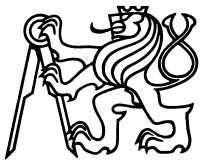
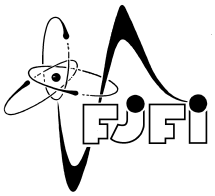


České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská



Metody strojového učení v částicové fyzice

Machine Learning Techniques in High Energy Physics

Diplomová práce

Autor: **Bc. Miroslav Kubů**

Vedoucí práce: **Ing. Petr Bouř**

Akademický rok: 2020/2021

České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská

Katedra: matematiky Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student:	Bc. Miroslav Kubů
Studijní program:	Aplikace přírodních věd
Obor:	Aplikované matematicko-stochastické metody
Název práce (česky):	Metody strojového učení v částicové fyzice
Název práce (anglicky):	Machine Learning Techniques in High Energy Physics

Pokyny pro vypracování:

1. Navažte na Váš výzkum prováděný v bakalářské práci "Konvoluční neuronové sítě v částicové fyzice" a ve výzkumném úkolu "Pokročilé techniky neuronových sítí v částicové fyzice".
2. Nastudujte a popište nejnovější techniky v problematice daných klasifikačních úloh při identifikaci částic na experimentu NOvA.
3. Kromě neuronových sítí se nyní soustřeďte i na odlišné typy klasifikačních algoritmů, např. Random Forest, AdaBoost, Gradient Boosting Machine atd. Vybrané algoritmy popište, srovnejte a okomentujte jejich vhodné užití.
4. Pokud to vybrané algoritmy umožňují, prostudujte rovněž jejich modifikace vedoucí k vyšší efektivitě výsledné klasifikace (např. specifické regularizační techniky).
5. S pomocí dostupných open source knihoven demonstrujte Vámi vybrané charakteristiky popisovaných klasifikačních algoritmů a otestujte vhodnost užití pro různé klasifikační úlohy.
6. V případě dostupnosti relevantních dat z částicové fyziky proveďte experimentální aplikaci Vámi popisovaných metod.

Doporučená literatura:

1. M. A. Acero et al., (NOvA Collaboration). First measurement of neutrino oscillation parameters using neutrinos and antineutrinos by NOvA. FERMILAB-PUB-19-272-ND, 2019. Arxiv: 1906.04907.
2. F. Psihas et al., Context-Enriched Identification of Particles with a Convolutional Network for Neutrino Events. FERMILAB-PUB-19-258-PPD, 2019. Arxiv: 1906.00713.
3. I. Goodfellow, Y. Bengio, A. Courville, Deep Learning. MIT Press, 2016.
4. A. Aurisano et al., A Convolutional Neural Network Neutrino Event Classifier. Journal of Instrumentation 11, 2016, P09001.
5. P. Baldi, P. Sadowski, D. Whiteson, Searching for Exotic Particles in High Energy Physics with Deep Learning. Nature Commun. 5, 2014.
6. M. Bishop, Neural Networks for Pattern Recognition. Oxford press, 1995

Jméno a pracoviště vedoucí diplomové práce:

Ing. Petr Bouř
Katedra matematiky , FJFI ČVUT v Praze, Trojanova 13, 120 00 Praha 2

Jméno a pracoviště konzultanta:

Datum zadání diplomové práce: 31.10.2019

Datum odevzdání diplomové práce: 4.5.2020

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 15. října 2019

.....
garant oboru

.....
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta jaderná a fyzikálně inženýrská
-1-

.....
děkan

Poděkování:

Chtěl bych zde poděkovat především svému školiteli Ing. Petru Bouřovi za jeho nekonečnou trpělivost, ochotu a cenné rady při vedení diplomové práce.

Čestné prohlášení:

Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 6. ledna 2021 Miroslav Kubů

Název práce:

Metody strojového učení v částicové fyzice

Autor: Miroslav Kubů

Obor: Aplikované matematicko-stochastické metody

Druh práce: Diplomová práce

Vedoucí práce: Ing. Petr Bouř, ČVUT v Praze, FJFI, KM

Abstrakt: Klasifikace částicových interakcí je jednou ze základních úloh analýzy dat v částicové fyzice. V neutrinové fyzice je přitom tato úloha velice podobná úloze rozpoznávání obrazu. Z tohoto důvodu se v této práci zajímáme o klasifikaci neutrinových interakcí v detektorech experimentu NOvA laboratoře Fermilab pomocí konvolučních neuronových sítí (CNN). Právě CNN jsou v současnosti díky své schopnosti extrahovat vizuální příznaky ze surových obrazových dat používány pro úlohu klasifikace neutrinových interakcí na experimentu NOvA. Pro tuto úlohu proto představujeme porovnání moderních CNN architektur. Výslednou přesnost klasifikace přitom porovnáváme s architekturou tzv. CVN, aktuálně používané na experimentu NOvA [2]. Dále představujeme modely kombinující extrakci vizuálních příznaků skrze CNN a jejich klasifikaci skrze modely založené na technice rozhodovacích stromů.

Klíčová slova: částicová fyzika, konvoluční neuronové sítě, neutrino, residuální neuronové sítě, rozhodovací stromy, rozpoznávání obrazu, strojové učení

Title:

Machine Learning Techniques in High Energy Physics

Author: Miroslav Kubů

Abstract: Classification of particle interactions is a crucial task in high energy physics. In neutrino physics, this classification task is similar to image recognition tasks in many ways. Therefore, we are interested in the application of convolutional neural networks (CNN) to the classification of neutrino interactions for data from the NOvA experiment. In particular, CNN can extract visual features from image pixel maps at different scales and use these features for particle identification. We present a comparison of the state-of-the-art CNN architectures for neutrino interaction classification. Furthermore, we compare these models to the so-called CVN architecture used at NOvA. Moreover, we construct hybrid models using ensemble decision tree classifiers on top of the features extracted by our CNN model.

Key words: high energy physics, convolutional neural networks, neutrino, residual neural networks, decision trees, image recognition, machine learning

Obsah

Seznam zkratek

ANN umělé neuronové sítě (artificial neural networks)

CC charged current

CNN konvoluční neuronové sítě (convolutional neural networks)

CVN convolutional visual network

DL hluboké učení (deep learning)

FNAL Fermi National Accelerator Laboratory

GD gradient descent

HEP částicová fyzika (high energy physics)

ML strojové učení (machine learning)

MLP multi-layer perceptron

NC neutral current

NOvA NuMI Off-axis ν_e Appearance

NuMI neutrinos at the main injector

PDF hustota pravděpodobnosti (probability density function)

ResNet residuální neuronové sítě

SGD stochastic gradient descent

Úvod

Klasifikace částicových interakcí je jednou z klíčových úloh v částicové fyzice (high energy physics, HEP). Vzhledem k velkému množství dat se k tomuto účelu s ohledem na automatizaci úlohy často používají metody strojového učení (machine learning, ML). V rámci této práce se proto zaměřujeme na aplikaci ML metod pro klasifikaci dat z HEP.

Naším cílem je provést klasifikaci na datech z neutrinového experimentu NOvA v americké laboratoři Fermi National Accelerator Laboratory (FNAL). Úloha klasifikace neutrinových interakcí se velice podobá úloze rozpoznávání obrazu. Z tohoto důvodu jsou pro tyto účely v současnosti používány modely konvolučních neuronových sítí (CNN) [2]. Právě CNN jsou tak středobodem našeho zájmu. CNN jsou totiž schopné ze surových obrazových dat extrahovat relevantní vizuální příznaky, s jejichž využitím jsou následně data klasifikována. V této práci proto navazujeme na teoretické poznatky o klasifikaci neutrinových interakcí pomocí CNN z [16], přičemž nabyté poznatky dále rozšiřujeme o pokročilé CNN architektury a další ML techniky s cílem využití pro úlohu vícerozměrné statistické klasifikace dat z experimentu NOvA.

Práce obsahuje teoretickou a implementační část. V úvodní kapitole 1 popisujeme technické pozadí experimentu NOvA, druhy pozorovatelných neutrinových interakcí a podobu dat. Dále v kapitole 2 rozšiřujeme úlohu binární statistické klasifikace z [16] na úlohu vícerozměrné statistické klasifikace. V kapitole 3 pak přímo navazujeme na teoretické poznatky o CNN z [16] a rozšiřujeme je o moderní architektury využívající tzv. inception moduly či residuální učení. Tyto techniky jsou buď částečně použity pro klasifikaci dat na experimentu NOvA, nebo se o jejich použití v současnosti uvažuje. Následně v kapitole 4 představujeme další klasifikační ML modely. Popisujeme přitom koncept rozhodovacích stromů a jejich rozšíření využívající tzv. bagging či boosting. Tyto koncepty představujeme s cílem použití pro klasifikaci příznaků extrahovaných pomocí CNN.

V závěrečné kapitole 5 poté využíváme teoretické poznatky z předchozích kapitol k implementaci modelů pro klasifikaci neutrinových interakcí z experimentu NOvA. Porovnáváme přitom náš model residuální neuronové sítě (ResNet) s modelem tzv. convolutional visual network (CVN), v současnosti používaným na experimentu NOvA. Následně tvoříme hybridní model, který využívá model ResNet pro extrakci příznaků a model založený na rozhodovacích stromech pro jejich klasifikaci. Celkově takto konstruujeme tři hybridní modely využívající pro klasifikaci příznaků koncepty tzv. náhodného lesa, AdaBoost a Gradient Boosting. Výslednou přesnost klasifikace hybridních modelů poté porovnáváme s přesností modelů ResNet a CVN.

Kapitola 1

Metody klasifikace dat v rámci experimentu NOvA

V úvodní kapitole přibližujeme fyzikální pozadí a technické detaily experimentu NOvA. Právě data z experimentu NOvA totiž budeme v závěrečné kapitole 5 analyzovat. Z tohoto důvodu přibližujeme jednotlivé typy neutrinových interakcí a způsoby jejich zpracování jako obrazových dat v detektorech experimentu NOvA. Dále pokračujeme popisem úlohy klasifikace dat z experimentu NOvA s důrazem na aktuálně používané modely konvolučních neuronových sítí (CNN).

1.1 Úvod do fyziky neutrin

NuMI Off-axis ν_e Appearance (NOvA) je jedním z experimentů americké laboratoře Fermi National Accelerator Laboratory (FNAL) využívajících neutrinový svazek neutrinos at the main injector (NuMI). Cílem tohoto výzkumu je popis chování elementárních částic zvaných neutrina. Neutrina jsou částice, dle standardního modelu [1] patřící do skupiny leptonů, které vznikají při jaderných reakcích. Elektrický náboj neutrin je nulový, a proto na ně nepůsobí elektromagnetická síla. Neutrina nepodléhají silné jaderné interakci a v porovnání s jinými elementárními částicemi se vyznačují extrémně malou hmotností. Kvůli tomu neutrina prakticky vůbec nereagují s okolním prostředí a je velice obtížné je detekovat. Neutrinové detektory jsou tak zpravidla velice rozměrná zařízení, která jsou navíc s ohledem na odstínění ostatních částic často ukryta pod zemí. Právě velké rozměry detektorů jsou nezbytné pro zvýšení četnosti detekce takto málo interagující částice.

V současnosti pozorujeme tři typy (též označované jako vůně) neutrin - mionové neutrino ν_μ, elektronové neutrino ν_e a tauonové neutrino ν_τ. Ke každé vůni následně existuje též příslušné antineutrino. Unikátní vlastností neutrin je kvantově mechanický jev tzv. oscilace neutrin, během něhož neutrino mění svou vůni. Základní myšlenku modelování oscilace je možno popsat pomocí soustav tzv. směšovacích rovnic (v angličtině mixing equations) (1.1), (1.2) a (1.3) [27], které s využitím Diracovy notace charakterizují jednotlivé tzv. vlastní stavy vůní ν_e, ν_μ a ν_τ na základě superpozice tzv. vlastních stavů hmotnosti ν_1, ν_2 a ν_3.

$$|\nu_e\rangle = U_{11}\,|\nu_1\rangle + U_{12}\,|\nu_2\rangle + U_{13}\,|\nu_3\rangle\,, \tag{1.1}$$

$$|\nu_\mu\rangle = U_{21}\,|\nu_1\rangle + U_{22}\,|\nu_2\rangle + U_{23}\,|\nu_3\rangle\,, \tag{1.2}$$

$$|\nu_\tau\rangle = U_{31}\,|\nu_1\rangle + U_{32}\,|\nu_2\rangle + U_{33}\,|\nu_3\rangle\,. \tag{1.3}$$

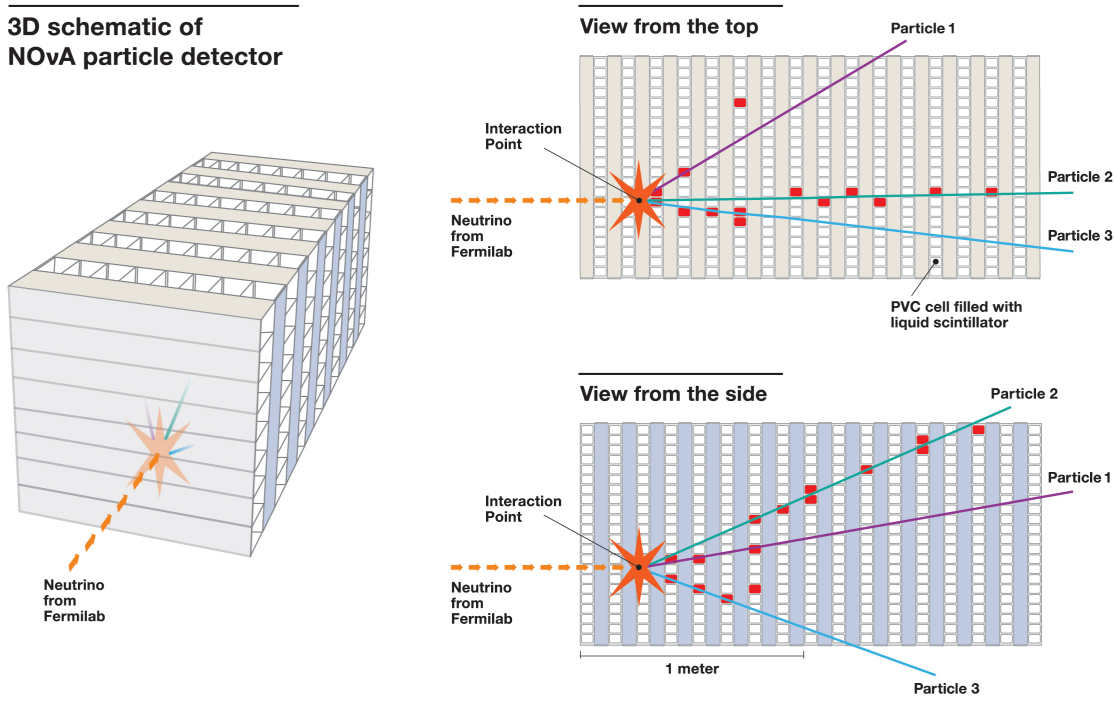
Koeficienty příslušné vlastním stavům hmotnosti odpovídají pravděpodobnostem pozorování vlastního stavu vůně s příslušnou hmotností. Jejich hodnoty poté odpovídají prvkům matice $\boldsymbol{U}$ vzniklé násobením tzv. matic rotací $\nu_1 \to \nu_2$, $\nu_2 \to \nu_3$ a $\nu_3 \to \nu_1$. Toto maticové násobení lze zapsat jako

$$\boldsymbol{U} = \begin{pmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{23} & s_{23} \\ 0 & -s_{23} & c_{23} \end{pmatrix} \begin{pmatrix} c_{31} & 0 & s_{31}e^{-i\delta} \\ 0 & 1 & 0 \\ -s_{31}e^{i\delta} & 0 & c_{31} \end{pmatrix}, \tag{1.4}$$

kde s využitím notace [27] značíme $c_{ij} = \cos(\theta_{ij})$ a $s_{ij} = \sin(\theta_{ij})$ pro příslušné indexy $i, j \in \{1, 2, 3\}$. Tzv. směšovací úhly θ_{12}, θ_{23} a θ_{31} poté určují pravděpodobnosti přechodů stavů neutrin a jejich hodnoty jsou společně s velikostí tzv. fázového faktoru δ cílem experimentálního i teoretického výzkumu na experimentu NOvA. Detailní popis procesu oscilace neutrin je možno najít v [27].

1.2 Experiment NOvA

Hlavní myšlenkou experimentu NOvA je vyslání neutrinového svazku NuMI skrze dvojici funkčně identických detektorů a porovnání složení svazku v obou detektorech. Svazek vysílaný z FNAL je z drtivé většiny tvořený mionovými neutriny ν_μ s příměsí malého množství elektronových neutrin ν_e a stopového množství neutrin tauonových ν_τ. První, tzv. blízký detektor, leží v podzemí blízko zdroje svazku v areálu FNAL. Jeho cílem je tedy přesný popis složení neutrinového svazku. Druhý, tzv. vzdálený detektor, leží v laboratoři Ash River ve státě Minnesota. Svazek tak mezi blízkým a vzdáleným detektorem urazí zhruba 800 kilometrů dlouhou trasu skrze skalnaté podloží. Vzdálený detektor tak může monitorovat změny ve složení svazku způsobené oscilací $\nu_\mu \to \nu_e$. Jak již bylo naznačeno, oba detektory jsou funkčně velice podobné. Kostra detektorů je tvořena systémem vertikálně a horizontálně položených trubek z PVC. Jak ukazuje obrázek 1.1, paralelní buňky jsou poté uspořádány do rovnoběžných desek, ze kterých je detektor složen. Jednotlivé desky jsou tedy poskládány za sebe takovým způsobem, aby bylo možno pořizovat snímky shora i ze strany detektoru. Rovinu pro snímky ze strany detektoru označujeme jako X-view, zatímco rovinu pro pohled shora nazýváme Y-view. Buňky z PVC jsou následně vyplněny tekutým scintilátorem. Nabité částice prolétávající detektorem poté reagují se scintilátorem v podobě emise fotoelektronů. Ty jsou následně smyčkou optických vláken zachyceny, díky čemuž lze rekonstruovat podobu částicové interakce v detektoru napříč buňkami.



Obrázek 1.1: Schéma konstrukce neutrinových detektorů experimentu NOvA. Obrázek vlevo zachycuje strukturu detektoru poskládaného z PVC buněk s indikací směru neutrinového svazku vedeného z FNAL. Obrázky vpravo poté ukazují zachycenou interakci z pohledů X-view (dole) a Y-view (nahoře). Na obrázcích je vyznačen bod interakce (Interaction Point) částic z neutrinového rozpadu s detektorem a směr při interakci vzniklých částic (Particle). Zvýrazněné buňky poté reprezentují buňky, ve kterých částice předala dostatečné množství energie pro zachycení signálu čítacím senzorem. Převzato z [2].

Prvotní snímky z detektorů experimentu NOvA tak odpovídají pixelovým mapám, v nichž intenzita jednotlivých pixelů reprezentuje počty emitovaných fotoelektronů. Počty emitovaných fotoelektronů v dané buňce lze poté skrze proces tzv. absolutní kalibrace [4] přepočítat na množství energie, jež částice ztratila v buňce. Díky tomu máme pro všechny buňky k dispozici faktor, kterým můžeme množství emitovaných fotoelektronů v buňkách porovnávat ve fyzikálně smysluplných jednotkách (eV). Bližší popis a technické detaily týkající se procesu kalibrace jsou k dispozici v [4]. Ukázka výsledné pixelové mapy pro X-view je zobrazena na obrázku 1.2.

1.3 Využití CNN pro klasifikaci částicových interakcí

V detektorech nelze pozorovat neutrina přímo, ale jejich přítomnost lze zjistit detekováním rozpadu na pozorovatelné částice. Vůně neutrina muže být determinována na základě tzv. charged current (CC) interakcí. Tyto interakce jsou doprovázeny výskytem elektronu v případě ν_e, mionu u ν_μ a tauonu pro ν_τ. Interakce, při kterých nelze určit vůni neutrina, jsou poté označovány jako tzv. neutral current (NC) interakce. Ve vzdáleném

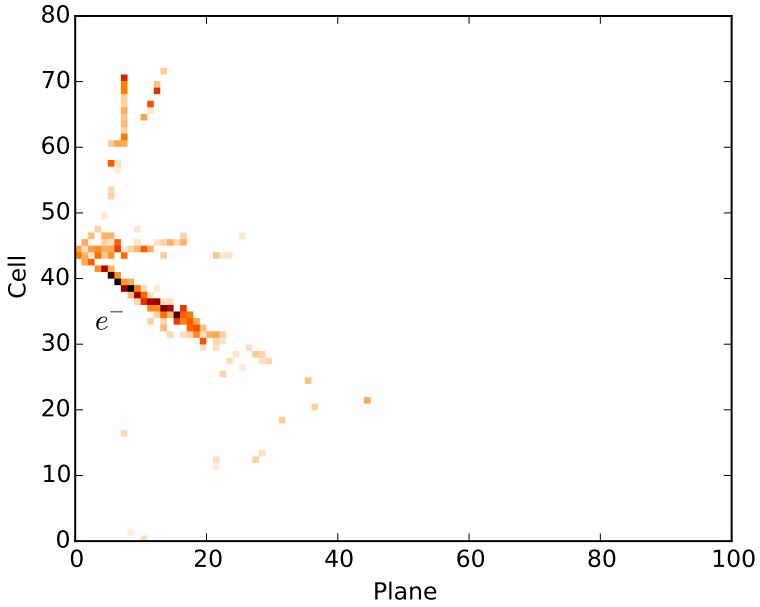
Interakce	Produkty interakce	Topologie trajektorie v detektoru
ν_μ CC	mion + hadron	Dlouhá rovná trajektorie odpovídající minimálně ionizujícímu mionu.
ν_e CC	elektron + hadron	Sprška částic s větším rozptylem, typicky kratší trajektorie.
ν_τ CC	tauon + hadron	Tauony se velice rychle rozpadají, a proto místo nich detektor snímá elektrony, miony, piony či neutrina vzniklé jejich rozpadem.
ν NC	neutrino + hadron	Leptony v NC interakcích jsou v detektoru neviditelná neutrina, a snímek tak zachycuje pouze hadronovou složku interakce.
	kosmické záření	Částice z kosmického záření (např. miony) typicky dopadají ze shora pod jiným úhlem, než částice ze svazku.

Tabulka 1.1: Přehled zkoumaných interakcí a odpovídajících topologií trajektorií v detektoru [2].

detektoru umístěném na povrchu poté můžeme pozorovat též částice z kosmického záření, a to zejména kosmické miony. Tyto částice však zpravidla přicházejí z jiných úhlů než částice ze svazku. Správnou identifikací výše popsaných interakcí je poté možno porovnávat složení svazku v obou detektorech a zkoumat tak proces neutrinové oscilace [2]. Jak uvádíme v tabulce 1.1, každá CC a NC interakce se v detektoru vyznačuje jinou topologií dráhy interagujících částic. Přestože obecně lze popsat trajektorii každé interakce v detektoru, v praxi může stále docházet k chybné klasifikaci částicové interakce. Nabité piony v NC interakcích například často trajektorií připomínají miony, čímž může dojít k záměně za ν_μ CC interakci. Klasifikace interakcí v detektoru je tak netriviální, ale nedílnou součástí analýzy neutrinových procesů.

Uvnitř detektorů standardně probíhá hned několik interakcí současně. Jednotlivé interakce jsou proto separovány a zobrazeny na samostatných snímcích o rozměrech 80×100 pixelů. Podobu takových snímků ukazujeme na obrázcích 1.2 a 1.3. Oříznutí snímku na uvedené rozměry má za efekt snížení rozměrů obrázku při zachování příznaků všech kýžených CC i NC interakcí. Jen zanedbatelná část interakcí se totiž s takto vymezenými rozměry nevejde do finálního snímku. Před použitím klasifikačních algoritmů jsou nicméně snímky interakce dále upraveny. První deska snímku reprezentovaná prvním bodem na horizontální ose poté odpovídá první desce detektoru, v níž byla zaznamenána interakce. Veškeré interakce jsou tak v rámci pixelových map zarovnány horizontálně doleva. Centrování pro výšku snímku je poté prováděno takovým způsobem, aby horní i spodní polovina snímku obsahovaly stejný počet buněk zachycujících signál. Snímek interakce je tak vertikálně centrovaný na střed. Centrování je prováděno především s cílem zkvalitnit

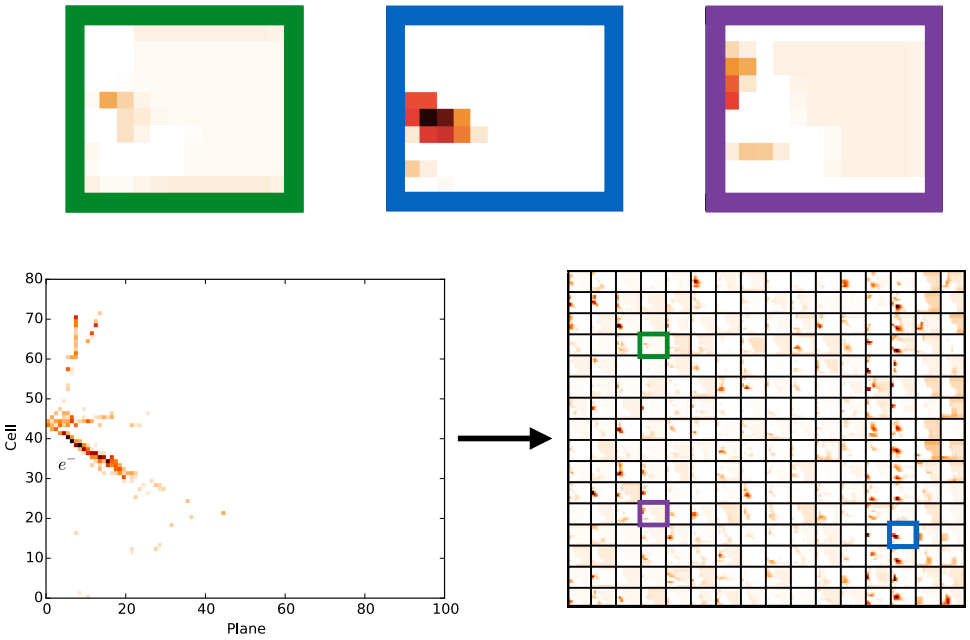
podobu vstupních dat pro klasifikační algoritmus. V tuto chvíli jsme dospěli do fáze, kdy jsme schopni jednotlivé pozorované interakce interpretovat v podobě dvou pixelových map reprezentující snímky neutrinových interakcí pro pohled X-view a Y-view.



Obrázek 1.2: Kalibrovaná pixelová mapa ν_e CC interakce s vyznačeným emitovaným elektronem e^- pozorované ve vzdáleném detektoru pro X-view. Osa Cell přitom označuje indexované buňky a osa Plane jednotlivé desky detektoru. Barevné stupně jednotlivých pixelů odpovídají normované hodnotě energie odevzdané v dané části detektoru. Tmavší barva tedy odpovídá větší hodnotě předané energie v dané buňce. Převzato z [21].

V rámci [16] jsme představili princip fungování, výhodné vlastnosti a především vysoký potenciál CNN pro úlohy rozpoznávání obrazu. CNN jsou pro tyto úlohy přirozeně adaptovány a v současnosti dominují mezi modely používanými na poli rozpoznávání obrazu [22]. Právě to je motivací pro použití CNN k rozpoznávání částicových interakcí z pixelových map namísto na experimentu NOvA dříve používaných technik jako multi-layer perceptron (MLP) či tzv. library event matching [5], [3]. Tyto techniky využívaly na vstupu manuálně určené expertní příznaky, které bylo třeba vyextrahovat ze snímků. Motivací pro použití CNN je v této fázi snaha o minimalizaci ztráty informace ze snímků způsobené nutností rekonstrukce fyzikálních příznaků následně použitých jako vstupy pro dříve používané metody. Při tom je využíváno schopnosti CNN samostatně extrahovat vizuální příznaky z obrazu [22]. Extrahovanými příznaky přitom rozumíme numerické charakteristiky popisující obraz, tedy v našem případě neutrinové interakce. Jak ukazuje obrázek 1.3, pro každou interakci jsou specifické různé příznaky, které je CNN schopná rozpoznat. Tyto vizuální příznaky přitom dříve zmíněné konvenční techniky klasifikace nemusejí odhalit. Konkrétní architektury CNN používané v experimentu NOvA se zakládají na modelech používaných na poli rozpoznávání obrazu. Každé pozorování neutrinové in-

terakce nicméně odpovídá hned dvojici snímků pro oba pohledy detektoru, a architektura CNN tak musí být upravena pro dvojici vstupních snímků. Velice významnou architekturou CNN používanou při analýze dat z experimentu NOvA je především tzv. CVN, jež je jednou z vůbec prvních CNN používaných pro rekonstrukci neutrinových dat z obrazových snímků [2]. Právě tuto architekturu budeme v kapitole 5 používat jako referenční model pro porovnávání výsledků klasifikace neutrinových interakcí.



