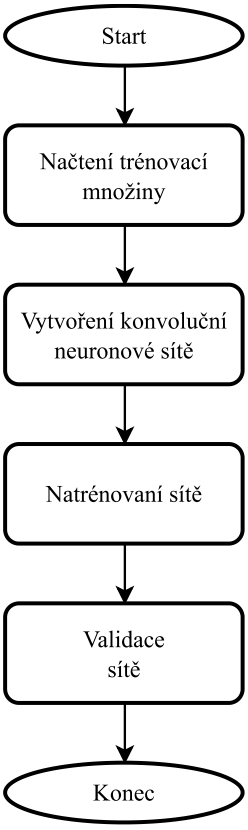
Nakonec je k této úloze je vytvořena laboratorní práce pro optimalizace neuronové sítě genetickým algoritmem. Zadání laboratorní úlohy a návod formou Live Skripu pomůže studentům se zpracování této úlohy. Budou obeznámení o tom, jak se optimalizuje síť genetickými algoritmy v prostředí MATLAB a jaký vliv má na výsledek změna několika parametrů jako je počet generací nebo počet neuronů v neuronové síti. Laboratorní práce se nachází v příloze.

8 Klasifikace akustických signálů konvoluční neuronovou sítí

Další úloha klasifikace strojovým učením se zabývá klasifikací jednodimenzionálních signálů akustických konvoluční neuronovou sítí (dále jen CNN). Princip učení a následné klasifikace u CNN spočívá v nalezení příznaků v konvolučních vrstvách a následnou klasifikací ve výstupních vrstvách jako je plně propojená vrstva nebo klasifikační vrstva. CNN může obsahovat od jednotek až po stovky vrstev, z nichž se učí rozpoznávat různé příznaky signálu. Na každý tréninkový signál se aplikují filtry s různou velikostí a výstup každé konvoluční vrstvy se použije jako vstup do další vrstvy [52].

Mezi hlavní vrstvy CNN patří **konvoluční vrstva**, která obsahuje konvoluční filtry a každý z nich aktivuje určité příznaky signálu. Dále je to **rektifikovaná lineární jednotka** (ReLu), která umožnuje rychlejší a efektivnější trénink. Provádí tzv. aktivaci tím, že záporné hodnoty převádí na nulu a pouze zachovává kladné hodnoty tedy do další vrstvy posílá aktivované příznaky. Další podstatná vrstva je **sdružovací vrstva**, která provádí nelineární downsampling a tím snižuje počet parametrů, které se sít musí naučit. Jako poslední vrstva je v CNN **klasifikační vrstva** nebo **vrstva softmax**, které klasifikují data z naučených příznaků z mnoha předchozích vrstev [52].

Pro cíle úlohy je síť natrénovaná při odlišně velkých trénovacích množinách a při několika nastaveních délky trénovaní. Jednotlivé konfigurace jsou validovány na validačních datech a je vyhodnocena přesnost klasifikace sítě. Základní myšlenka vytváření algoritmu pro klasifikaci signálů konvoluční neuronovou sítí je na Obr. 43.



Obr. 43 Diagram vývoje CNN [zdroj vlastní]

8.1 Akustická data

Akustické signály jsou získány z veřejné databáze Free Spoken Digit Dataset (FSDD) [53]. Sada jednoduchých zvukových dat sestávající z nahrávek mluvených cifer 0 až 9 v souborech wav s frekvencí 8 kHz. Nahrávky jsou ořezány, tak aby na jejich začátku a konci byl minimální úsek ticha.

Databáze obsahuje 3000 zvukových nahrávek od šesti řečníků. Každý řečník namluvil všech deset číslovek, tedy 50 opakovaní každé číslovky. Nahrávky jsou v anglickém jazyce.

Při testování jednotlivých konfigurací sítě se mění velikost trénovací a validační množiny. Obě množiny obsahují vždy stejný celkový počet signálů, mění se pouze poměr, s jakým jsou rozděleny. Jsou testovány tři různé poměry a to 3:7, kde trénovací množina obsahovala 30 % signálů a validační množina 70 % signálů z celkového počtu, dále to jsou poměry 5:5 a 2:8.

8.2 Architektura CNN

Ke klasifikaci zvukových nahrávek je použita konvoluční neuronová síť, která obsahuje 6 vrstev (10 vrstev i se sdružovacími vrstvami) viz Obr. 44. Význam vrstev v síti [54]:

- **Vstupní vrstva**: vkládá do sítě data a aplikuje normalizaci dat. Při vytváření se určuje vstupní velikost dat, $delkaSignalu = 8192$ a určuje počet kanálů, $kanal = 1$.
- **Konvoluční vrstvy**: vrstva se učí příznaky lokalizované v oblastech. Při vytváření vrstev se určuje velikost oblasti neboli filtru, kde pro první dvě konvoluční vrstvy je velikost filtru $velikostFiltru = 5$ a pro ostatní $velikostFiltru = 3$. Pro každou oblast se v průběhu trénovaní vypočte bodový součin vah a signálu. Sada vah, která se aplikuje na oblast v signálů, se nazývá filtr. Filtr se pohybuje podél signálu a opakuje stejný výpočet pro každou oblast. Jinými slovy, filtr konvoluje signál. Jak se filtr pohybuje podél signálu, používá pro konvoluci stejnou sadu vah a vytváří mapu příznaků. Každá mapa příznaků je výsledkem konvoluce s použitím jiné sady vah. Počet map příznaků je tedy roven počtu filtrů, kde je zpočátku $početFiltru = 12$, a pak se zvyšuje na hodnoty 24 a 48. Délka signálu je po konvoluci stejná, neboť se využívá metoda same padding.
- **Normalizační vrstva**: využívá se k urychlení trénování CNN a snížení citlivosti na inicializaci sítě, běžně je mezi konvoluční vrstvou a nelineární vrstvou, jako je vrstva ReLU.
- **Vrstva rektifikované lineární jednotky (ReLU)**: provádí na každém prvku vstupu prahovou operaci (25).

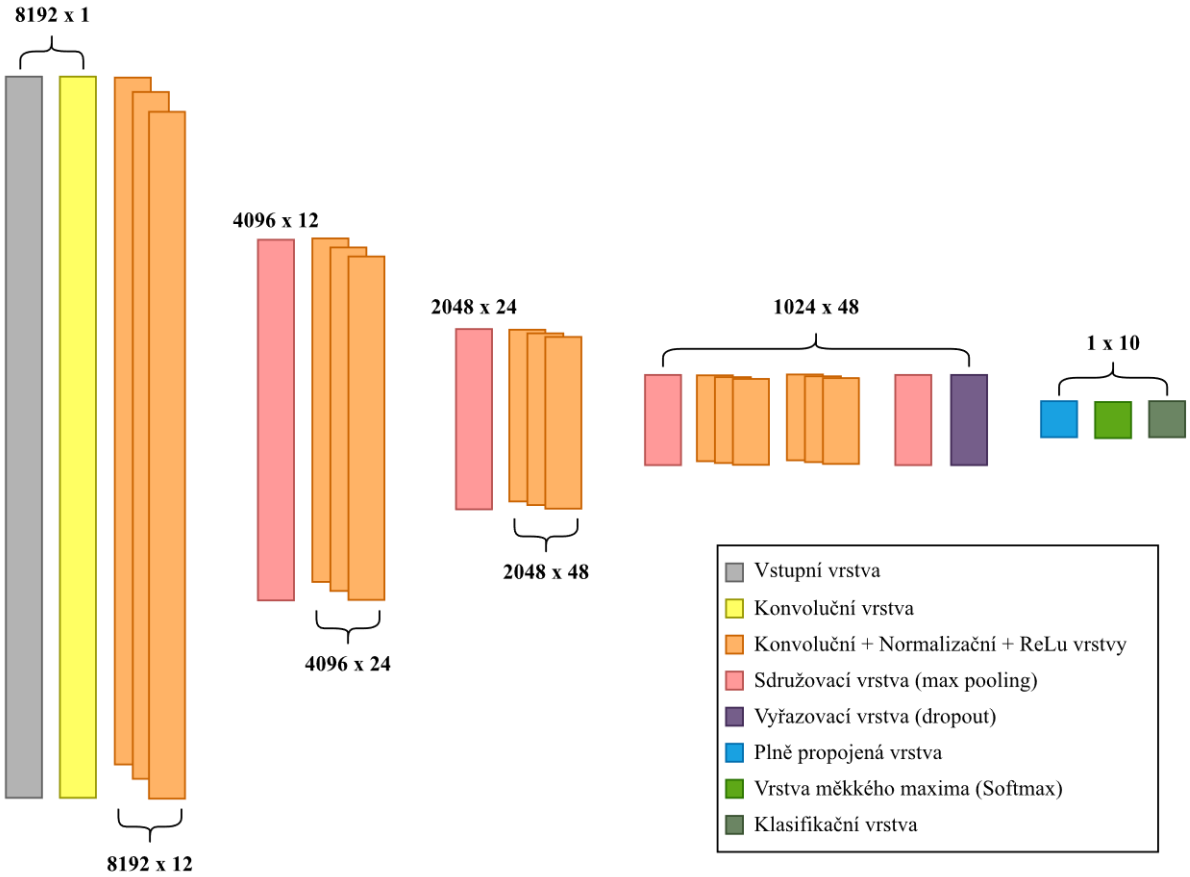
$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{25}$$

- **Sdružovací vrstva (max pooling)**: vykonává převzorkování rozdělením vstupu na oblasti sdružování a následným výpočtem maxima každé oblasti. Snižuje počet spojení do následujících vrstev. Neprovádí žádné učení, ale snižuje počet parametrů, které je třeba naučit v následujících vrstvách. Pomáhá také omezit nadměrné přizpůsobování.
- **Vyřazovací vrstva (dropout)**: náhodně nastaví vstupní prvky na nulu s pravděpodobností $P_d = 0{,}2$. Tato operace účinně mění základní architekturu sítě mezi iteracemi a pomáhá zabránit nadměrnému přizpůsobení sítě.

- **Plně propojená vrstva**: neurony se zde propojují se všemi neurony v předchozí vrstvě. Tato vrstva kombinuje všechny příznaky naučené předchozími vrstvami napříč obrazem a identifikuje větší vzory. Kombinuje příznaky pro klasifikaci signálu. Při vytvoření se určuje počet tříd v datech, $pocetTrid = 10$.
- **Vrstva měkkého maxima (softmax)**: aplikuje softmax funkci na vstup, kde funkce softmax je aktivační funkcí výstupní jednotky.
- **Klasifikační vrstva**: počítá ztrátu křížové entropie pro klasifikační a vážené klasifikační úlohy se vzájemně se vylučujícími třídami. Převezme hodnoty z funkce softmax a přiřadí každý vstup do jedné z K vzájemně se vylučujících tříd pomocí funkce křížové entropie pro kódovací schéma 1 z K (26).

$$loss = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K} w_i t_{ni} \ln y_{ni} \tag{26}$$

Kde N je počet vzorků, K je počet tříd, w_i je váha pro třídu i, t_{ni} je ukazatel, že n-tý vzorek patří do i-té třídy, y_{ni} je výstup pro vzorek n pro třídu i, což je v tomto případě hodnota z funkce softmax, tedy y_{ni} je pravděpodobnost, že síť přiřadí n-tý vstup třídě i.



Obr. 44 Architektura CNN vytvořené v prostředí MATLAB [zdroj vlastní]

8.3 Nastavení CNN

V síti se nastavují parametry jako je optimalizační algoritmus, míra učení, velikost mini dávky, míchání a počet epoch.

Optimalizační algoritmus je nastaven na typ *adam*. Míra učení je $\eta = 10^{-4}$. Ta ovlivňuje vývoj vah, při příliš nízkých hodnotách probíhá trénink zdlouhavě, naopak při příliš vysokých hodnotách dochází ke špatným výsledům. Velikost mini dávky (MiniBatch) je $miniBatch = 50$, což je podmnožina trénovací množiny a využívá se k vyhodnocení gradientu ztrátové funkce a aktualizaci vah. A míchání je nastaveno na zamíchání trénovací množiny pro každou epochu.

Hlavním parametr, který byl zkoumaný pro jednotlivé nastavení sítě, je počet epoch. Epocha je celý průchod tréninkového algoritmu celou tréninkovou množinou. Ty se skládají z iterací, což je jeden krok v algoritmu gradient descent směrem k minimalizaci ztrátové funkce pomocí mini dávky. Počet epoch je nastaven na $epocha = 5, 10, 15, 30$ [54]. K přehlednému zobrazení všech konfigurací a jejich parametrů slouží Tab. 4.

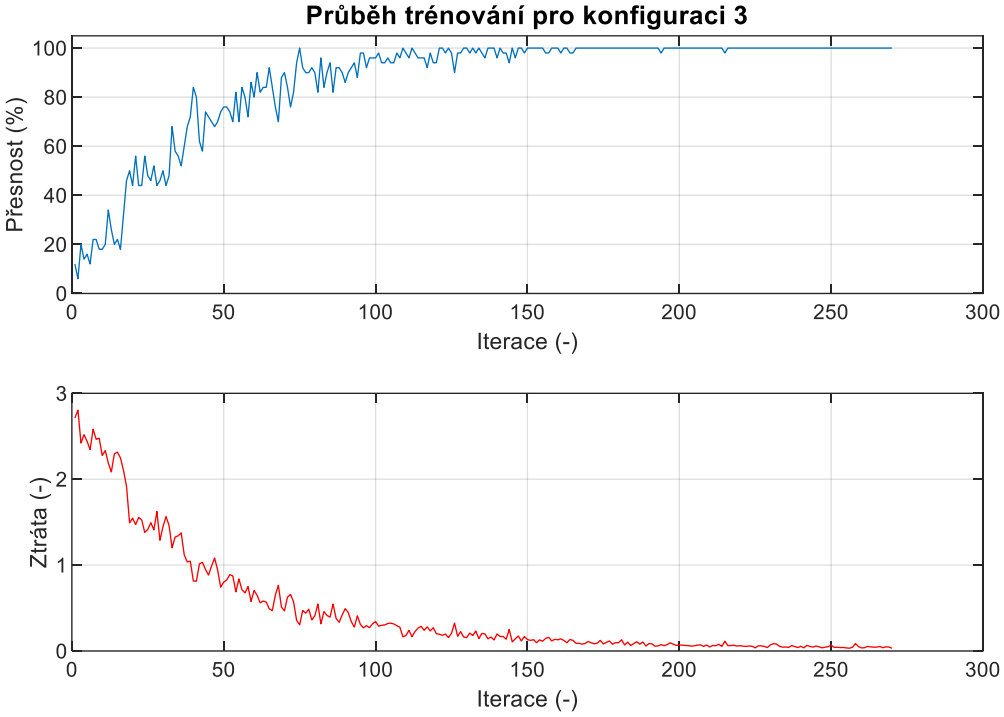
Tab. 4 Paramenty nastavení pro všechny konfigurace CNN [zdroj vlastní]

Pořadí konfigurací	Poměr množin testování / validace	Počet epoch
Konf 1	0,3 / 0,7	5
Konf 2	0,3 / 0,7	10
Konf 3	0,3 / 0,7	15
Konf 4	0,3 / 0,7	30
Konf 5	0,5 / 0,5	5
Konf 6	0,5 / 0,5	10
Konf 7	0,5 / 0,5	15
Konf 8	0,5 / 0,5	30
Konf 9	0,8 / 0,2	5
Konf 10	0,8 / 0,2	10
Konf 11	0,8 / 0,2	15
Konf 12	0,8 / 0,2	30

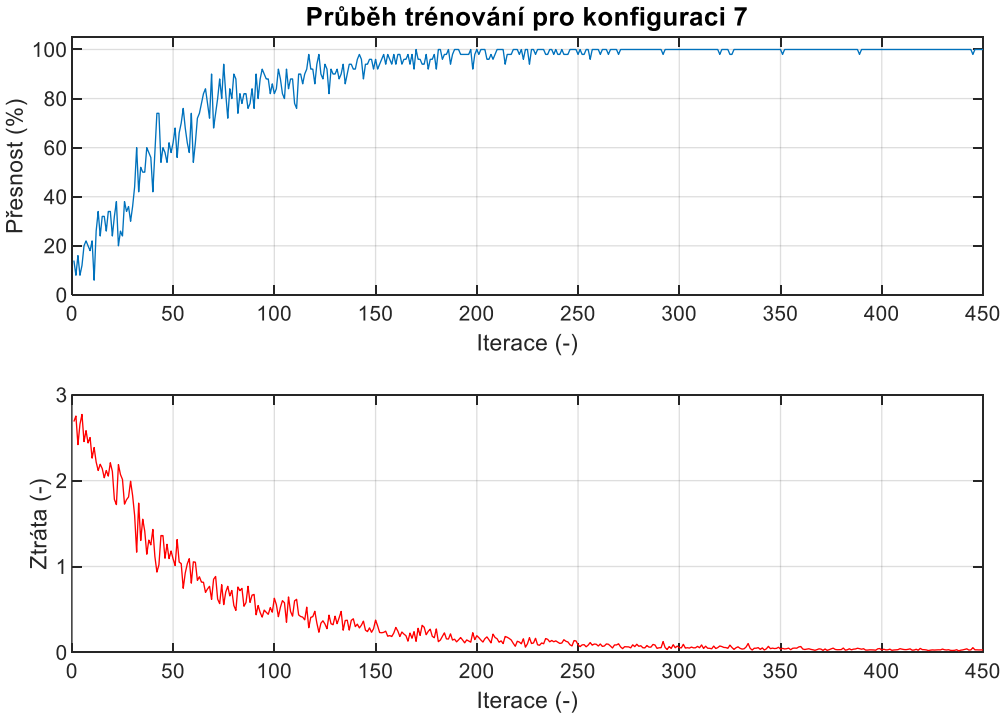
8.4 Průběh trénovaní CNN

Za pomocí aktualizace vah se síť postupně učila a přizpůsobovala vstupním akustickým signálům. Na grafech níže jsou zobrazeny průběhy trénovaní pro tři různé konfigurace. Průběhy jsou zobrazeny pro konfiguraci 3 (Obr. 45), konfiguraci 7 (Obr. 46) a konfiguraci 11 (Obr. 47). Všechny konfigurace mají stejný počet epoch 15 a různý počet poměru trénovací a validační množiny, přesné nastavení je v Tab. 4. Grafy všech průběhů je možné si prohlédnout v Příloha 4.

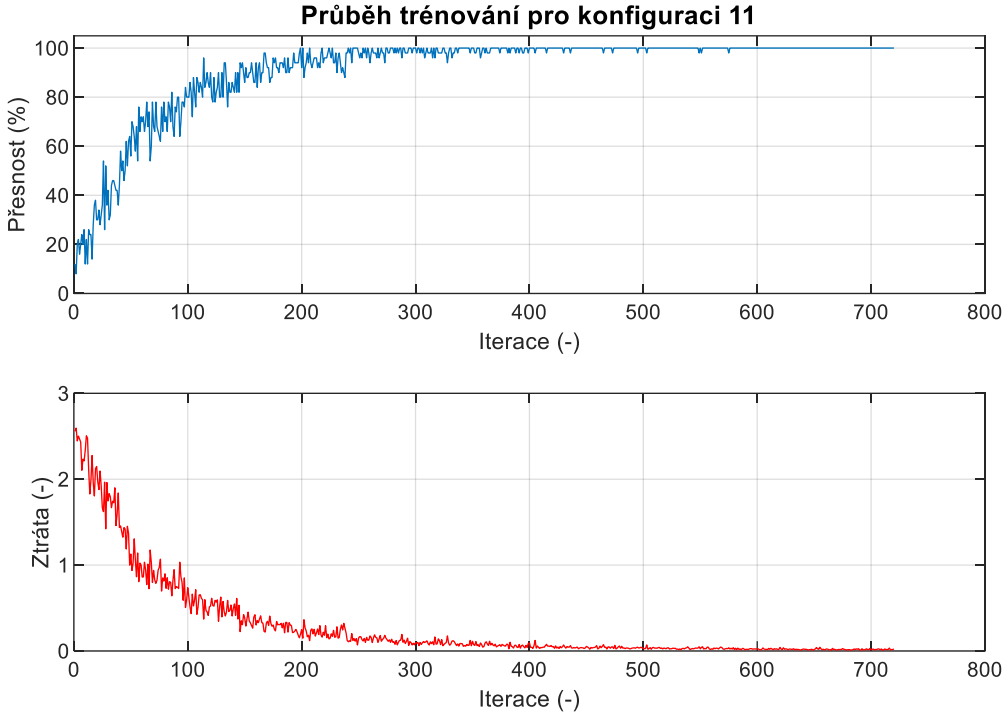
Horní graf průběhu trénovaní představuje přesnost trénovaní neboli jak přesně dokáže klasifikovat data v průběhu tréningu a dolní graf je ztráta neboli ztrátová funkce což je chybovost.



Obr. 45 Průběh trénovaní pro konfiguraci 3, CNN [zdroj vlastní]



Obr. 46 Průběh trénovaní pro konfiguraci 7, CNN [zdroj vlastní]



Obr. 47 Průběh trénovaní pro konfiguraci 11, CNN [zdroj vlastní]

Na grafech lze vidět, že pro stejný počet epoch se zvětšuje množství iterací v jednotlivých epochách s vetší trénovací množinou. Také se dá říci, že při 15 epochách, je přesnost klasifikace trénovací množiny na konci trénovaní okolo 100 %.

8.5 Validace CNN

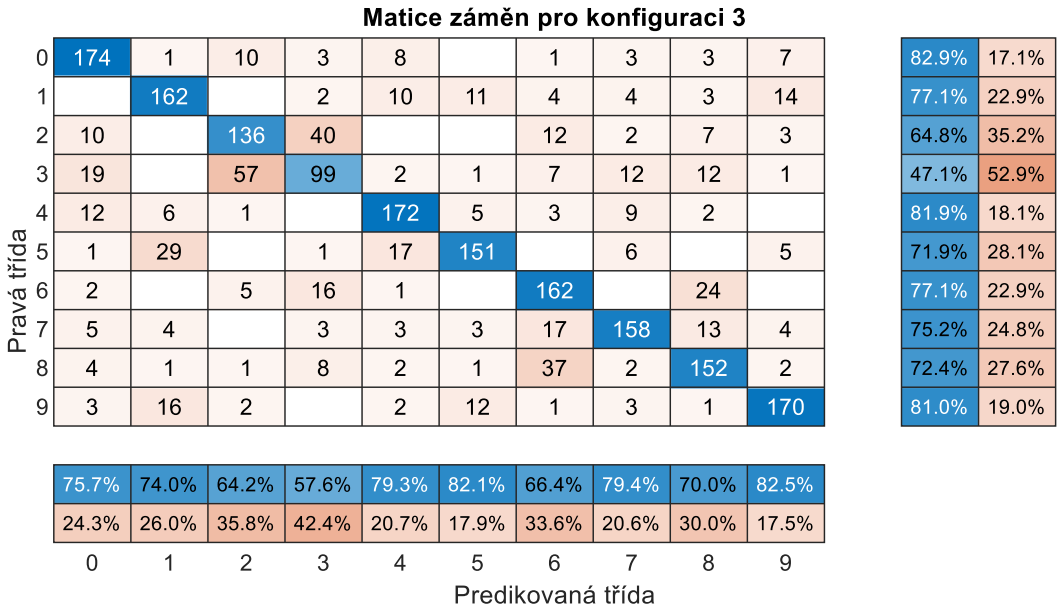
Jako jedno z posledních se validuje přesnost natrénování jednotlivých konfigurací nastavení CNN. Síť klasifikuje data určená pro validaci a její výsledek, predikovaná třída, se porovná s jejich pravou třídou, která jim byla na začátku určena. Celková procentuální přesnost sítě se určí dle vzorce (27). Kde A je procentuální přesnost, n_c je počet správných predikcí a n_t je celkový počet predikcí.

$$A = \frac{n_c}{n_t} \cdot 100 \tag{27}$$

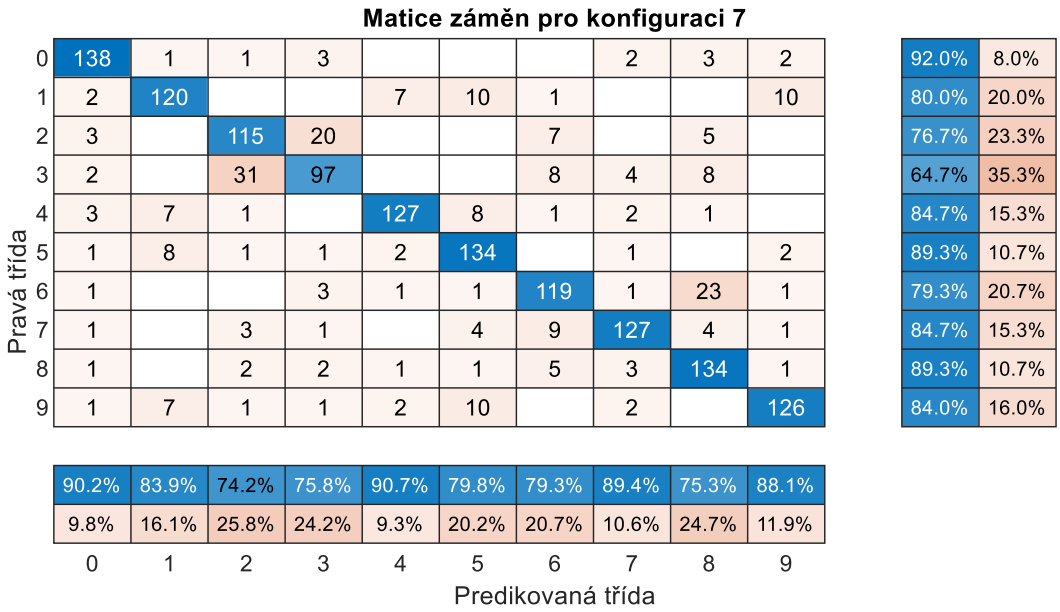
8.5.1 Matice záměn CNN

K lepšímu přehledu, jak síť klasifikovala validační data do jednotlivých tříd slouží matice záměn (confusion matrix). Obsahuje validační data a informaci o tom, jak je konvoluční síť klasifikovala. Z toho se vypočte úspešnost správně určení jednotlivých tříd. Na ose y je pravá třída, jenž byla přiřazena datům na počátku. Na ose x je predikovaná třída, kterou přiřadila akustickým signálům natrénovaná síť.

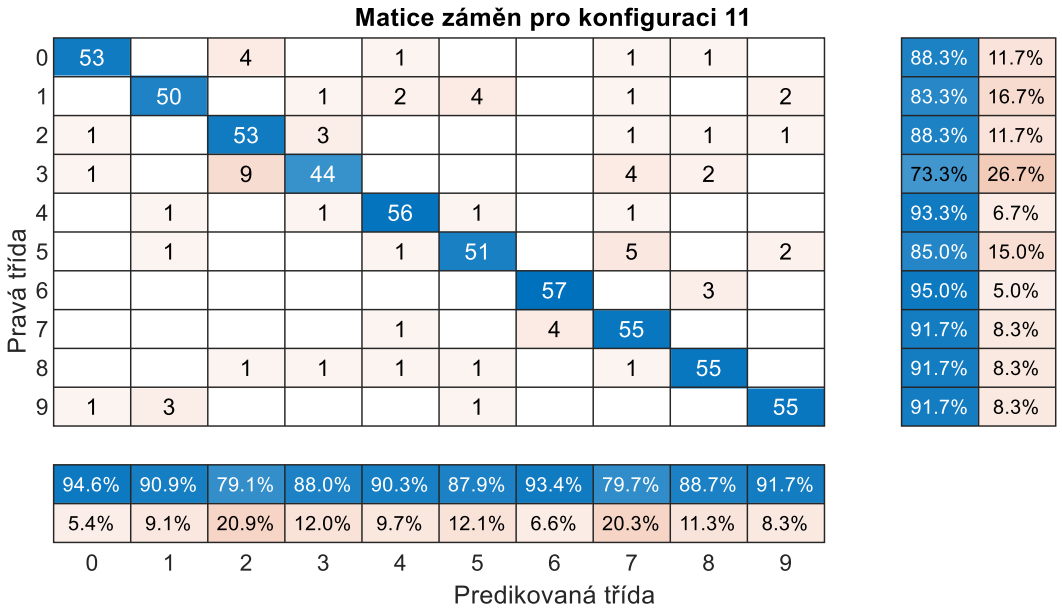
Pro ukázku matic záměn jsou vybrány stejné konfigurace z Tab. 4 jako pro průběh tréninku, konfigurace 3 (Obr. 48), konfigurace 7 (Obr. 49) a konfigurace 11 (Obr. 50). Matice záměn všech konfigurací je možné si prohlédnout v Příloha 5.



Obr. 48 Matice záměn pro konfiguraci 3, CNN [zdroj vlastní]



Obr. 49 Matice záměn pro konfiguraci 7, CNN [zdroj vlastní]



Obr. 50 Matice záměn pro konfiguraci 11, CNN [zdroj vlastní]

8.6 Vyhodnocení CNN

Vyhodnocení přesnosti klasifikace konvoluční neuronovou sítí audio nahrávek je provedeno pro každou konfiguraci sítě. V Tab. 5 jsou přesnosti klasifikace jednotlivých tříd CNN, a v Tab. 6 je celková přesnost a zjištěná výpočetní náročnost natrénovaní sítě.

U klasifikace do jednotlivých tříd lze vidět, že číslice 2 a 3 mají nejhorší přesnost určení, kde dokonce číslice 3 neměla v první konfiguraci ani 50% přesnost. Z prozkoumání matic záměn, jde říct, že právě tyto dvě třídy byly často zaměňovány jedna za druhou. Příkladem toho může být matice záměn konfigurace 3 (Obr. 48) i 7 (Obr. 49). Další číslicí, která se špatně klasifikovala je 6, kde v počátcích nedosahovala o moc lepších výsledků než číslice 2 a 3, ale od konfigurace 10 patří naopak mezi nejlépe klasifikované číslice. Číslice 1 a 8 můžeme zařadit mezi průměrně dobře klasifikované a číslice 5 a 7 mezi lehce nadprůměrně dobře klasifikované.

Jako velmi dobrou klasifikaci můžeme označit číslice 0, 4 a 9. Číslice 9 je nejlépe klasifikovaná při nejnižším počtu trénovacích dat, avšak i při vyšším počtu je stále jedna z lepších. Číslice 4 se nejlépe klasifikovala při středním počtu trénovacích dat i částečně při nejvyšším počtu trénovacích dat. Průměrně její přesnost klasifikace dopadla nejlépe s 87,32% přesností. Číslice 0 se nejlépe určila pouze při největším počtu trénovacích dat, přesto i v předcházejících konfiguracích dopadala dobře.

Celková přesnost CNN se s rostoucí trénovací množinou a počtem epoch zvětšovala a tedy konfigurace 1 má nejmenší celkovou přesnost a konfigurace 12 jí má největší. Lze také zpozorovat, že i menší počet trénovacích dat s 30 epochami dopadl lépe než vetší počte trénovacích dat s 5 epochami, avšak výpočetní náročnost je delší.

Jako nejoptimálnější nastavení CNN se může jevit konfigurace 10, která i přes dobrou celkovou přesnost má oproti nejlepší konfiguraci 12 několikrát kratší výpočetní náročnost.

Tab. 5 Přesnost klasifikace jednotlivých tříd CNN [zdroj vlastní]

Pořadí konfigurací	Přesnost určení jednotlivých tříd (%)									
	0	1	2	3	4	5	6	7	8	9
Konf 1	71,20	65,70	56,30	**44,70**	64,50	75,60	55,60	69,60	59,80	**80,70**
Konf 2	81,50	71,00	66,70	**52,20**	77,50	**82,50**	72,80	81,80	65,90	**82,50**
Konf 3	75,70	74,00	64,20	**57,60**	79,30	82,10	66,40	79,40	70,00	**82,50**
Konf 4	86,00	78,50	71,80	**70,50**	86,20	86,50	73,90	77,20	79,20	**87,00**
Konf 5	82,80	66,80	**59,30**	64,80	**90,60**	89,30	76,20	84,50	67,00	80,30
Konf 6	89,90	72,30	**68,40**	69,50	**94,80**	77,10	72,20	83,30	76,80	91,90
Konf 7	90,20	83,90	**74,20**	75,80	**90,70**	79,80	79,30	89,40	75,30	88,10
Konf 8	92,40	91,00	82,60	**75,00**	87,70	91,80	80,80	82,30	80,10	**92,90**
Konf 9	85,20	84,20	**78,30**	79,60	**91,40**	86,80	**78,30**	86,70	82,70	83,60
Konf 10	93,00	88,10	**80,30**	83,30	**98,30**	90,50	90,20	85,10	87,70	87,90
Konf 11	**94,60**	90,90	**79,10**	88,00	90,30	87,90	93,40	79,70	88,70	91,70
Konf 12	**98,30**	87,10	**81,80**	91,80	96,50	88,70	90,20	91,90	85,90	94,80
Průměr	86,73	79,46	**71,92**	**71,07**	**87,32**	84,88	77,44	82,58	76,59	86,99

Tab. 6 Celková přesnost klasifikace CNN [zdroj vlastní]

Pořadí konfigurací	Celková přesnost (%)	Výpočetní náročnost (min)
Konf 1	**64,24**	00:44
Konf 2	73,00	01:07
Konf 3	73,14	01:34
Konf 4	79,62	02:58
Konf 5	75,20	01:00
Konf 6	79,53	01:46
Konf 7	82,47	02:31
Konf 8	85,53	04:50
Konf 9	83,50	01:33
Konf 10	**88,33**	**02:45**
Konf 11	88,17	03:59
Konf 12	**90,50**	07:41

8.7 Vytvoření laboratorní úlohy CNN

Posledním bodem této kapitoly je vytvoření laboratorní práce pro klasifikaci akustických dat konvoluční neuronovou sítí. Prostřednictvím zadání úlohy a návodu formou Live Skripu budou studenti schopni načíst a pracovat s akustickými daty. Dále se seznámí s vytvářením konvolučních neuronových sítí v prostředí MATLAB a s jejich nastavením. Jejich hlavním úkolem v této úloze je otestování několika různých nastavení CNN a jaký vliv má tato nastavení na výslednou přesnost klasifikace. Laboratorní práce se nachází v příloze.

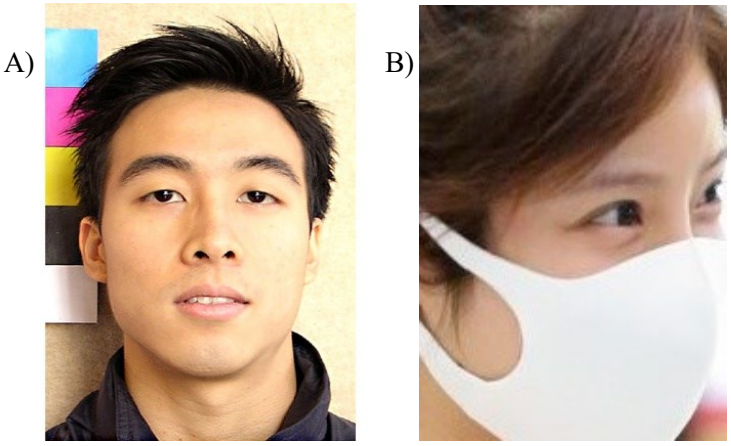
9 Klasifikace obrazů s použitím předučené sítě GoogLeNet

Poslední problematika klasifikace strojovým učením se zabývá klasifikací obrazů předučenou hlubokou konvoluční neuronovou sítí GoogLeNet. Princip učení a klasifikace je stejný jako u běžné CNN z předchozí úlohy, tedy nalezení příznaků v konvolučních vrstvách a následnou klasifikací ve výstupních vrstvách. Předučená síť GoogLeNet potřebuje k lepší klasifikaci menší trénovací množinu, než pokud by bylo úkolem trénovat zcela novou konvoluční síť, která by začala s učením od nuly.

Pro účely práce je předučená síť GoogLeNet natrénovaná při odlišně velkých trénovacích množinách a při několika různých délkách trénovaní. Jednotlivé konfigurace jsou validovány na validačních datech a je vyhodnocena přesnost klasifikace sítě.

9.1 Obrazová data

Data pro úlohu jsou získány z veřejných zdrojů. Dataset obsahuje fotky obličejů s rouškou a bez roušky (Obr. 51). Dataset se skládá celkem z 3833 fotek obličejů z toho je 1915 s rouškou a 1918 bez roušky. Fotografie jsou nafoceny z různých úhlů a mají rozdílné velikosti, přičemž některé snímky jsou natočeny, posunuty, s pozadím nebo obsahují více tváří a případně i jiné předměty.



Obr. 51 Ukázka obrazových dat, A) bez roušky, B) s rouškou [zdroj vlastní]

Pro rozšíření datasetu se na data aplikovala augmentace pomocí polohové transformace dat. Na data se aplikovala augmentace rotace obrazu, horizontálního a vertikálního posunu.

Stejně jakou u předcházející úlohy se při testování jednotlivých konfigurací sítě mění velikost trénovací a validační množiny. Obě množiny obsahují vždy stejný celkový počet obrazů, pouze se mění poměr s jakým jsou rozděleny. Jsou testovány tři různé poměry a to 3:7, kde trénovací množina obsahovala 30 % signálů a validační množina 70 % signálů z celkového počtu, dále to jsou poměry 5:5 a 2:8.

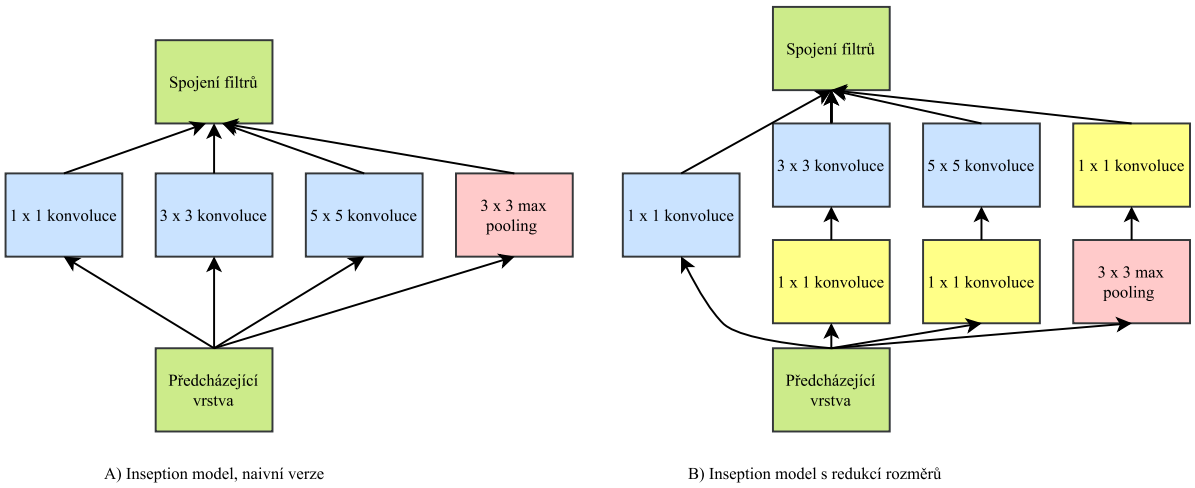
9.2 Architektura GoogLeNet

GoogLeNet je hluboká konvoluční neuronová síť založená na architektuře Inception. Využívá moduly Inception (Obr. 52), které síti umožňují volit mezi několika velikostmi konvolučních filtrů v každém bloku. Síť Inception stohuje tyto moduly na sebe, přičemž se občas vyskytnou sdružovací vrstvy max pooling s krokem 2, které snižují rozlišení sítě na polovinu [55].

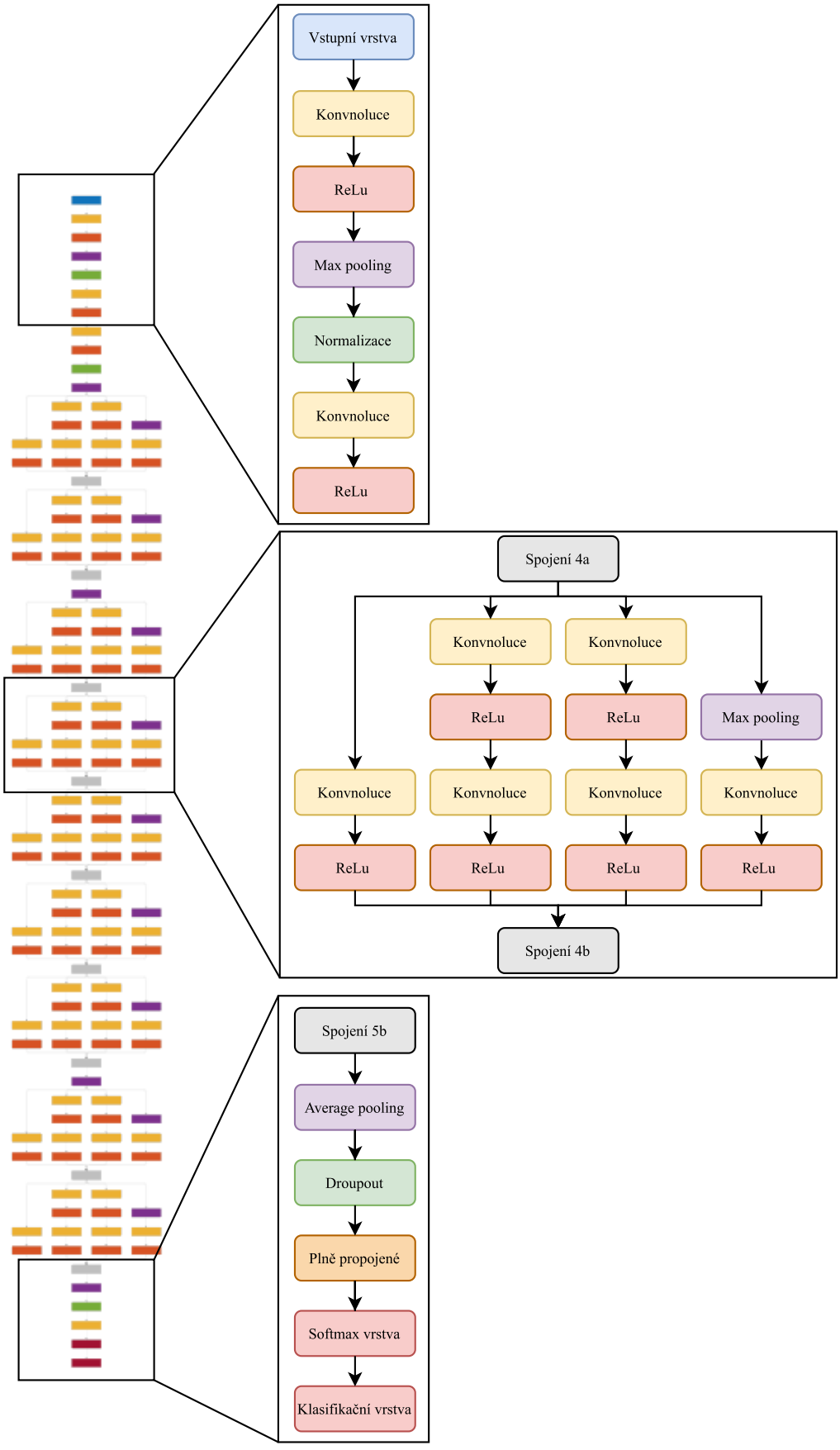
V architektuře GoogLeNet následuje po všech konvolučních vrstvách, včetně tech uvnitř modulů Inception, vrstva ReLu. Velikost vstupní vrstvy je 224x224x3, proto jsou vstupní obrazy upraveny na požadovaný rozměr a snímky zůstávají v RGB barvách neboli potřebují tři kanály [55].