Obrázek 1.3: Snímek z obrázku 1.2 zobrazený společně se sadou příznakových map extrahovaných z CVN. Zvýrazněné příznakové mapy se ukázaly být citlivé na ν_μ CC interakce (vyznačeno zeleně), ν_e CC interakce (modře) a NC interakce (fialově). Převzato z [21].

Kapitola 2

Úloha statistické klasifikace

Jak již bylo naznačeno v kapitole 1, jednou ze základních úloh při analýze dat nejen na experimentu NOvA je správná kategorizace zaznamenaných interakcí do přesně definovaných tříd. V případě experimentu NOvA je tak například nezbytné rozlišit interakce třídy pozadí ν_μ a třídy signálu ν_e s cílem měřit oscilace $\nu_\mu \to \nu_e$. V obecnějším případě je poté třeba rozlišit interakce všech pěti tříd popsaných v tabulce 1.1. V této kapitole tedy ilustrujeme koncept statistické klasifikace, jenž jsme podrobně definovali v [16]. Zprvu představujeme problematiku na úloze binární statistické klasifikace a následně představujeme zobecnění na vícerozměrnou statistickou klasifikaci. Cílem této sekce je především definovat klasifikační metriky, s jejichž využitím budeme následně v kapitole 5 schopni porovnávat úspěšnost klasifikace různých algoritmů.

2.1 Statistická klasifikace

Uvažujme vektor $\boldsymbol{x} \in \mathbb{R}^d$, $d \in \mathbb{N}$ jako realizaci náhodné veličiny $\boldsymbol{X}$. Dále nechť jsou $\omega_1, \ldots, \omega_{\mathscr{L}}$ disjunktní třídy pozorování splňující $\sum_{i=1}^{\mathscr{L}} P(\omega_i) = 1$ a $P(\omega_i) > 0$ pro všechna $i \in \{1, \ldots, \mathscr{L}\}$, kde $\mathscr{L} \in \mathbb{N}$ je počet tříd. Potom klasifikací do $\mathscr{L}$ tříd myslíme úlohu rozhodování, zda pozorování $\boldsymbol{x}$ přísluší k i-té třídě dle pravidla

$$\boldsymbol{x} \in \omega_i \iff P(\omega_i \mid \boldsymbol{x}) > P(\omega_j \mid \boldsymbol{x}), \tag{2.1}$$

pro všechna $j \in \{1, \ldots, \mathscr{L}\}$, kde $j \neq i$. Aposteriorní pravděpodobnosti z rovnice (2.1) lze dále odvodit na základě Bayesovy věty jako

$$P(\omega_i \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \omega_i) P(\omega_i)}{p(\boldsymbol{x})}, \tag{2.2}$$

kde

$$p(\boldsymbol{x}) = \sum_{i=1}^{\mathscr{L}} p(\boldsymbol{x} \mid \omega_i) P(\omega_i) > 0, \tag{2.3}$$

je hustota pravděpodobnosti (probability density function, PDF) pro náhodnou veličinu $\boldsymbol{X}$ a $p(\boldsymbol{x} \mid \omega_i)$ je PDF pro náhodnou veličinu $\boldsymbol{X}$ podmíněnu příslušností ke třídě ω_i.

2.2 Binární klasifikace

V této části navazujeme na rigorózně zavedený proces binární statistické klasifikace z [16] a pokračujeme v zavádění metrik, jež na konci práce použijeme pro měření úspěšnosti klasifikace jednotlivých algoritmů. Označme $P = \text{card}(\omega_S)$ celkový počet pozorování třídy signálu ω_S a $N = \text{card}(\omega_B)$ celkový počet pozorování třídy pozadí ω_B. Potom TP (true positive) je počet správně klasifikovaných pozorování třídy ω_S a FN (false negative) počet pozorování třídy ω_S nesprávně klasifikovaných jako ω_B. Analogicky buď TN (true negative) počet správně klasifikovaných pozorování třídy ω_B a FP (false positive) počet pozorování ω_B nesprávně klasifikovaných jako ω_S. Zřejmě tedy platí

$$P = TP + FN, \tag{2.4}$$

$$N = TN + FP. \tag{2.5}$$

S touto notací lze nyní zavést metriku accuracy (ACC) předpisem

$$ACC = \frac{TP + TN}{P + N}. \tag{2.6}$$

ACC tedy udává poměr správně klasifikovaných pozorování ku celkovému počtu pozorování. Získáváme tak velice intuitivní metriku, jejíž použitelnost nicméně klesá u nevybalancovaných datových množin. V oblastech jako je HEP totiž typicky platí $P \ll N$, a ani vysoká hodnota ACC tak obecně nemusí zohledňovat nízkou klasifikační úspěšnost na méně početné třídě.

Z tohoto důvodu zavádíme tzv. senzitivitu (též označovanou jako recall, true positive rate (TPR) nebo signal efficiency) jako

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}. \tag{2.7}$$

Senzitivita tak odpovídá poměru počtu správně klasifikovaných pozorování třídy signálu ku celkovému počtu pozorování třídy signálu. Oproti ACC tak tato metrika lépe zohledňuje úspěšnost klasifikace na pozorováních příslušných ke třídě signálu. Dále zavádíme metriku precision neboli positive predictive value (PPV) jako

$$PPV = \frac{TP}{TP + FP}. \tag{2.8}$$

Podobně jako v případě senzitivity i precision slouží k přesnějšímu popisu klasifikace na třídě signálu. Na rozdíl od senzitivity nicméně precision odpovídá poměru počtu správně klasifikovaných pozorování třídy signálu ku počtu pozorování klasifikovaných jako signál. V případech, kdy není jasná preference senzitivity nad precision či naopak, lze použít smíšené metriky. Proto zavádíme tzv. F1 skóre, jež odpovídá harmonickému průměru senzitivity a precision, tedy

$$F1 = 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR}. \tag{2.9}$$

F1 skóre v sobě zahrnuje senzitivitu i precision. Díky tomu představuje kompromis mezi oběma metrikami.

2.3 Vícerozměrná klasifikace

V případě klasifikace do $\mathscr{L} > 2$ tříd je třeba koncepty metrik z předchozí sekce zobecnit. K tomu zavádíme pojem matice záměn $\boldsymbol{C} \in \mathbb{N}^{\mathscr{L}\times\mathscr{L}}$, jejíž (i,j)-tý prvek C_{ij} je definovaný jako počet pozorování třídy ω_i klasifikovaných jako pozorování třídy ω_j pro $i,j \in \{1,\ldots,\mathscr{L}\}$. Diagonální prvky matice tedy odpovídají počtům správně klasifikovaných pozorování daných tříd, zatímco mimodiagonální prvky odpovídají počtům špatně klasifikovaných pozorování. Zároveň navíc můžeme z matice záměn přímo vyčíst za jaké třídy byla špatně klasifikovaná pozorování skutečné třídy predikována.

S takto zavedenou maticí záměn nyní lze definovat ACC pro klasifikaci do $\mathscr{L}$ tříd jako poměr součtu diagonálních a všech prvků matice záměn, tedy

$$ACC = \frac{\sum_{i=1}^{\mathscr{L}} C_{ii}}{\sum_{i,j=1}^{\mathscr{L}} C_{ij}}. \tag{2.10}$$

Podobně jako v případě binární klasifikace nicméně ACC při nevybalancovanosti tříd $\omega_1,\ldots,\omega_{\mathscr{L}}$ není zcela vypovídající o přesnosti klasifikace jednotlivých tříd. Z tohoto důvodu rozšiřujeme senzitivitu zavedenou rovnicí (2.7) na senzitivitu třídy ω_i jako poměr počtu správně klasifikovaných pozorování třídy ω_i vůči celkovému počtu pozorování třídy ω_i, tedy

$$TPR_i = \frac{C_{ii}}{\sum_{j=1}^{\mathscr{L}} C_{ij}}. \tag{2.11}$$

Analogicky lze zavést též precision třídy ω_i jako poměr počtu správně klasifikovaných pozorování třídy ω_i ku počtu pozorování klasifikovaných jako třída ω_i

$$PPV_i = \frac{C_{ii}}{\sum_{j=1}^{\mathscr{L}} C_{ji}}, \tag{2.12}$$

a F1 skóre pro třídu ω_i jako

$$F1_i = 2 \cdot \frac{PPV_i \cdot TPR_i}{PPV_i + TPR_i}. \tag{2.13}$$

Ekvivalentní zavedení výše vypsaných metrik bychom získali také, pokud bychom na třídu ω_i pohlíželi jako na třídu signálu ω_S a na sjednocení zbylých tříd $\bigcup_{i\neq j}\omega_j$ jako na třídu pozadí ω_B. Pro $\mathscr{L} = 2$ poté zavedení metrik pro obecný počet tříd přechází na zavedení pro úlohu binární klasifikace z předchozí sekce.

Ilustrujme nyní výše popsanou problematiku na příkladu klasifikace do tříd ω_1, ω_2 a ω_3 ze simulovaných dat z 2D Gaussova rozdělení se středními hodnotami

$$\mu_{\omega_1} = \begin{pmatrix} 0 \\ -15 \end{pmatrix}, \qquad \mu_{\omega_2} = \begin{pmatrix} 0 \\ 15 \end{pmatrix}, \qquad \mu_{\omega_3} = \begin{pmatrix} 15 \\ 0 \end{pmatrix}, \tag{2.14}$$

a stejnou kovarianční maticí

$$\boldsymbol{\Sigma} = \begin{pmatrix} 15 & 0 \\ 0 & 15 \end{pmatrix}. \tag{2.15}$$

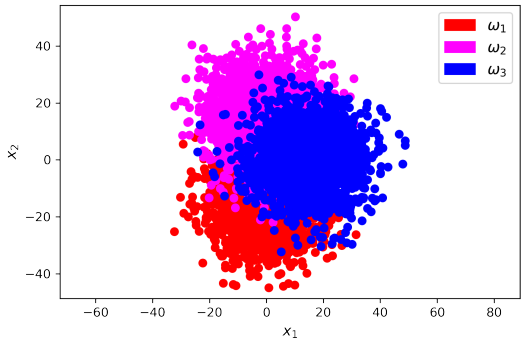
Pro každou třídu tímto způsobem generujeme 2000 pozorování, která ilustrujeme na obrázku 2.1a. Pro ukázku vyhodnocování úspěšnosti klasifikace nyní data separujeme lineárním klasifikátorem. Na obrázcích 2.1b, 2.1c a 2.1d poté pro $i \in \{1, 2, 3\}$ vykreslujeme histogramy odhadů aposteriorních pravděpodobností $P(\omega_i|\boldsymbol{x})$, tzv. control ploty. V našem případě tak vidíme, že na obrázku 2.1d není vrchol histogramu pro třídu ω_3 tak výrazný jako na ostatních control plotech. To indikuje nižší jistotu klasifikátoru pro klasifikaci do této třídy. Na obrázcích 2.1e a 2.1f poté vykreslujeme výsledky klasifikace zapsané v matici záměn, resp. matici záměn normalizované počtem pozorování v jednotlivých třídách. Matice záměn tedy dokazují, že třída ω_3 byla klasifikována s nižší přesností nežli zbylé třídy. Detailnější přehled metrik pro klasifikaci do jednotlivých tříd poté uvádíme v tabulce 2.1. V ní uvádíme též vážený průměr pro jednotlivé metriky, jenž zohledňuje počty pozorování v jednotlivých třídách. Označíme-li tedy $\mathrm{card}(\omega_i)$ počet pozorování i-té třídy, pak můžeme vážený průměr například pro precision $\overline{TPR}$ spočítat jako

$$\overline{TPR} = \frac{\sum_{i=1}^{\mathscr{L}} \mathrm{card}(\omega_i) TPR_i}{\sum_{i=1}^{\mathscr{L}} \mathrm{card}(\omega_i)}. \tag{2.16}$$

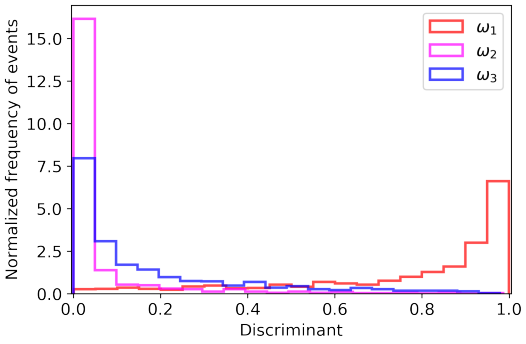
V tomto ukázkovém případě je počet pozorování v jednotlivých třídách stejný, a vážený průměr tak přechází v průměr aritmetický. V praxi se nicméně častěji setkáváme s nevybalancovaností tříd, a proto obecně pracujeme s váženým průměrem.

Třída	precision	recall	F1
ω_1	0.79	0.85	0.82
ω_2	0.84	0.80	0.82
ω_3	0.75	0.73	0.74
Vážený průměr	0.79	0.79	0.79

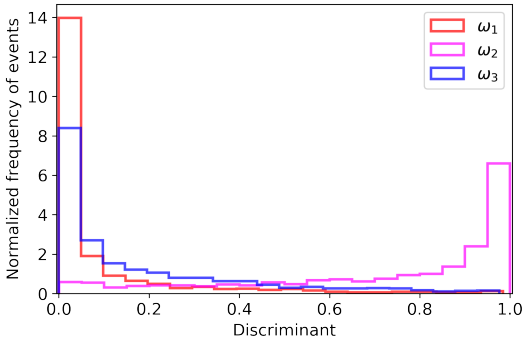
Tabulka 2.1: Výsledky klasifikace 2D simulovaných dat.



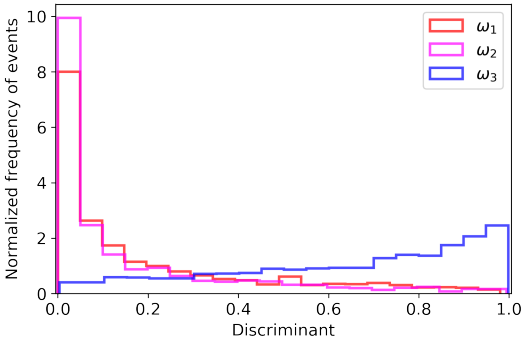
(a) simulovaná data



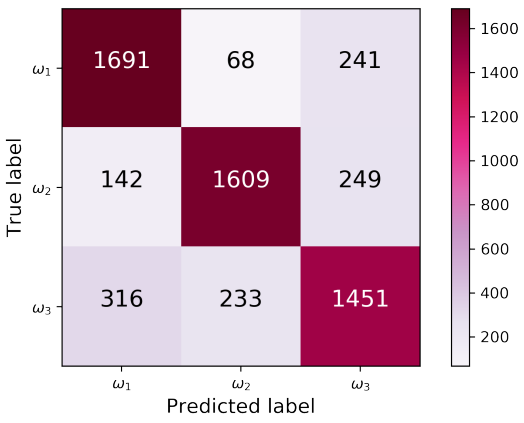
(b) odhad $P(\omega_1 \mid \boldsymbol{x})$



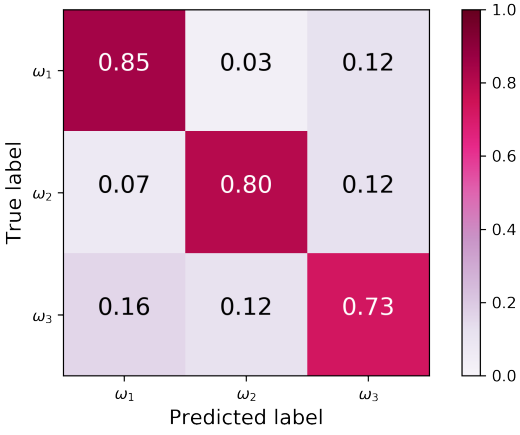
(c) odhad $P(\omega_2 \mid \boldsymbol{x})$



(d) odhad $P(\omega_3 \mid \boldsymbol{x})$



(e) matice záměn



(f) normalizovaná matice záměn

Obrázek 2.1: Ukázka klasifikace do tříd ω_1, ω_2 a ω_3 na simulovaných datech.

Kapitola 3

Umělé neuronové sítě

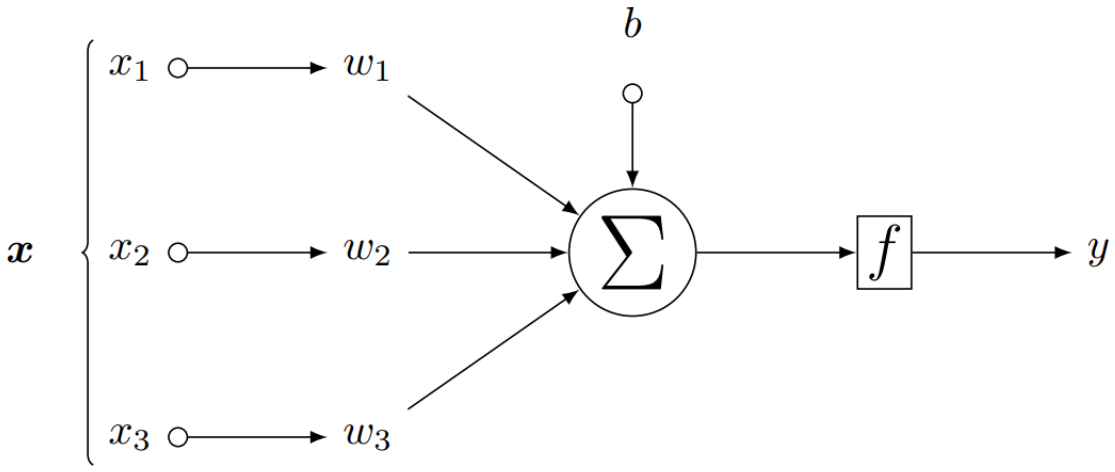
Zatímco v předchozí kapitole jsme popsali princip a metodiku vyhodnocování statistické klasifikace, nyní se zaměříme na popis klasifikačních algoritmů. Konkrétně se zaměřujeme na metody machine learning (ML), neboli česky strojového učení. Tyto metody dokážou za pomocí dostupných dat natrénovat velice komplexní klasifikační vzory. Významnou skupinou klasifikačních algoritmů jsou tzv. artificial neural networks (ANN), neboli umělé neuronové sítě. Především konvoluční neuronové sítě (CNN) jsou poté velmi rozšířeným a úspěšným přístupem k řešení úlohy klasifikace obrazových dat. CNN jsou totiž schopné extrahovat ze snímku příznaky, s nimiž je možno následně snímek klasifikovat. Z tohoto důvodu jsou také CNN metodou hojně využívanou pro klasifikaci obrazových dat na experimentu NOvA. Pokročilé ANN modely schopné zachytit i velice komplikované vzory poté označujeme jako metody používající techniky hlubokého učení (deep learning, DL) [10]. V této kapitole tak navazujeme na podrobně popsané teoretické koncepty ANN a CNN v práci [16], které rozšiřujeme pro úlohu klasifikace do více než dvou tříd. Po zavedení základního značení, konceptů architektury a učení neuronových sítí se zaměříme na pokročilé architektury CNN a regularizační techniky zabraňující přeučení modelů. Všechny tyto poznatky přitom představujeme s cílem použít je v kapitole 5 pro klasifikaci neutrinových interakcí na datech z experimentu NOvA.

3.1 Umělé neuronové sítě (ANN)

ANN jsou komplexní modely tvořené základními komponenty, tzv. neurony. Uvažujme vstupní vektor $\boldsymbol{x} \in \mathbb{R}^d$, $d \in \mathbb{N}$. Neuron s vektorem vah $\boldsymbol{w} \in \mathbb{R}^d$, prahem $b \in \mathbb{R}$ a aktivační funkcí $f : \mathbb{R} \to \mathbb{R}$ pak transformuje vstupní vektor $\boldsymbol{x}$ na výstupní hodnotu $y \in \mathbb{R}$ způsobem

$$y = f(\boldsymbol{w}^\intercal \boldsymbol{x} + b) = f\left(\sum_{i=1}^{d} w_i x_i + b\right). \tag{3.1}$$

Výraz $\xi = \sum_{i=1}^{d} w_i x_i + b$ nazýváme potenciálem neuronu. Matematický model neuronu ilustrujeme na obrázku 3.1. Jak jsme rigorózně definovali v [16], jednotlivé neurony lze formovat do vrstev, přičemž výstup vrstvy předchozí používáme jako vstup pro následující vrstvu. První vrstvu ANN nazýváme vrstvou vstupní a poslední vrstvu modelu nazýváme



Obrázek 3.1: Model neuronu se vstupy x_1, x_2, x_3, vahami w_1, w_2, w_3, prahem b a aktivační funkcí f. Transformací popsanou rovnicí (3.1) poté vzniká výstup y.

vrstvou výstupní. Zbylé vrstvy nazýváme vrstvami skrytými. Schéma ANN tvořené třemi skrytými vrstvami zobrazujeme na obrázku 3.2.

Konkrétní předpis funkce f poté záleží na povaze úlohy. Typicky se pro skryté vrstvy ANN používají jiné druhy aktivačních funkcí než pro vrstvy výstupní. Pro klasifikační účely totiž od výstupů výstupní vrstvy požadujeme interpretaci v podobě odhadu pravděpodobnosti. Jednotlivé druhy používaných aktivačních funkcí představíme v následujících odstavcích.

Definice 3.1.1. Nechť $\xi \in \mathbb{R}$ je potenciál neuronu, potom skokovou aktivační funkci $f : \mathbb{R} \to \{0, 1\}$ definujeme jako

$$f(\xi) = \begin{cases} 1 & \xi \geq 0, \\ 0 & \text{jinak.} \end{cases} \tag{3.2}$$

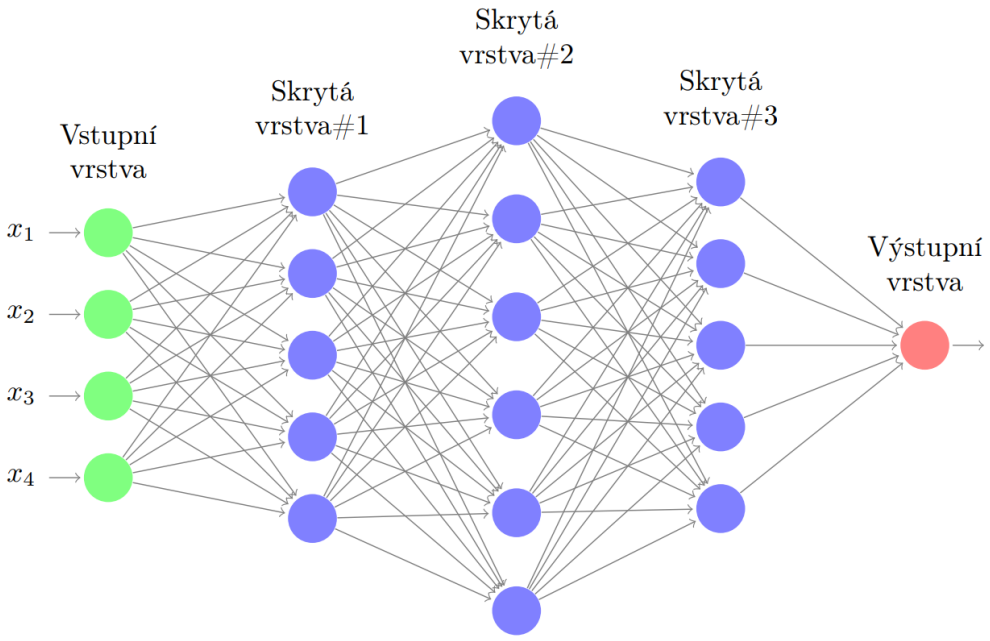
Skoková aktivační funkce tak vyšle tzv. signál z neuronu pouze při překročení prahu

$$\sum_{i=1}^{d} w_i x_i \geq -b. \tag{3.3}$$

ANN tvořenou jediným neuronem se skokovou aktivační funkcí nazýváme perceptron [6]. V praxi se nicméně jeví užitečnější používat spojitě diferencovatelné aktivační funkce, proto skokovou funkci uvádíme pouze z ilustrativních důvodů.

Definice 3.1.2. Nechť $\xi \in \mathbb{R}$ je potenciálem neuronu, pak sigmoidální aktivační funkcí $f : \mathbb{R} \to (0, 1)$ myslíme funkci

$$f(\xi) = \frac{1}{1 + e^{-\xi}}. \tag{3.4}$$



Obrázek 3.2: Model ANN tvořené vstupní vrstvou o čtyřech neuronech, třemi skrytými vrstvami a výstupní vrstvou o jednom neuronu. Všechny neurony ve skrytých vrstvách jsou přitom propojeny s každým neuronem předešlé a následující vrstvy. Model ANN tvořený seskupením neuronů dle (3.1) do vrstev bývá také označován jako multi-layer perceptron (MLP), neboli vícevrstvý perceptron [6].

Jak ilustrujeme na obrázku 3.3, sigmoidální funkce zobrazuje vstupní potenciály na interval $(0, 1)$. Díky tomu se sigmoidální funkce zpravidla používá ve výstupní vrstvě pro úlohu binární klasifikace, jelikož její výstup lze v případě jednoho výstupního neuronu interpretovat jako pravděpodobnost příslušnosti ke třídě signálu. Pro případ klasifikace do $\mathscr{L} > 2$ tříd lze sigmoidální funkci zobecnit na tzv. softmax funkci. Softmax funkcí f myslíme funkci $f : \mathbb{R}^{\mathscr{L}} \to (0, 1)^{\mathscr{L}}$ definovanou po složkách jako

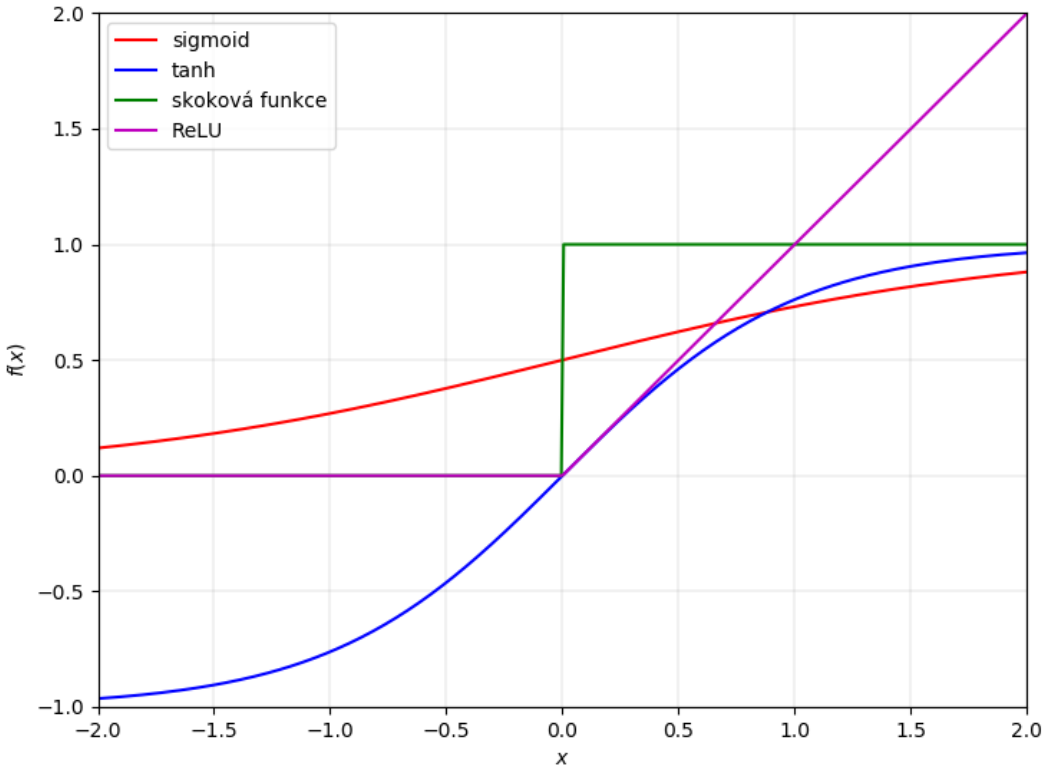
$$(f(\boldsymbol{\xi}))_j = \frac{e^{\xi_j}}{\sum_{k=1}^{\mathscr{L}} e^{\xi_k}}, \quad j \in \{1, \ldots, \mathscr{L}\}, \tag{3.5}$$

kde $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_{\mathscr{L}})$ je vektor potenciálů na výstupní vrstvě. Pro složky výstupu $\boldsymbol{y} = f(\boldsymbol{\xi})$ tedy platí $0 < y_i < 1$ a $\sum_{i=1}^{\mathscr{L}} y_i = 1$. Poté lze j-tý výstup softmax funkce interpretovat jako pravděpodobnost příslušnosti pozorování $\boldsymbol{x}$ k j-té třídě, $j \in \{1, \ldots, \mathscr{L}\}$. Z tohoto důvodu se podobně jako sigmoidální funkce používají ve výstupní vrstvě sítě, tentokráte ovšem pro případ klasifikace do $\mathscr{L} > 2$ tříd.

Definice 3.1.3. Nechť $\xi \in \mathbb{R}$ je potenciálem neuronu, pak rectified linear unit (ReLU) funkci $f : \mathbb{R} \to \mathbb{R}_0^+$ definujeme jako

$$f(\xi) = \begin{cases} \xi & \xi \geq 0, \\ 0 & \text{jinak.} \end{cases} \tag{3.6}$$

Výhodou ReLU funkce je díky podobnosti s identitou její snadný výpočet a skutečnost, že pro vysoké hodnoty ξ se derivace funkce f neblíží nule (viz tzv. vanishing gradient problem [14]). Z těchto důvodů se ReLU funkce často používá ve skrytých vrstvách ANN o velkém počtu vrstev a neuronů.

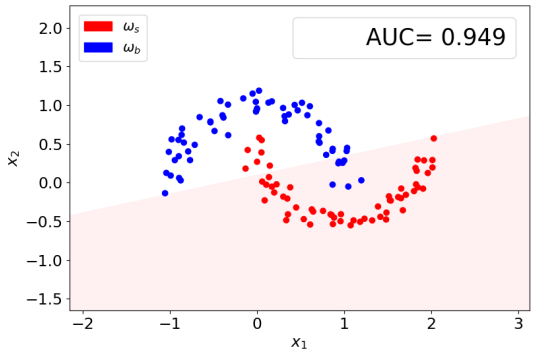


Obrázek 3.3: Porovnání jednotlivých aktivačních funkcí.

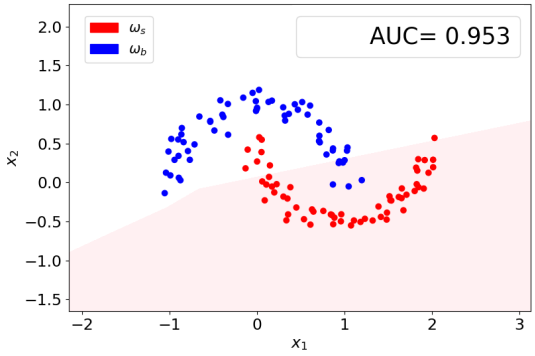
3.2 Učení neuronových sítí

Učení neuronových sítí je proces určení volných parametrů modelu (vah a prahů) takovým způsobem, aby klasifikátor přiřazoval vstupům správnou příslušnost ke třídě. Systém volných parametrů budeme pro přehlednost značit jako $\boldsymbol{\theta} = \big((\boldsymbol{w}^1, b^1), \ldots, (\boldsymbol{w}^k, b^k)\big)$, kde $k \in \mathbb{N}$. Uvažujeme přitom případ učení s učitelem, kde v případě klasifikace do $\mathscr{L} > 2$ tříd používáme trénovací množinu $((\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n))$. V takovém případě pro i-té trénovací pozorování $\boldsymbol{x}_i$ udává vektor $\boldsymbol{y}_i \in \{0,1\}^{\mathscr{L}}$ indikaci příslušnosti ke třídě ω_j, $j \in \mathscr{L}$ jako

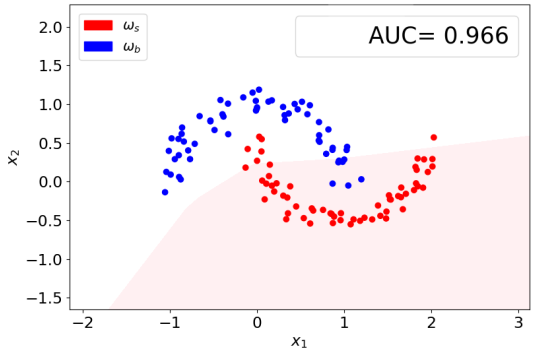
$$(\boldsymbol{y}_i)_j = \begin{cases} 1 & \text{pro } \boldsymbol{x}_i \in \omega_j, \\ 0 & \text{pro } \boldsymbol{x}_i \notin \omega_j, \end{cases} \tag{3.7}$$



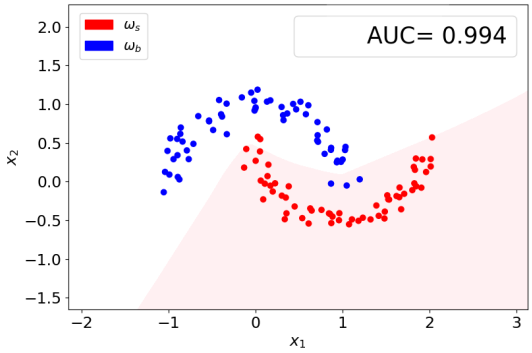
(a) 1 skrytá vrstva o 2 neuronech



(b) 1 skrytá vrstva o 5 neuronech



(c) 1 skrytá vrstva o 10 neuronech



(d) 1 skrytá vrstva o 50 neuronech

Obrázek 3.4: Ukázka klasifikace simulovaných 2D dat pomocí ANN s jednou skrytou vrstvou o různém počtu neuronů. Obrázek zobrazuje data s indikací množin a nelineární separační hranici. Model s větším množstvím neuronů, tedy i větším množstvím parametrů modelu, dokázal lépe zachytit reálný vzor v datech, a dosáhl tak vyšší hodnoty metriky AUC [16].

kde $(\boldsymbol{y}_i)_j$ je j-tá složka vektoru $\boldsymbol{y}_i$. S využitím trénovací množiny je naším cílem nastavit parametry modelu $\boldsymbol{\theta}$ takovým způsobem, aby byl na základě pravidel naučených na trénovací množině schopný správně predikovat i na datech, která při učení nebyla použita.

3.2.1 Ztrátová funkce

Uvažujme výstup modelu $f(\boldsymbol{x}_i;\boldsymbol{\theta}) \in (0,1)^{\mathscr{L}}$ pro případ klasifikace se softmax funkcí. Výstup modelu tedy reprezentuje vektor predikovaných pravděpodobností příslušností k jednotlivým třídám. Odchylku predikce $f(\boldsymbol{x}_i;\boldsymbol{\theta})$ od skutečné indikace $\boldsymbol{y}_i$ poté měříme s využitím tzv. ztrátové funkce $L : \mathbb{R}^{\mathscr{L}} \times \mathbb{R}^{\mathscr{L}} \to \mathbb{R}^{+}$. Pro případ vícerozměrné klasifikace poté používáme kategorickou křížovou entropickou funkci L_{CCE} definovanou jako

$$L_{CCE}\left(f(\boldsymbol{x}_i;\boldsymbol{\theta}),\boldsymbol{y}_i\right) = -\sum_{j=1}^{\mathscr{L}} (\boldsymbol{y}_i)_j \log\left(f(\boldsymbol{x}_i;\boldsymbol{\theta})\right)_j . \tag{3.8}$$

S využitím vhodné ztrátové funkce poté definujeme empirickou rizikovou funkci J jako

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L\left(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i\right), \tag{3.9}$$

což odpovídá odhadu střední hodnoty ztrátové funkce L napříč $n \in \mathbb{N}$ trénovacími pozorováními [16]. Úloha učení modelu poté přechází v úlohu minimalizace cílové funkce J.

3.2.2 Gradient descent

Typicky používanou metodou pro minimalizaci J je metoda největšího gradientního spádu, tzv. gradient descent (GD). Ta spočívá v iterativní úpravě parametrů modelu pomocí malých kroků ve směru největšího záporného gradientu empirické rizikové funkce dle vzorce

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \varepsilon\frac{1}{n}\nabla_{\boldsymbol{\theta}}\sum_{i=1}^{n} L\left(f(\boldsymbol{x}_i;\boldsymbol{\theta}^{(k)}), \boldsymbol{y}_i\right), \tag{3.10}$$

kde $\varepsilon > 0$ je rychlost učení neboli learning rate a $k \in \mathbb{N}$ je index iterace. Pro $k = 1$ lze parametry $\boldsymbol{\theta}^{(1)}$ inicializovat například výběrem z normálního rozdělení o nulové střední hodnotě. Pro výpočet gradientů ztrátové funkce se používá algoritmus tzv. backpropagation, který jsme detailně popsali v [16]. V rámci GD využíváme v každém kroku celou množinu $n \in \mathbb{N}$ trénovacích pozorování. Proto jej v této podobě označujeme též jako batch GD.

Označením stochastic gradient descent (SGD) poté pro přehlednost odkazujeme na proces, kdy dochází k aktualizaci parametrů po výpočtu gradientu skrz jedno náhodně vybrané trénovací pozorování [17]. V takovém případě tedy rovnice (3.10) přechází na tvar

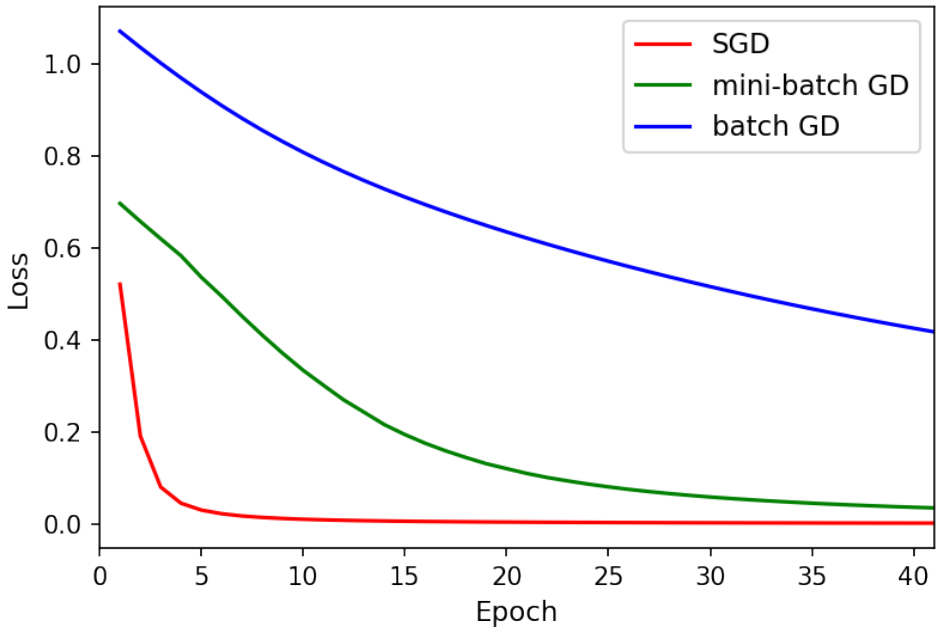
$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \varepsilon\nabla_{\boldsymbol{\theta}} L\left(f(\boldsymbol{x}_i;\boldsymbol{\theta}^{(k)}), \boldsymbol{y}_i\right), \tag{3.11}$$

kde $i \in \{1, \ldots, n\}$ je index náhodně vybraného pozorování z trénovacího vzorku. V případě SGD je výpočetní náročnost jednoho kroku výrazně nižší než pro GD, protože výpočet je prováděn jen pro jedno pozorování. Díky rychlým jednotlivým krokům se tak s použitím SGD lze rychleji dostat do blízkého okolí minima J. Pro jeden trénovací cyklus (neboli epochu) však oproti GD namísto jednoho kroku opakujeme krok pro každé pozorování z trénovací množiny. Ačkoliv je tak výpočet jednoho kroku výrazně rychlejší a je možno rychleji se dostat do okolí minima J, časový úsek pro jednu trénovací epochu může být celkově výrazně delší.

Oba tyto přístupy tak proto může být výhodné skloubit optimalizací pomocí tzv. mini-batch GD [11]. Při mini-batch GD je pro aktualizaci parametrů použita vždy pouze vybraná menší várka (mini-batch) trénovacích dat. S použitím mini-batch GD tedy vystává požadavek na další uživatelem volený hyperparametr modelu v podobě velikosti várky. Uvažujme pro k-tou iteraci várku $\mathcal{B}^{(k)} = \left((\boldsymbol{x}_{i_1}, \boldsymbol{y}_{i_1}), \ldots, (\boldsymbol{x}_{i_m}, \boldsymbol{y}_{i_m})\right)$, přičemž platí $i_1, \ldots, i_m \in \{1, \ldots, n\}$, $m \in \mathbb{N}$ za podmínky $m < n$ a pro $j \neq l$ platí $i_j \neq i_l$ pro všechna $j, l \in \{1, \ldots, m\}$. Potom se tedy pravidlo pro aktualizaci parametrů popsané rovnicí (3.10) změní na

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \varepsilon\frac{1}{m}\nabla_{\boldsymbol{\theta}}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}^{(k)}} L\left(f(\boldsymbol{x};\boldsymbol{\theta}^{(k)}), \boldsymbol{y}\right), \tag{3.12}$$

kde $m = \text{card}(\mathcal{B}^{(k)})$ označuje počet pozorování v jedné várce. Díky častějším aktualizacím parametrů má mini-batch GD možnost rychleji dojít do bodu minima cílové funkce nežli GD. Navíc oproti GD se mini-batch GD dokáže snáze vymanit z lokálního minima. Empiricky se přitom ukazuje [11], že menší várky vedou k rychlejší konvergenci cílové funkce co do počtu trénovacích cyklů. Porovnání průběhu minimalizace cílové funkce pro různé varianty GD ilustrujeme na obrázku 3.5.



Obrázek 3.5: Obrázek ilustruje průběh minimalizace cílové funkce s použitím různých variant GD. Pozorování bylo provedeno v rámci klasifikace na množině IRIS [9] na ANN tvořené třemi skrytými vrstvami po 10 neuronech. Z obrázku je patrná pomalá konvergence batch GD. Oproti tomu SGD i mini-batch GD konvergují k minimu J rychleji.

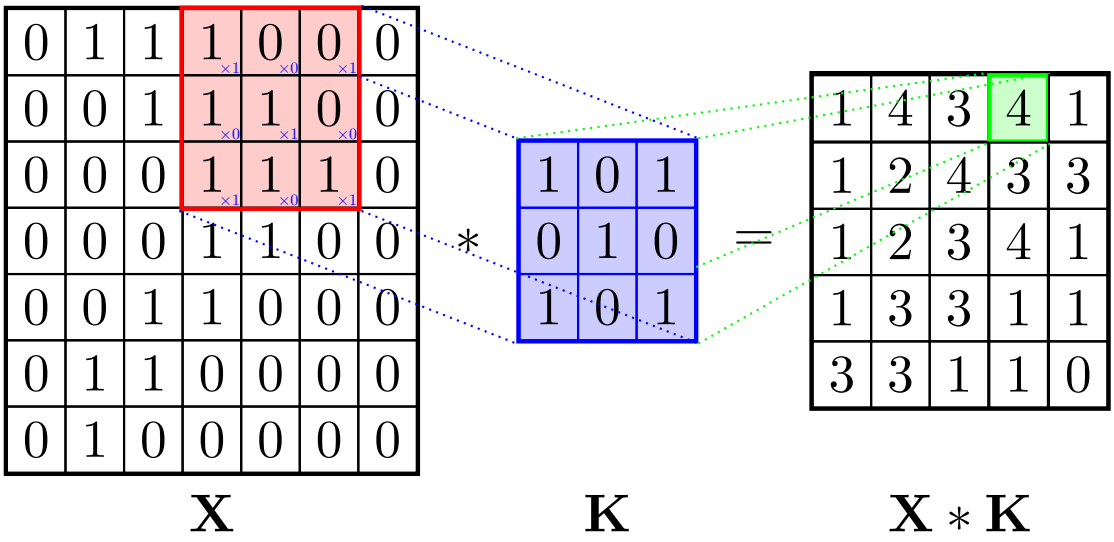
3.3 Konvoluční neuronové sítě (CNN)

CNN jsou rozšířením konceptu ANN, které dokáže lépe zachytit příznaky v obrazových vstupech. Oproti klasickým ANN používají CNN pro transformaci dat namísto maticového násobení operaci konvoluce. Konvoluci funkcí $f, g : \mathbb{R} \to \mathbb{R}$ ve spojitém případě definujeme jako

$$(f * g)(x) = \int_{\mathbb{R}} f(y)g(x - y)\mathrm{d}y. \tag{3.13}$$

V praxi nicméně budeme pracovat s diskrétními 2D obrazovými daty. V takovém případě konvoluci funkcí $\boldsymbol{X} : \mathbb{Z}^2 \to \mathbb{R}^2$ a $\boldsymbol{K} : \mathbb{Z}^2 \to \mathbb{R}^2$ definujeme jako

$$(\boldsymbol{X} * \boldsymbol{K})(i, j) = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} \boldsymbol{X}(i - m, j - n)\boldsymbol{K}(m, n), \tag{3.14}$$

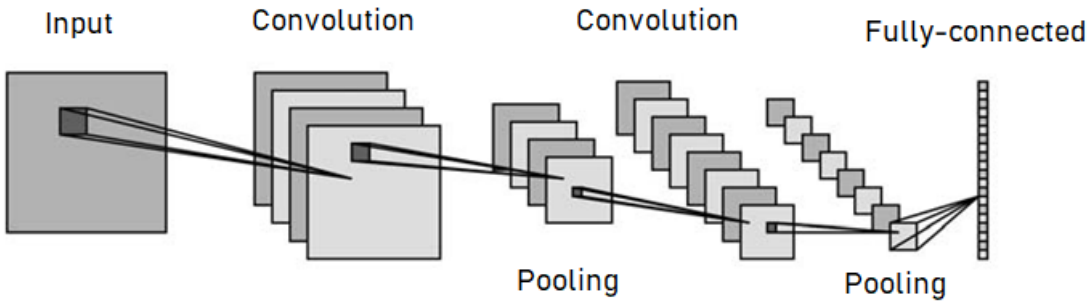


Obrázek 3.6: Schéma konvoluce dle rovnice (3.14). Červeně vyznačené pole odpovídá části vstupního snímku $\boldsymbol{X}$, ze kterého po konvoluci s modře označeným konvolučním jádrem $\boldsymbol{K}$ vzniká zeleně vyznačený prvek na výstupní příznakové mapě.

kde $i, j \in \mathbb{Z}$ jsou indexy výstupního snímku. Pro obrazová data konečných rozměrů poté nekonečná sumace přechází v sumaci konečnou, viz [16]. Funkce $\boldsymbol{X}$ v takovém případě reprezentuje vstupní obrázek a $\boldsymbol{K}$ je tzv. konvoluční jádro neboli kernel. Právě prvky konvolučního jádra jsou volné parametry pro natrénování modelu a jsou analogií k vahám v případě ANN. Rozměry a počet konvolučních jader jsou uživatelem zadaný parametr, standardně tak pro každou konvoluční vrstvu používáme hned několik konvolučních jader. Schéma konvoluce popsané rovnicí (3.14) ilustrujeme na obrázku 3.6.

Výstup konvoluce (tzv. příznaková mapa) je následně podobně jako v případě ANN po prvcích transformován aktivační funkcí. Zároveň navíc můžeme výstupy takto vytvořené konvoluční vrstvy použít jako vstupy pro vrstvu následující a získat tak hlubokou CNN. Pro klasifikaci nakonec každý pixel výstupu poslední konvoluční vrstvy propojujeme skrze skrytou tzv. plně propojenou vrstvu (též fully-connected layer) s vrstvou výstupní. Výstup modelu poté stejně jako v případě ANN po použití aktivační funkce sigmoid či softmax odpovídá pravděpodobnosti příslušnosti k dané třídě. Schéma tohoto popisu ilustrujeme na obrázku 3.7.

Kromě konvolucí a transformace aktivační funkcí je v architekturách CNN též používán tzv. pooling, jak jsme již podrobně popsali v [16]. Ten slouží ke snížení dimenze příznakových map vhodnou transformací. Standardně je používán tzv. max pooling, který příznakovou mapu rozdělí na pole o uživatelem daných rozměrech. Z každého pole je poté na výstup přenesen pouze prvek o maximální hodnotě. Alternativou za max pooling je tzv. average pooling, který místo nejvyšší hodnoty vybere aritmetický průměr prvků z pole.



Obrázek 3.7: Schéma CNN. Vstup (Input) je po sérii konvolučních vrstev (Convolution) a poolingu skrze plně propojenou vrstvu (Fully-connected) spojen s výstupní vrstvou. Architekturu CNN přitom ilustrujeme na jednokanálovém vstupu, stejně jako je tomu v případě dat z experimentu NOvA.

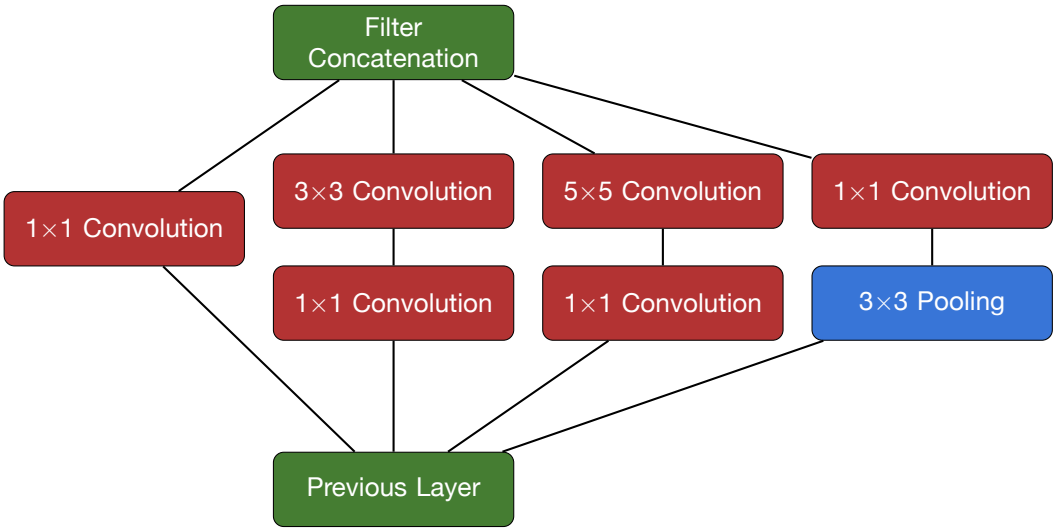
3.4 Pokročilé techniky

Pro komplexní úlohy využívající velké množství dat o vysoké dimenzi je pro dosažení kýžených výsledků nutno rozšířit architektury CNN popsané v předchozí sekci. V této sekci proto popisujeme dvě modifikace CNN, jež jsou v současnosti hojně využívané na poli rozpoznávání obrazu. Tyto techniky následně v kapitole 5 používáme v architekturách modelů pro klasifikaci neutrinových interakcí z experimentu NOvA.

3.4.1 Inception

Jednou z možností, jak zvýšit přesnost klasických CNN, je zavedení tzv. inception modulů publikovaných v rámci modelu GoogLeNet [24]. Na základě testování na porovnávacích benchmark datových množinách lze s inception moduly dosáhnout výrazně lepších výsledků oproti základním CNN. Právě modelem GoogLeNet je inspirovaný model CVN používaný na klasifikaci neutrinových interakcí na experimentu NOvA.

Architektura inception modulu použitého v CVN je znázorněna na obrázku 3.8. V rámci každého modulu jsou jednotlivé příznakové mapy paralelně zpracovány čtyřmi větvemi skrytých vrstev. Na počátku je provedena konvoluce s jádry o rozměrech $(1, 1)$, počet výstupních příznakových map pak totiž odpovídá počtu $(1, 1)$ filtrů. Pracujeme-li s vysokým množstvím vstupních příznakových map, lze tímto krokem snížit počet vstupů do následujících konvolučních vrstev. Tímto způsobem lze tedy snížit výpočetní náročnost modelu [21]. Zatímco v první větvi již nejsou použity další operace, ve druhé a třetí větvi jsou vstupy transformovány konvolucemi s konvolučními filtry o rozměrech $(3, 3)$ a $(5, 5)$. Poslední větev poté před konvolucí s filtrem $(1, 1)$ používá max pooling o rozměrech $(3, 3)$. Díky takto popsané architektuře složené ze čtyř unikátních paralelních sérií skrytých vrstev disponuje inception modul vyšší flexibilitou pro extrakci příznaků ze vstupu nežli standardní konvoluční vrstva. Pro jediný vstup totiž inception modul vytvoří výstupní příznakové mapy hned pro čtyři větve s různými konvolučními vrstvami. Model tak zís-



Obrázek 3.8: Schéma inception modulu používaného v architekturách GoogLeNet a CVN. Buňka Previous Layer zde reprezentuje výstupy z předchozí vrstvy, které jsou následně použity jako vstupy do čtyř paralelních vrstev. Buňka Filter Concatenation poté symbolizuje poskládání takto vzniklých příznakových map do jedné série. Převzato z [21].

kává větší kapacitu použitím paralelních větví namísto zvětšováním hloubky. Z výsledných sérií příznakových map ze všech čtyř větví je následně vytvořen jeden velký systém příznakových map. Ten je použit jako vstup do další vrstvy. Na jednotlivé inception moduly se poté v rámci sítě můžeme dívat analogicky jako na standardní skryté vrstvy.

3.4.2 Residuální neuronové sítě (ResNet)

Další modifikací architektury CNN vedoucí k vyšší přesnosti klasifikace pro komplexní a obsáhlé datové množiny jsou tzv. residuální neuronové sítě (ResNet) [13]. Tato technika byla vyvinuta s cílem odstranit problémy při učení ANN s řádově desítkami skrytých vrstev. U takto hlubokých ANN se často projevoval tzv. vanishing gradient problem [14] a problém degradace [13], kdy další přidávání skrytých vrstev snižuje klasifikační přesnost modelu. V této sekci popisujeme základní myšlenku a značení ResNet. Pro popis ResNet budeme hojně používat pojem tzv. mapování vstupu. To odpovídá transformaci vstupu skrze určitou sérii vrstev CNN. Uvažujeme-li vstup $\boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2}$, pak mapováním $\mathcal{F}(\boldsymbol{X}) : \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}^{d_1 \times d_2}$ myslíme transformaci vstupu $\boldsymbol{X}$ určitou sérií vrstev sítě. Pro jednoduchost nyní předpokládáme stejné rozměry vstupní a výstupní matice. Hodnoty $d_1, d_2 \in \mathbb{N}$ tedy odpovídají dimenzím vstupu, respektive mapování. Uvažujeme pouze jednokanálové obrazové vstupy ve stupních šedi. Mapování $\mathcal{F}(\boldsymbol{X})$ nemusí být výstupem celé CNN, ale značíme jím i transformaci do úrovně skrytých vrstev. S využitím této notace nyní přistupujeme k popisu residuálního učení.