Síť je navržena s ohledem na výpočetní efektivitu a praktičnost, takže inferenci lze provádět na jednotlivých zařízeních, včetně těch s omezenými výpočetními zdroji, zejména s malou pamětí. Síť má hloubku 22 vrstev, pokud počítáme pouze vrstvy s parametry (nebo 27 vrstev, pokud se počítají i sdružovací vrstvy). Celkový počet vrstev (nezávislých stavebních bloků) použitých pro konstrukci sítě je přes 120 (Obr. 53) [55].

Ve výchozím nastevaní jsou plně propojené vrstvy a klasifikační vrstva nastaveny na 1000 tříd. Před natrénováním sítě se tyto vrstvy upravily na požadované dvě třídy.



Obr. 52 Inseption model, A) naivní verze, B) s redukcí rozměrů [55]



Obr. 53 Architektura GoogLeNet, [zdroj vlastní]

9.3 Nastavení GoogLeNet

V síti se nastavují parametry jako je optimalizační algoritmus, míra učení, velikost mini dávky, způsob promíchání a počet epoch.

Optimalizační algoritmus je nastaven na typ *adam*. Míra učení je $\eta = 10^{-5}$. Velikost mini dávky je $miniBatch = 20$. Promíchávání dat je nastaveno na zamíchání trénovací množiny pro každou epochu. Hlavním parametr, která byl zkoumaný pro jednotlivé nastavení sítě je počet epoch. Počet epoch je nastaven na $epocha = 5, 10, 15, 30$.

K přehlednému zobrazení všech konfigurací a jejich parametrů slouží Tab. 7.

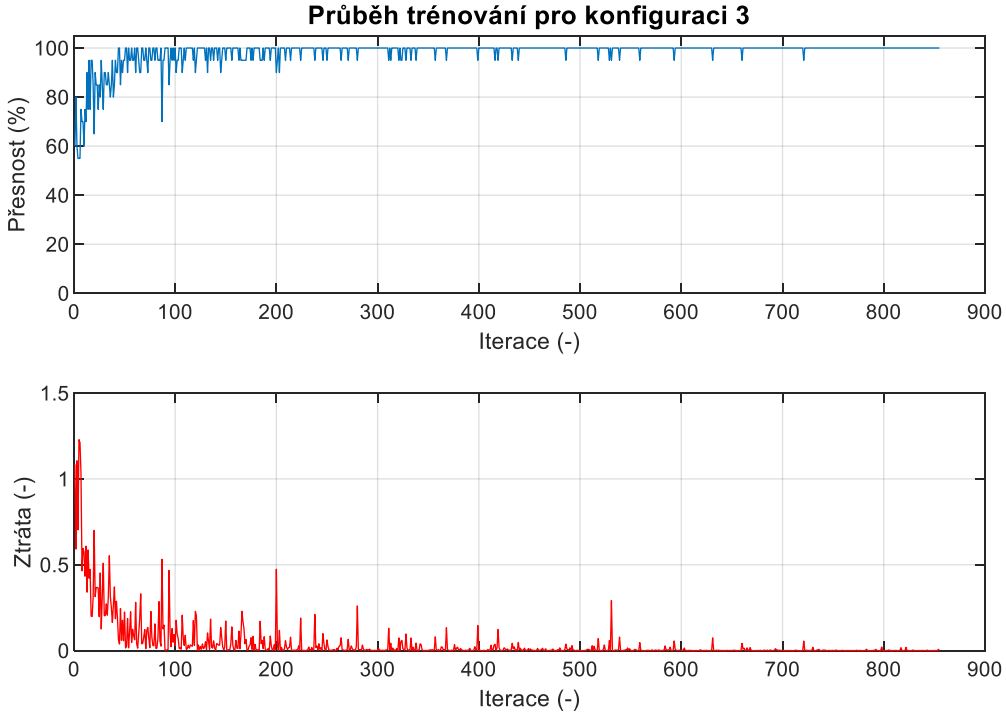
Tab. 7 Paramenty nastavení pro všechny konfigurace GoogLeNet [zdroj vlastní]

Pořadí konfigurací	Poměr množin testování / validace	Počet epoch
Konf 1	0,3 / 0,7	5
Konf 2	0,3 / 0,7	10
Konf 3	0,3 / 0,7	15
Konf 4	0,3 / 0,7	30
Konf 5	0,5 / 0,5	5
Konf 6	0,5 / 0,5	10
Konf 7	0,5 / 0,5	15
Konf 8	0,5 / 0,5	30
Konf 9	0,8 / 0,2	5
Konf 10	0,8 / 0,2	10
Konf 11	0,8 / 0,2	15
Konf 12	0,8 / 0,2	30

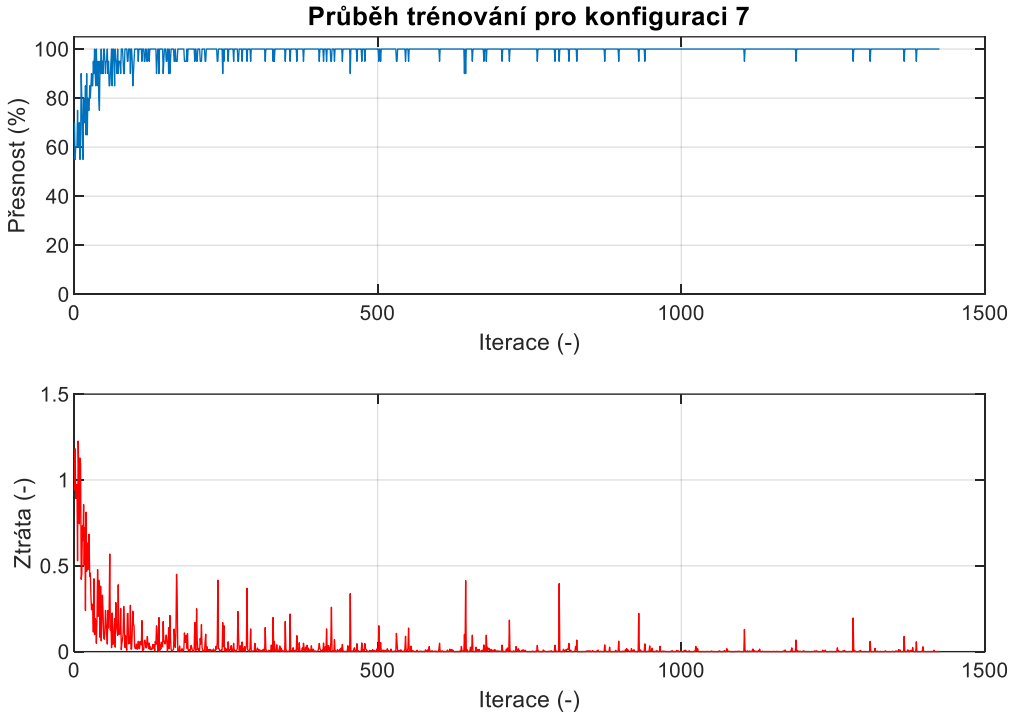
9.4 Průběh trénovaní GoogLeNet

Prostřednictvím aktualizace vah se síť postupně učila a adaptovala vstupním obrazům. Na grafech níže jsou zobrazeny průběhy trénovaní pro tři různé konfigurace. Průběhy jsou zobrazeny pro konfiguraci 3 (Obr. 45), konfiguraci 7 (Obr. 46) a konfiguraci 11 (Obr. 47). Všechny konfigurace mají stejný počet epoch 15 a různý počet poměru trénovací a validační množiny, přesné nastavení je uvedeno v Tab. 4. Grafy všech průběhů je možné si prohlédnout v Příloha 6.

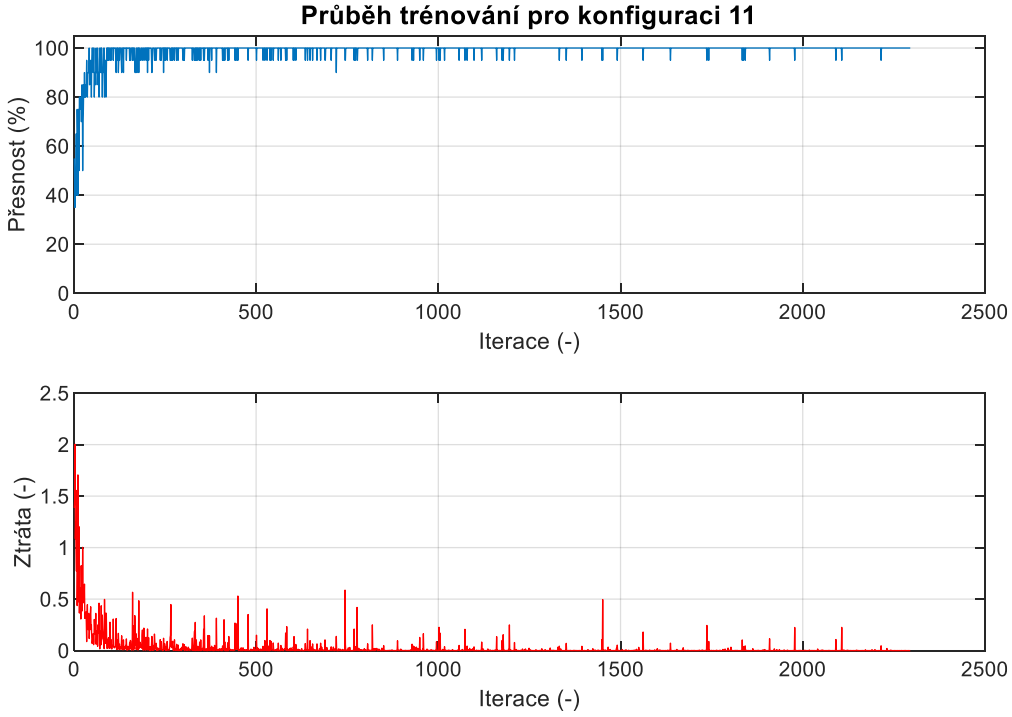
Horní graf průběhu trénovaní představuje přesnost trénovaní tedy jak přesně dokáže klasifikovat data v průběhu tréningu a v dolním grafu je zobrazena ztráta při učení.



Obr. 54 Průběh trénovaní pro konfiguraci 3, GoogLeNet [zdroj vlastní]



Obr. 55 Průběh trénovaní pro konfiguraci 7, GoogLeNet [zdroj vlastní]



Obr. 56 Průběh trénovaní pro konfiguraci 11, GoogLeNet [zdroj vlastní]

Na grafech je možno pozorovat, že pro stejný počet epoch (15) se zvětšuje množství iterací v jednotlivých epochách s větší trénovací množinou. U většiny je už při počátku přesnost klasifikace trénovacích dat přes 50 % a i u nejmenší trénovací množiny (konfigurace 3) je přesnost při trénovaní přibližně rovna 100 %.

9.5 Validace GoogLeNet

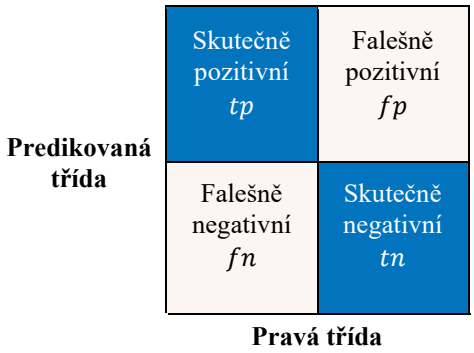
V jednom z posledních kroků úlohy se ověřuje přesnost nastavení jednotlivých konfigurací předučené sítě GoogLeNet. Pro vypočtení přesnosti a dalších měr hodnocení úspěšnosti klasifikace validačních dat jsou použita zejména data z matice záměn.

9.5.1 Matice záměn GoogLeNet

I zde je vytvořena matice záměn z klasifikovaných validačních dat, ukázka obecné matice je na Obr. 57. Na ose *y* je pravá třída, jenž byla přiřazena datům na počátku. Na ose *x* je predikovaná třída, kterou přiřadila snímkům natrénovaná síť. Matice ukazuje:

- **Skutečně pozitivní** (true positive) – množství snímků s rouškou správně klasifikovaných jako snímky s rouškou.
- **Falešně pozitivní** (false positive) – množství snímků s rouškou špatně klasifikovaných jako snímky bez roušky.
- **Falešně negativní** (false negative) – množství snímků bez roušky špatně klasifikovaných jako snímky s rouškou.

- **Skutečně negativní** (true negative) – množství snímků bez roušky správně klasifikovaných jako snímky bez roušky.



Obr. 57 Ukázka matice záměn [zdroj vlastní]

Ze získaných hodnot matice záměn se vypočte míry hodnocení úspěšnosti klasifikace validačních dat:

- **Senzitivita** je úspěšnost, s níž se správně určila třída podle vztahu (28).

$$senzitivita = \frac{tp}{tp + fn} \cdot 100 \tag{28}$$

- **Specifita** je úspěšnost, s níž se správně neurčila třída dle vztahu (29).

$$specifita = \frac{tn}{tn + fp} \cdot 100 \tag{29}$$

- **FPR** je míra falešné pozitivity (false positive rate), počet falešně pozitivních ku celkovému počtu skutečně falešných (30).

$$FPR = \frac{fp}{tn + fp} \cdot 100 \tag{30}$$

- **Preciznost** je počet skutečně pozitivních výsledků ku všem skutečným výsledkům (i špatným) dle vzorce (31).

$$preciznost = \frac{tp}{tp + fp} \cdot 100 \tag{31}$$

- **Recall** je počet skutečně pozitivních výsledků ku všem snímkům co měly být klasifikovány jako pozitivní dle vztahu (32).

$$recall = \frac{tp}{tp + fn} \cdot 100 \tag{32}$$

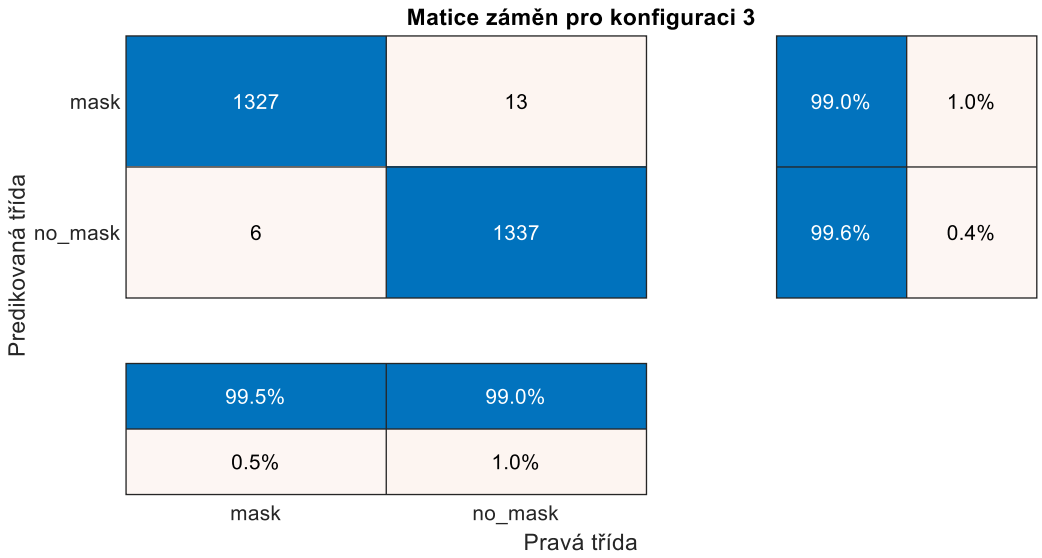
- **F1-score** je harmonický průměr preciznosti a recall vypočtený dle vztahu (33).

$$F1 = \frac{2 \cdot preciznost \cdot recall}{preciznost + recall} \cdot 100 \tag{33}$$

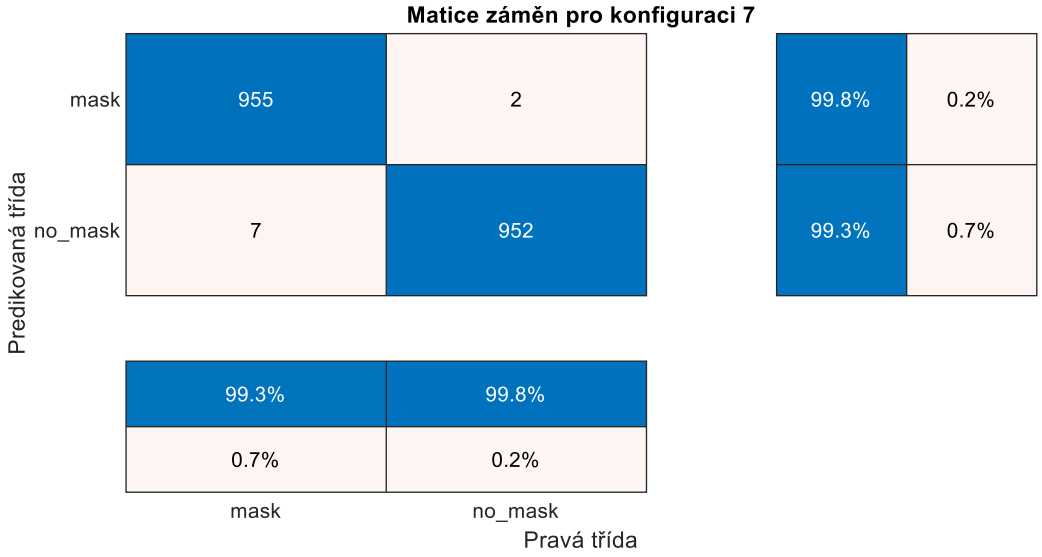
- **Přesnost** je úspěšnost správně určení třídy podle vztahu (34).

$$presnost = \frac{tp + tn}{tp + fp + fn + tn} \cdot 100 \tag{34}$$

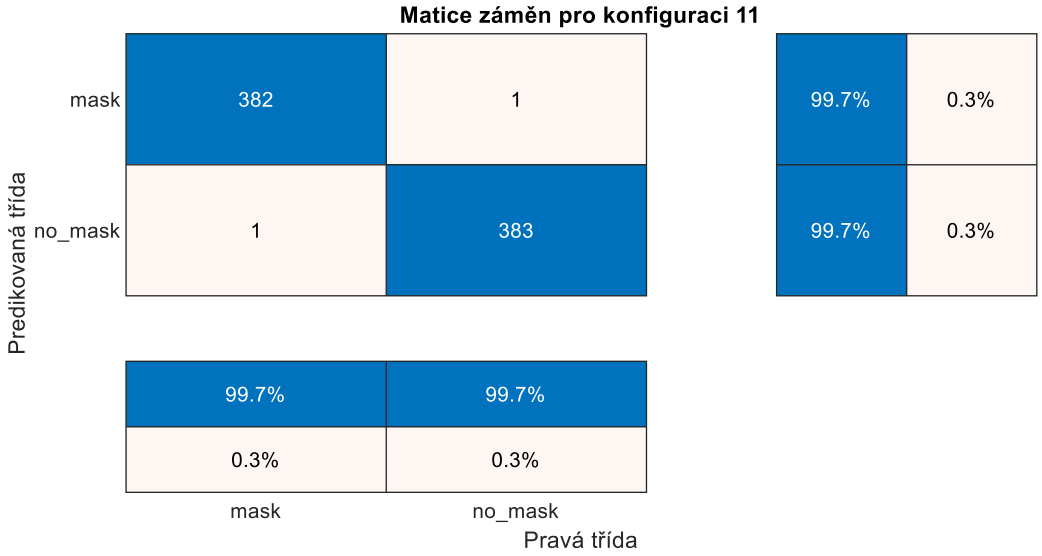
Pro ukázka matic záměn jsou vybrány stejné konfigurace z Tab. 4 jako pro průběh tréninku, konfigurace 3 (Obr. 58), konfigurace 7 (Obr. 59) a konfigurace 11 (Obr. 60). Matice záměn všech konfigurací je možné si prohlédnout v Příloha 7.



Obr. 58 Matice záměn pro konfiguraci 3, GoogLeNet [zdroj vlastní]



Obr. 59 Matice záměn pro konfiguraci 7, GoogLeNet [zdroj vlastní]



Obr. 60 Matice záměn pro konfiguraci 11, GoogLeNet [zdroj vlastní]

9.6 Vyhodnocení GoogLeNet

Vyhodnocení přesnosti klasifikace obrazů hlubokou konvoluční neuronovou sítí GoogLeNet je provedeno pro každou konfiguraci sítě. V Tab. 8 jsou míry přesnosti klasifikace a jištěná výpočetní náročnost natrénovaní sítě.

Z tabulky vychází, že jednotlivá nastavení neuronové sítě mají velmi malý vliv na výslednou přesnost. Ta vychází v intervalu <98,96; 99,87>. I přes malý vliv různý nastavení sítě mají nejvelší vliv jen na výpočetní náročnost, která dosahuje až přibližně 22 minut při nejvyšším nastavení. Pokud dáme v úvahu dobu trvání, jako nejoptimálnější nastavení neuronové sítě se jeví konfigurace 1 s nejmenší trénovací množinou a s pěti epochami.