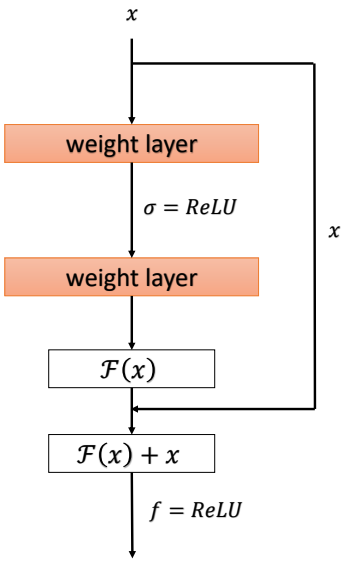
Cílem trénování CNN je naučit model mapování vedoucí na výstup $\boldsymbol{y} \in \{0,1\}^{\mathscr{L}}$, kde $\mathscr{L} \in \mathbb{N}$ je počet tříd. Uvažujme, že toto cílené mapování je po jisté sérii skrytých vrstev popsáno transformací $\mathcal{H}(\boldsymbol{X}) : \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}^{d_1 \times d_2}$. Základní myšlenkou residuálního učení je naučit se namísto cíleného popisného mapování $\mathcal{H}(\boldsymbol{X})$ vstupu $\boldsymbol{X}$ tzv. residuální mapování ve tvaru

$$\mathcal{F}(\boldsymbol{X}) = \mathcal{H}(\boldsymbol{X}) - \boldsymbol{X}. \tag{3.15}$$

Rozdíl v (3.15) je myšlen po prvcích maticově. Z tohoto důvodu jsme předpokládali stejné dimenze vstupu a výstupu mapování. Obecně lze ale vhodnou transformací vztah (3.15) upravit i pro obecné rozměry [13]. Výzkum ukazuje, že série skrytých vrstev nedokáže dostatečně přesně mapovat identity. Proto může být vhodnější použít právě residuální mapování, které tento problém odstraňuje. V extrémním případě, kdy je optimální mapování identitou, se poté ukazuje, že je skrze řadu vrstev snazší modelovat nulové residuum nežli identitu [13], [12]. Tímto způsobem tak může dojít k omezení vlivu problému degradace. Myšlenku residuálního učení popsanou rovnicí (3.15) lze v praxi implementovat za pomocí tzv. shortcut connections [13]. Po úpravě dostáváme z rovnice (3.15) vztah

$$\mathcal{H}(\boldsymbol{X}) = \mathcal{F}(\boldsymbol{X}) + \boldsymbol{X}. \tag{3.16}$$

Shortcut connections poté schematicky znázorňují přičtení vstupu $\boldsymbol{X}$ k residuálnímu mapování $\mathcal{F}(\boldsymbol{X})$ v rámci tzv. residuálních bloků. Residuálními bloky myslíme sérii po sobě jdoucích vrstev, na které skrze shortcut connections použijeme techniku residuálního učení dle rovnice (3.15). Schematicky je model residuálního bloku vykreslen na obrázku 3.9.

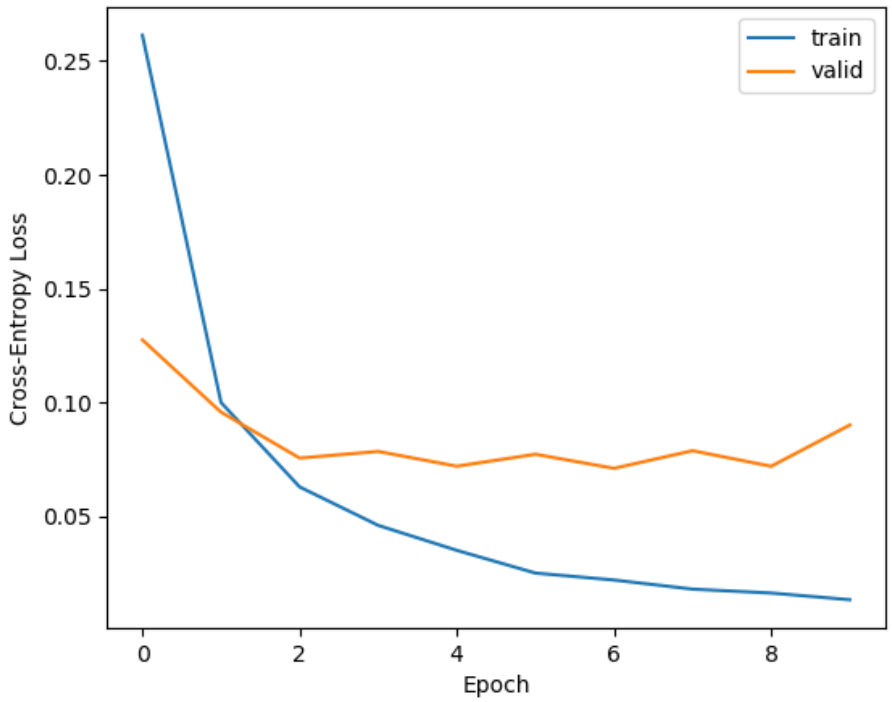


Obrázek 3.9: Schéma principu ResNet použité v [13]. Šipka vpravo znázorňuje shortcut connection. Aktivační funkce f poté provádí transformaci výstupu residuálního bloku $\mathcal{F}(\boldsymbol{X}) + \boldsymbol{X}$. Funkce f přitom obecně nemusí být stejná jako aktivační funkce σ uvnitř bloku, což zdůrazňujeme odlišným značením funkcí.

3.5 Regularizační techniky

Zásadním požadavkem na aplikovatelný model je schopnost generalizovat, tedy predikovat na trénovacích datech s podobnou přesností jako na datech, která během trénování nebyla použita. Jak již bylo zmíněno v sekci 3.2, pro trénování modelu používáme tzv. trénovací množinu dat. V praxi navíc pro kontrolu generalizace modelu během učení pozorujeme přesnost modelu na tzv. validační množině dat. Ve chvíli, kdy se přestane snižovat empirická riziková funkce pro validační množinu, začíná se model tzv. přetrénovávat (nastává tzv. overfitting). To znamená, že model přestává generalizovat pro nová data, přestože přesnost na trénovacích datech se stále zvyšuje. Empirická riziková funkce se na trénovací množině sice napříč iteracemi stále snižuje, ale přestává být dobrým odhadem skutečné rizikové funkce. Právě výsledky klasifikace na validační množině jsou tak pro nás klíčové při ladění hyperparametrů modelu.

Ztrátě generalizace modelu lze předcházet volbou rozsáhlé a reprezentativní trénovací množiny či technikami zabraňujícími přetrénování modelu úpravou modelu nebo procesu učení. Modifikacím architektury modelu či procesu učení vedoucím k prevenci přeučení modelu říkáme regularizační techniky [6]. V této sekci popisujeme regularizační techniky, které následně používáme jako prevenci proti přetrénování CNN během analýzy dat z experimentu NOvA v kapitole 5.



Obrázek 3.10: Zatímco modře označená empirická riziková funkce pro trénovací data klesá, oranžově značená empirická riziková funkce pro validační data po druhém trénovacím cyklu (Epoch) osciluje. To značí, že přesnost klasifikátoru na validačních datech se nezvyšuje, a model se tedy přetrénovává.

3.5.1 Early stopping

Jednou z velice intuitivních regularizačních technik je tzv. early stopping [20]. Během early stopping probíhá ukládání parametrů modelu v rámci každé trénovací epochy. Při dosažení tzv. ukončovacího kritéria je učení modelu ukončeno a early stopping uloží model s nejvýhodnějšími parametry. V praxi může být učení sítě zastaveno v k-té iteraci pro $k \in \mathbb{N}$, pokud se empirická riziková funkce J pro validační množinu v následující epoše zvýší, tedy pokud

$$J^{(k)} < J^{(k+1)}. \tag{3.17}$$

Uplatníme-li tedy early stopping na učení ilustrovaném na obrázku 3.10, učení bude ukončeno po druhé epoše a výsledný model bude obsahovat parametry právě z druhé epochy. V praxi takto definované kritérium nicméně může vlivem ojedinělé oscilace funkce J trénovací proces předčasně ukončit. Proto se zavádí parametr tolerance early stopping $c \in \mathbb{N}$. V takovém případě je učení v epoše k ukončeno až poté, co funkce J na validační množině dat v následujících c epochách nezačne klesat, tedy pokud

$$J^{(k)} < J^{(k+i)}, \tag{3.18}$$

pro všechna $i \in \{1, \ldots, c\}$. Tímto způsobem tak zamezíme předčasnému ukončení trénování způsobeného nahodilou oscilací empirické rizikové funkce pro validační data.

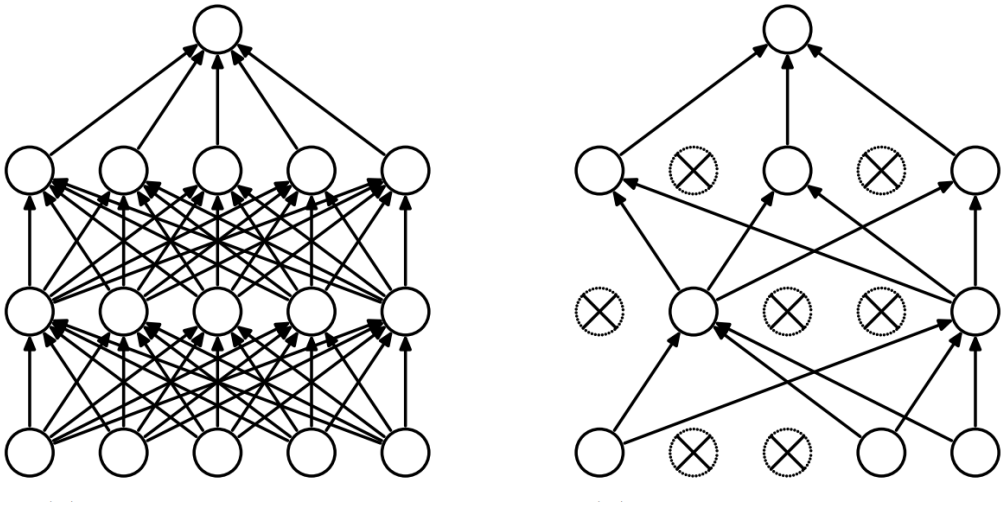
3.5.2 Dropout

Další regularizační technikou snižující riziko přetrénování modelu je tzv. dropout [23]. Hlavní myšlenkou metody dropout je zabránění přílišné adaptace parametrů na trénovací data deaktivací náhodně vybraných neuronových spojů ve skryté vrstvě pro konkrétní trénovací cyklus. V silnější verzi je poté možno deaktivovat namísto spojů přímo celé neurony. Tímto způsobem tak lze upravit ANN po vzoru schématu na obrázku 3.11. V každém trénovacím cyklu je pozměněna architektura modelu, což snižuje tendenci modelu k přetrénování. Zároveň tímto způsobem lze snížit počet trénovaných parametrů, a zrychlit tak učení modelu. To je typicky žádáno v plně propojených vrstvách CNN, do kterých vede vysoké množství spojů z předešlých konvolučních vrstev. Tuto metodu tak popisujeme právě s cílem použití v plně propojených vrstvách CNN. Pro matematický popis deaktivace neuronových spojů uvažujme síť s $\mathcal{L} \in \mathbb{N}$ skrytými vrstvami, kde $l \in \{1, \ldots, \mathcal{L}\}$ značí index vrstvy. Buď $\boldsymbol{x}^l \in \mathbb{R}^d$ vektorem výstupů z l-té vrstvy. Pak pro libovolný neuron lze výstup neuronu y^{l+1} neregularizované sítě vypočíst jako

$$y^{l+1} = f(\boldsymbol{w}^{l+1\mathsf{T}}\boldsymbol{x}^l + b^{l+1}), \tag{3.19}$$

kde f, $\boldsymbol{w}^{l+1}$ a b^{l+1} značí postupně aktivační funkci, vektor vah a práh neuronu. V případě použití regularizace dropout s uživatelem zvolenou pravděpodobností $p \in (0, 1)$ jednotlivé spoje mezi neurony přerušíme náhodným vynulováním složek $\boldsymbol{w}^{l+1} = (w_1^{l+1}, \ldots, w_d^{l+1})$, $d \in \mathbb{N}$. Uvažujeme tedy vektor vah $\tilde{\boldsymbol{w}}^{l+1}$ vybraný po složkách jako

$$\tilde{w}_i^{l+1} = \begin{cases} w_i^{l+1} & \text{s pravděpodobností } 1-p, \\ 0 & \text{s pravděpodobností } p, \end{cases} . \tag{3.20}$$



Obrázek 3.11: Obrázek vpravo ilustruje použití metody dropout na modelu ANN vykresleném vlevo. Spoje mezi neurony jsou znázorněny šipkami. Obrázek tak ilustruje, že s využitím metody dropout napříč modelem lze výrazně snížit počet parametrů při zachování komplexity modelu. Převzato z [23].

kde $i \in \{1, \ldots, d\}$. Složky $\tilde{w}_i^{l+1}$ tedy nabývají buď původních hodnot w_i^{l+1}, nebo hodnot nulových. To přitom dle Bernoulliho rozdělení s parametrem p. V každém trénovacím cyklu jsou poté jednotlivé neurony s pravděpodobností p deaktivovány vynulováním svých vstupů, zatímco s pravděpodobností $q = 1-p$ se vstupy neuronů nezmění. Hyperparametr p tak můžeme volit unikátně pro každou skrytou vrstvu s technikou dropout. Výstup neuronu lze poté získat jako

$$y^{l+1} = f\left(\left(\tilde{\boldsymbol{w}}^{l+1}\right)^{\mathsf{T}} \boldsymbol{x}^l + b^{l+1}\right). \tag{3.21}$$

Matematický popis deaktivace celých neuronů je analogickým způsobem popsán v [23].

3.5.3 Batch normalization

Učení ANN může komplikovat fakt, že se změnami velikostí parametrů v jednotlivých vrstvách se mění distribuce jejich výstupů, a tedy vstupů do následujících vrstev. Tento jev vyžaduje vysokou opatrnost ohledně volby inicializace parametrů a velikosti learning rate. Vzhledem k tomu, že se během tréninku parametry v jednotlivých vrstvách neustále adaptují na jiná rozdělení vstupů, příliš vysoká hodnota learning rate či neopatrně inicializované parametry mohou ve skrytých vrstvách způsobit nestabilitu při učení modelu a explozi hodnot gradientů. Uživatel je tak nucen volit nižší hodnotu learning rate, čímž se však celkově prodlužuje délka učení modelu. V rámci [15] byl nicméně představen koncept normalizace várek trénovacích dat (v angličtině tzv. batch normalization), který umožňuje stabilnější průběh trénování s menší citlivostí na volbu learning rate a inicializaci vah.

Nechť je dán obecný trénovací systém $n \in \mathbb{N}$ vstupů $\boldsymbol{x}_i \in \mathbb{R}^d$, $d \in \mathbb{N}$, do jisté vrstvy sítě, přičemž $i \in \{1, \ldots, n\}$. Uvažujeme přitom d dimenzionální vstupy $\boldsymbol{x}_i = \left(x_i^1, x_i^2, \ldots, x_i^d\right)$.

V rámci normalizace pro každé $k \in \{1, \dots, d\}$ normalizujeme každý prvek

$$\hat{x}_i^k = \frac{x_i^k - \hat{\mu}^k}{\hat{\sigma}^k}, \tag{3.22}$$

kde

$$\hat{\mu}^k = \frac{1}{n} \sum_{i=1}^{n} x_i^k, \tag{3.23}$$

$$\hat{\sigma}^k = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(x_i^k - \hat{\mu}^k\right)^2}. \tag{3.24}$$

V této fázi nicméně skrze normalizaci ovlivňujeme reprezentační schopnosti modelu. Z tohoto důvodu v rámci batch normalization pro vstup $\boldsymbol{x}_i \in \mathbb{R}^d$ po normalizaci z (3.22) používáme též škálování a posun parametry $\gamma^k, \beta^k \in \mathbb{R}$ v podobě

$$a_i^k = \gamma^k \hat{x}_i^k + \beta^k. \tag{3.25}$$

Parametry γ^k, β^k jsou natrénovány během učení společně s dalšími parametry modelu. Díky škálování dle (3.25) je poté možno upravit normalizaci ve prospěch rychlejší minimalizace cílové funkce oproti (3.22). V extrémním případě pro $\gamma^k = \hat{\sigma}^k$ a $\beta^k = \hat{\mu}^k$ přechází transformace v identitu.

Jak jsme popsali v sekci 3.2, v praxi nicméně jednotlivé dopředné kroky často provádíme namísto s celou trénovací sadou dat po sériích trénovacích várek. Uvažujeme tedy várku $m \in \mathbb{N}, m < n$ pozorování $\mathcal{B} = (\boldsymbol{x}_{i_1}, \dots, \boldsymbol{x}_{i_m})$, kde $i_j \in \{1, \dots, n\}$ pro $j \in \{1, \dots, m\}$ dle popisu v sekci 3.2. Pak provádíme normalizaci pozorování z $\mathcal{B}$ analogicky po vzoru rovnic (3.22) a (3.25). Pro normalizaci $\mathcal{B}$ tedy napočítáme výrazy

$$\hat{\mu}_{\mathcal{B}}^k = \frac{1}{m} \sum_{j=1}^{m} x_{i_j}^k, \tag{3.26}$$

$$\hat{\sigma}_{\mathcal{B}}^k = \sqrt{\frac{1}{m} \sum_{j=1}^{m} \left(x_{i_j}^k - \hat{\mu}_{\mathcal{B}}^k\right)^2}. \tag{3.27}$$

S využitím vztahů (3.26) a (3.27) nyní můžeme provést transformaci dle [15] jako

$$\hat{x}_{i_j}^k = \frac{x_{i_j}^k - \hat{\mu}_{\mathcal{B}}^k}{\hat{\sigma}_{\mathcal{B}}^k}, \tag{3.28}$$

$$a_{i_j}^k = \gamma^k \hat{x}_{i_j}^k + \beta^k, \tag{3.29}$$

Celou transformaci $x_{i_j}^k \to a_{i_j}^k$ poté zapisujeme jako

$$\mathrm{BN}_{\gamma^k, \beta^k}(x_{i_j}^k) = a_{i_j}^k. \tag{3.30}$$

Pro přehlednost notace popisujeme skrze rovnici (3.22) normalizaci vstupů ANN s vektorovým vstupem $\boldsymbol{x}$. V případě CNN je várka $\mathcal{B}$ tvořena $m \in \mathbb{N}$ vstupními maticemi.

Požadujeme přitom, aby všechny prvky na stejné příznakové mapě byly normalizovány stejným způsobem pro zachování interpretace vstupu jako obrázku. Z tohoto důvodu normalizujeme každý prvek vstupu napříč všemi prvky všech příznakových map ve várce.

Mimo rychlejší minimalizace empirické rizikové funkce funguje batch normalization i jako regularizační metoda. V rovnicích (3.26) a (3.27) jsou odhady výběrových statistik počítány vždy pouze pro danou várku $\mathcal{B}$, nikoliv pro celou trénovací množinu. Díky tomu dochází k vnášení šumu během každé normalizace, což zvyšuje generalizační potenciál modelu. Tento regularizační efekt roste pro menší velikosti várek. S postupným růstem mohutnosti várky $\mathcal{B}$ vzhledem k mohutnosti celé trénovací množiny dochází k útlumu šumu, a tedy i k útlumu regularizačních vlastností normalizace.

Kapitola 4

Metody strojového učení založené na rozhodovacích stromech

Kromě neuronových sítí lze pro klasifikační účely použít řadu dalších ML technik. V této sekci proto popisujeme další ML algoritmy založené na schématu tzv. rozhodovacího stromu. Podobně jako v předchozí kapitole 3, i v tomto případě se pro učení modelů používá trénovací množina dat. DL modely jako CNN jsou schopné si ze surových obrazových dat samostatně extrahovat vhodné příznaky pro následnou klasifikaci. Obecně je ovšem pro ML modely nutné vstupní obrazové příznaky extrahovat manuálně. Pro extrakci příznaků ze surových obrazových dat je nicméně možné použít konvoluční vrstvy CNN. Algoritmy popsané v této kapitole proto prezentujeme právě s motivací použít je pro klasifikaci na příznacích extrahovaných z modelů CNN. Algoritmy založenými na rozhodovacích stromech tak můžeme nahradit plně propojenou vrstvu CNN.

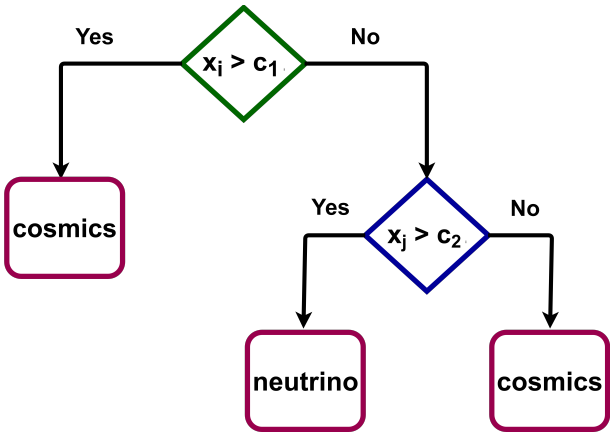
4.1 Rozhodovací stromy

Jedním z často používaných klasifikačních algoritmů je rozhodovací strom [6]. Namísto komplexních maticových operací a nelineárních transformací v případě neuronových sítí rozhodovací stromy klasifikují vstupní pozorování na základě série rozhodovacích procesů. Rozhodovací proces uvnitř rozhodovacího stromu je možno definovat jako binární porovnání, zda jistý prvek x_j vstupního vektoru $\boldsymbol{x} \in \mathbb{R}^d$, $d \in \mathbb{N}$, splňuje pro parametr $c_k \in \mathbb{R}$, $k \in \mathbb{N}$ podmínku

$$x_j > c_k, \tag{4.1}$$

pro $j \in \{1, \ldots, d\}$. Tímto způsobem tak data rozřazujeme do dvou tzv. rozhodovacích regionů $R_1 = \{\boldsymbol{x} | x_j > c_k\}$ a $R_2 = \{\boldsymbol{x} | x_j \leq c_k\}$. Index j a hodnotu c_k přitom volíme s cílem co nejlépe v daném kroku dle rozhodovacího pravidla separovat data, což lze v praxi provést skrze minimalizaci jistého klasifikačního kritéria. Tím může být například entropie. První takto vybrané binární rozhodování nazýváme kořen stromu. Podle splnění rozhodovací podmínky v kořenu nicméně můžeme pro jednotlivé rozhodovací regiony pokračovat dále po tzv. rozhodovacích hranách k dalším rozhodovacím uzlům. Uzel, ze kterého nevychází další rozhodovací hrany, nazýváme list. Právě počet listů poté určuje počet rozhodovacích

regionů na konci rozhodovacího stromu. Strukturu rozhodovacích stromů ilustrujeme na příkladu separace pozorování neutrin od kosmického záření na obrázku 4.1.



Obrázek 4.1: Schéma rozhodovacího stromu na ilustrativním příkladu binární klasifikace. Zelená buňka znázorňuje kořen stromu, modrá buňka rozhodovací uzel a fialové buňky listy. Výrazy x_i a x_j jsou složky vstupního vektoru $\boldsymbol{x}$.

Počet úrovní stromu počítaný od kořene po nejhlubší list nazýváme hloubkou stromu. Maximální hloubka stromu je uživatelem volený hyperparametr a určuje komplexitu modelu. S větší hloubkou modelu totiž roste také počet uzlů obsahujících rozhodovací pravidla. Díky tomuto principu tak rozhodovací stromy nabízejí oproti ANN možnost jednoduché interpretace modelu. Je totiž možno jednoduše zjistit na základě jakých podmínek model klasifikuje vstupní pozorování.

Popsané schéma sepíšeme matematicky. Uvažujme rozhodovací strom s $M \in \mathbb{N}$ listy určujícími rozhodovací regiony $R_1, \ldots, R_M$. Označme nyní $N_m \in \mathbb{N}$ jako počet trénovacích pozorování $\boldsymbol{x}_i$ v regionu R_m, tedy

$$N_m = \text{card}\left\{\boldsymbol{x}_i \in R_m | i \in \{1, \ldots, n\}\right\}, \tag{4.2}$$

kde $m \in \{1, \ldots, M\}$ a $n \in \mathbb{N}$ je velikost trénovací množiny. Potom můžeme pro list m v rozhodovacím stromu odhadnout pravděpodobnost příslušnosti pozorování z regionu R_m ke třídě ω_j jako

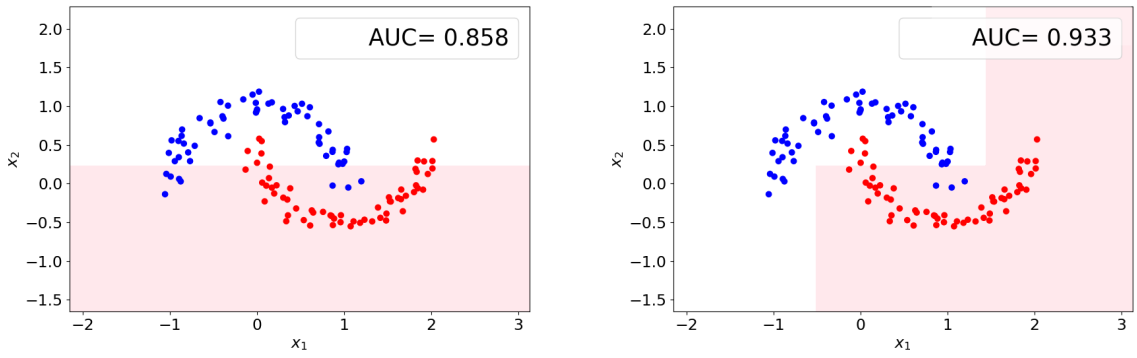
$$\hat{p}_{j,m} = \frac{1}{N_m} \sum_{\substack{\boldsymbol{x}_i \in R_m \\ i \in \{1,\ldots,n\}}} \mathrm{I}(\boldsymbol{x_i} \in \omega_j), \tag{4.3}$$

kde $j \in \{1, \ldots, \mathscr{L}\}$ označuje index třídy a

$$\mathrm{I}(\boldsymbol{x_i} \in \omega_j) = \begin{cases} 1 \text{ pokud } \boldsymbol{x_i} \in \omega_j, \\ 0 \text{ jinak }, \end{cases} \tag{4.4}$$

je identifikátor třídy ω_j. Kategorickou entropii pro list m lze spočíst jako

$$H^{(m)} = -\sum_{j=1}^{\mathscr{L}} \hat{p}_{j,m} \log(\hat{p}_{j,m}). \tag{4.5}$$



(a) maximální hloubka 1

(b) maximální hloubka 2

(c) maximální hloubka 5

(d) maximální hloubka 10

Obrázek 4.2: Ukázka klasifikace simulovaných 2D dat pomocí rozhodovacích stromů s různými hloubkami. Obrázek zobrazuje data s indikací množin a separační hranici. Model s větší hloubkou dokázal lépe zachytit reálný vzor v datech, a dosáhl tak vyšší hodnoty klasifikační metriky AUC. Na případě rozhodovacího stromu o hloubce 5 a více nicméně pozorujeme přetrénování modelu. Rozhodovací stromy totiž v těchto případech zřejmě přestávají zachytávat skutečný vzor v datech. Na obrázku je též patrný odlišný tvar rozhodovací hranice oproti rozhodovací hranici ANN ilustrované na obrázku 3.4 pro stejnou úlohu.

Budeme-li pokračovat v růstu rozhodovacího stromu, právě na základě minimalizace entropie v uzlu budeme vybírat další rozhodovací pravidlo. Tímto způsobem můžeme pokračovat v rozdělování uzlů až do dosažení maximální hloubky stromu. V praxi nicméně z listu vytváříme další uzel pouze když tím dosáhneme snížení entropie o zadanou hraniční hodnotu. Ta může být volena jako uživatelský hyperparametr. Struktura stromů tak nemusí být symetrická. Predikce listu m rozhodovacího stromu h je poté pro $j, k \in \{1, \ldots, \mathscr{L}\}$ určena jako

$$h_k^{(m)} = \begin{cases} 1 \text{ pro } k = \operatorname{argmax}_j \hat{p}_{j,m}, \\ 0 \text{ jinak }, \end{cases} \tag{4.6}$$

což je ekvivalentní predikci třídy $\hat{y}^{(m)} \in \{1, \ldots, \mathscr{L}\}$ pro list m jako

$$\hat{y}^{(m)} = \underset{j}{\operatorname{argmax}}\, \hat{p}_{j,m}. \tag{4.7}$$

Detailnější popis učení rozhodovacích stromů je k dispozici v [6]. Rozhodovací hranice u rozhodovacích lesů jsou určeny na základě rozhodovacích regionů R_m. To ilustrujeme na příkladu binární klasifikace na simulovaných datech na obrázku 4.2. Modifikované rozhodovací stromy se pro jednoduché úlohy, jako je například právě separace CC interakcí od kosmického záření, používají i na experimentu NOvA [2].

4.2 Bagging a náhodné lesy

Rozhodovací stromy jsou pro hledání komplikovaných klasifikačních vzorů často nevhodné kvůli sklonu k overfittingu. V praxi proto používáme další rozšiřující techniky vedoucí k lepším klasifikačním vlastnostem modelu jako například tzv. náhodné lesy [8]. Náhodné lesy se skládají ze systému několika rozhodovacích stromů. Ty slouží jako tzv. slabé klasifikátory, jejichž složením získáváme tzv. silný klasifikátor v podobě náhodného lesa. Slabým klasifikátorem přitom myslíme rozhodovací strom s alespoň o trochu větší přesností než náhodný klasifikátor. Každý rozhodovací strom v náhodném lese predikuje pro dané pozorování výslednou třídu a nejčastěji predikovaná třída se stane predikovanou třídou náhodného lesa.

Klíčovým požadavkem pro správnou funkčnost náhodných lesů je nekorelovanost slabých klasifikátorů [6]. Díky nekorelovanosti rozhodovacích stromů totiž slabé klasifikátory nebudou kopírovat stejné chyby a s vyšším počtem slabých klasifikátorů bude možné, aby byly chybné klasifikátory tzv. přehlasovány. K tomu je využíváno silné závislosti rozhodovacích stromů na vstupních trénovacích datech. Složení dat použitých pro trénování modelu totiž dokáže zásadně ovlivnit parametry natrénovaného rozhodovacího stromu. Pokud tak jednotlivé rozhodovací stromy budeme trénovat na vzájemně rozdílných trénovacích množinách, můžeme se tak přiblížit jejich nekorelovanosti.

První možností jak se přiblížit nekorelovanosti rozhodovacích stromů je odebrání náhodně vybraných pozorování z trénovací množiny. Každý strom tak trénujeme na náhodně vybrané podmnožině trénovacích pozorování. Tuto techniku nazýváme bagging. Uvažujme trénovací množinu $n \in \mathbb{N}$ pozorování $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$. Nechť přitom $\boldsymbol{x}_i \in \mathbb{R}^d$ a $\boldsymbol{y}_i \in \{0,1\}^{\mathscr{L}}$, kde $d \in \mathbb{N}$ a $\mathscr{L} \in \mathbb{N}$ označuje počet tříd. Pro každý slabý klasifikátor z náhodného lesa o $T \in \mathbb{N}$ rozhodovacích stromech poté provádíme uniformně náhodný výběr s opakováním n pozorování z trénovací množiny. Pro rozhodovací strom o indexu $t \in \{1, \ldots, T\}$ takto vzniká množina $\mathcal{B}_t = \big((\boldsymbol{x}_{i_1}, \boldsymbol{y}_{i_1}), \ldots, (\boldsymbol{x}_{i_n}, \boldsymbol{y}_{i_n})\big)$, kde $i_j \in \{1, \ldots, n\}$ pro $j \in \{1, \ldots, n\}$. Tento systém tedy obsahuje n pozorování z trénovací množiny bez podmínky jejich unikátnosti. Soubor $\mathcal{B}_t$ má tedy stejný počet pozorování jako původní trénovací množina, některá pozorování z trénovací množiny se v něm však mohou opakovat. Rozhodovací stromy $h_1, \ldots, h_T$ z náhodného lesa tedy trénujeme postupně na různě poskládaných množinách $\mathcal{B}_1, \ldots, \mathcal{B}_T$, což ovlivní parametry náhodných lesů, a vede tak k jejich nekorelovanosti.

Další možností jak znáhodnit strukturu stromů v náhodném lese je omezení počtu vstupních příznaků, tedy omezení dimenze vstupu. Pak ze vstupních trénovacích pozo-

rování $\boldsymbol{x}_i \in \mathbb{R}^d$ náhodným výběrem $d' < d$ příznaků vytvoříme nové vstupní pozorování $\boldsymbol{x}'_i \in \mathbb{R}^{d'}$. Každý slabý klasifikátor přitom může používat jinou sadu d' příznaků. Tento proces můžeme matematicky zapsat s využitím výběrové funkce $S_t(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^{d'}$, jež pro $t \in \{1, \ldots, T\}$ vybere z $\boldsymbol{x} \in \mathbb{R}^d$ pouze d' dimenzí. Po použití techniky bagging a snížení počtu vstupních příznaků poté můžeme získat predikci náhodného lesa $F_T(\boldsymbol{x}) : \mathbb{R}^d \to \{0, 1\}^{\mathscr{L}}$ složeného ze systému T rozhodovacích stromů $h_t : \mathbb{R}^{d'} \to \{0, 1\}^{\mathscr{L}}$ jako

$$F_T(\boldsymbol{x}) = \underset{\boldsymbol{y}}{\operatorname{argmax}} \sum_{t=1}^{T} \mathrm{I}\left(\boldsymbol{y} = h_t\left[S_t(\boldsymbol{x})\right]\right), \tag{4.8}$$

kde strom h_t je natrénován na datech ze souboru $\mathcal{B}_t$. Výsledný vektor F_T tedy odpovídá indikaci predikované třídy.

4.3 Boosting

Kromě baggingu se používá další metoda rozšiřující rozhodovací stromy, tzv. boosting. Myšlenkou boostingu je využití sady slabých klasifikátorů k vytvoření jednoho silného modelu. Pro tyto účely nám opět postačí například sada rozhodovacích stromů, jež fungují lépe než náhodný klasifikátor. Oproti baggingu se nicméně liší způsob učení silného klasifikátoru. V této práci se zaměříme na algoritmy AdaBoost a Gradient Boosting.

4.3.1 AdaBoost

Technika AdaBoost [28] spočívá v trénování silného klasifikátoru $F_T : \mathbb{R}^d \to \mathbb{R}^{\mathscr{L}}$ složeného jako váženého součtu $T \in \mathbb{N}$ slabých klasifikátorů $h_t(\boldsymbol{x})$ s koeficienty $\alpha_t \in \mathbb{R}^+$ jako

$$F_T(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}), \tag{4.9}$$

kde $\boldsymbol{x} \in \mathbb{R}^d$, $d \in \mathbb{N}$, je vstupní pozorování a $\mathscr{L} \in \mathbb{N}$ je počet tříd. Jako slabý klasifikátor lze poté typicky použít rozhodovací strom s nízkou hloubkou. Právě druh slabého klasifikátoru a jejich počet T jsou uživatelem zadávané hyperparametry. Každý slabý klasifikátor poté produkuje výstup $h_t(\boldsymbol{x}_i)$ pro každé pozorování $\boldsymbol{x}_i$ o indexu $i \in \{1, \ldots, n\}$ z trénovací sady $n \in \mathbb{N}$ pozorování. Další iterativně přidaný rozhodovací strom h_{t+1} se snaží vylepšit silný klasifikátor F_t z předchozí iterace. Slabé klasifikátory s vyšší přesností pak skrze větší hodnoty α_t přispívají silnému klasifikátoru více nežli hůře fungující slabé klasifikátory. Zlepšit silný klasifikátor lze typicky přidáním slabého klasifikátoru, který dokáže správně zařadit dříve špatně klasifikovaná data. Takové slabé klasifikátory jsou poté zvýhodněny větší hodnotou koeficientu α_t. Specifikum algoritmu AdaBoost je totiž adaptace nově přidaných slabých klasifikátorů na dříve nesprávně klasifikovaná pozorování. Na základě chybovosti klasifikace pozorování $\boldsymbol{x}_i$ je totiž možno $\boldsymbol{x}_i$ přidat váhu $w_{i,t} \in \mathbb{R}^+$. Váhy $w_{i,t}$ pro index pozorování $i \in \{1, \ldots, n\}$ a index iterace $t \in \{1, \ldots, T\}$ inicializujeme rovnoměrně jako

$$w_{i,t} = \frac{1}{n}. \tag{4.10}$$

V pozdějších iteracích pak můžeme zvětšit váhu špatně klasifikovaným pozorováním. Nově přidanými klasifikátory je poté možno zaměřit se právě na špatně klasifikovaná pozorování. To v praxi provádíme takovým způsobem, že po přidání slabého klasifikátoru o indexu $t \in \{1, \ldots, T\}$ napočítáme váženou chybu ϵ_t [28] na špatně klasifikovaných datech jako

$$\epsilon_t = \frac{\sum_{i=1}^{n} w_{i,t} \mathrm{I}(h_t(\boldsymbol{x_i}) \neq \boldsymbol{y}_i)}{\sum_{i=1}^{n} w_{i,t}}, \tag{4.11}$$

s jejíž využitím určíme koeficient slabého klasifikátoru α_t pro $\mathscr{L} \geq 2$ tříd jako

$$\alpha_t = \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) + \log(\mathscr{L} - 1). \tag{4.12}$$

Důvod této volby α_t je netriviální, odvození je rozepsáno v [28]. Následně přepočítáme váhy pro další iteraci dle vzorce

$$w_{i,t+1} = w_{i,t} \mathrm{e}^{\alpha_t \mathrm{I}(h_t(\boldsymbol{x_i}) \neq \boldsymbol{y}_i)}, \tag{4.13}$$

a po normalizaci na $\sum_{i=1}^{n} w_{i,t+1} = 1$ cyklus opakujeme až do dosažení maximálního počtu slabých klasifikátorů T. Výslednou indikaci predikované třídy $\hat{\boldsymbol{y}} \in \{0,1\}^{\mathscr{L}}$ poté určíme jako

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}} \sum_{t=1}^{T} \alpha_t \mathrm{I}\left(\boldsymbol{y} = h_t(\boldsymbol{x})\right). \tag{4.14}$$

Nutným předpokladem pro funkčnost modelu je nicméně schopnost slabých modelů rozhodovat lépe než náhodný klasifikátor. Horším než náhodným klasifikátorům by potom dle (4.12) byl přiřazován záporný koeficient $\alpha_t < 0$ a algoritmus by zkolaboval.

4.3.2 Gradient Boosting

Další metodou využívající sadu rozhodovacích stromů k vytvoření jednoho silného klasifikátoru je tzv. Gradient Boosting [18]. Podobně jako AdaBoost, i Gradient Boosting modeluje silný klasifikátor $F_T(\boldsymbol{x})$ jako váženou sumu $T \in \mathbb{N}$ slabých klasifikátorů s koeficienty $\alpha_t > 0$ jako

$$F_T(\boldsymbol{x}) = \sum_{t=1}^{T} f_t(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}), \tag{4.15}$$

kde používáme stejného značení jako v případě AdaBoost. Uvažujeme opět trénovací množinu $n \in \mathbb{N}$ pozorování $\boldsymbol{x}_i \in \mathbb{R}^d$ s indikacemi třídy $\boldsymbol{y}_i \in \{0,1\}^{\mathscr{L}}$, kde $i \in \{1, \ldots, n\}$. Cílem této metody je naleznout klasifikátor F_T jako model minimalizující sumu ztrátových funkcí $L : \mathbb{R}^{\mathscr{L}} \times \mathbb{R}^{\mathscr{L}} \to \mathbb{R}^{+}$ napříč trénovacími daty, tj.

$$F_T = \underset{\tilde{F}}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(\boldsymbol{y}_i, \tilde{F}(\boldsymbol{x}_i)\right). \tag{4.16}$$

Jako ztrátovou funkci L je pro tuto úlohu z netriviálních důvodů vysvětlených v [18] vhodné použít exponenciální ztrátovou funkci

$$L(\boldsymbol{y}_i, \tilde{F}(\boldsymbol{x}_i)) = \exp\left\{-\frac{1}{\mathscr{L}} \boldsymbol{y}_i^{\intercal} \tilde{F}(\boldsymbol{x}_i)\right\}. \tag{4.17}$$

Podobně jako v případě AdaBoost se i v tomto případě s každým dalším přidaným slabým klasifikátorem snažíme minimalizovat chybu silného klasifikátoru. Slabé klasifikátory tedy opět iterativně přidáváme dle předpisu

$$F_t(\boldsymbol{x}) = F_{t-1}(\boldsymbol{x}) + \alpha_t h_t(\boldsymbol{x}). \tag{4.18}$$

Tuto minimalizační optimalizační úlohu lze poté podobně jako v případě ANN řešit za pomocí metody gradient descent. Na počátku trénování model $F_0 \in \mathbb{R}^{\mathscr{L}}$ inicializujeme konstantou jako

$$F_0 = \underset{\tilde{F} \in \mathbb{R}^{\mathscr{L}}}{\operatorname{argmin}} \sum_{i=1}^{n} L(\boldsymbol{y}_i, \tilde{F}). \tag{4.19}$$

Úlohu zadanou rovnicí (4.19) lze v praxi řešit například metodou tzv. exhaustive search [19]. Naším cílem je další slabé klasifikátory h_t přidávat způsobem

$$F_t = F_{t-1} + \underset{h_t}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(\boldsymbol{y}_i, F_{t-1}(\boldsymbol{x}_i) + h_t(\boldsymbol{x}_i)\right). \tag{4.20}$$

To je však obecně neřešitelná optimalizační úloha, proto pokračujeme minimalizací ve směru nejvyššího gradientního spádu dle rovnice

$$F_t = F_{t-1} - \alpha_t \sum_{i=1}^{n} \nabla_{F_{t-1}} L\left(\boldsymbol{y}_i, F_{t-1}(\boldsymbol{x}_i)\right), \tag{4.21}$$

kde $\nabla_{F_{t-1}} L$ je gradient ztrátové funkce L podle F_{t-1}. Přibližné rovnosti ve vztahu 4.21 lze nyní docílit natrénováním dalšího slabého klasifikátoru h_t na záporný gradient ztrátové funkce L. Zavedeme-li nyní tzv. pseudoresidua

$$\boldsymbol{r}_{i,t} = -\left[\frac{\partial L(\boldsymbol{y}_i, F(\boldsymbol{x}_i))}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x}_i)=F_{t-1}(\boldsymbol{x}_i)}, \tag{4.22}$$

pak se platnosti (4.21) můžeme přiblížit trénováním slabého klasifikátoru h_t na trénovacích datech $\{(\boldsymbol{x}_i, \boldsymbol{r}_{i,t})\}_{i=1}^{n}$. Koeficienty α_t pro slabý klasifikátor h_t určujeme na základě pravidla

$$\alpha_t = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(\boldsymbol{y}_i, F_{t-1} - \alpha \nabla_{F_{t-1}} L\left(\boldsymbol{y}_i, F_{t-1}(\boldsymbol{x}_i)\right)\right), \tag{4.23}$$

které při trénování na pseudoresidua přechází do tvaru

$$\alpha_t = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(\boldsymbol{y}_i, F_{t-1} + \alpha h_t(\boldsymbol{x}_i)\right). \tag{4.24}$$

Rovnice (4.24) tedy zadává jednoduchou optimalizační úlohu řešitelnou například metodou exhaustive search. Na závěr cyklu aktualizujeme silný klasifikátor F_t jako

$$F_t = F_{t-1} + \alpha_t h_t. \tag{4.25}$$

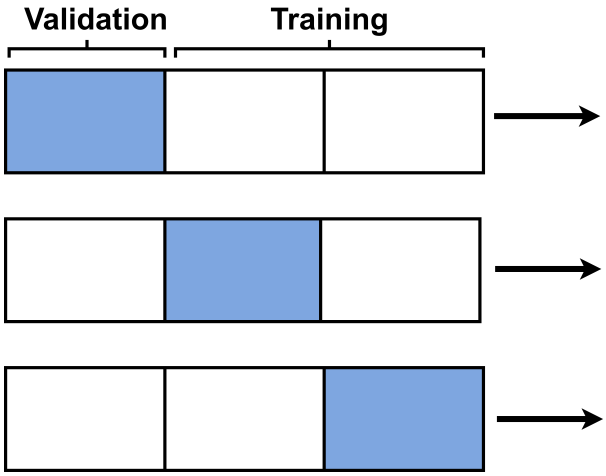
Tímto způsobem trénování opakujeme dokud není dosažen maximální počet slabých klasifikátorů T. S větším počtem slabých klasifikátorů se tak zvyšuje přesnost modelu na

trénovacích datech. Výslednou predikovanou třídu poté podobně jako v předchozím případě určíme jako analogicky dle rovnice (4.14) jako v případě AdaBoost.

Pro přehlednost praktické implementace algoritmu ve stručnosti ještě jednou opakujeme použití výše odvozených kroků v praxi. Pro implementaci modelu Gradient Boosting po inicializaci F_0 dle (4.19) prvně vypočteme pseudoresidua dle (4.22). Následně na takto vypočtená pseudoresidua natrénujeme slabý klasifikátor h_t, čímž bude přibližně platit vztah (4.21). Se znalostí na residuích natrénovaného stromu h_t dále dopočítáme příslušnou váhu α_t dle (4.24). Na závěr aktualizujeme silný klasifikátor F_t dle (4.25) a celý cyklus opakujeme do dosažení $t = T$.

4.4 Křížová validace a výběr hyperparametrů

Protože ML modely mohou být náchylné k overfittingu, je vhodné přetrénovanost modelu kontrolovat tzv. křížovou validací. Ta spočívá v rozdělení trénovacích dat do $k \in \mathbb{N}$ stejně velkých exkluzivních trénovacích podmnožin. Poté se model k-krát natrénuje na kombinacích $(k-1)$ podmnožin sloučených dohromady, zatímco jedna zbývající validační podmnožina slouží pro vyhodnocování klasifikace. Schéma křížové validace ilustrujeme na obrázku 4.3.



Obrázek 4.3: Schéma křížové validace pro rozdělení do $k = 3$ skupin. Model je celkově trénován třikrát, přičemž modře označená část trénovacích dat je vyhrazena pro validaci (validation) modelu. Zbylá část dat je určena pro natrénování modelu (training). Celková přesnost modelu je poté vypočtena jako aritmetický průměr výsledků na k kombinacích trénovacích dat.

Použijeme-li notaci z předchozí sekce, trénováním na k trénovacích podmnožinách získáváme modely $F_1, \ldots, F_k$, jejichž přesnost měříme postupně na validačních množinách $\mathcal{B}_1, \ldots, \mathcal{B}_k$. K měření přesnosti modelů lze použít například ztrátovou funkci L nebo metriky popsané v kapitole 2. Obecně budeme toto kritérium úspěšnosti klasifikace nazývat skóre a označovat jako score. Označíme-li score_j skóre modelu F_j na validační množině $\mathcal{B}_j$, $j \in \{1, \ldots, k\}$, pak celkové skóre je počítáno jako aritmetický průměr skóre modelů

na k validačních sadách

$$\text{score} = \frac{1}{k} \sum_{j=1}^{k} \text{score}_j. \tag{4.26}$$

Pokud tak model na nějaké podmnožině dat $\mathcal{B}_j$ nefunguje dostatečně dobře, je penalizováno jeho celkové skóre.

Přesnost klasifikace výše popsaných ML modelů je silně závislá na volbě hyperparametrů, jako je hloubka stromu či počet použitých slabých klasifikátorů. Optimální volba hyperparametrů je nicméně silně vázána na konkrétní povahu úlohy a dat. Optimální hodnoty hyperparametrů tak může být poměrně obtížné odhadnout. Nejjednodušším řešením, jak se riziku špatné volby hyperparametrů vyhnout, je tzv. grid search. Tato metoda spočívá v natrénování modelu na všech kombinacích hyperparametrů z uživatelem zadané mřížky. Zásadní nevýhodou této metody je výpočetní náročnost. Při pokrytí dostatečného rozsahu hodnot hyperparametrů se u modelů s vyšší dimenzí hyperparametrického prostoru dostáváme na velice vysoké množství kombinací. V případě použití křížové validace je následně počet modelů k natrénování ještě umocněn. Úspornější metodou je tak tzv. randomized search. Tato metoda podobně jako grid search vyžaduje uživatelem zadaný prostor přípustných hyperparametrů. Namísto zkoušení všech kombinací nicméně randomized search prochází uživatelem zvolený pevný počet náhodně vybraných kombinací hyperparametrů. Díky tomu tak může randomized search prohledat rozsáhlý prostor hyperparametrů při zachování libovolného počtu kombinací. Při zvolení příliš nízkého počtu kombinací nicméně může randomized search výrazně minout okolí globálně optimální kombinace hyperparametrů. Celkově tak trénujeme modely pro sadu různých kombinací hyperparametrů, přičemž jejich přesnost hodnotíme na základě skóre z křížové validace. Výsledný model poté vybereme jako model s nejlepším skóre.

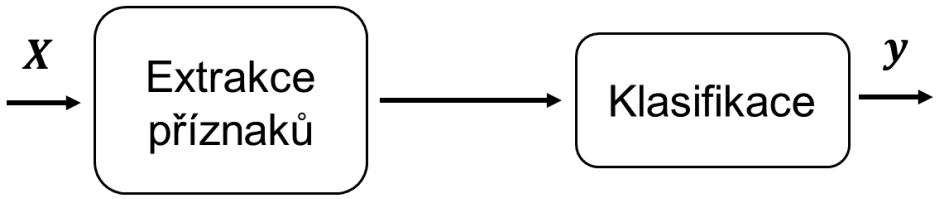
Kapitola 5

Klasifikace neutrinových interakcí z experimentu NOvA

V této kapitole využíváme poznatky z předchozích kapitol k analýze simulovaných interakcí ze vzdáleného detektoru experimentu NOvA. Konkrétně provádíme úlohu klasifikace do tříd ν_μ, ν_e, ν_τ, NC a kosmického záření popsaných v kapitole 1. Jak jsme již v úvodní kapitole uvedli, snímky interakcí z detektorů experimentu NOvA mají povahu obrazových dat. Z tohoto důvodu pro klasifikaci používáme CNN popsané v kapitole 3. Ty se dokážou ze surových obrazových dat naučit příznaky typické pro jednotlivé topologie trajektorií a na jejich základě klasifikovat vstupní snímky.

Pro srovnání nejdříve sestavujeme základní model CNN tvořený dvěma konvolučními vrstvami. Na tomto modelu budeme ilustrovat, jak veliký vliv má na přesnost klasifikace použití komplexních sítí s technikami popsanými v sekci 3.4. Pro porovnání používáme referenční model CVN využívající metody inception modulů popsaných v sekci 3.4.1. Tento model je díky dobrému poměru přesnosti klasifikace a počtu parametrů používán pro klasifikaci neutrinových interakcí na experimentu NOvA [2]. Dále konstruujeme model ResNet využívající residuálního učení popsaného v sekci 3.4.2. Naším cílem poté bude přiblížit se klasifikační přesnosti CVN, a to převážně pro klíčové třídy ν_μ a ν_e. Jak jsme již uvedli v kapitole 1, experiment NOvA se zaměřuje na pozorování oscilace $\nu_\mu \rightarrow \nu_e$, a právě proto jsou pro nás výsledky na třídách ν_μ a ν_e zásadní.

Model ResNet dále rozšiřujeme s využitím technik ML popsaných v kapitole 4. Zatímco konvoluční vrstvy v ResNet slouží k extrakci příznaků, závěrečná plně propojená vrstva slouží ke klasifikaci do jednotlivých tříd s využitím právě získaných příznaků. CNN jsou pro rozpoznávání obrazových dat vhodné právě díky schopnosti extrakce příznaků. Co se samotné klasifikace s nalezenými příznaky týče, ML metody ze sekce 4 mohou nabídnout vyšší přesnost klasifikace než skrytá plně propojená vrstva. Proto dále implementujeme hybridní model, který využije extrahovaných příznaků z konvolučních vrstev ResNet a následně pro jejich klasifikaci namísto plně propojené vrstvy používá popsané ML metody. Konkrétně se zaměřujeme na náhodné lesy, AdaBoost a Gradient Boosting. Myšlenku hybridního modelu ilustrujeme na obrázku 5.1. Podobné schéma klasifikace s využitím příznaků extrahovaných z CNN bylo použito v [25], kde vedlo ke zlepšení klasifikace oproti samotnému modelu CNN. Tuto myšlenku prakticky realizujeme tak, že z modelu ResNet



Obrázek 5.1: Schéma kombinace CNN pro extrakci příznaků ze vstupu $\boldsymbol{X}$ a klasifikátoru pro jejich klasifikaci.

natrénovaného na příslušných datech vyjmeme příznakové mapy z poslední konvoluční vrstvy a použijeme je jako vstupy do ML klasifikátoru.

5.1 Data

Celkově máme k dispozici 3 miliony vstupních snímků vygenerovaných ze 3300 simulovaných množin dat. Jak již bylo popsáno v kapitole 1, neutrinový svazek je standardně tvořen mionovými neutriny ν_μ. Simulace v takovém případě z neutrinových interakcí obsahují převážně interakce ν_μ s malým množstvím ν_e a stopovým množstvím ν_τ. Z tohoto důvodu byla použita též data ze simulací pro jiná nastavení svazku obsahující převažující zastoupení zbylých neutrinových interakcí. Tímto způsobem tak snižujeme nevybalancovanost datové množiny. Mimo neutrinové interakce zachycují simulace též interakce z kosmického záření. Ačkoliv to v praxi tvoří drtivou většinu dat zaznamenaných detektorem, v naší datové množině pevně nastavujeme zastoupení kosmického záření na 10 %. Částice kosmického záření totiž do detektoru typicky vlétají shora pod zcela jinými úhly než částice ze svazku, díky čemuž je lze snáze klasifikovat. Pro natrénování modelu proto není nezbytné tak vysoké množství pozorování z této třídy. Kompletní zastoupení jednotlivých tříd v datové množině je uvedeno v tabulce 5.1. Každé pozorování se skládá

	ν_μ CC	ν_e CC	ν_τ CC	NC	kosmika
Počet pozorování	0.8M	0.8M	0.3M	0.8M	0.3M

Tabulka 5.1: Zastoupení jednotlivých tříd v datové množině.

ze dvojice snímků pro oba pohledy o rozměrech 100×80 pixelů. Intenzita jednotlivých pixelů $I \in \{0, \ldots, 255\}$ poté odpovídá normovanému počtu fotoelektronů emitovaných v příslušné buňce detektoru. Pro trénování modelů používáme 80 % trénovacích dat. Dalších 10 % používáme jako validační množinu pro ladění hyperparametrů modelu. Se zbylými 10 % dat model během trénování nepřijde do styku, a používáme je proto pro výsledné testování přesnosti klasifikace modelu. Celkově tak datovou sadu rozdělujeme na

tři disjunktní množiny, které dále označujeme jako trénovací (training), validační (valid) a testovací (test).

5.2 Modely

Pro všechny modely používáme jako optimalizační proces mini-batch GD s learning rate $\varepsilon = 0.002$. Trénovací várky se skládají z 16 vstupních pozorování. Vzhledem k povaze úlohy volíme jako ztrátovou funkci kategorickou křížovou entropii. Pro ukončení trénování modelu používáme metodu early stopping, která učení přeruší, pokud se v 10 po sobě jdoucích epochách nesníží hodnota empirické rizikové funkce pro validační množinu. Toto nastavení hyperparametrů vychází z volby hyperparametrů pro trénování CVN z experimentu NOvA. Maximální počet trénovacích epoch je poté vzhledem k výpočetní náročnosti omezen na 100. Ve všech skrytých vrstvách používáme ReLU aktivační funkci. Jako aktivační funkce pro výstupní vrstvu je poté vzhledem ke klasifikační povaze úlohy použita funkce softmax. Jejím výstupem je poté vektor, jehož složky odpovídají pravděpodobnosti příslušnosti k jednotlivým třídám.