Tab. 8 Přesnost klasifikace sítě GoogLeNet [zdroj vlastní]

Pořadí konfigurací	Senzitivita (%)	Specifita (%)	FPR (%)	Preciznost (%)	Recall (%)	F1-score (%)	Přesnost (%)	Výpočetní náročnost (min)
Konf 1	**99,62**	**98,96**	**1,04**	**98,96**	**99,62**	**99,29**	**99,29**	**01:42**
Konf 2	99,77	99,26	0,74	99,25	99,77	99,51	99,52	02:53
Konf 3	99,55	99,04	0,96	99,03	99,55	99,29	99,29	04:12
Konf 4	99,55	99,70	0,30	99,70	99,55	99,63	99,63	08:08
Konf 5	**98,55**	**99,37**	**0,63**	**99,37**	**98,55**	**98,96**	**98,96**	02:26
Konf 6	99,48	99,58	0,42	99,58	99,48	99,53	99,53	04:37
Konf 7	99,27	99,79	0,21	99,79	99,27	99,53	99,53	06:51
Konf 8	99,69	99,69	0,31	99,69	99,69	99,69	99,69	13:35
Konf 9	98,96	99,48	0,52	99,48	98,96	99,22	99,22	03:50
Konf 10	**100,00**	**99,74**	**0,26**	**99,74**	**100,00**	**99,87**	**99,87**	07:25
Konf 11	99,74	99,74	0,26	99,74	99,74	99,74	99,74	10:58
Konf 12	99,74	99,74	0,26	99,74	99,74	99,74	99,74	21:41

9.7 Vytvoření laboratorní úlohy GoogLeNet

Ke klasifikaci je vytvořena laboratorní úloha pro klasifikaci obličejů předučenou sítí GoogLeNet. Zadání laboratorní úlohy a návod ve formě Live Scriptu obsahují principy načtení a práce s předučenou sítí GoogLeNet. Studenti se seznámí s tím, jaký vliv má změna nastavení neuronových sítí jako je počet epoch nebo velikost trénovací množiny, na jednotlivé míry přesnosti. Laboratorní práce se nachází v příloze.

Závěr

Hlavní náplní diplomové práce bylo vytvořit čtyři příkladné laboratorní úlohy se zaměřením na klasifikaci základním perceptronem, optimalizaci neuronové sítě a klasifikaci akustického signálu konvoluční neuronovou sítí a klasifikaci obrazů hlubokou konvoluční neuronovou předučenou sítí GoogLeNet. Ke všem laboratorním úlohám byly vytvořeny Live Scripty obsahující krok po kroku popis vytvoření algoritmů.

V první laboratorní úloze vznikly dvě metody natrénování perceptronu pro klasifikaci vytvořených dat. První metoda se zabývala natrénováním prostřednícím implementovaných funkcí v MATLAB. Metoda sloužila jako reference k tomu, jak ve druhé metodě vytvořit vlastní funkci pro přesné natrénování perceptronu. Obě metody byly natrénované se stoprocentní úspěšností klasifikace při tréninku. K otestování metod byly vytvořeny nová náhodná data, které byly klasifikovány.

V druhé laboratorní úloze na optimalizaci vah neuronové sítě pomocí genetického algoritmu, se zkoumal vliv parametrů jako je počet neuronů ve skryté vrstvě neuronové sítě, tak počet generací a velikost populace u genetických algoritmů. Vzniklo 27 konfigurací obsahující tyto různé nastavení. Důležitou sledovanou proměnou byla fitness funkce, která určuje, jak dobře neuronová sít data klasifikovala, na základě výpočtu MSE z predikované a pravé třídy. Nejvhodnější konfigurace je konfigurace 24 s nastavením 10 neuronů, 200 generací a populace o velikosti 500. Konfigurace 24 měla nejmenší fitness funkci $f(x) = 0{,}021$, tedy dosáhla nejlepší přesnosti klasifikace. Pokud by se brala v potaz výpočetní doba tak by jako nejvhodnější konfigurace byla označena konfigurace 5 s 5 neurony, 50 generacemi o 500 jedincích. Její fitness funkce dosahovala $f(x) = 0{,}081$, tedy o něco menší než konfigurace 24, ale výpočetní čas byl z cca 14 minut zkrácen na cca tři minuty.

Ve třetí laboratorní úloze zabývající se klasifikací audio signálů konvoluční neuronovou sítí se sledovalo ovlivňování celkové přesnosti klasifikace různou volbou parametrů, kterými byly poměr velikostí trénovací a validační množiny a počet epoch. Vzniklo 12 konfigurací obsahující tyto různé nastavení. Audio signály obsahují vyslovení cifer nula až devět v anglickém jazyce a cílem konvoluční neuronové sítě bylo tato data správně klasifikovat do daných číselných tříd. Po natrénování sítě se síť validovala na validační množině a vznikla matice záměn a výpočet celkové přesnosti. Matice záměn představuje výsledek klasifikace validační množiny, kde jsou data z predikované třídy porovnána vzhledem k pravé třídě. Z výsledků je patrné, že největší problém pro klasifikaci dělaly cifry dva a tři, kde z matic záměn, lze vidět, že všem konfiguracím se tyto cifry často zaměňovali jedna za druhou. Nejlépe dopadly klasifikace cifer 0, 4 a 9. Jako nejvhodnější je označena konfigurace 12 s největší trénovací množinou a 30 epochami a s celkovou přesností 90,50 %. Pokud by se brala v úvahu doba trénovaní tak by byla jako nejvhodnější konfigurace označená konfigurace 10 s největší trénovací množinou a 10 epochami. Její celková přesnost je 88,33 %, tedy o málo menší než konfigurace 12, ale výpočetní čas byl z cca 22 minut zkrácen na cca dvě minuty.

V poslední úloze na klasifikaci obličejů s rouškou a bez roušky je použita hluboká konvoluční neuronová předučená sít GoogLeNet. Měnily se parametry natrénovaní sítě, jako je poměr velikosti trénovací množiny k validační množině a počet epoch, kde se zkoumal vliv těchto parametrů na výslednou přesnost sítě. Vzniklo 12 konfigurací obsahující tyto různé nastavení. Po natrénování sítě se síť validovala na validační množině a vznikla matice záměn z nichž se vypočetly míry přesnosti.

Z výsledků vyšlo, že volba parametrů má minimální vliv na přesnost klasifikace neuronovou sítí, neboť přesnost jednotlivých konfigurací se nachází v intervalu <98,96; 99,84>. Jako nejlepší dopadla konfigurace 10 s přesností 99,84 %. Pokud by se brala v úvahu výpočetní náročnost tak by jako nejvhodnější konfigurace byla označená konfigurace 1 s nejmenší trénovací množinou a 5 epochami. Její přesnost je 99,29 %, tedy jen nepatrně menší než konfigurace 10, ale výpočetní čas byl z cca sedmi minut zkrácen na cca dvě minuty.

V návaznosti na tuto práci je možné prozkoumat další metody strojového učení jako je např. segmentace medicínských obrazu neuronovou sítí U-net anebo klasifikace metodou podpůrných vektorů a posoudit jejich vlastnosti a vhodnost pro výuku.

Literatura

[1] JANOUŠOVÁ, Eva a Jiří HOLČÍK. Vícerozměrné metody pro analýzu a klasifikaci dat. In: *Matematická biologie: e-learningová učebnice* [online]. Brno: Masarykova univerzita, 2015. Dostupné také z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--klasifikace

[2] ASIRI, Sidath. Machine Learning Classifiers. In: *Towards Data Science* [online]. Canada: Medium. Dostupné také z: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

[3] HOLČÍK, Jiří. *Analýza a klasifikace dat*. Brno: Akademické nakladatelství CERM, 2012. ISBN 978-80-7204-793-2.

[4] PALUSZEK, Michael a Stephanie THOMAS. *MATLAB Machine Learning*. Berkeley, CA: Apress, 2017. ISBN 978-148-4222-508.

[5] LASKOV, Pavel, Patrick DÜSSEL, Christin SCHÄFER a Konrad RIECK. Learning Intrusion Detection: Supervised or Unsupervised?. *Image Analysis and Processing – ICIAP 2005* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, 50-57. Lecture Notes in Computer Science. ISBN 978-3-540-28869-5. Dostupné z: doi:10.1007/11553595_6

[6] Unsupervised Machine Learning. In: *JavaTpoint* [online]. Noida, c2011-2021. Dostupné také z: https://www.javatpoint.com/unsupervised-machine-learning

[7] KOTSIANTIS, S. B. Supervised Machine Learning: A Review of Classification Techniques. *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies* [online]. 2007, (160), 3-24. ISSN 0922-6389. Dostupné také z: https://www.researchgate.net/publication/220166738_Supervised_Machine_Learning_A_Review_of_Classification_Techniques

[8] Supervised Machine Learning. In: *JavaTpoint* [online]. Noida, c2011-2021. Dostupné také z: https://www.javatpoint.com/supervised-machine-learning

[9] Data Augmentation: How to use Deep Learning when you have Limited Data—Part 2. In: *Nanonets* [online]. Nano Net Technologies Inc., 2021. Dostupné také z: https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

[10] *Statistics and Machine Learning Toolbox™: User's Guide* [online]. Version 12.2. Natick: The MathWorks, c1998-2021. Dostupné také z: https://www.mathworks.com/help/pdf_doc/stats/stats.pdf

[11] TANG, Bo a Haibo HE. ENN: Extended Nearest Neighbor Method for Pattern Recognition [Research Frontier]. *IEEE Computational Intelligence Magazine* [online]. 2015, **10**(3), 52-60. ISSN 1556-603X. Dostupné z: doi:10.1109/MCI.2015.2437512

[12] Support Vector Machine Algorithm. In: *JavaTpoint* [online]. JavaTpoint, c2011-2021. Dostupné také z: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[13] VONDRÁK, Ivo. *Neuronové sítě*. Ostrava: VŠB - Technická univerzita Ostrava, 2009.

[14] VOLNÁ, Eva. *Neuronové sítě 1*. Druhé. Ostrava: Ostravská univerzita v Ostravě, 2008.

[15] SMOLA, Alex a S.V.N. VISHWANATHAN. *Introduction to Machine Learning*. Cambridge: Cambridge University Press, 2008. ISBN 0 521 82583 0.

[16] BLAHA, Milan. Umělá inteligence. In: *Matematická biologie: e-learningová učebnice* [online]. Brno: Masarykova univerzita, 2015. Dostupné také z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence

[17] Neural Networks. In: *IBM* [online]. IMB, 2022. Dostupné také z: https://www.ibm.com/cloud/learn/neural-networks#toc-neural-net-u3voPJVU

[18] AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?. In: *IBM* [online]. IMB, 2022. Dostupné také z: https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

[19] CHOLLET, François. *Deep learning v jazyku Python: knihovny Keras, Tensorflow*. Praha: Grada Publishing, 2019. Knihovna programátora (Grada). ISBN 978-802-4731-001.

[20] ALBAWI, Saad, Tareq Abed MOHAMMED a Saad AL-ZAWI. Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)* [online]. IEEE, 2017, 1-6. ISBN 978-1-5386-1949-0. Dostupné z: doi:10.1109/ICEngTechnol.2017.8308186

[21] SAHA, Sumit. A Comprehensive Guide to Convolutional Neural Networks. In: *Towards Data Science* [online]. Canada: Towards Data Science Inc. Dostupné také z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[22] RAY, Sujan, Khaldoon ALSHOUILIY a Dharma P. AGRAWAL. Dimensionality Reduction for Human Activity Recognition Using Google Colab. *Information* [online]. 2021, **12**(1). ISSN 2078-2489. Dostupné z: doi:10.3390/info12010006

[23] The Architecture of Lenet-5. In: *Analytics Vidhya* [online]. Analytics Vidhya, c2013-2022. Dostupné také z: https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/

[24] WEI, Jerry. AlexNet: The Architecture that Challenged CNNs. In: *Towards Data Science* [online]. Canada: Towards Data Science Inc. Dostupné také z: https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951

[25] ALAKE, Richmond. Deep Learning: GoogLeNet Explained. In: *Towards Data Science* [online]. Canada: Towards Data Science Inc. Dostupné také z: https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765

[26] FENG, Vincent. An Overview of ResNet and its Variants. In: *Towards Data Science* [online]. Canada: Towards Data Science Inc. Dostupné také z: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

[27] BOESCH, Gaudenz. VGG Very Deep Convolutional Networks. In: *Viso.ai* [online]. viso.ai, 2022. Dostupné také z: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/

[28] TYAGI, Mrinal. Image Segmentation: Part 1. In: *Towards Data Science* [online]. Canada: Towards Data Science Inc. Dostupné také z: https://towardsdatascience.com/image-segmentation-part-1-9f3db1ac1c50

[29] What Is Image Segmentation?: 3 things you need to know. In: *MathWorks* [online]. United States: The MathWorks, Inc., c1994-2022. Dostupné také z: https://www.mathworks.com/discovery/image-segmentation.html#how-it-works

[30] LEE, Cheolha Pedro. *Robust Image Segmentation using Active Contours: Level Set Approaches*. Raleigh, 2005. Disertace. North Carolina State University.

[31] MUTHUKRISHNAN, R a M RADHA. Edge Detection Techniques For Image Segmentation. *International Journal of Computer Science and Information Technology* [online]. 2011, **3**(6), 259-267. ISSN 09754660. Dostupné z: doi:10.5121/ijcsit.2011.3620

[32] TYAGI, Mrinal. Image Segmentation: Part 2. In: *Towards Data Science* [online]. Canada: Towards Data Science Inc. Dostupné také z: https://towardsdatascience.com/image-segmentation-part-2-8959b609d268

[33] HEMALATHA, R.J., T.R. THAMIZHVANI, A. Josephin Arockia DHIVYA, Josline Elsa JOSEPH, Bincy BABU a R. CHANDRASEKARAN. Active Contour Based Segmentation Techniques for Medical Image Analysis. *Medical and Biological Image Analysis* [online]. InTech, 2018. ISBN 978-1-78923-330-8. Dostupné z: doi:10.5772/intechopen.74576

[34] BOESCH, Gaudenz. Image Segmentation Using Deep Learning: Quick Guide. In: *Viso.ai* [online]. viso.ai, 2022. Dostupné také z: https://viso.ai/deep-learning/image-segmentation-using-deep-learning/

[35] LE, James. How to do Semantic Segmentation using Deep learning. In: *Nanonets* [online]. Nano Net Technologies Inc., 2021. Dostupné také z: https://nanonets.com/blog/how-to-do-semantic-segmentation-using-deep-learning/

[36] RONNEBERGER, Olaf, Philipp FISCHER a Thomas BROX. *U-Net: Convolutional Networks for Biomedical Image Segmentation* [online]. Dostupné z: doi:arXiv:1505.04597v1

[37] CHEN, Liang-Chieh, George PAPANDREOU, Iasonas KOKKINOS, Kevin MURPHY a Alan L. YUILLE. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs* [online]. Dostupné z: doi:arXiv:1606.00915v2

[38] WU, Huikai, Junge ZHANG, Kaiqi HUANG, Kongming LIANG a Yizhou YUD. *FastFCN: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation* [online]. Dostupné z: doi:arXiv:1903.11816v1

[39] SILVA, Pedro, Eduardo LUZ, Elizabeth WANNER, David MENOTTI a Gladston MOREIRA. QRS Detection in ECG Signal with Convolutional Network. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* [online]. Cham: Springer International Publishing, 2019, 802-809. Lecture Notes in Computer Science. ISBN 978-3-030-13468-6. Dostupné z: doi:10.1007/978-3-030-13469-3_93

[40] YıLDıRıM, Özal, Paweł PŁAWIAK, Ru-San TAN a U. Rajendra ACHARYA. Arrhythmia detection using deep convolutional neural network with long duration ECG signals. *Computers in Biology and Medicine* [online]. 2018, **102**, 411-420. ISSN 00104825. Dostupné z: doi:10.1016/j.compbiomed.2018.09.009

[41] CAI, Wenjie a Danqin HU. QRS Complex Detection Using Novel Deep Learning Neural Networks. *IEEE Access* [online]. 2020, **8**, 97082-97089. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2020.2997473

[42] CAMPS, Julià, Blanca RODRIGUEZ a Ana MINCHOLE. *Deep Learning Based QRS Multilead Delineator in Electrocardiogram Signals* [online]. -. Dostupné z: doi:10.22489/CinC.2018.292

[43] HABIB, Ahsan, Chandan KARMAKAR a John YEARWOOD. Impact of ECG Dataset Diversity on Generalization of CNN Model for Detecting QRS Complex. *IEEE Access* [online]. 2019, **7**, 93275-93285. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2019.2927726

[44] XIANG, Yande, Zhitao LIN a Jianyi MENG. Automatic QRS complex detection using two-level convolutional neural network. *BioMedical Engineering OnLine* [online]. 2018, **17**(1). ISSN 1475-925X. Dostupné z: doi:10.1186/s12938-018-0441-4

[45] YANG, Ruixin a Yingyan YU. Artificial Convolutional Neural Network in Object Detection and Semantic Segmentation for Medical Imaging Analysis. *Frontiers in Oncology* [online]. 2021, **11**. ISSN 2234-943X. Dostupné z: doi:10.3389/fonc.2021.638182

[46] WANG, Jiazhuo, John D. MACKENZIE, Rageshree RAMACHANDRAN a Danny Z. CHEN. A Deep Learning Approach for Semantic Segmentation in Histology Tissue Images. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016* [online]. Cham: Springer International Publishing, 2016, 176-184. Lecture Notes in Computer Science. ISBN 978-3-319-46722-1. Dostupné z: doi:10.1007/978-3-319-46723-8_21

[47] JIANG, Feng, Aleksei GRIGOREV, Seungmin RHO, Zhihong TIAN, YunSheng FU, Worku JIFARA, Khan ADIL a Shaohui LIU. Medical image semantic segmentation based on deep learning. *Neural Computing and Applications* [online]. 2018, **29**(5), 1257-1265. ISSN 0941-0643. Dostupné z: doi:10.1007/s00521-017-3158-6

[48] OUAHABI, Abdeldjalil a Abdelmalik TALEB-AHMED. Deep learning for real-time semantic segmentation: Application in ultrasound imaging. *Pattern Recognition Letters* [online]. 2021, **144**, 27-34. ISSN 01678655. Dostupné z: doi:10.1016/j.patrec.2021.01.010

[49] ROTH, Holger R., Chen SHEN, Hirohisa ODA, Masahiro ODA, Yuichiro HAYASHI, Kazunari MISAWA a Kensaku MORI. *Deep learning and its application to medical image segmentation* [online]. Dostupné z: doi:10.11409/mit.36.63

[50] Perceptron Neural Networks. In: *MathWorks* [online]. United States: The MathWorks, Inc., c1994-2022. Dostupné také z: https://www.mathworks.com/help/deeplearning/ug/perceptron-neural-networks.html

[51] How the Genetic Algorithm Works. In: *MathWorks* [online]. United States: The MathWorks, Inc., c1994-2022. Dostupné také z: https://www.mathworks.com/help/gads/how-the-genetic-algorithm-works.html

[52] What is a Convolutional Neural Network?. In: *MathWorks* [online]. United States: The MathWorks, Inc., c1994-2022. Dostupné také z: https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html

[53] JACKSON, Zohar, César SOUZA, Jason FLAKS, Yuxin PAN, Hereman NICOLAS a Adhish THITE. *Jakobovski/free-spoken-digit-dataset: v1.0.8* [online]. Zenodo. Dostupné z: doi:10.5281/zenodo.1342401

[54] Specify Layers of Convolutional Neural Network. In: *MathWorks* [online]. United States: The MathWorks, Inc., c1994-2022. Dostupné také z: https://www.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html

[55] SZEGEDY, Christian, Wei LIU, Yangqing JIA et al. *Going Deeper with Convolutions* [online]. Dostupné z: doi:10.48550/arXiv.1409.4842

Seznam příloh

Příloha I **Příloha v IS Edison**

Tato příloha obsahuje všechny skripty s použitými algoritmy, Live Scripty, vstupní data, naučené sítě a výsledky. Algoritmy jsou přehledně organizovány do relativně krátkých skriptů v samostatných souborech s pomocnými komentáři. Skripty byly vytvořeny a testovány v programu MATLAB ve verzi R2021a.

Příloha I.I **Adresář Laboratorní úloha 1**

Obsahuje zadání laboratorní úlohy *LU1_zadání.pdf*. Dále se adresář dělí na Laboratorní úloha 1a a Laboratorní úloha 1b.

- Podadresář **Laboratorní úloha 1a**:
  - *LU1a_Perceptron.mlx* – Live Script, který slouží jako návod pro vypracování laboratorní úlohy metodou implementace funkce MATLAB krok po kroku.
  - *LU1a_perceptron.mat* – obsahuje vytvořený a natrénovaný perceptron spolu s trénovací množinou.
  - *LU1a_novyBod.mat* – obsahuje natrénovaný perceptron a nová data pro klasifikaci.
  - *LU1a_klasifikacePerceptronem.fig* – graf zobrazující trénovací množinu s klasifikační přímkou. Pro otevření v prostřední MATLAB.
  - *LU1a_novyBod.fig* – graf zobrazující nová data s klasifikační přímkou získanou perceptronem. Pro otevření v prostřední MATLAB.
- Podadresář **Laboratorní úloha 1b**:
  - *LU1b_Perceptron.mlx* – Live Script, který slouží jako návod pro vypracování laboratorní úlohy metodou vytvoření vlastní funkce pro výpočet perceptronu krok po kroku.
  - *LU1b_perceptron.mat* – obsahuje vytvořený a natrénovaný perceptron spolu s trénovací množinou.
  - *LU1b_novyBod.mat* – obsahuje natrénovaný perceptron a nová data pro klasifikaci.
  - *LU1b_klasifikacePerceptronem.fig* – graf zobrazující trénovací množinu s klasifikační přímkou. Pro otevření v prostřední MATLAB.
  - *LU1b_novyBod.fig* – graf zobrazující nová data s klasifikační přímkou získanou perceptronem. Pro otevření v prostřední MATLAB.

Příloha I.II **Adresář Laboratorní úloha 2**

Obsahuje zadání laboratorní úlohy *LU2_zadání.pdf*. Pak obsahuje Skripty a podadresář s výsledky.

- *LU2_pracovni_GA.m* – pracovní skript pro výpočet všech nastavení.
- *LU2_genetickyAlgoritmus.mlx* – Live Script pro vypracování laboratorní úlohy.
- *LU2_tabulka_GA.m* – pomocný skript.
- Podadresář **LU2_Výsledky**:
  - *LU2_Score.xlsx* – obsahuje všechny hodnoty fitness funkce pro celou populaci ve všech generacích, pro všechny konfigurace.

- *LU2_Score_best.xlsx* – obsahuje nejlepší hodnoty fitness funkce pro všechny generace, pro všechny konfigurace.
- *LU2_Konfigurace_rozpis.xlsx* – obsahuje seznam konfigurací, jejich nastavení, nejlepší a průměrnou hodnotu fitness funkce a potřebný čas.
- *LU2_konf_x_genetickyAlgoritmus.mat* – uložená neuronové sítě s nejlepší hodnotou fitness funkce, kde x značí konfiguraci.
- *LU2_konf_x_mean.fig* – graf průběhu nejlepší hodnoty fitness funkce vzhledem ke generaci, kde x značí konfiguraci. Pro otevření v prostřední MATLAB.
- *LU2_konf_x_best.fig* – graf průběhu průměrné hodnoty fitness funkce vzhledem ke generaci, kde x značí konfiguraci. Pro otevření v prostřední MATLAB.
- *LU2_time* – obsahuje výpis z konzole s dobou trvání algoritmů.

Příloha I.III **Adresář Laboratorní úloha 3**

Obsahuje zadání laboratorní úlohy *LU3_zadání.pdf*. Pak obsahuje Skripty a podadresář s výsledky.

- *LU3_pracovni_CNN.m* – pracovní skript pro výpočet všech nastavení.
- *LU3_CNN.mlx* – Live Script pro vypracování laboratorní úlohy.
- *LU3_ helpergenLabels.m* – pomocný skript.
- *LU3_ helperReadSPData.m* – pomocný skript.
- Podadresář **LU3_Průběh:**
  - *LU3_konf_x_Training_progress.fig* – graf průběhu trénovaní, kde *x* značí konfiguraci. Pro otevření v prostřední MATLAB.
  - *LU3_konf_x_Training_progress.png* – graf průběhu trénovaní, kde *x* značí konfiguraci.
- Podadresář **LU3_Výsledky**:
  - *LU3_Training_Data.xlsx* – obsahuje soupis všech konfigurací a průběhy trénovaní u jednotlivých konfigurací.
  - *LU3_konf_x_network.mat* – natrénovaná neuronová síť s trénovací a validační množinou, kde *x* značí konfiguraci.
  - *LU3_konf_x_Conf_chart.fig* – graf matice záměn, kde *x* značí konfiguraci. Pro otevření v prostřední MATLAB.
  - *LU3_konf_x_Conf_chart.png* – graf matice záměn, kde *x* značí konfiguraci.
- Podadresář **recordings:**
  - Obsahuje databázi zvukových nahrávek použitých k natrénovaní sítě.

Příloha I.IV **Adresář Laboratorní úloha 4**

Obsahuje zadání laboratorní úlohy *LU4_zadání.pdf*. Pak obsahuje Skripty a podadresář s výsledky.

- *LU4_pracovni_GoogLeNet.m* – pracovní skript pro výpočet všech nastavení.
- *LU4_GoogLeNet.mlx* – Live Script pro vypracování laboratorní úlohy.

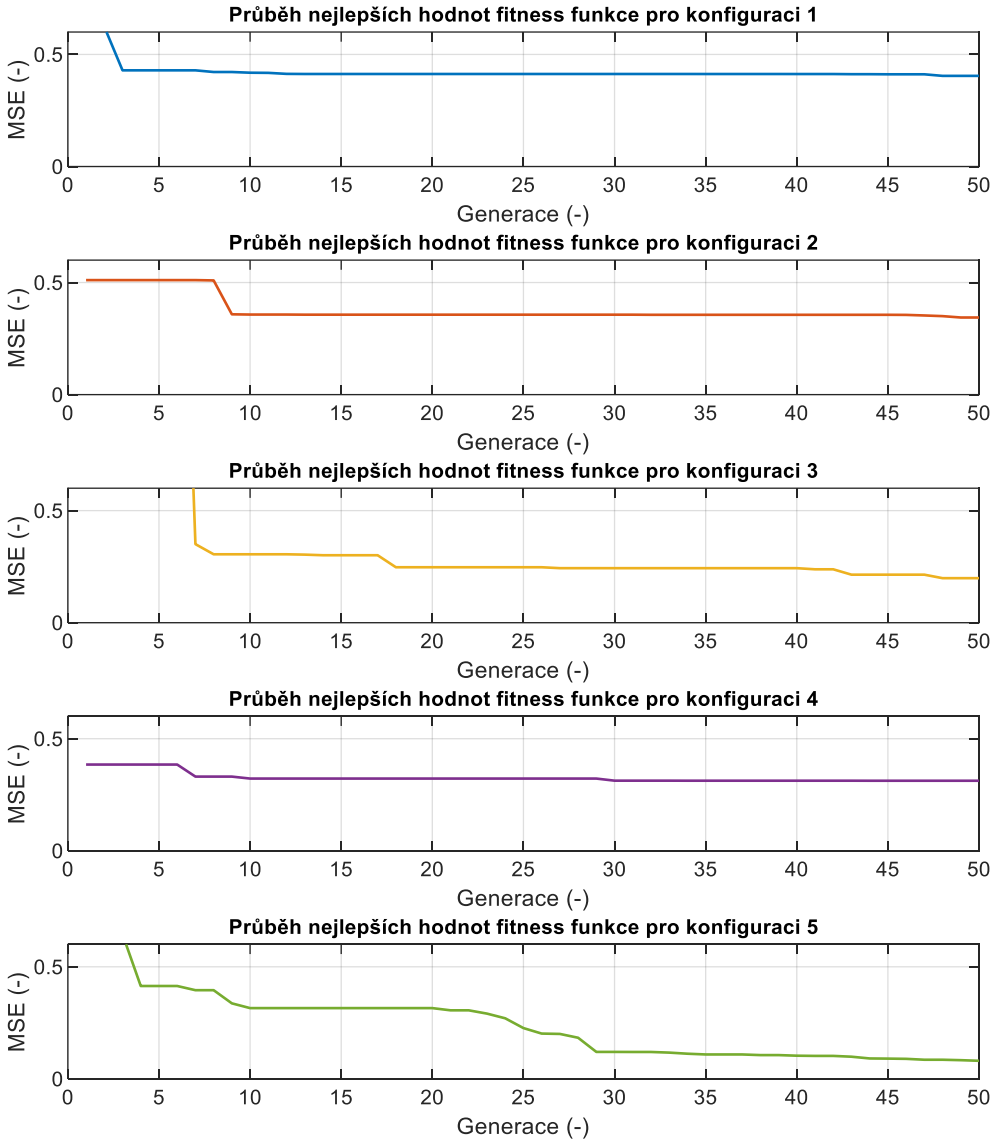
- Podadresář **LU4_Průběh:**
  - *LU4_konf_x_Training_progress.fig* – graf průběhu trénovaní, kde *x* značí konfiguraci. Pro otevření v prostřední MATLAB.
  - *LU4_konf_x_Training_progress.png* – graf průběhu trénovaní, kde *x* značí konfiguraci.
- Podadresář **LU4_Výsledky**:
  - *LU4_Training_Data.xlsx* – obsahuje soupis všech konfigurací a průběhy trénovaní u jednotlivých konfigurací.
  - *LU4_konf_x_network.mat* – natrénovaná neuronová síť s trénovací a validační množinou, kde x značí konfiguraci.
  - *LU4_konf_x_Conf_chart.fig* – graf matice záměn, kde x značí konfiguraci. Pro otevření v prostřední MATLAB.
  - *LU4_konf_x_Conf_chart.png* – graf matice záměn, kde *x* značí konfiguraci.
- Podadresář **Face_mask:**
  - Obsahuje databázi obrazů použitých pro natrénovaní neuronové sítě.

Příloha II **Grafická příloha**

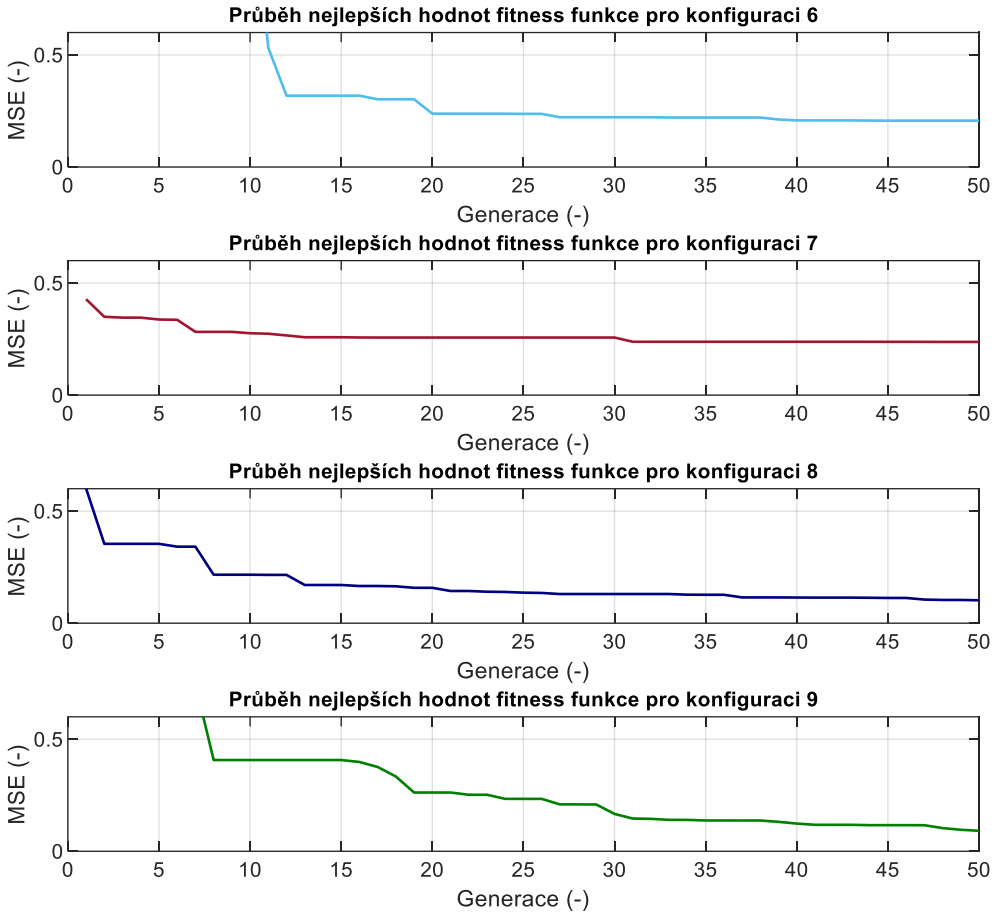
Tato příloha obsahuje grafy k jednotlivým laboratorním úlohám, které jsou použité nebo významné v této diplomové práci. V první části jsou zobrazeny grafy k druhé laboratorní úloze o optimalizaci neuronové sítě genetickým algoritmem, které vykreslují nejlepší průběh fitness funkce pro všechny konfigurace zvlášť. V druhé části jsou grafy ke třetí laboratorní úloze. Tyto grafy obsahují průběhy trénovaní sítě a matice záměn pro všechny konfigurace. V poslední části jsou grafy ke čtvrté úloze, které rovněž zobrazují průběhy trénovaní sítě a matice záměn pro všechny konfigurace.

Příloha II.I **Grafy k laboratorní úloze 2**

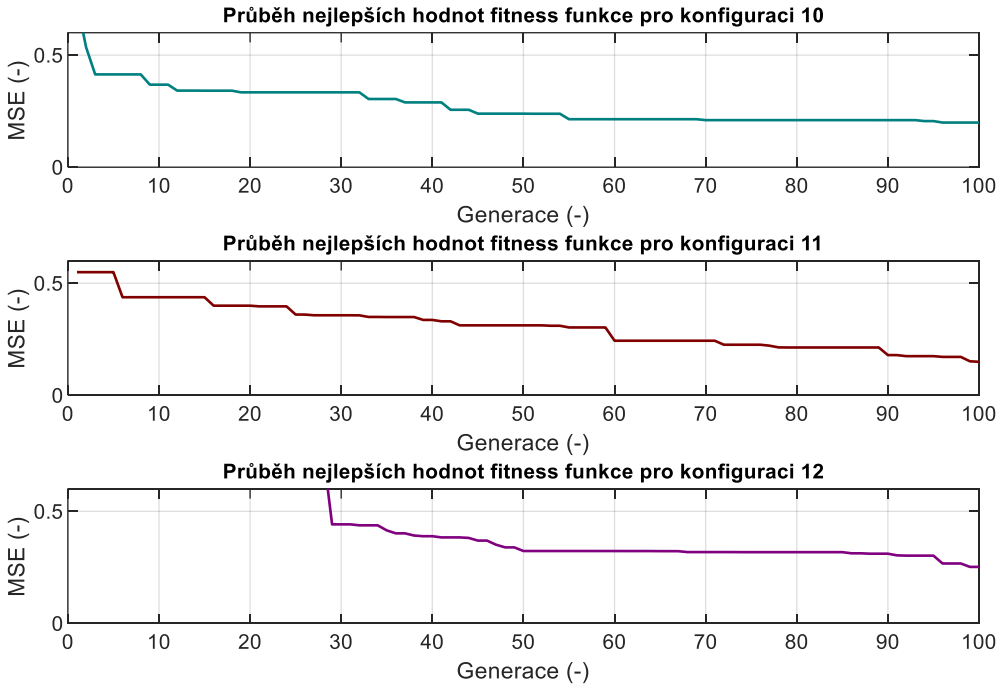
V této části příloh jsou grafy nejlepšího průběhu fitness funkce pro všechny konfigurace zvlášť. Všechny grafy jsou oříznuty na ose y, aby byly v rozsahu 0 až 0,6. V tomto formátu jde lépe vidět rozdíl v průběhu fitness funkce jednotlivých konfigurací. Celý průběh lze najít v elektronické příloze.



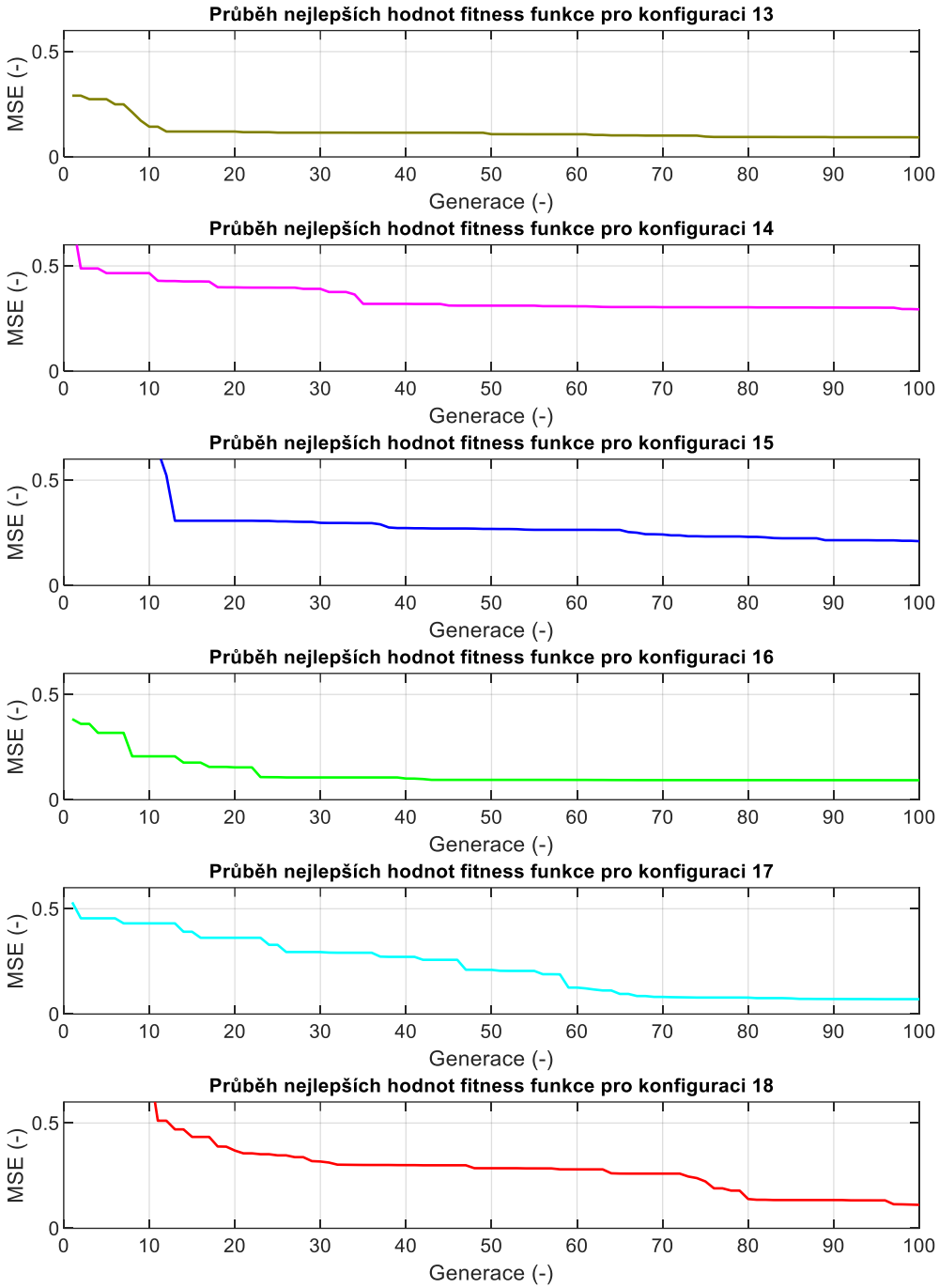
Příloha 1 Průběhy nejlepších hodnot fitness funkce pro jednotlivé konfigurace s 50 generacemi



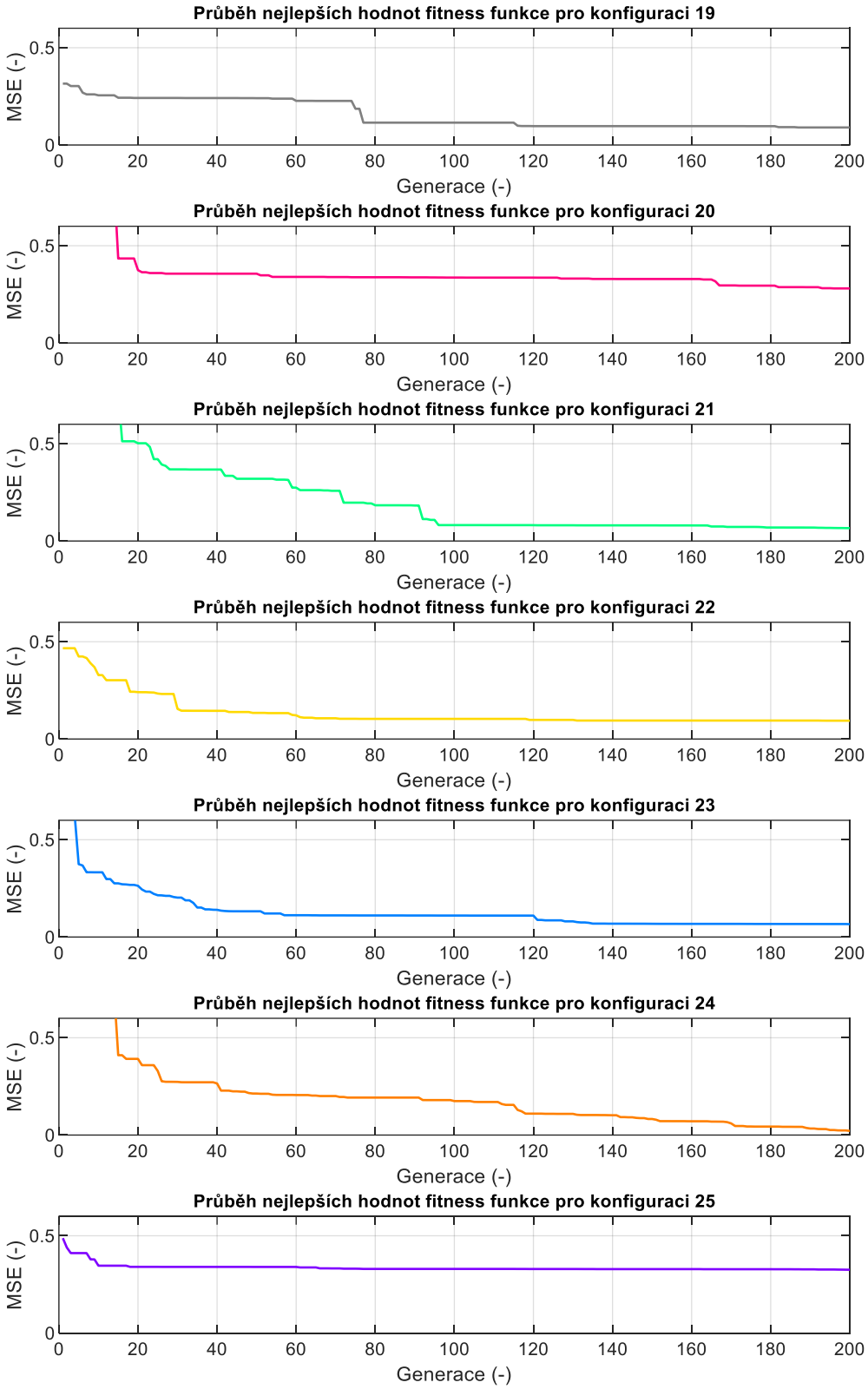
Příloha 1 Průběhy nejlepších hodnot fitness funkce
pro jednotlivé konfigurace s 50 generacemi – pokračování