5.2.1 Ilustrativní CNN

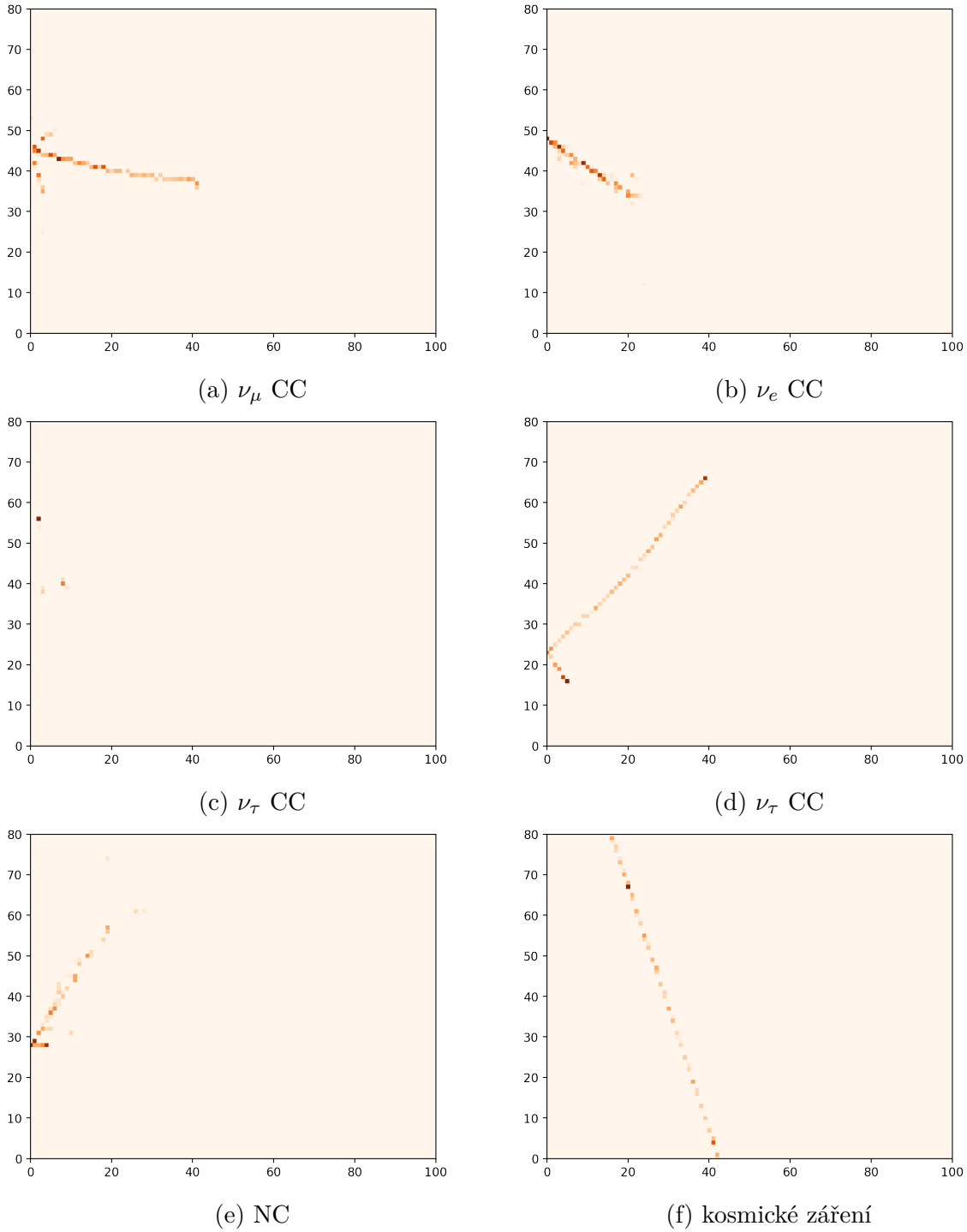
Jako první model konstruujeme jednoduchou ilustrativní CNN se dvěma skrytými konvolučními vrstvami s 32, respektive 64 konvolučními jádry a plně propojenou vrstvou o 128 neuronech. Tuto volbu hyperparametrů jsme zvolili s cílem získat výpočetně jednoduchý model pro následné porovnání s pokročilejšími architekturami CNN. Model nejprve zpracovává oba vstupy paralelně a poté skládá vzniklé příznakové mapy do druhé konvoluční vrstvy. Celkový počet volných parametrů pro trénování je cca 1 milion. Architekturu modelu CNN ilustrujeme na obrázku 5.4.

5.2.2 CVN

Jako referenční model používáme CVN. Ten využívá techniku inception modulů ve verzi popsané na obrázku 3.8. Model zpracovává vstupní snímky skrze dvě paralelně vedené větve tvořené třemi konvolučními vrstvami a jedním inception modulem. Následně jsou příznakové mapy spojeny do jedné velké série a zpracovány dalším inception modulem. Po každé konvoluční vrstvě CVN též následuje max pooling s jádrem o rozměrech 2×2. Výstupy posledního inception modulu jsou poté přímo převedeny na výstupní vrstvu o 5 neuronech se softmax funkcí. Celkově má tento model 1 milion parametrů pro trénování, tedy cca stejně jako předchozí model CNN. Přesné technické detaily CVN jsou k dispozici [2].

5.2.3 ResNet

Model CVN budeme dále porovnávat s vlastním modelem ResNet využívajícím residuální učení. Jak ilustrujeme na obrázku 5.5a, model ResNet oba vstupy nejdříve zpracovává skrze dvě paralelně vedené větve. Po sérii dvou residuálních bloků tvořených konvolučními



(a) ν_μ CC

(b) ν_e CC

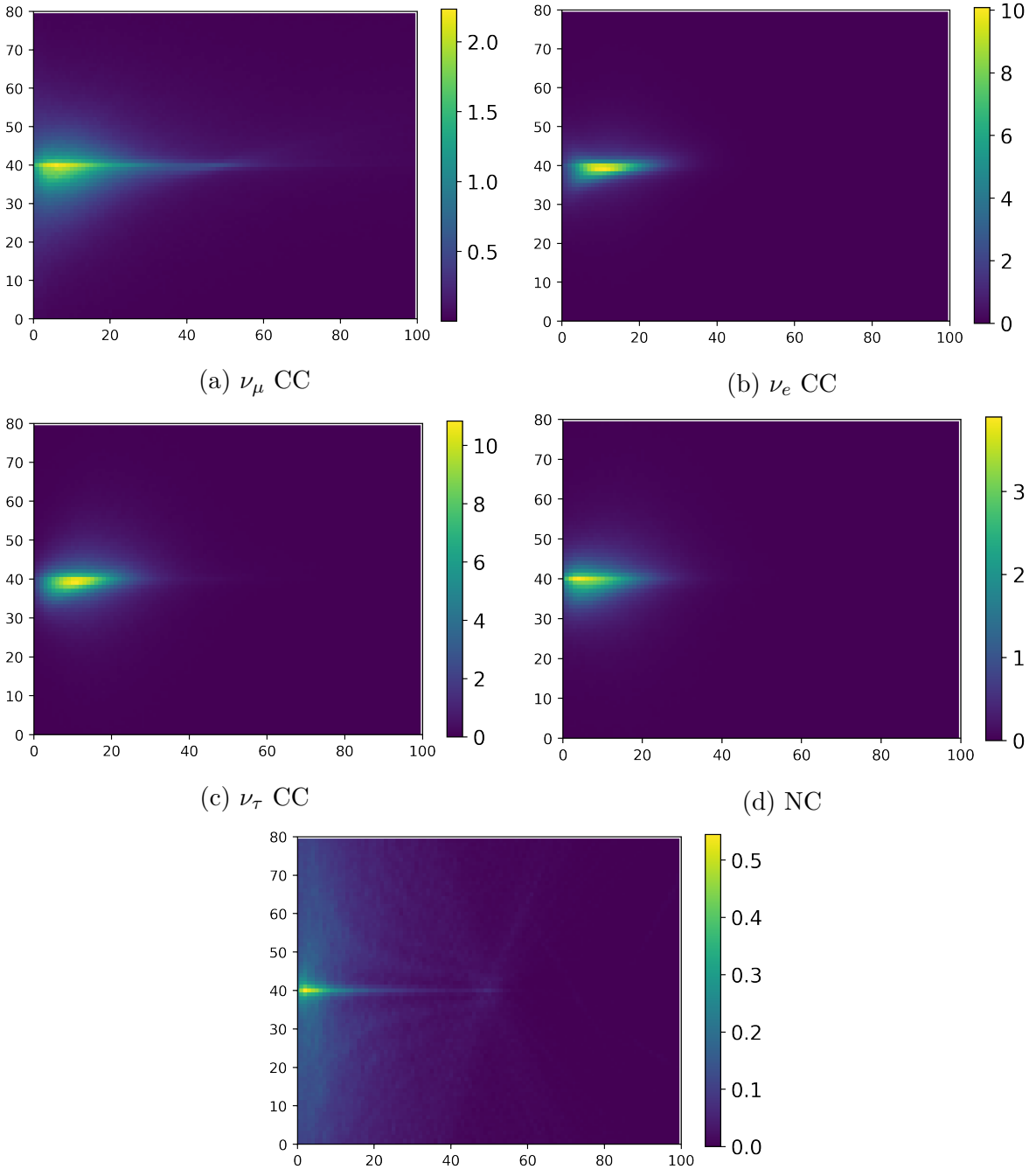
(c) ν_τ CC

(d) ν_τ CC

(e) NC

(f) kosmické záření

Obrázek 5.2: Ukázka obrazových dat pro jednotlivé třídy z X-view. Horizontální osa odpovídá hloubce detektoru, zatímco vertikální výšce detektoru. Intenzita pixelů odpovídá energii předané v dané buňce normované na hodnoty od 0 do 255. Na obrázcích 5.2c a 5.2d ilustrujeme problematiku nejednoznačnosti topologií drah při ν_τ CC interakci, kde při rozpadu mohou vznikat různé částice.



(a) ν_μ CC

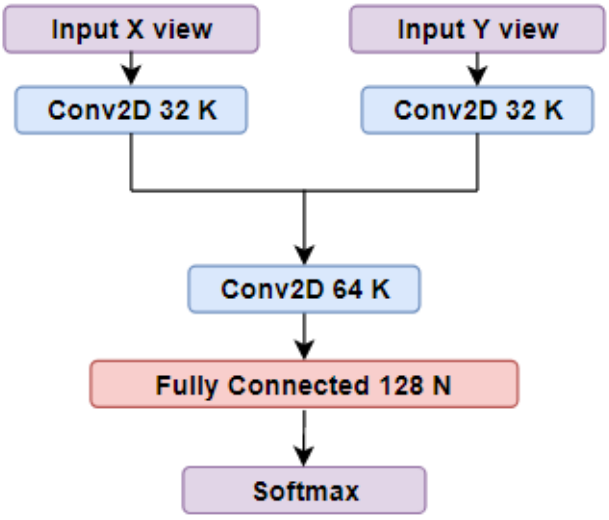
(b) ν_e CC

(c) ν_τ CC

(d) NC

(e) kosmické záření

Obrázek 5.3: Průměry snímků počítané napříč trénovací množinou pro jednotlivé třídy z X-view. Nejvyšší hodnoty pohybující se v levé části uprostřed obrázků odpovídají centrování snímků. Z obrázku je patrné, že ν_μ CC a kosmické částice ztrácejí energii pomaleji a projdou hlouběji do detektoru. Zároveň lze pozorovat, že částice z kosmického záření typicky do detektoru vlétají pod odlišnými úhly než částice při zbylých interakcích. To se na tomto obrázku projevuje nižším kontrastem hodnot uprostřed a na krajích ve vertikálním směru snímku.



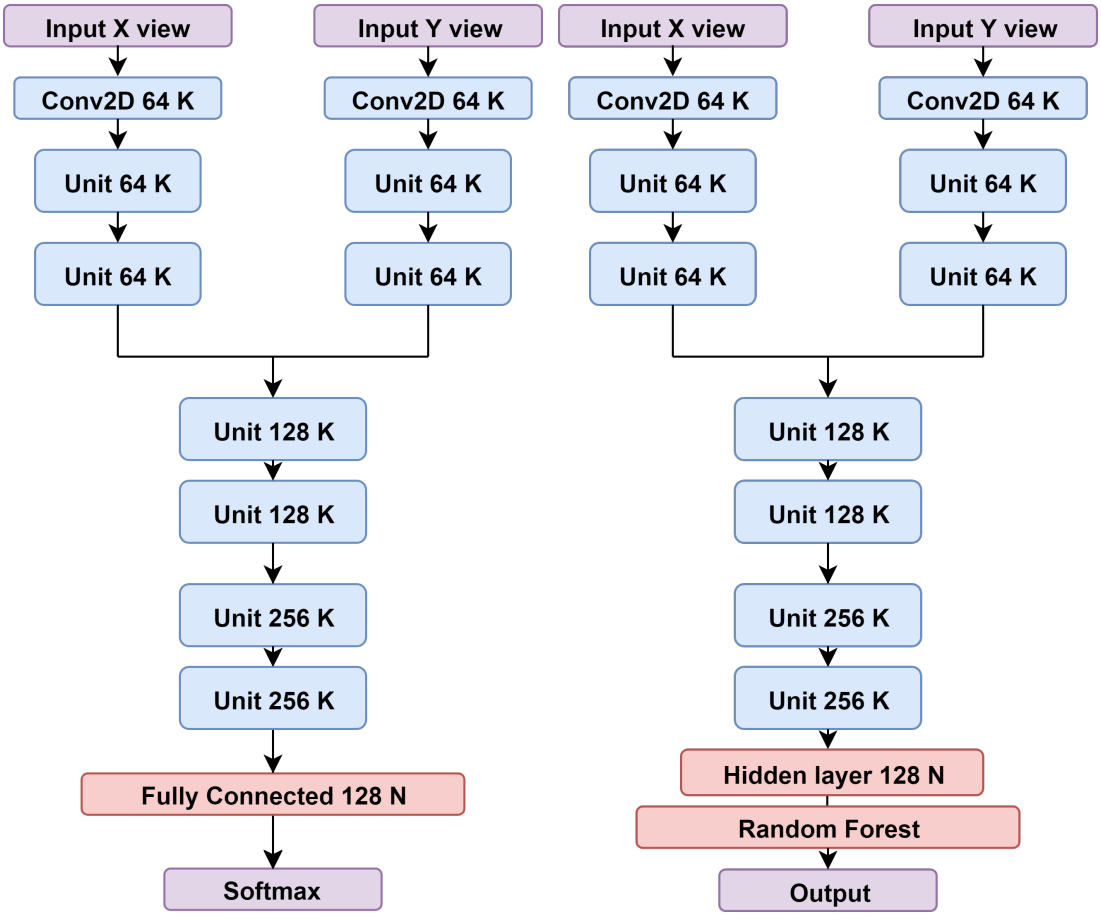
Obrázek 5.4: Schéma modelu CNN. Dvojice vstupů pro X-view (Input X view) a Y-view (Input Y view) je zpracována paralelními konvolučními vrstvami o 32 konvolučních jádrech. Po spojení dvou paralelních větví je použita další konvoluční vrstva o 64 kernelech a plně propojená vrstva (Fully Connected) se 128 neurony.

vrstvami se 64 kernely skládáme příznakové mapy z obou větví do jedné série příznakových map, které dále prochází residuálními bloky s konvolučními vrstvami o 128, respektive 256 konvolučními jádry. Poslední série příznakových map je poté skrze plně propojenou vrstvu o 128 neuronech spojena s výstupní vrstvou tvořenou 5 neurony. V rámci plně propojené vrstvy zároveň používáme techniku dropout s parametrem $p = 0.4$. Celkově má model ResNet 4.6 milionu parametrů k natrénování. Proto očekáváme větší kapacitu modelu ve srovnání s CVN.

5.2.4 ML metody pro klasifikaci příznaků z ResNet

Model ResNet dále modifikujeme s využitím ML klasifikátorů popsaných v sekci 4. Pro trénování ML metod používáme jako vstupy příznakové mapy z natrénovaného modelu ResNet. Vzhledem k výpočetní náročnosti převádíme extrahované příznaky skrze jednu skrytou vrstvu o 128 neuronech. Tím zredukujeme počet vstupů do ML algoritmů na 128 z původních cca 3000. Oproti klasické plně propojené vrstvě nicméně již na výstup skryté vrstvy neaplikujeme aktivační funkci, význam této vrstvy spočívá čistě v redukci dimenze vstupu do klasifikátoru. Schematicky takto vzniklý model pro případ náhodného lesa ilustrujeme na obrázku 5.5b.

Pro ladění hyperparametrů ML algoritmů používáme metodu randomized search. Zároveň provádíme křížovou validaci pro 10 podmnožin vybíraných z trénovací a validační množiny. Klíčovou metrikou pro volbu hyperparametrů modelu je pro nás vážené F1 skóre. Celkově provádíme hledání napříč 10 náhodně inicializovanými sadami hyperparametrů. Toto omezení opět plyne z vysoké výpočetní náročnosti modelů spojené s použitím křížové validace.



(a) ResNet (b) ResNet + náhodný les

Obrázek 5.5: Porovnání architektur modelů ResNet (vlevo) a hybridního modelu ResNet (vpravo) využívajícího jako klasifikátor náhodný les (Random Forest). Buňkami Unit myslíme residuální bloky dle schématu na obrázku 3.9.

5.2.4.1 ResNet & náhodný les

Při učení náhodného lesa jsme pro počet rozhodovacích stromů vyzkoušeli hodnoty z množiny $\{20, 40, 60, 100, 200, 300, 500\}$. Pro maximální hloubku stromu byla použita mřížka $\{4, 6, \ldots, 20\}$. Nejlepších výsledků jsme dosáhli s náhodným lesem o 300 stromech s maximální hloubkou 14. Při použití větší hloubky se model začal stávat náchylnějším k přetrénování. Při malém počtu stromů jsme naopak obdrželi vysokou disproporci mezi výsledky na validační a testovací množině.

5.2.4.2 ResNet & AdaBoost

Dalším použitým klasifikátorem příznaků je AdaBoost. Ten je výpočetně náročnější než klasický náhodný les, a proto byl počet rozhodovacích lesů v algoritmů omezen na mřížku $\{20, 40, 60, 100, 200, 300, 400\}$ slabých klasifikátorů s hloubkou z mřížky $\{1, \ldots, 10\}$. Nejlepší model se skládá ze 300 rozhodovacích stromů o hloubce 8.

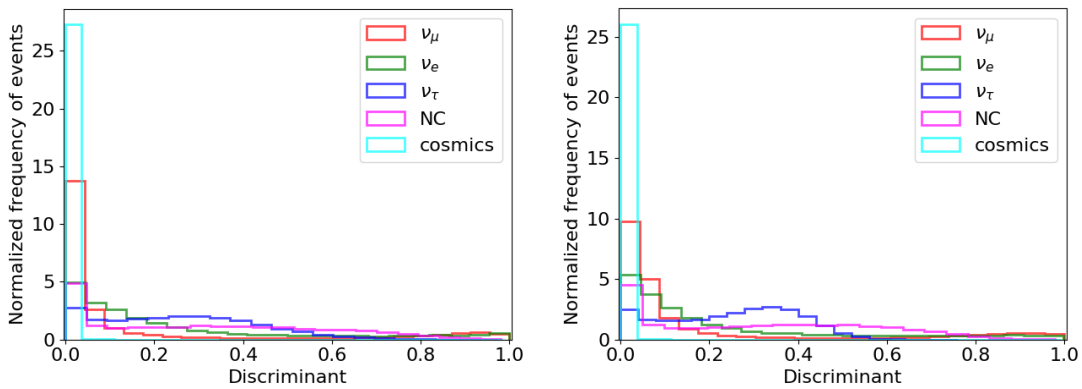
5.2.4.3 ResNet & Gradient Boosting

V případě metody Gradient Boosting jsme hledali počet slabých klasifikátorů na mřížce $\{20, 40, 60, 100, 200\}$. Snížení horní hranice pro počet stromů oproti předchozím modelům bylo voleno vzhledem k vyšší výpočetní náročnosti tohoto typu modelu. Hloubku klasifikátorů jsme omezili na mřížku $\{2, 4, \ldots, 16\}$. Oproti předchozím klasifikátorům se Gradient Boosting ukázal být nejnáchylnější k přetrénování. Výsledný model se skládá ze 150 stromů o hloubce 10.

5.3 Výsledky

5.3.1 Modely CNN

Výsledky modelů porovnáváme na základě matic záměn vykreslených na obrázku 5.7. Z obrázku pozorujeme, že všechny modely měly problémy s přesností klasifikace pro třídu ν_τ. Jak jsme naznačili již na obrázku 5.2, tento jev může být způsoben širokou škálou rozpadů ν_τ, a tedy i nejednoznačností trajektorií. Ve skutečnosti je nicméně pozorování ν_τ CC interakcí extrémně vzácné a oproti třídám ν_μ CC a ν_e CC pro nás tak jeho špatná klasifikace není příliš podstatná. Podíváme-li se podrobněji na obrázek 5.6, kde zobrazujeme histogramy pro výstupní skóre klasifikátorů při predikci na snímcích třídy ν_τ CC, pozorujeme, že se nepodařilo spolehlivě separovat pozorování třídy ν_τ CC od tříd ν_e CC a NC.

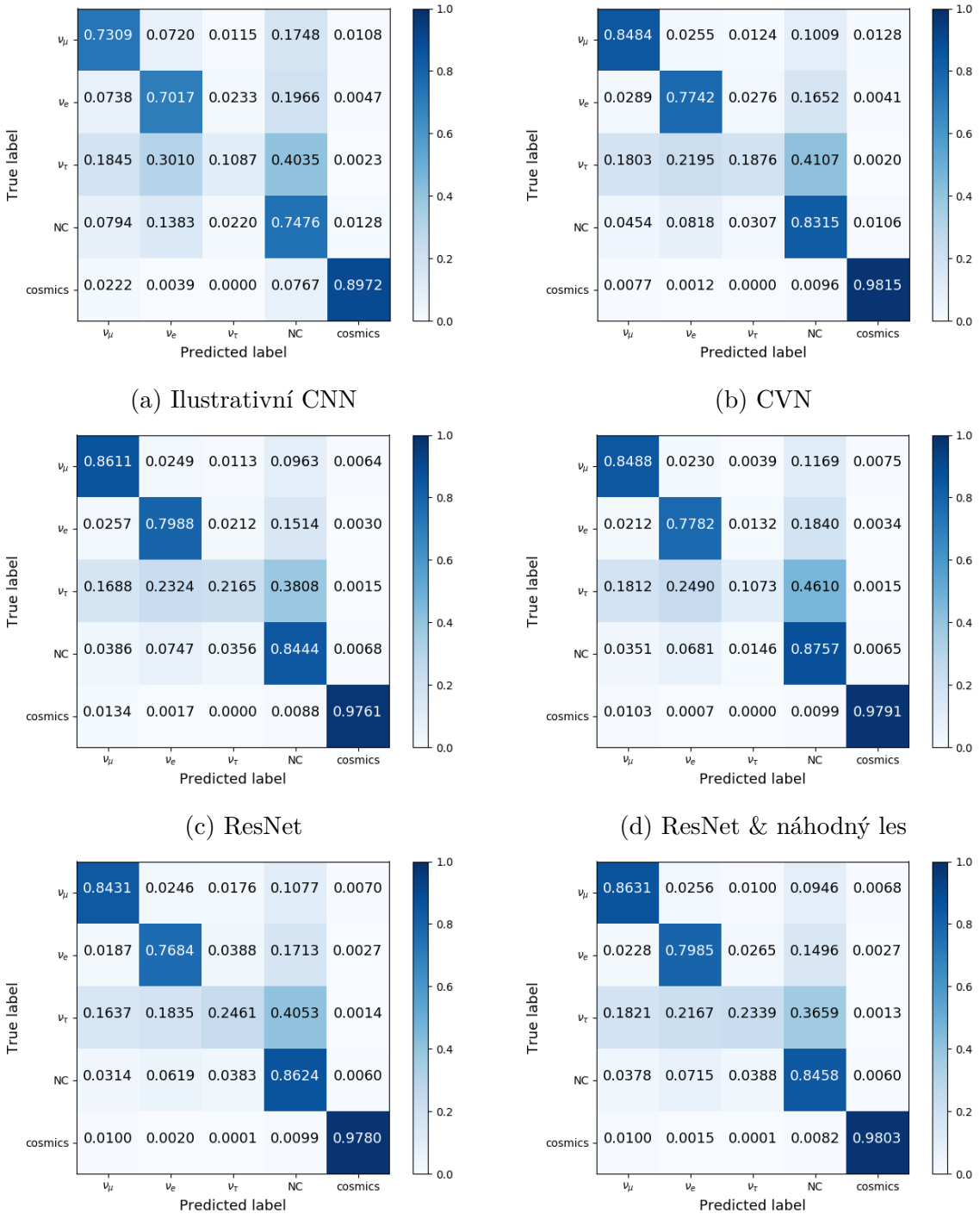


(a) ResNet (b) CVN

Obrázek 5.6: Histogramy výstupních hodnot pro pozorování s indikací třídy ν_τ.

Naopak všechny klasifikátory dokážou velmi dobře odhalit částice z kosmického záření. Jak jsme již ukázali na obrázku 5.3, částice z kosmického záření typicky přicházejí z jiných úhlů než vede svazek, a proto jsou pro klasifikátor dobře rozpoznatelné. Ačkoliv model ilustrativní CNN obsahuje stejný počet parametrů jako CVN, dosahuje výrazně horších výsledků, co se přesnosti klasifikace týče. Navzdory stejnému počtu parametrů tedy dokáže model CVN díky úspornějším inception modulů dosáhnout lepších výsledků, nežli obyčejný model CNN. Porovnáme-li mezi sebou pokročilejší klasifikátory, z matic

záměn na obrázku 5.7 vidíme, že model ResNet dokázal mírně zlepšit přesnost klasifikace ν_μ CC, ν_e CC a ν_τ CC a NC. Nevýhodou ResNet ovšem je, že pro dosažení těchto výsledků vyžadoval cca čtyřikrát více parametrů než CVN.



(a) Ilustrativní CNN

(b) CVN

(c) ResNet

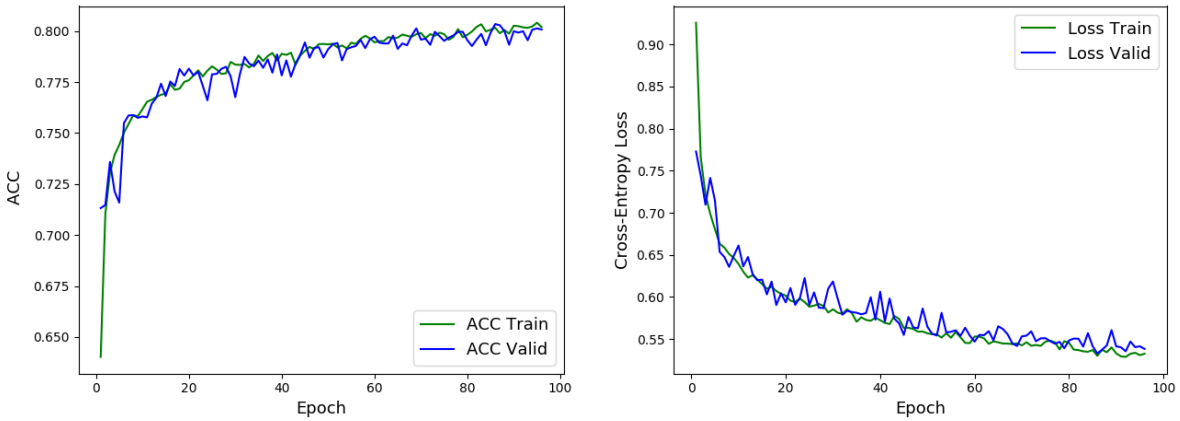
(d) ResNet & náhodný les

(e) ResNet & AdaBoost

(f) ResNet & Gradient Boosting

Obrázek 5.7: Matice záměn pro klasifikaci na testovacích datech.

Na obrázku 5.8 zobrazujeme průběh minimalizace empirické rizikové funkce společně s růstem ACC pro model ResNet. Malé rozdíly mezi empirickou rizikovou funkcí pro trénovací a validační data ukazují, že nedochází k přetrénování modelu.



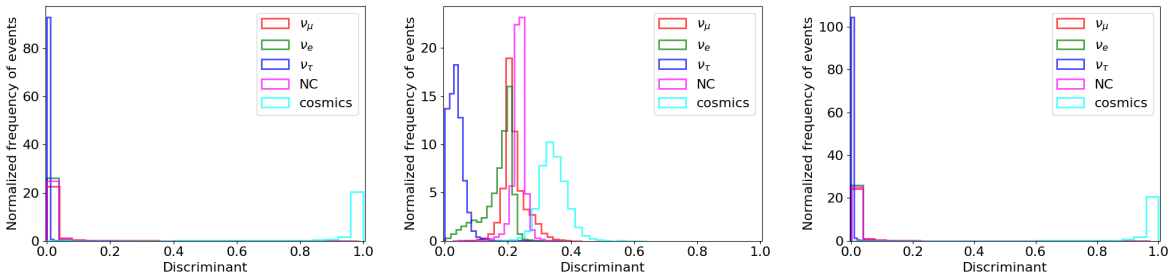
(a) ACC (b) Empirická riziková funkce

Obrázek 5.8: Ilustrace průběhu trénování pro model ResNet v rámci trénovacích cyklů (epoch). Průběh pro trénovací a validační data je na obou obrázcích velice podobný, nedochází tedy k overfittingu.