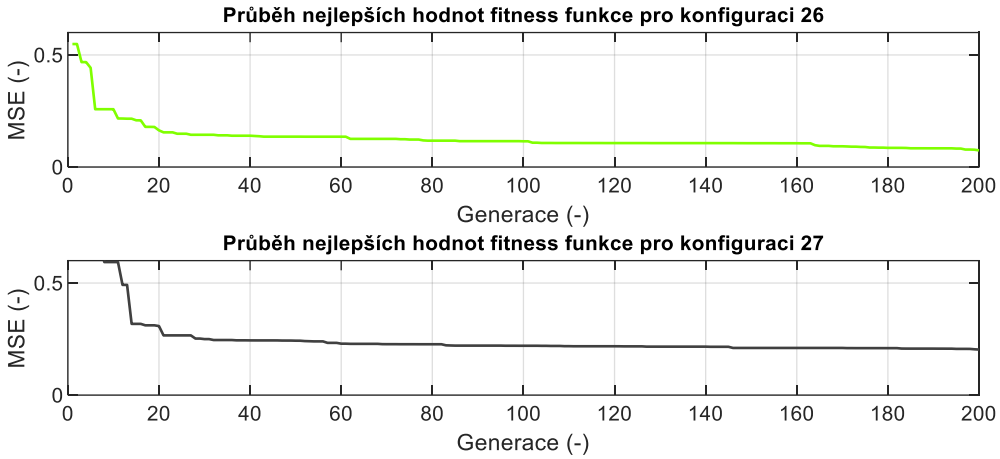
Příloha 2 Průběhy nejlepších hodnot fitness funkce
pro jednotlivé konfigurace se 100 generacemi



Příloha 2 Průběhy nejlepších hodnot fitness funkce
pro jednotlivé konfigurace se 100 generacemi – pokračování



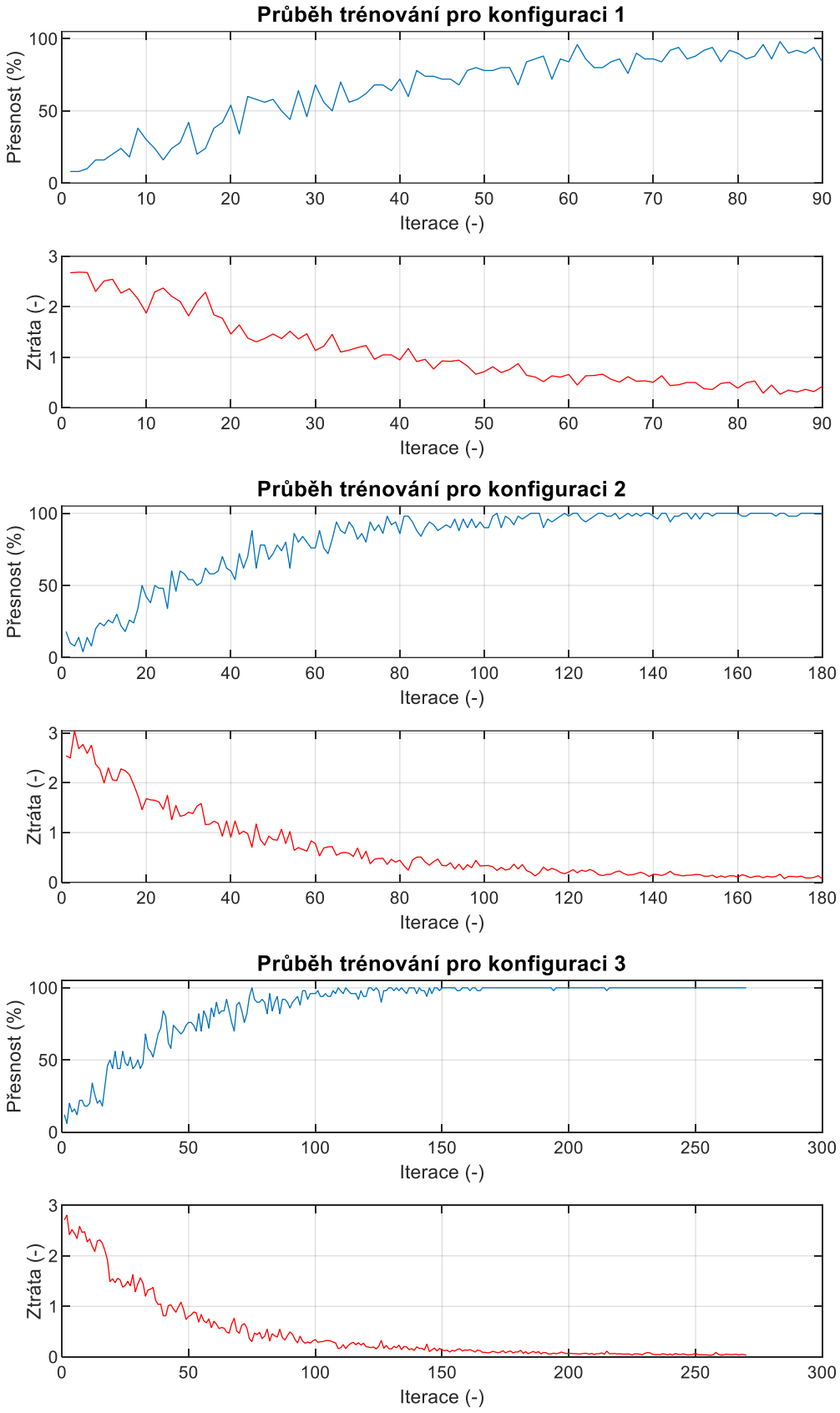
Příloha 3 Průběhy nejlepších hodnot fitness funkce pro jednotlivé konfigurace s 200 generacemi



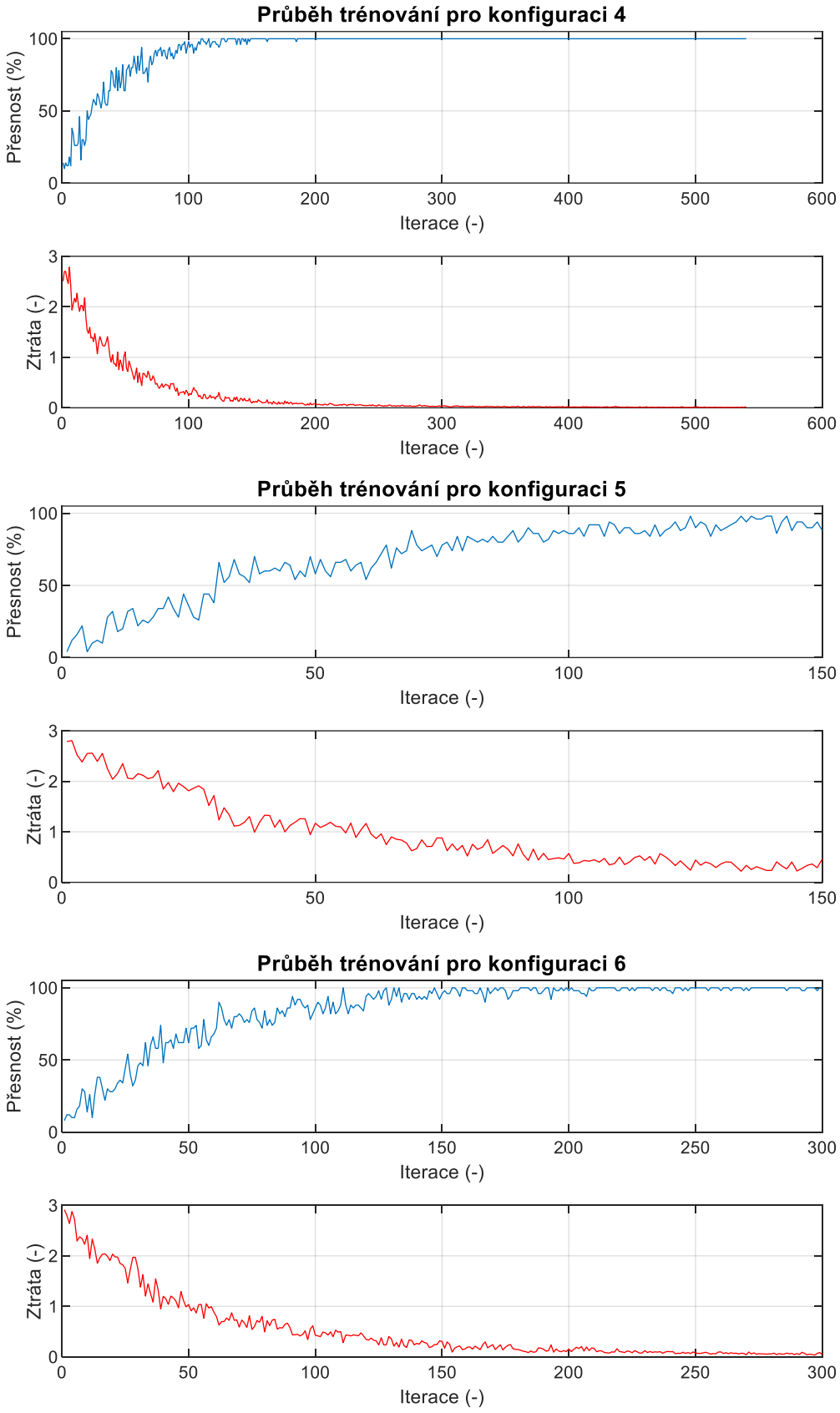
Příloha 3 Průběhy nejlepších hodnot fitness funkce
pro jednotlivé konfigurace s 200 generacemi – pokračování

Příloha II.II **Grafy k laboratorní úloze 3**

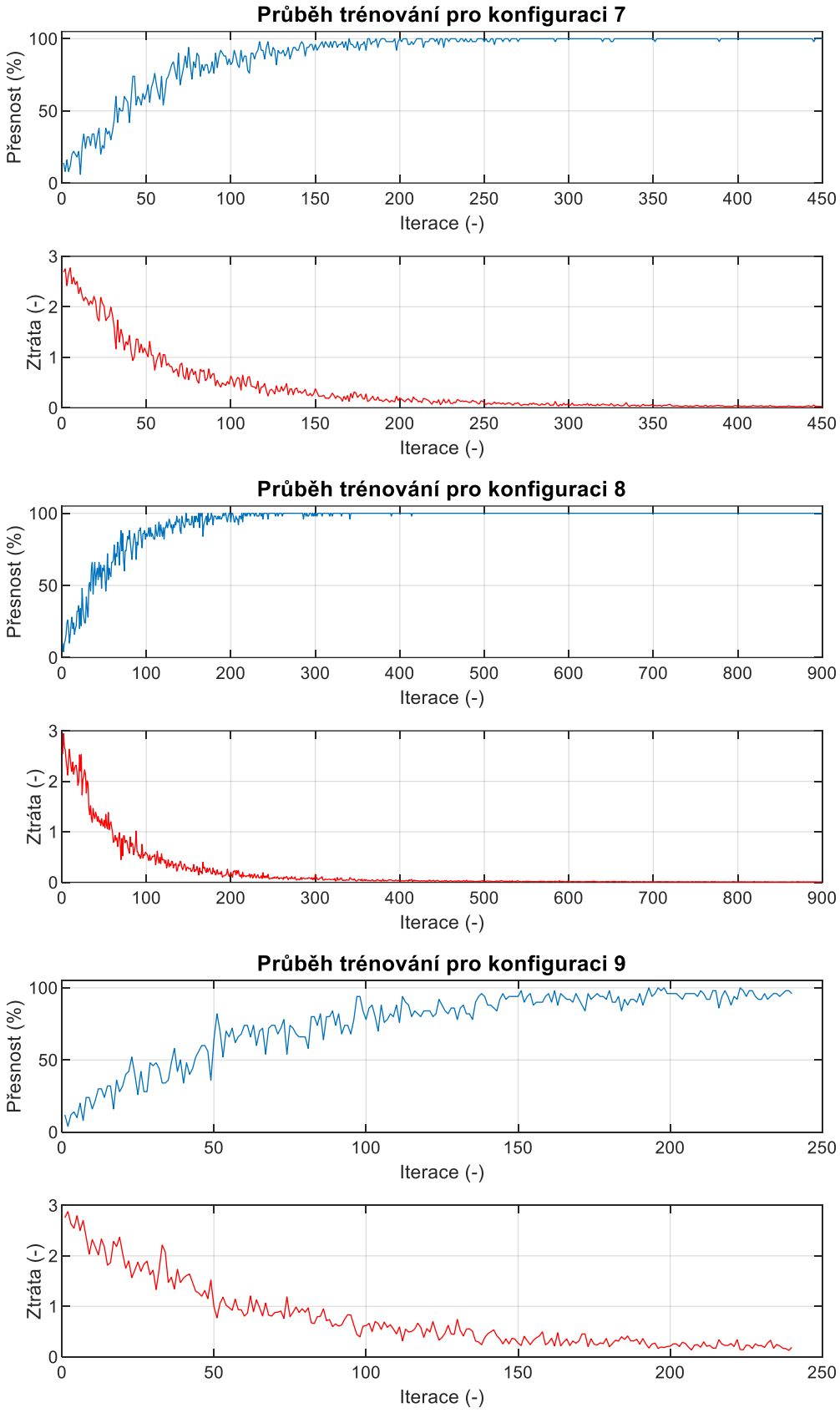
V této části jsou zpočátku grafy průběhu tréninku CNN pro jednotlivé konfigurace. Dále jsou obsaženy všechny matice záměn. Všechny grafy průběhu trénovaní a matice záměn jsou také uloženy pod názvy *LU3_konf_x_Training_progress* a *LU3_konf_x_Conf_chart*, kde *x* značí konfiguraci.



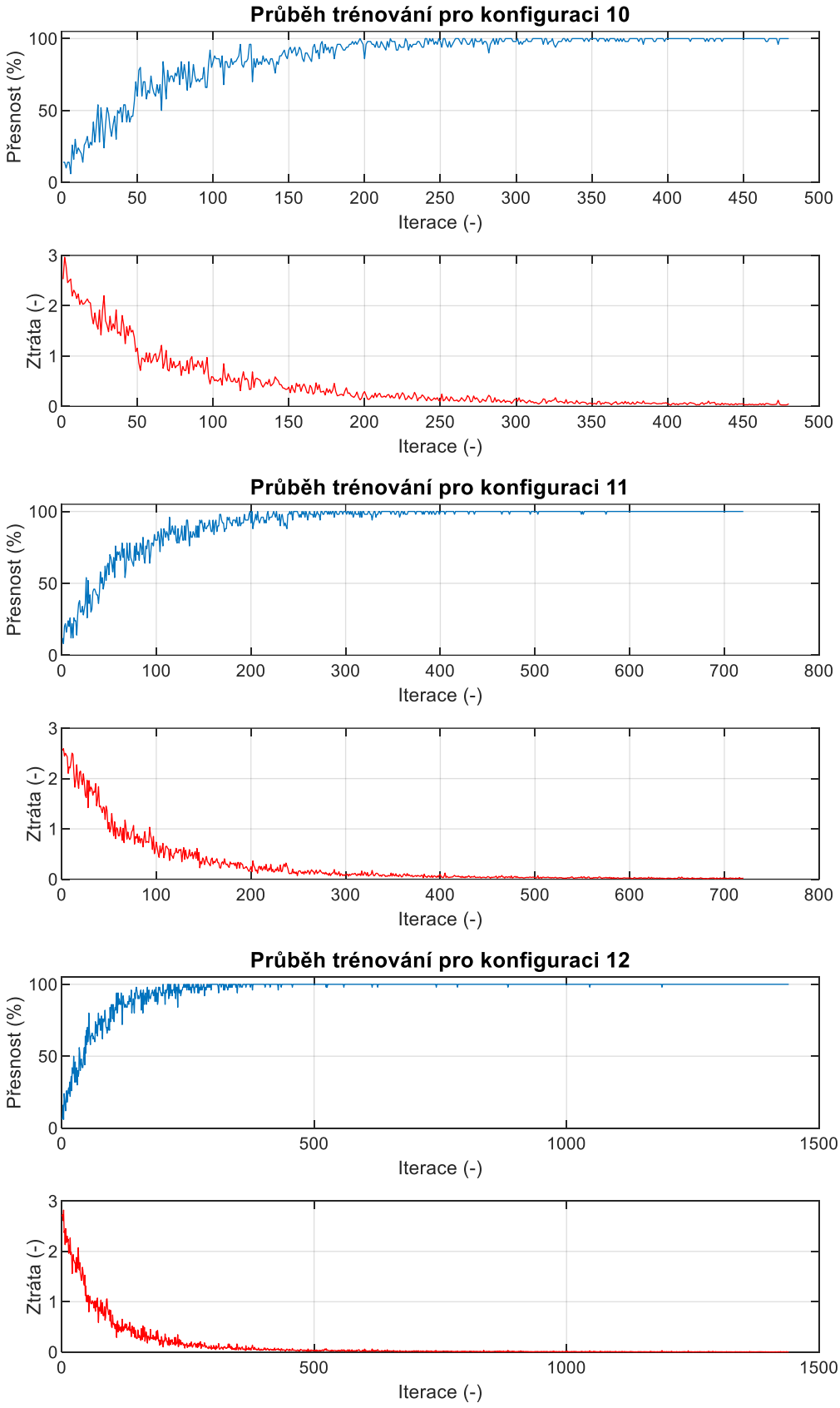
Příloha 4 Průběhy trénování CNN pro všechny konfigurace



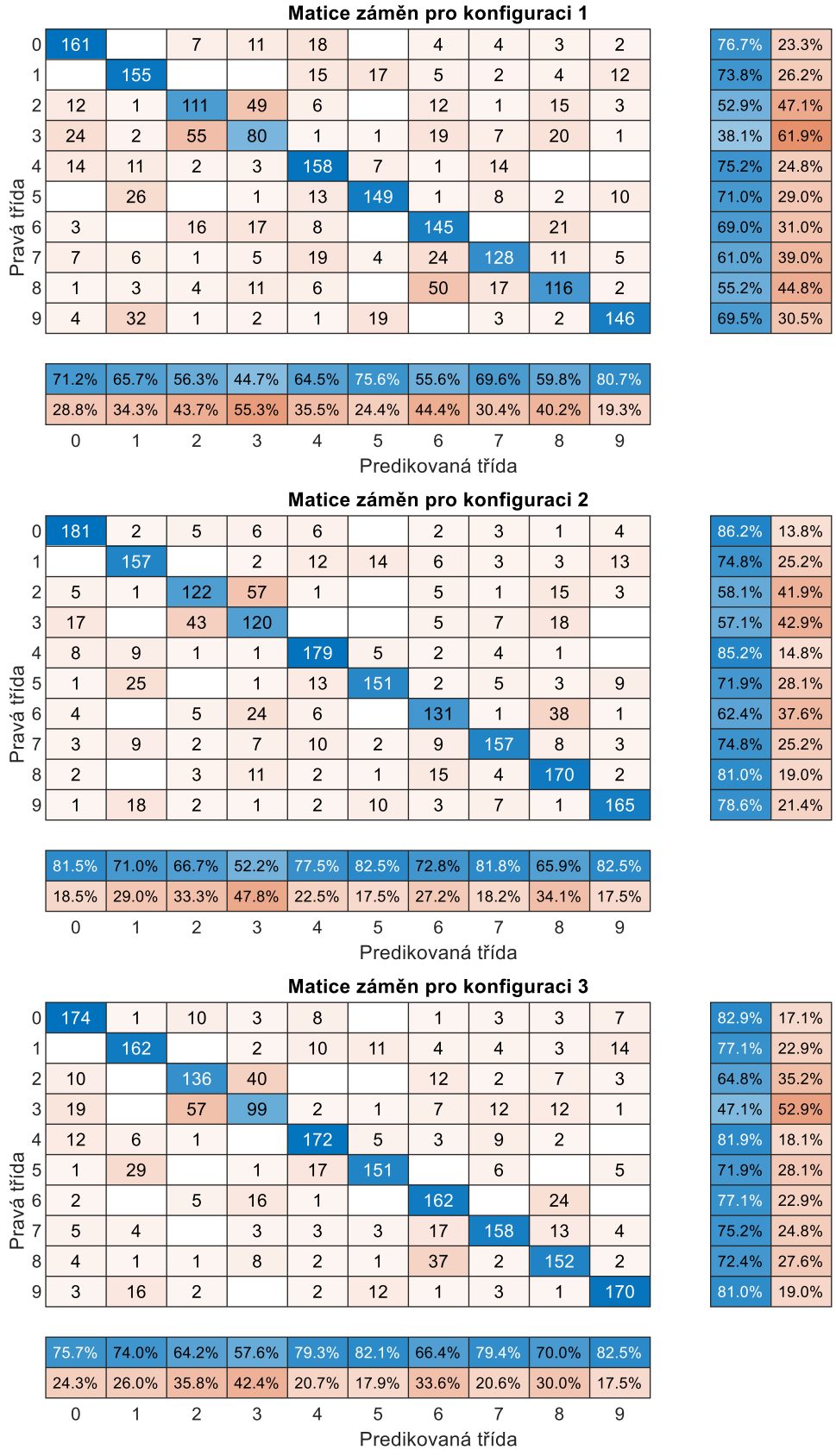
Příloha 4 Průběhy trénování CNN pro všechny konfigurace – pokračování



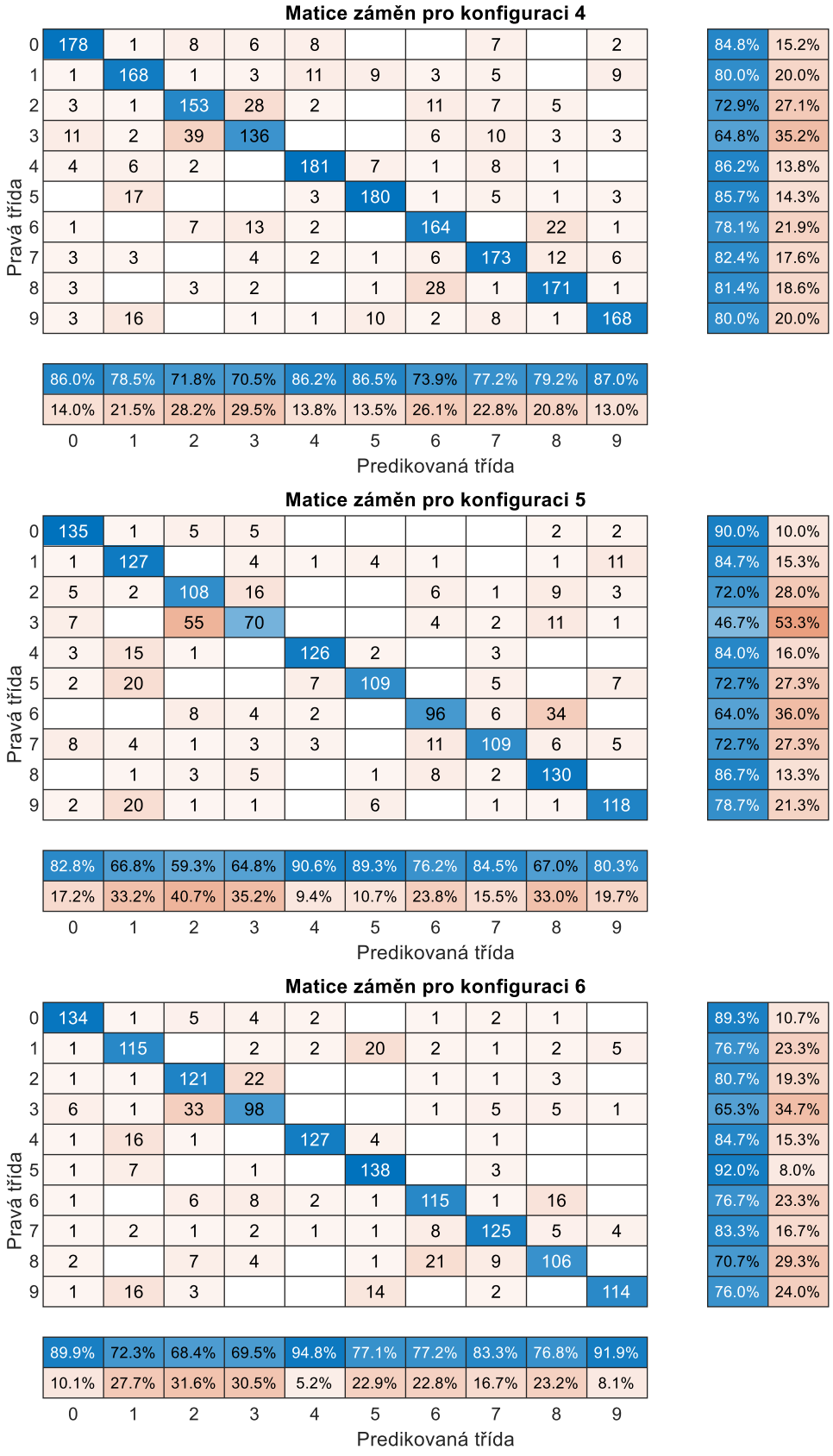
Příloha 4 Průběhy trénování CNN pro všechny konfigurace – pokračování



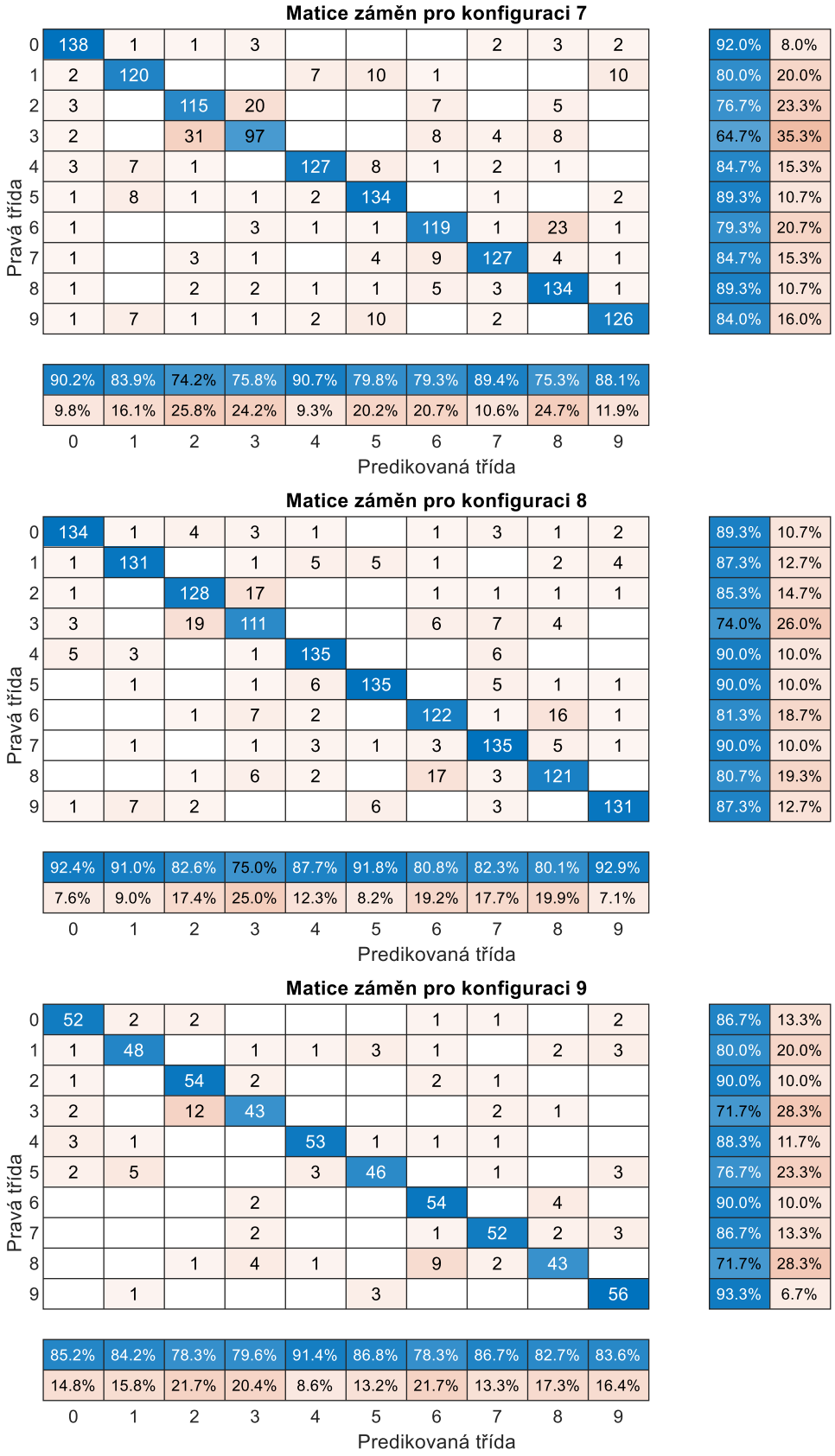
Příloha 4 Průběhy trénování CNN pro všechny konfigurace – pokračování



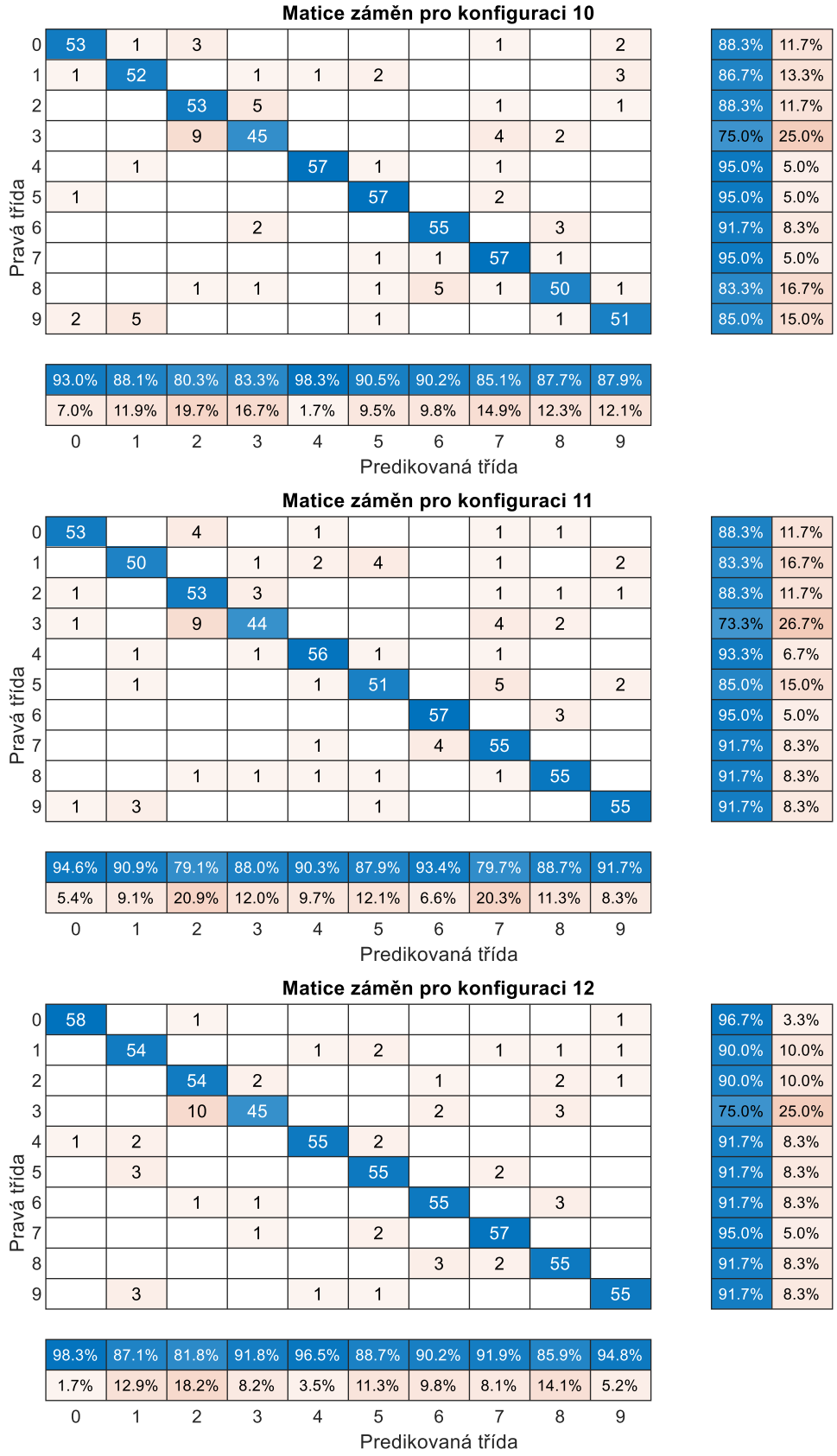
Příloha 5 Matice záměn CNN pro všechny konfigurace



Příloha 5 Matice záměn CNN pro všechny konfigurace – pokračování



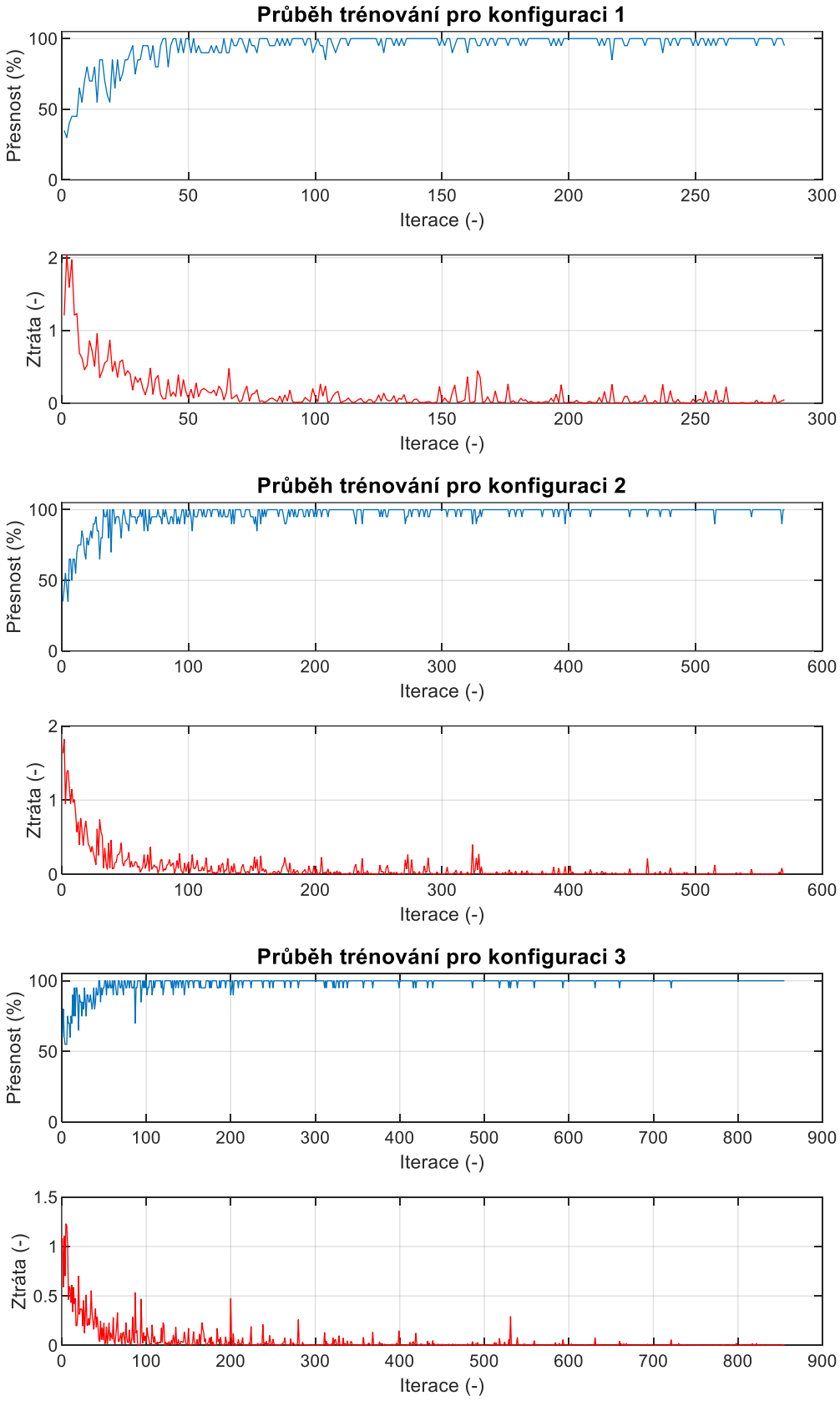
Příloha 5 Matice záměn CNN pro všechny konfigurace – pokračování



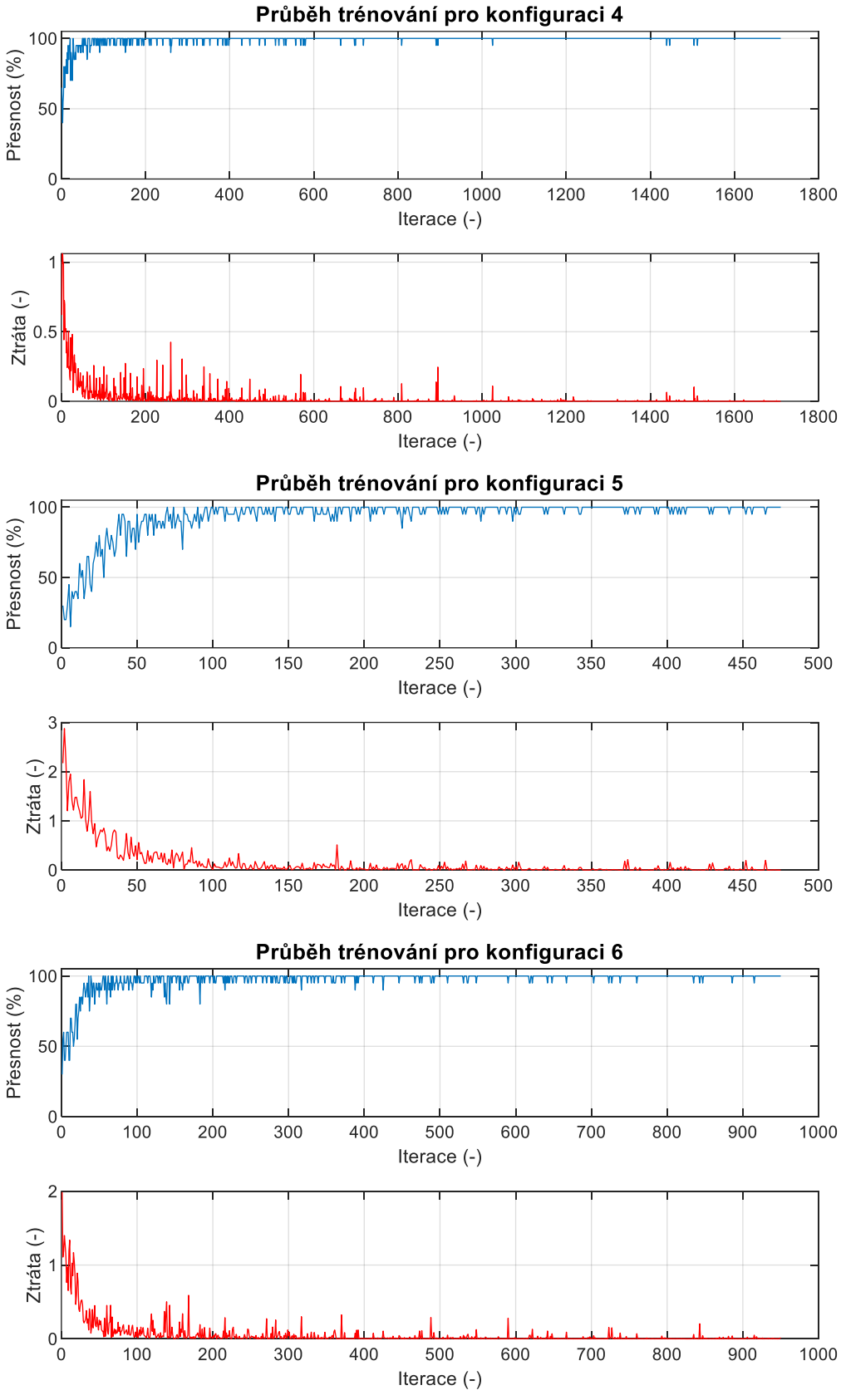
Příloha 5 Matice záměn CNN pro všechny konfigurace – pokračování

Příloha II.III **Grafy k laboratorní úloze 4**

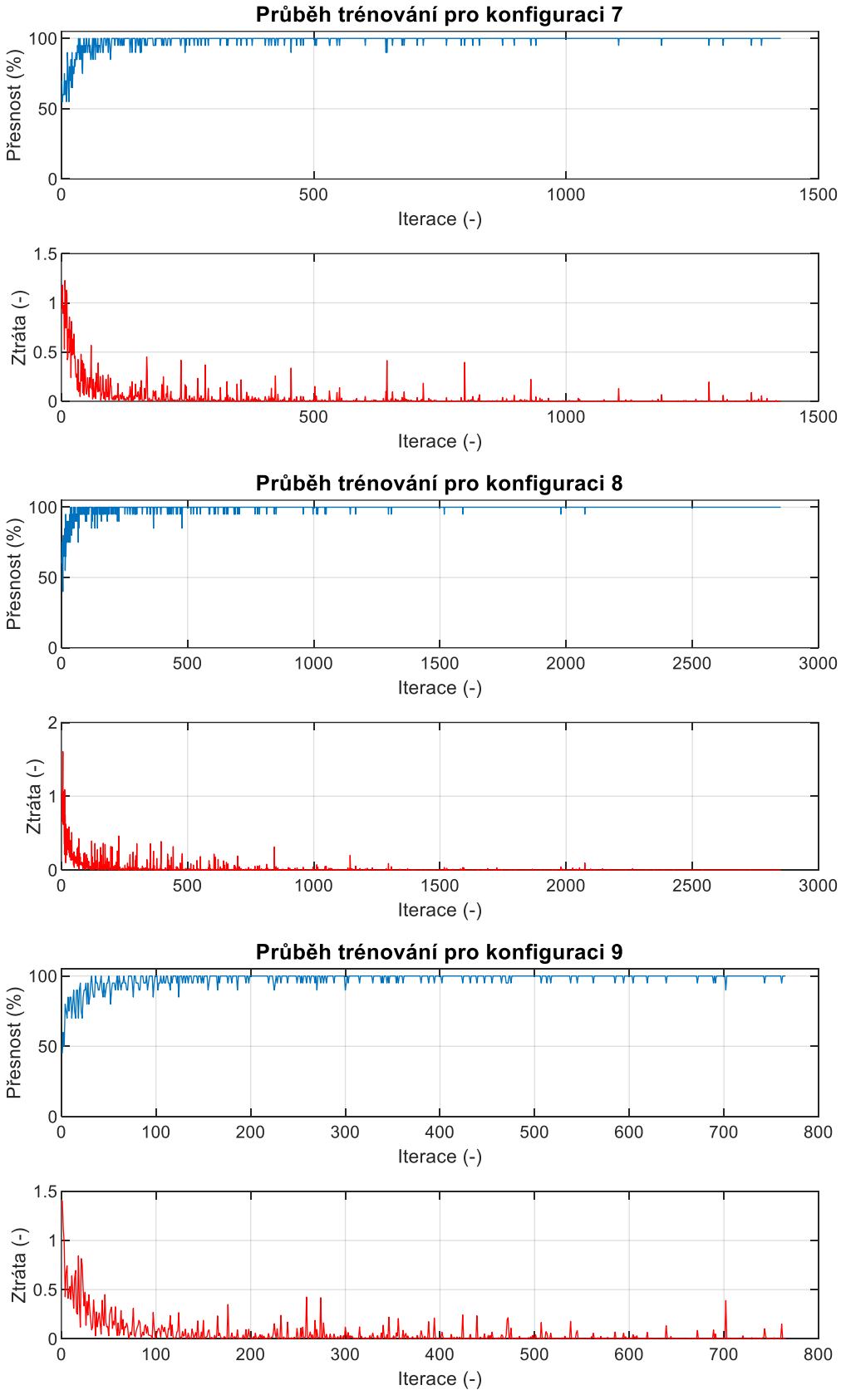
V této části jsou zpočátku grafy průběhu tréninku GoogLeNet pro jednotlivé konfigurace. Dále jsou pak všechny matice záměn. Všechny grafy průběhu trénovaní a matice záměn jsou také uloženy pod názvy *LU4_konf_x_Training_progress* a *LU4_konf_x_Conf_chart*, kde *x* značí konfiguraci.