5.3.2 Aplikace ML klasifikátorů

Jak dále ukazujeme na výsledcích na obrázku 5.7, vliv použití ML klasifikátorů na přesnost modelu se pro jednotlivé třídy liší. Co se týče klíčových tříd ν_μ a ν_e, aplikace náhodného lesa či metody AdaBoost jako klasifikátoru vedla ke zhoršení klasifikace. Přesnost klasifikace na třídě ν_μ se za pomocí klasifikátoru využívajícího Gradient Boosting podařilo vylepšit pouze o 0.2 %. Nebylo tedy dosaženo významného zpřesnění klasifikace. Pro třídu ν_τ nevedlo použití ML klasifikátorů k výraznému zlepšení. Co se nicméně týče třídy NC, všechny ML klasifikátory vedly ke zpřesnění klasifikace, v případě náhodného lesa poté dokonce velmi výrazně o 3.13 %. Pro třídu kosmického záření aplikace ML klasifikátorů na příznaky extrahované z ResNet sice vedla ke zlepšení výsledků oproti ResNet, přesnost na této třídě je nicméně stále nižší než pro CVN. S ohledem na důležitost jednotlivých tříd jsme tak dospěli pouze k velice malému zlepšení přesnosti klasifikace, a to za pomocí klasifikátoru s technikou Gradient Boosting. Pokud by pro nás byla významnější třída NC, mohlo by nicméně být výhodné využít klasifikátor s náhodným lesem.

Co se analýzy výstupů modelů týče, od zbylých modelů se výrazně liší rozdělení výstupů z modelu AdaBoost. Jak ilustrujeme na obrázku 5.9, těžiště rozdělení pro predikované pravděpodobnosti se totiž pozorovatelně blíží výstupu náhodného klasifikátoru. Oproti tomu rozdělení predikcí pravděpodobností příslušnosti ke správné třídě pro zbylé modely se vyjma špatně klasifikovatelné třídy ν_τ soustředí kolem 1. Vzhledem k volbě příslušnosti ke třídě jako maximální predikované pravděpodobnosti tento jev nicméně pozorovatelně neovlivňuje klasifikační metriky modelu AdaBoost. Tento jev se umocňoval na modelech s nízkou hloubkou klasifikátoru. Při volbě větší hloubky modelu byl



(a) ResNet (b) ResNet & AdaBoost (c) ResNet & Grad. Boosting

Obrázek 5.9: Histogramy výstupních hodnot predikovaných pro pozorování třídy kosmického záření. Přestože model AdaBoost na základě histogramu nepredikoval výrazně vyšší odhady pravděpodobnosti pro třídu kosmického záření, výsledné F1 skóre či *ACC* jsou srovnatelné se zbylými modely.

nicméně tento klasifikátor velmi náchylný k přetrénování. Kompletní výsledky klasifikace pro všechny testované modely jsou k dispozici v příloze 5.4.

5.4 Analýza příznaků a t-SNE

Aplikace ML klasifikátorů na příznaky extrahované poslední konvoluční vrstvou může dávat smysluplné výsledky pouze pokud s extrahovanými příznaky lze separovat jednotlivé třídy. Z tohoto důvodu v této sekci ilustrujeme separační potenciál vybraných příznaků pomocí metody t-SNE [26]. Ukázku příznaků z první konvoluční vrstvy modelu ResNet vykreslujeme na obrázku 5.10. Tyto obrázky odpovídají příznakům, na jejichž základě model klasifikuje do jednotlivých tříd.

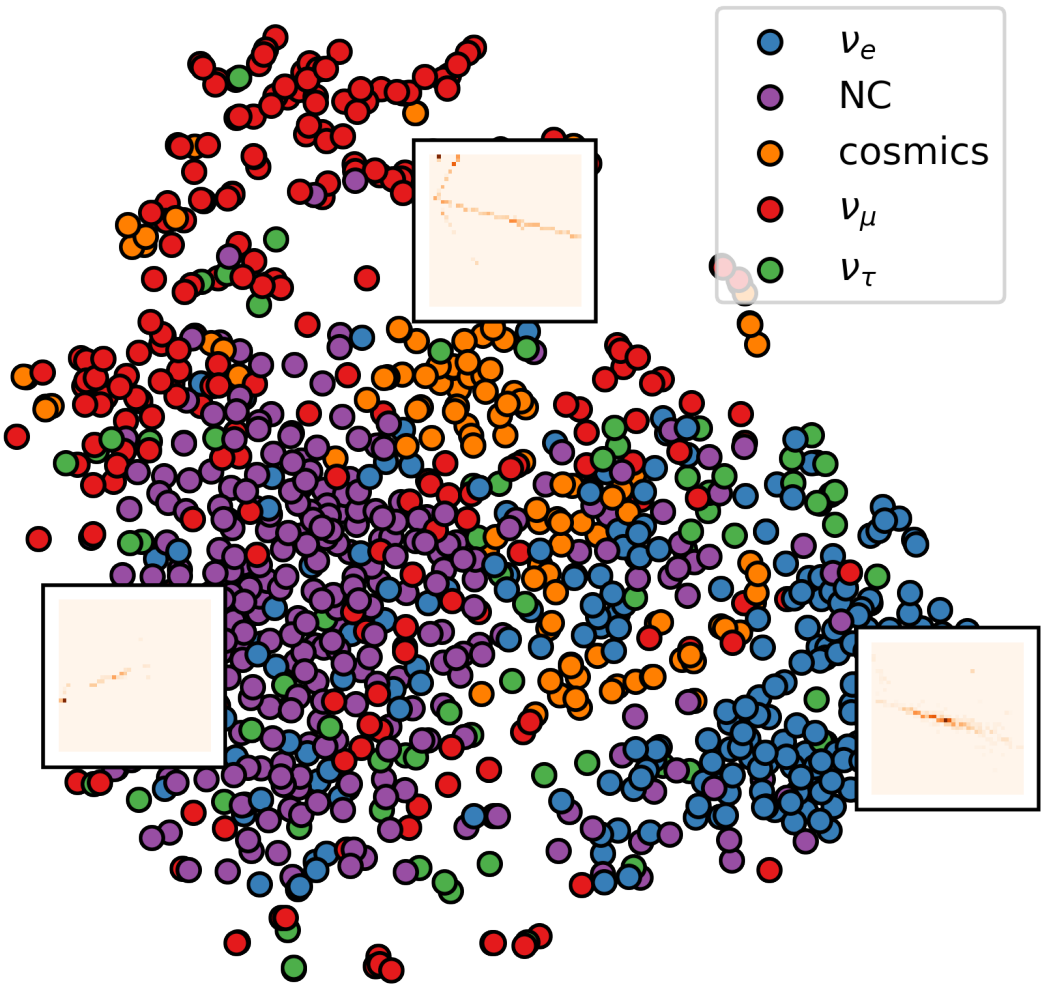
(a) (b)

Obrázek 5.10: Ukázka dvou vybraných příznaků z první konvoluční vrstvy modelu ResNet.

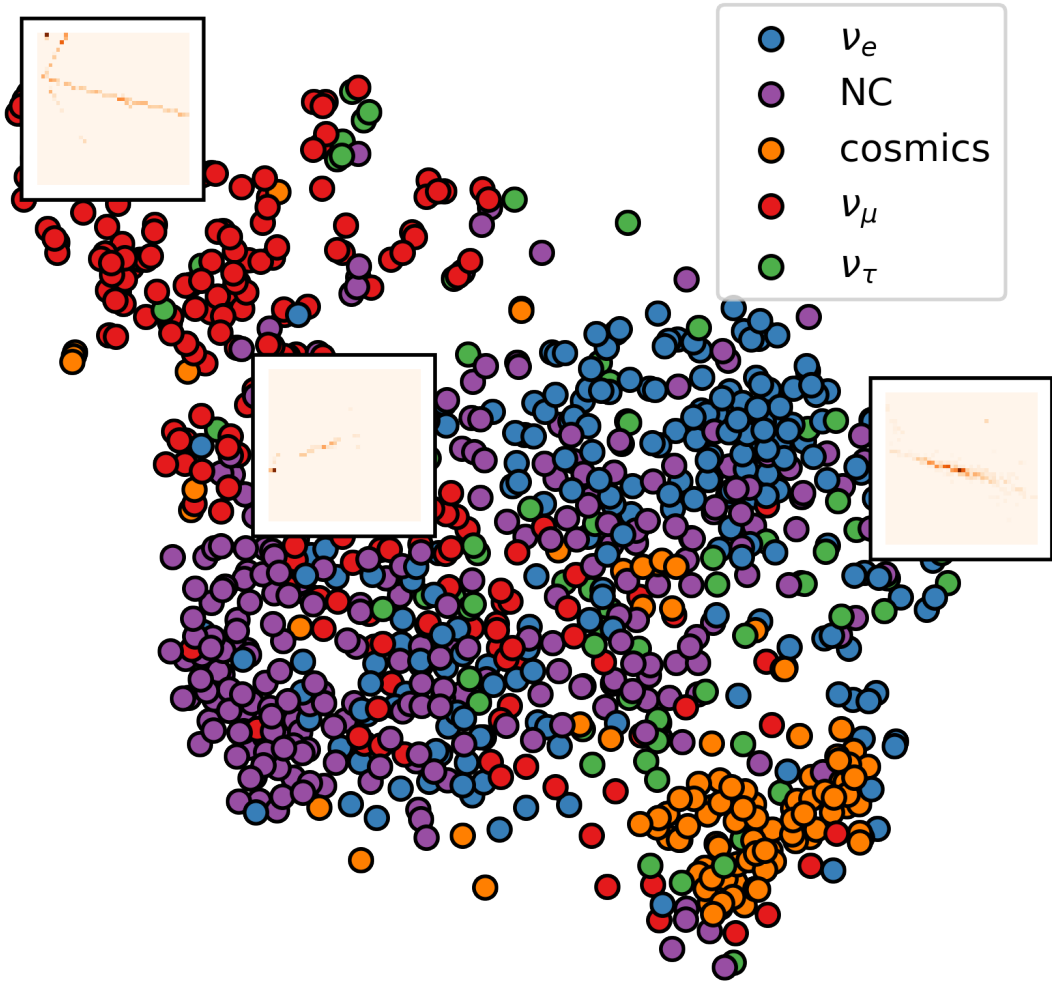
Rozhodování se sítě na základě vizuálních příznaků lze vizualizovat za pomocí metody t-SNE. Ta spočívá v projekci příznakových map ze CNN do prostoru o nízké dimenzi, tedy například do 2D prostoru. Příklad vykreslení t-SNE ukazujeme na obrázcích 5.11

a 5.12. Každý bod na obrázku odpovídá jednomu pozorování, celkově však pro přehlednost vykreslujeme pouze 1000 náhodně vybraných bodů. Barva poté indikuje příslušnost ke třídě. Dobrá separace dat v příslušné t-SNE projekci poté odpovídá dobře odděleným shlukům na obrázku. Příznak z obrázku 5.11 tak patrně dobře separuje třídy ν_μ a ν_e, ale není příliš úspěšný pro třídy ostatní. Příznak z obrázku 5.12 zase zjevně výborně separuje třídy ν_μ a kosmického záření. Zároveň však poměrně dobře separuje i třídy ν_e a NC. Toto pozorování tedy ukazuje na schopnost CNN extrahovat příznaky umožňující separaci dat do příslušných tříd. Na závěr na obrázku 5.13 ilustrujeme též t-SNE projekci pro příznaky z plně propojené vrstvy modelu ResNet. Na této projekci je nejlépe vidět separovatelnost pozorování tříd ν_μ a kosmického záření.

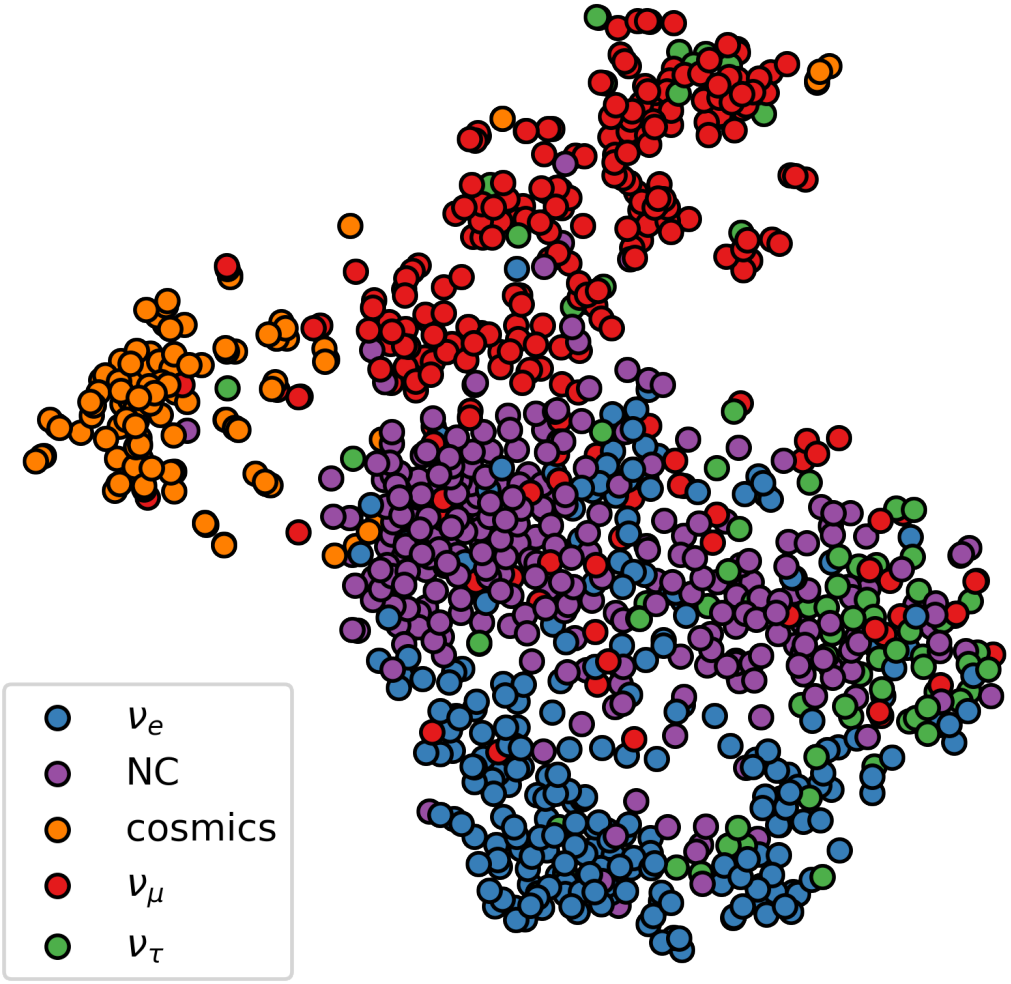
Celkově lze z poslední konvoluční vrstvy pro 256 konvolučních filtrů vykreslit 256 t-SNE projekcí. Při detailním pohledu bychom zjistili, že žádná z 256 t-SNE projekcí nenabízí dobrou separaci ν_τ. To odpovídá skutečnosti, že žádný z extrahovaných příznaků poslední konvoluční vrstvy ResNet není vhodný pro účinnou separaci třídy ν_τ.



Obrázek 5.11: Ukázka t-SNE projekcí příznaků.



Obrázek 5.12: Ukázka t-SNE projekcí příznaků.



Obrázek 5.13: Ukázka t-SNE projekcí příznaků z plně propojené vrstvy modelu ResNet.

Závěr

V této práci jsme demonstrovali potenciál ML metod pro klasifikaci obrazových dat z HEP. V kapitole 1 jsme popsali podstatu neutrinového experimentu NOvA a představili význam úlohy klasifikace neutrinových interakcí včetně analogií s úlohou rozpoznávání obrazu. Následně jsme v kapitole 2 navázali na práci [16] a rozšířili jsme koncepty binární statistické klasifikace na úlohu klasifikace do více tříd s odkazem na klasifikační úlohu pro experiment NOvA. Dále jsme v kapitole 3 navázali na poznatky o CNN z práce [16] a představili moderní CNN architektury. V kapitole 4 jsme potom představili další metody ML založené na rozhodovacích stromech s cílem jejich zakomponování do komplexních CNN architektur pro klasifikaci neutrinových interakcí.

V závěrečné kapitole 5 jsme využili nabyté teoretické znalosti z předchozích kapitol k úloze klasifikace neutrinových interakcí do tříd ν_μ, ν_e, ν_τ, NC a kosmického záření. K tomu jsme použili množinu simulovaných obrazových dat ze vzdáleného detektoru experimentu NOvA. Úspěšně jsme implementovali tři CNN modely s odlišnou architekturou. Pro první, ilustrativní implementaci, jsme sestrojili konvenční CNN model se dvěma konvolučními vrstvami. Dále jsme rekonstruovali na experimentu NOvA v současnosti používaný model CVN využívající tzv. inception moduly. Na závěr jsme sestrojili vlastní model ResNet využívající techniku tzv. residuálního učení. Na výsledné přesnosti klasifikace jsme ukázali na výrazně lepší výsledky pro modely CVN a ResNet oproti konvenčnímu modelu CNN. Na klíčových třídách ν_μ a ν_e dosáhl model ResNet ve srovnání s CVN zlepšení ACC z 84.84 % na 86.11 %, resp. z 77.42 % na 79.88 %. K tomuto zlepšení jsme nicméně potřebovali zhruba čtyřnásobný počet parametrů modelu. Zároveň jsme u všech modelů shodně pozorovali velice špatnou přesnost klasifikace na třídě ν_τ. Ukazuje se tedy, že CNN nejsou vhodné pro extrahování příznaků z rozpadových interakcí ν_τ. Třída ν_τ je nicméně ve skutečnosti zastoupena pouze stopovým množstvím pozorování a není přímým předmětem zájmu výzkumu na experimentu NOvA. Proto špatná klasifikace této třídy v praxi nijak nesnižuje použitelnost CNN pro klasifikaci neutrinových interakcí.

Dále jsme do architektury ResNet implementovali modely založené na rozhodovacích stromech. V tomto případě jsme použili konvoluční vrstvy ResNet k extrakci příznaků z obrazových dat a následně provedli jejich klasifikaci právě skrze modely založené na rozhodovacích stromech. Konkrétně jsme k tomuto účelu sestrojili hybridní modely kombinující ResNet a ML klasifikátory jako náhodný les, AdaBoost a Gradient Boosting. U takto konstruovaných modelů jsme nicméně nepozorovali významné zlepšení ani zhoršení přesnosti klasifikace. Vzhledem ke komplikovanosti konstrukce a trénování hybridních modelů se tedy zdá výhodnější použití CNN modelů s plně propojenou vrstvou.

Literatura

[1] Aitchison, I. - Hey, A. *Gauge Theories in Particle Physics: A Practical Introduction*, vol. 2., Institute of Physics, 2003. ISBN 0-7503-0950.

[2] Aurisano, A. et al. A Convolutional Neural Network Neutrino Event Classifier. *Journal of Instrumentation* 11 P09001, 2016.

[3] Backhouse, C. - Patterson, R. B. Library Event Matching event classification algorithm for electron neutrino interactions in the NOvA detectors. *Nuclear Inst. and Methods in Physics Research* , 2015.

[4] Backhouse, C. - Radovic, A. *The Attenuation and Threhold Correction of the NOvA detectors*. NOvA technote, 2017.

[5] Baird, M. et al. Event Reconstruction Techniques in NOvA. *Journal of Physics: Conference Series*, Vol. 664, 2015.

[6] Bishop, C. M. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[7] Bouř, P. *Vývoj statistických neparametrických a divergenčních metod pro zpracování dat z experimentu D0 a NOvA*. Diplomová práce, ČVUT v Praze, FJFI, 2016.

[8] Cutler, A. - Cutler, D. - Stevens, J. Random Forests. *Machine Learning - ML*, vol. 45, 2011. DOI 10.1007/978-1-4419-9326-75.

[9] Fisher, R. A. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics* (2), 1936.

[10] Goodfellow, I. - Bengio, Y. - Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org

[11] Goyal, P. et al. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *CoRR Journal*, vol. 1706.02677, 2017.

[12] He, K. and Sun, J. Convolutional neural networks at constrained time cost. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[13] He, K. - Zhang, X. - Ren, S. - Sun, J. Deep Residual Learning for Image Recognition *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, ISBN: 978-1-4673-8851-1.

[14] Hochreiter, S. The Vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems* 6, 1998.

[15] Ioffe, S. - Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of 2nd International Conference on International Conference on Machine Learning* - Vol. 37, 2015.

[16] Kubů, M. *Konvoluční neuronové sítě v částicové fyzice*. Bakalářská práce, ČVUT v Praze, FJFI, 2018.

[17] Le, V. Q. et al. On optimization methods for deep learning. *Proceedings of the 28th International Conference on International Conference on Machine Learning ICML 2011*, 2011, ISBN: 978-1-4503-0619-5.

[18] Natekin, A. - Knoll, A. Gradient Boosting machines, a Tutorial. *Frontiers in Neurorobotics*, Vol. 7, 2013.

[19] Nievergelt, J. Exhaustive Search, Combinatorial Optimization and Enumeration: Exploring the Potential of Raw Computing Power. *Lecture Notes in Computer Science 1963:87-125*, 2020.

[20] Prechelt, L. Early Stopping-But When? *Neural Networks: Tricks of the Trade*, 1998.

[21] Psihas, F. -Radovic, A. Assorted CVN Plots For Blessing. [online - 29.03.2019], 2018. https://nusoft.fnal.gov/nova/blessedplots/#!/doc/15639.

[22] Qin, Z. - Yu, F. - Liu, Ch. - Chen, X. How Convolutional Neural Networks See the World - A Survey of Convolutional Neural Network Visualisation Methods. *Journal of CoRR*, Vol. 1804.11191, 2018. http://arxiv.org/abs/1804.11191.

[23] Srivastava, N. - Hinton, G. - Krizhevsky, A. - Sutskever, I. - Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 2014.

[24] Szegedy, Ch. et al. Going Deeper with Convolutions. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[25] Thongsuwan, S. et al. ConvXGB: A new deep learning model for classification problems based on CNN and XGBoost. *Journal of Nuclear Engineering and Technology*, 2020.

[26] Van Der Maaten, L. Viualizing data using t-SNE. *Journal of Machine Learning Research 9*, 2008.

[27] Westerdale, S. *Neutrino Mass Problem: Masses and Oscillations*. Technical Report MIT Department of Physics, 2010.

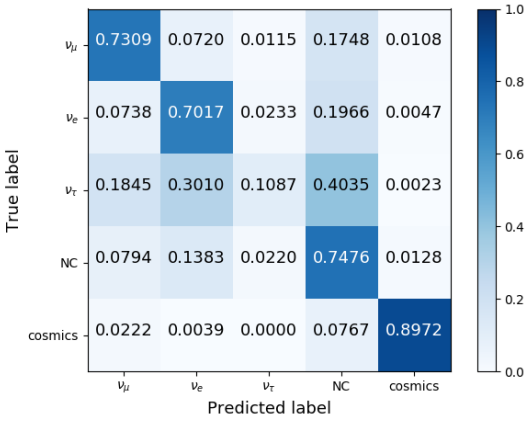
[28] Zhu J. et al. Multi-class AdaBoost. *Statistics and Its Interface*, Vol. 2, 2009.

Příloha - Výsledky klasifikace pro NOvA data

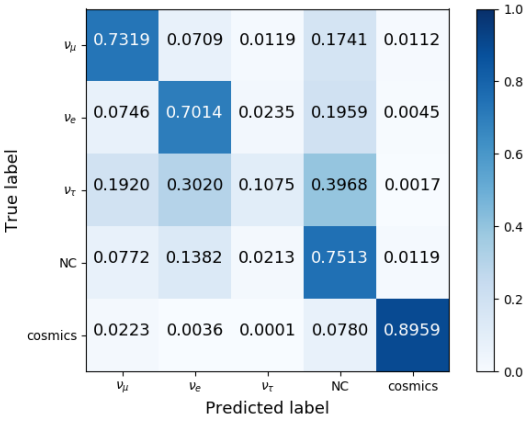
5.5 Ilustrativní CNN

	Test			Valid			Train		
Třída	prec.	recall	F1	prec.	recall	F1	prec.	recall	F1
ν_μ	0.766	0.731	0.748	0.765	0.732	0.748	0.767	0.734	0.750
ν_e	0.680	0.702	0.691	0.683	0.701	0.692	0.686	0.703	0.695
ν_τ	0.344	0.109	0.165	0.341	0.107	0.163	0.348	0.110	0.167
NC	0.629	0.748	0.683	0.630	0.751	0.685	0.630	0.752	0.686
Kosmika	0.916	0.897	0.907	0.919	0.896	0.907	0.920	0.899	0.909
Váž. průměr	0.685	0.698	0.685	0.686	0.699	0.686	0.688	0.701	0.688

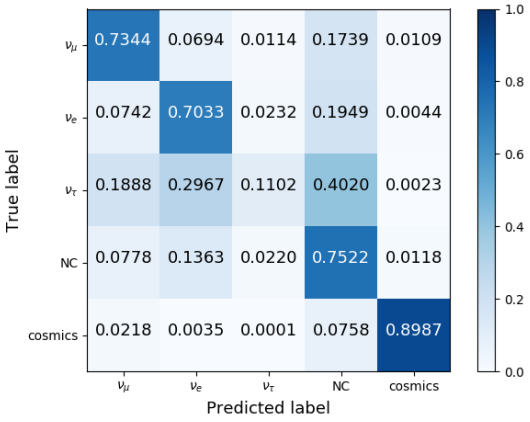
Tabulka 5.2: Výsledky pro model ilustrativní CNN na testovací, validační a trénovací množině.



(a) testovací data

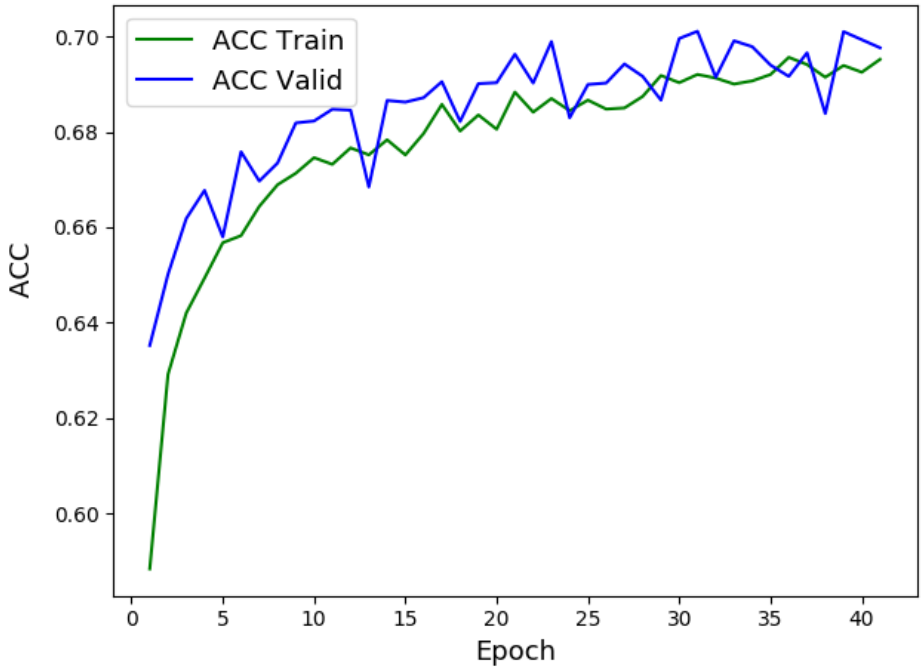


(b) validační data

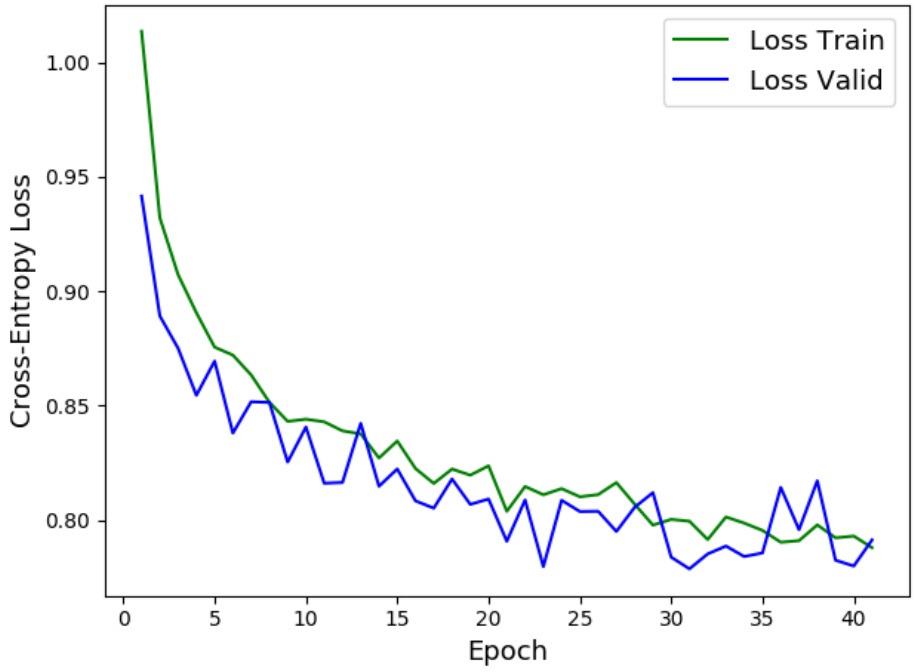


(c) trénovací data

Obrázek 5.14: Normalizovaná matice záměn pro model ilustrativní CNN. Diagonální hodnoty zvýrazněné barevnou škálou odpovídají *ACC* pro příslušnou třídu.