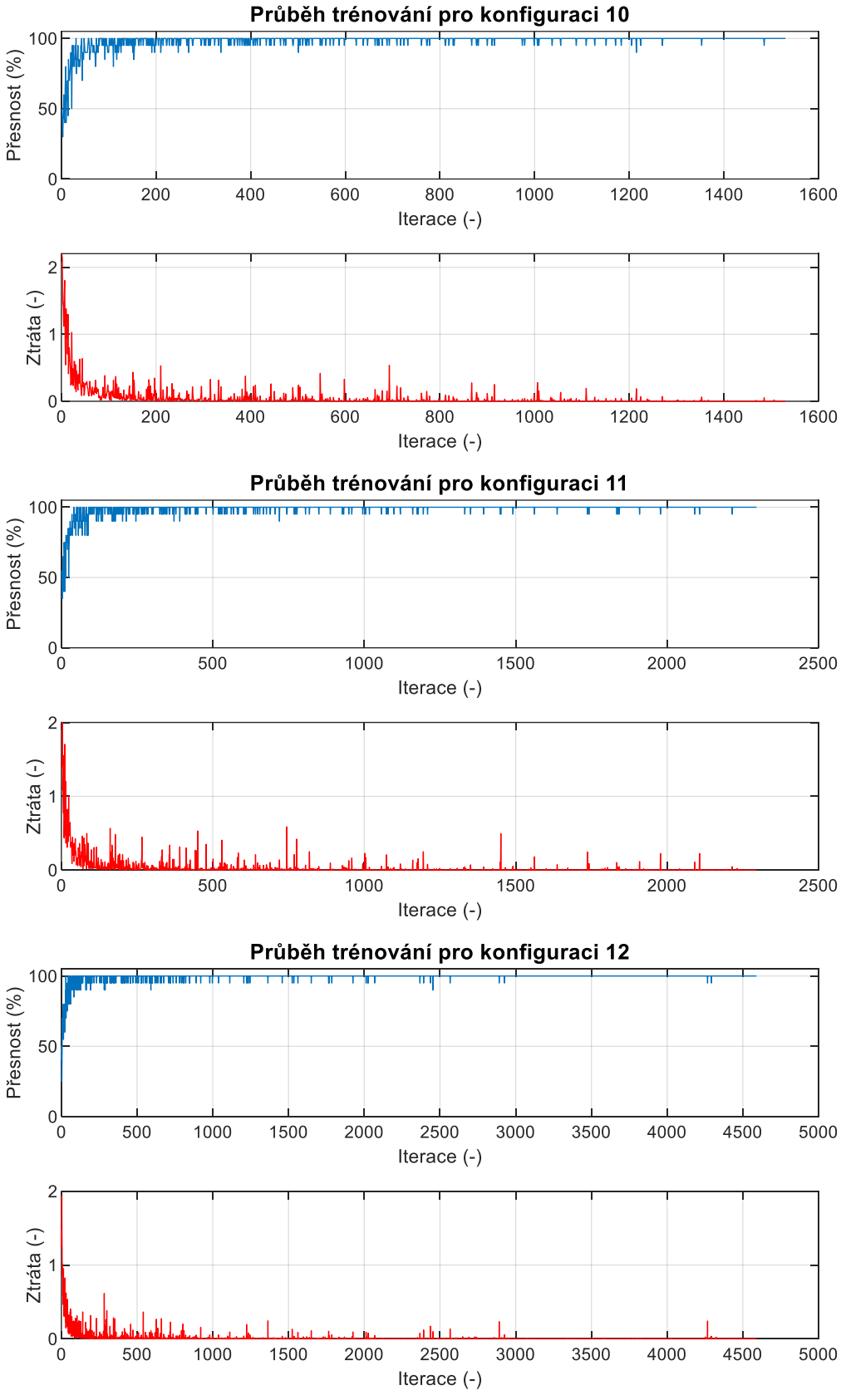
Příloha 6 Průběhy trénování GoogLeNet pro všechny konfigurace



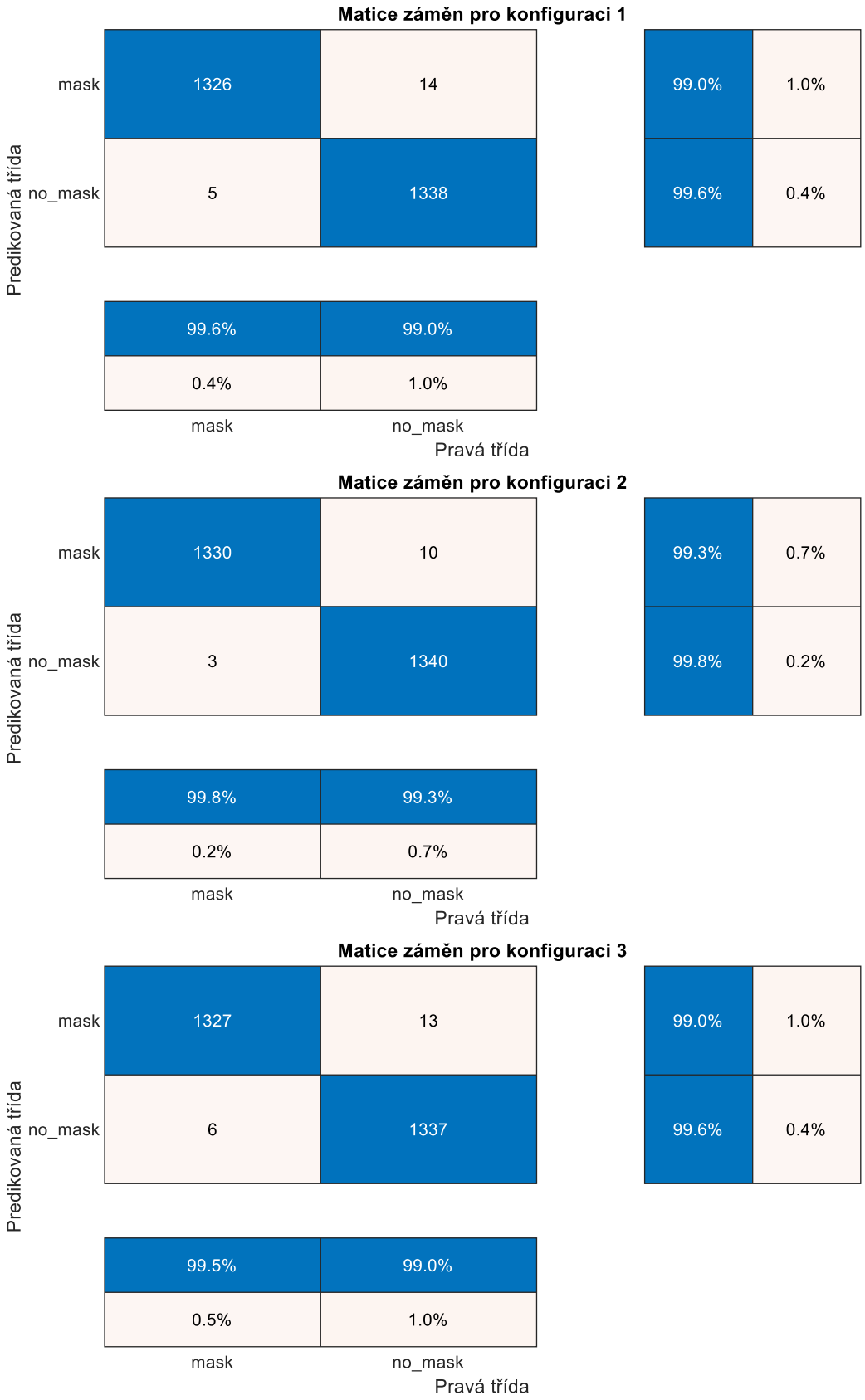
Příloha 6 Průběhy trénování GoogLeNet pro všechny konfigurace – pokračování



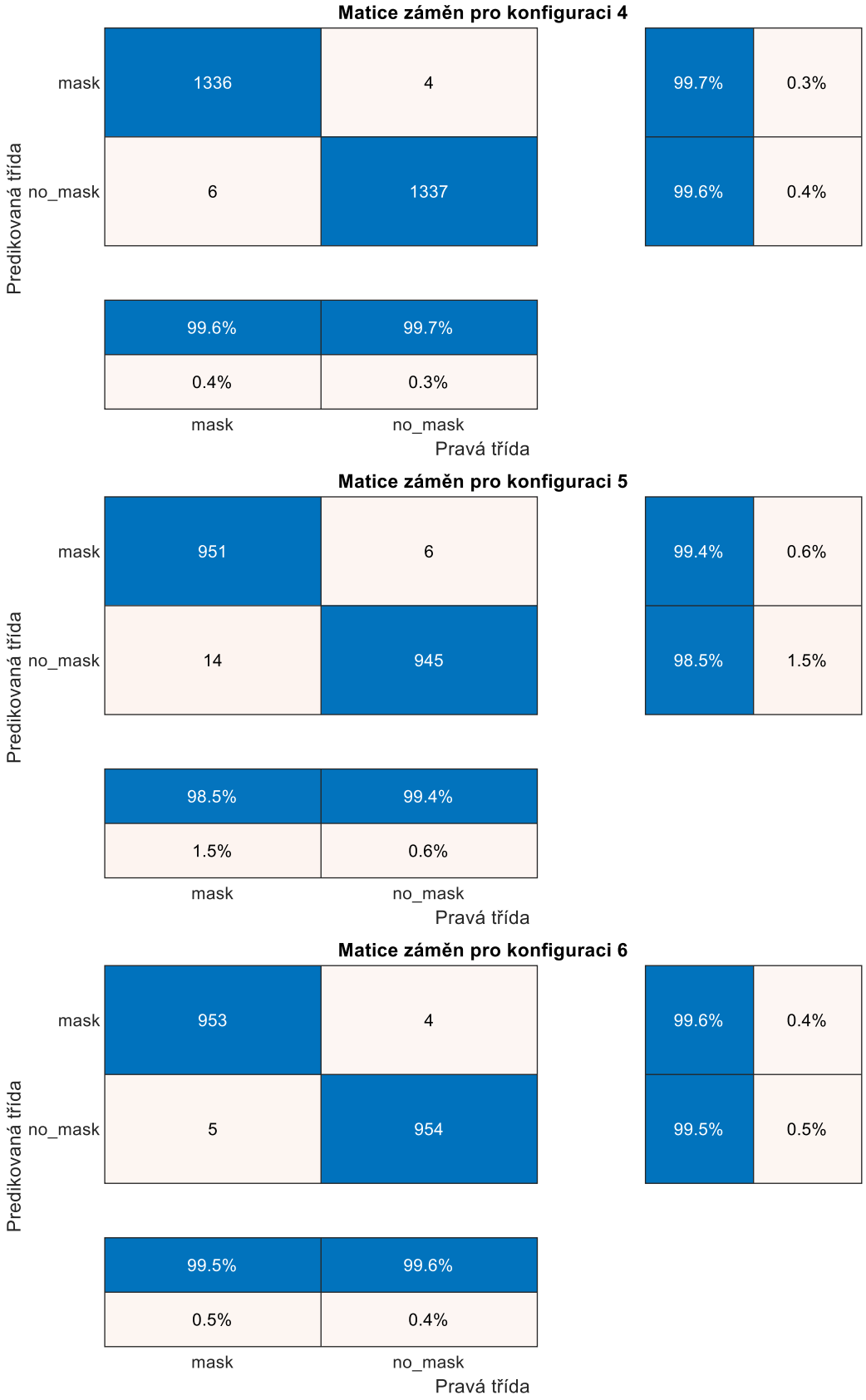
Příloha 6 Průběhy trénování GoogLeNet pro všechny konfigurace – pokračování



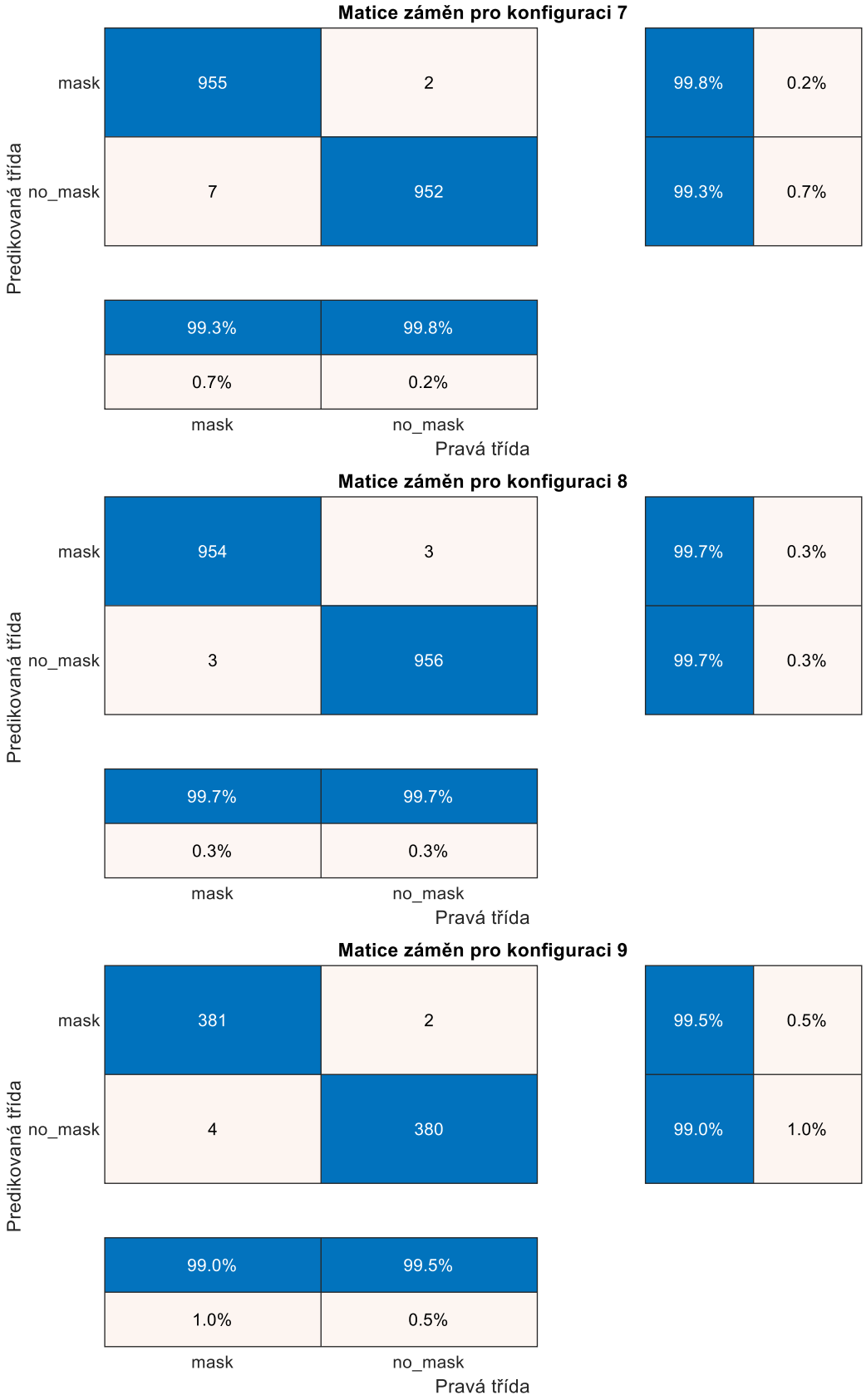
Příloha 6 Průběhy trénování GoogLeNet pro všechny konfigurace – pokračování



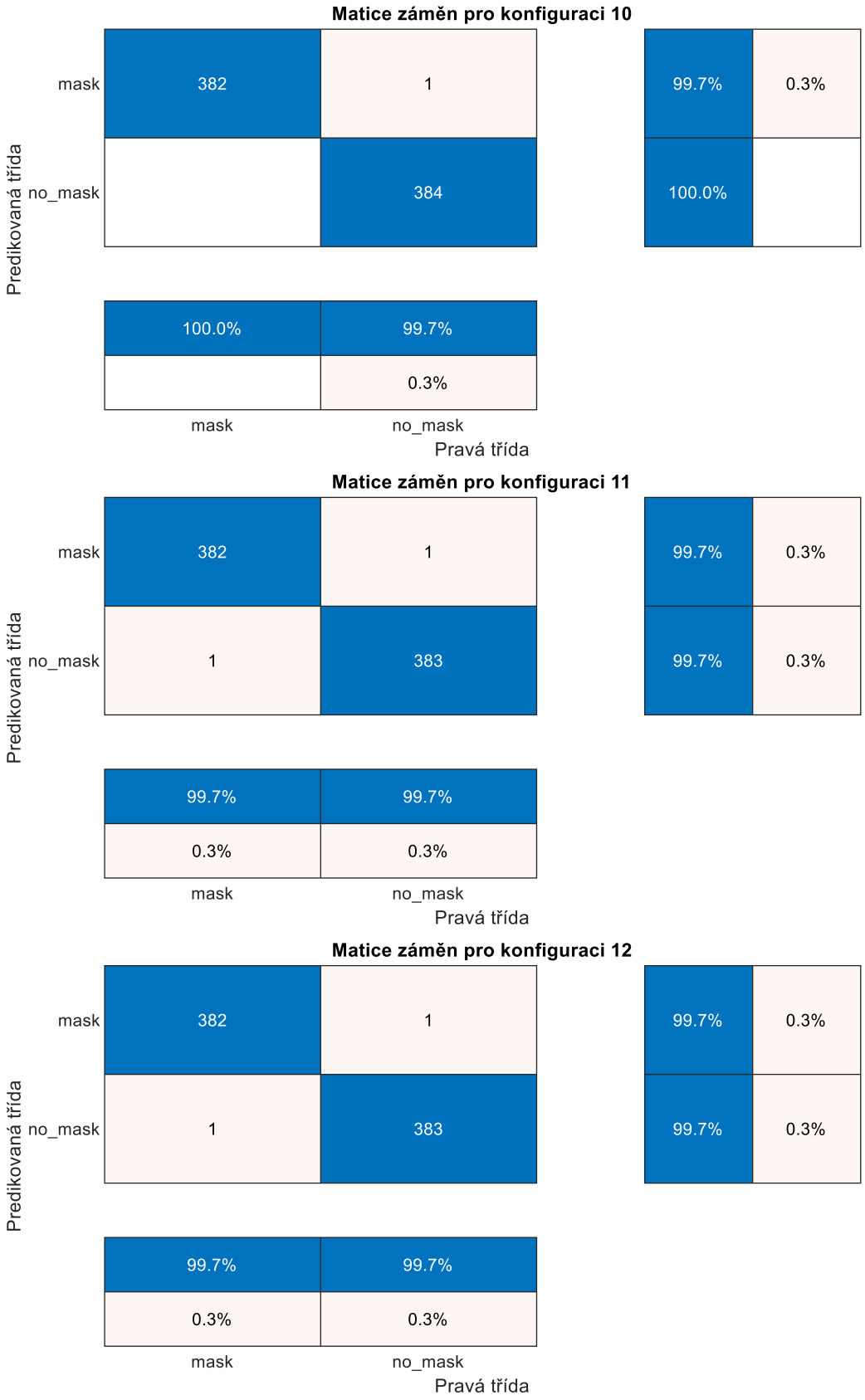
Příloha 7 Matice záměn GoogLeNet pro všechny konfigurace



Příloha 7 Matice záměn GoogLeNet pro všechny konfigurace –pokračování



Příloha 7 Matice záměn GoogLeNet pro všechny konfigurace –pokračování



Příloha 7 Matice záměn GoogLeNet pro všechny konfigurace –pokračování