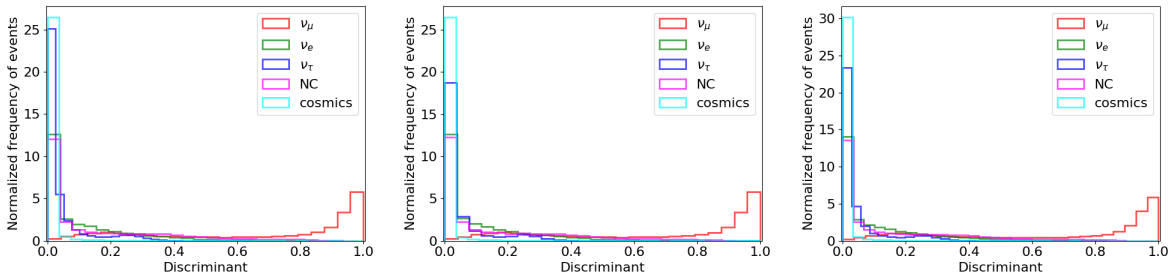


(a)

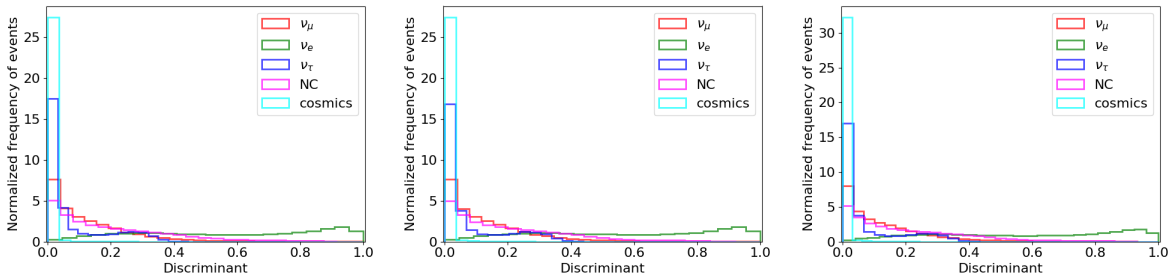


(b)

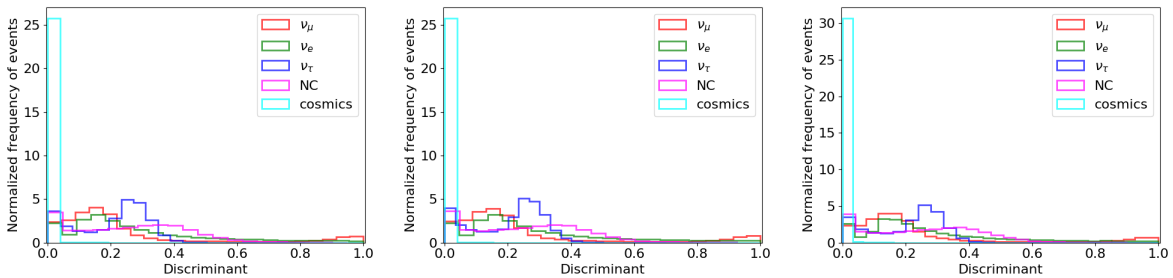
Obrázek 5.15: Ilustrace průběhu trénování pro ilustrativní CNN v rámci trénovacích cyklů (epoch).



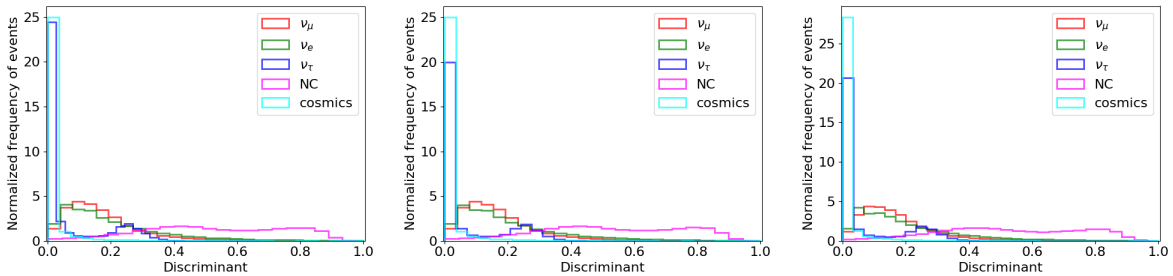
(a) test ν_μ (b) valid ν_μ (c) train ν_μ



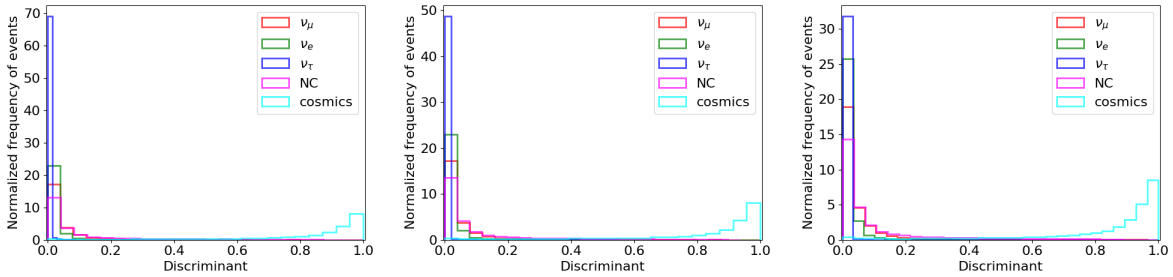
(d) test ν_e (e) valid ν_e (f) tran ν_e



(g) test ν_τ (h) valid ν_τ (i) train ν_τ



(j) test NC (k) valid NC (l) train NC



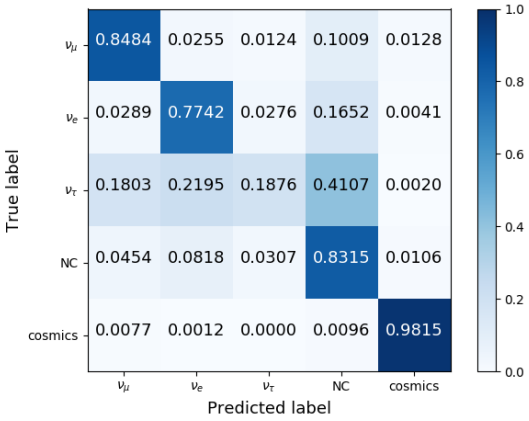
(m) test kosmika (n) valid kosmika (o) train kosmika

Obrázek 5.16: Control ploty pro ilustrativní CNN.

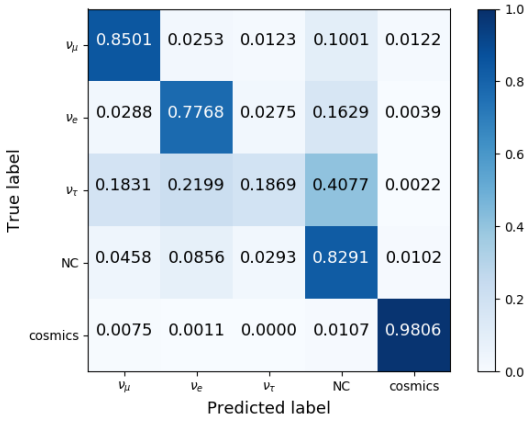
5.6 CVN

	Test			Valid			Train		
Třída	prec.	recall	F1	prec.	recall	F1	prec.	recall	F1
ν_μ	0.863	0.848	0.856	0.862	0.850	0.856	0.862	0.851	0.856
ν_e	0.803	0.774	0.788	0.802	0.777	0.789	0.805	0.777	0.791
ν_τ	0.419	0.188	0.259	0.423	0.187	0.259	0.428	0.191	0.264
NC	0.715	0.831	0.769	0.715	0.829	0.768	0.716	0.831	0.769
Kosmika	0.926	0.982	0.953	0.929	0.981	0.954	0.927	0.982	0.954
Váž. průměr	0.775	0.787	0.777	0.775	0.788	0.777	0.777	0.789	0.778

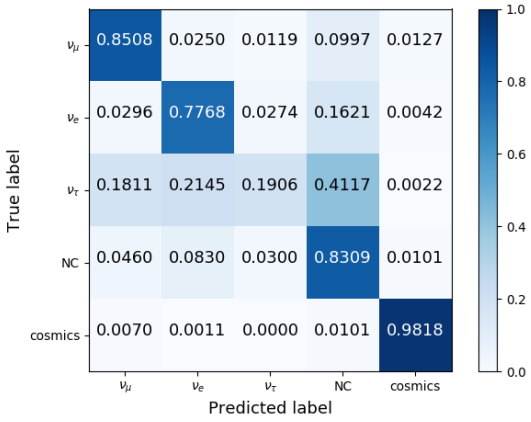
Tabulka 5.3: Výsledky pro model CVN na testovací, validační a trénovací množině.



(a) testovací data

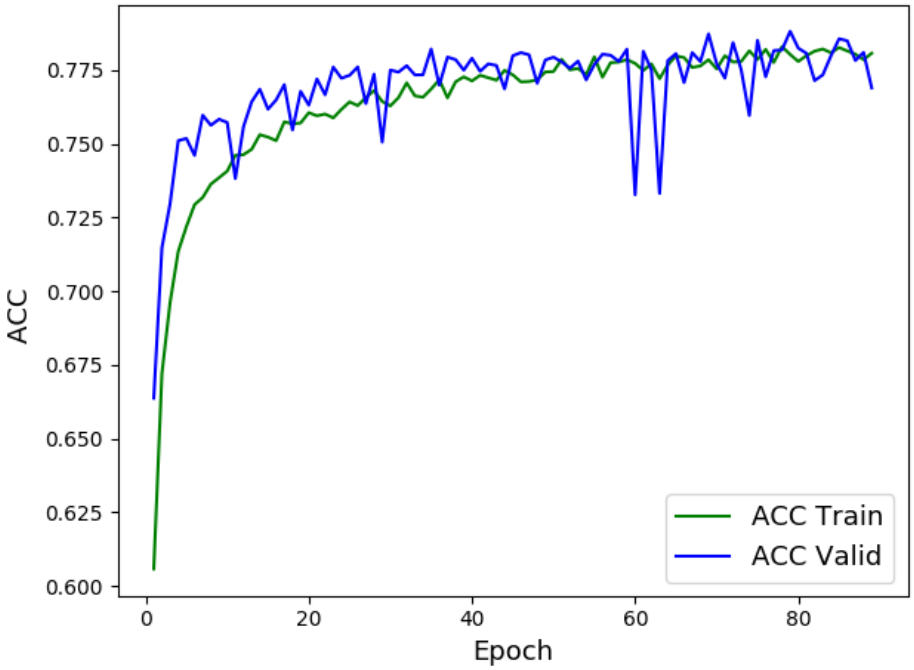


(b) validační data

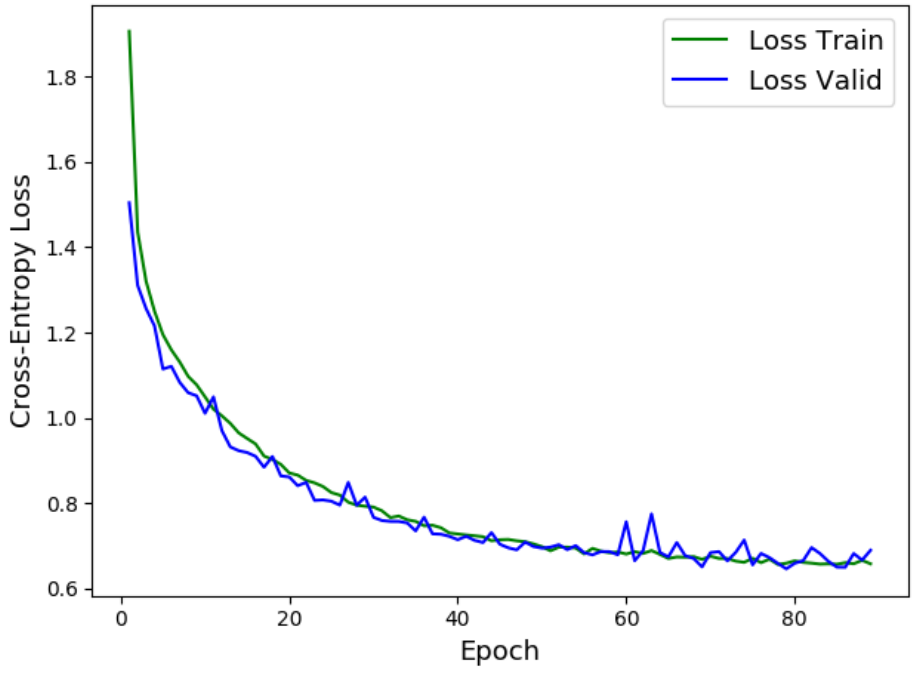


(c) trénovací data

Obrázek 5.17: Normalizovaná matice záměn pro model CVN. Diagonální hodnoty zvýrazněné barevnou škálou odpovídají *ACC* pro příslušnou třídu.

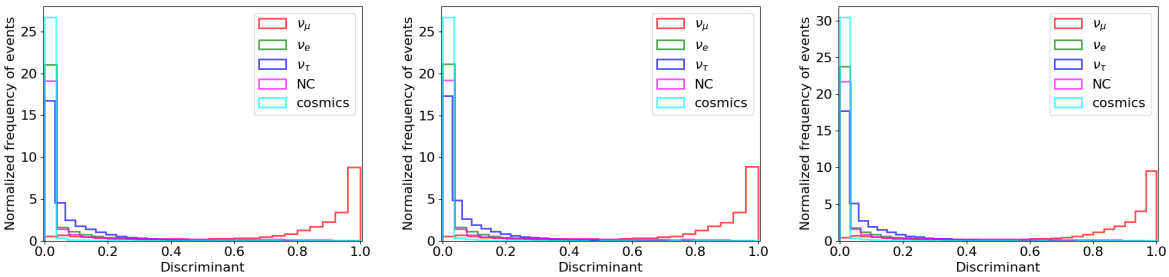


(a)

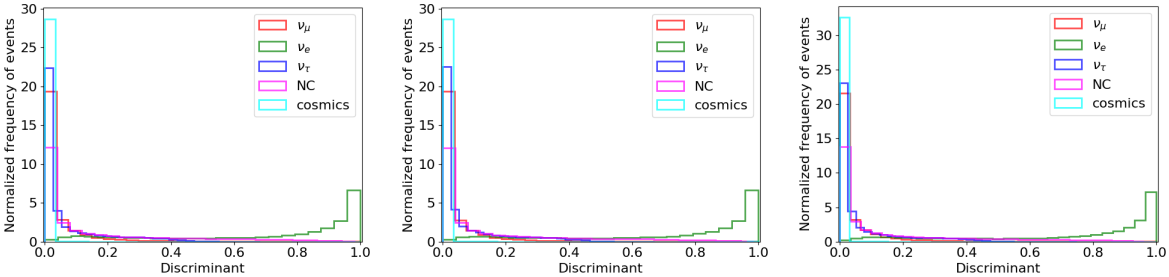


(b)

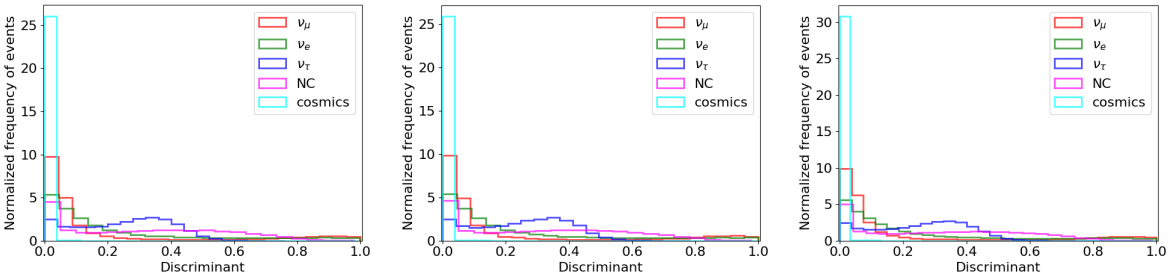
Obrázek 5.18: Ilustrace průběhu trénování pro model CVN v rámci trénovacích cyklů (epoch).



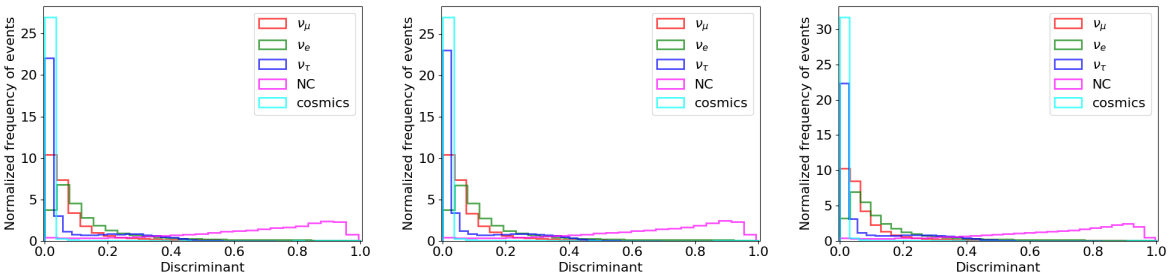
(a) test ν_μ (b) valid ν_τ (c) train ν_τ



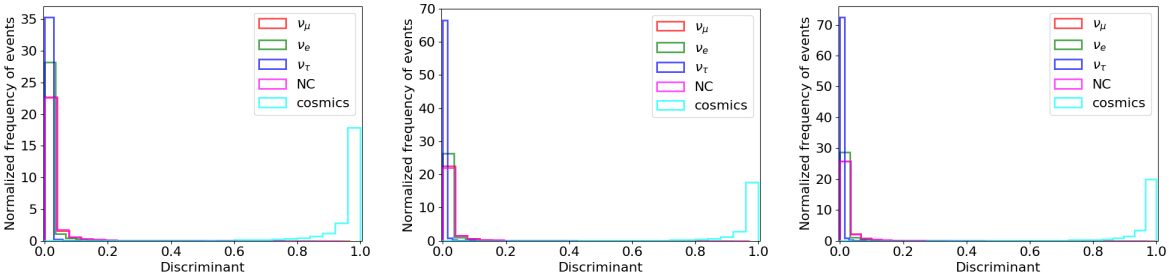
(d) test ν_e (e) valid ν_e (f) train ν_e



(g) test ν_τ (h) valid ν_τ (i) train ν_τ



(j) test NC (k) valid NC (l) train NC



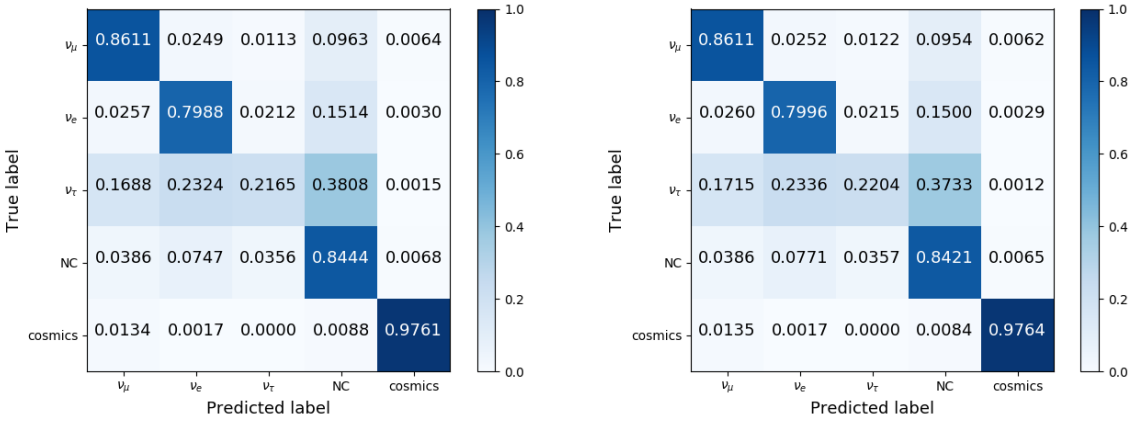
(m) test kosmika (n) valid kosmika (o) train kosmika

Obrázek 5.19: Control ploty pro CVN.

5.7 ResNet

	Test			Valid			Train		
Třída	prec.	recall	F1	prec.	recall	F1	prec.	recall	F1
ν_μ	0.875	0.861	0.868	0.874	0.861	0.868	0.876	0.863	0.869
ν_e	0.812	0.799	0.806	0.811	0.800	0.805	0.816	0.802	0.809
ν_τ	0.460	0.217	0.294	0.459	0.220	0.298	0.471	0.224	0.303
NC	0.733	0.844	0.785	0.734	0.842	0.784	0.736	0.847	0.787
Kosmika	0.954	0.976	0.965	0.956	0.976	0.966	0.957	0.978	0.967
Váž. průměr	0.792	0.803	0.793	0.792	0.803	0.793	0.795	0.805	0.796

Tabulka 5.4: Výsledky pro model ResNet na testovací, validační a trénovací množině.

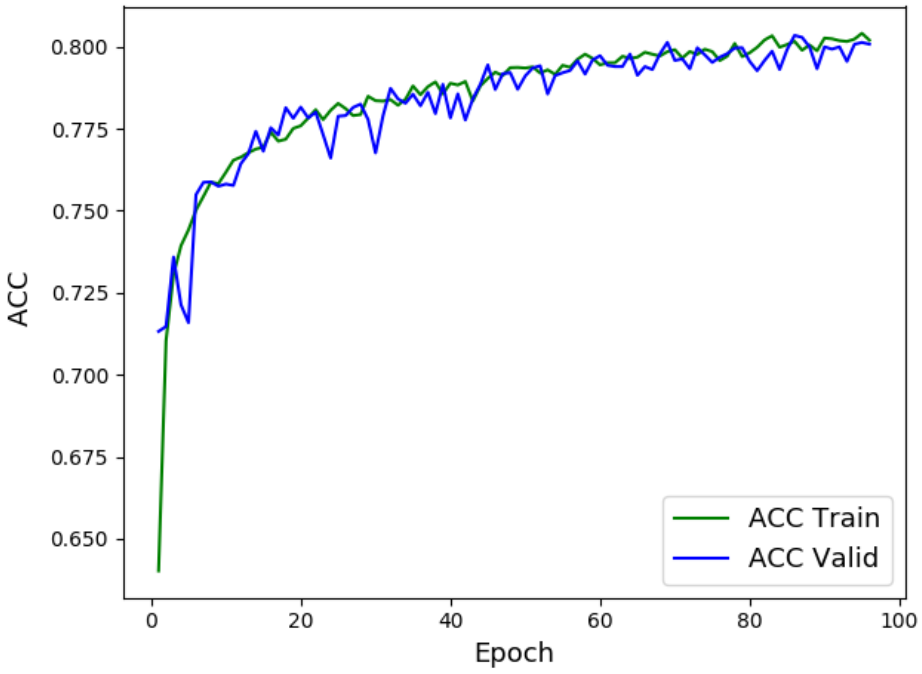


(a) testovací data

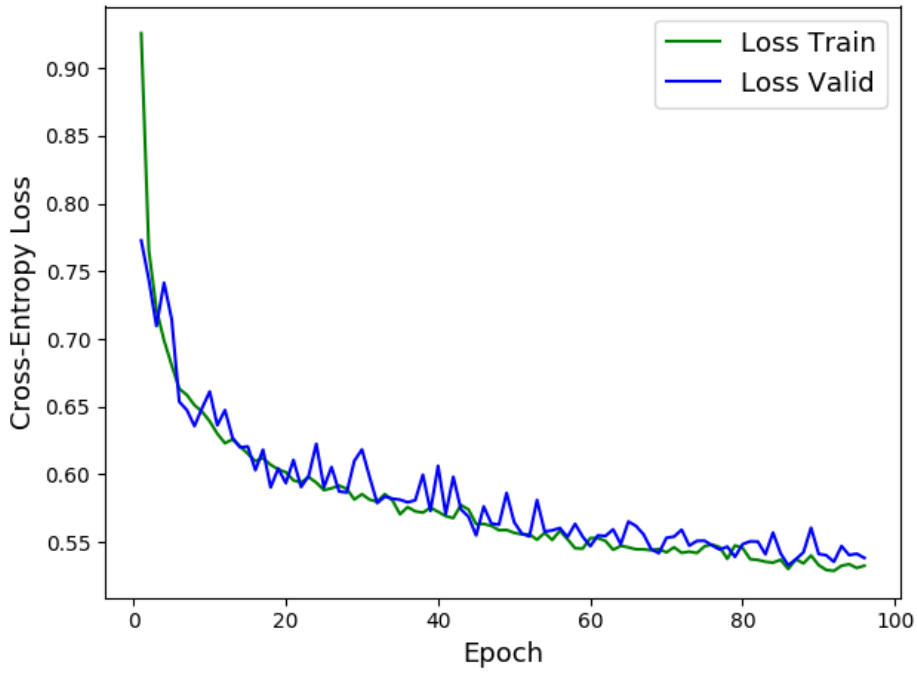
(b) validační data

(c) trénovací data

Obrázek 5.20: Normalizovaná matice záměn pro model ResNet. Diagonální hodnoty zvýrazněné barevnou škálou odpovídají *ACC* pro příslušnou třídu.

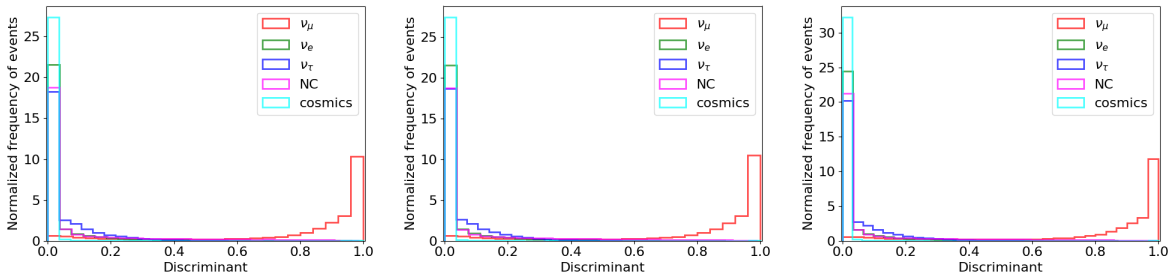


(a)

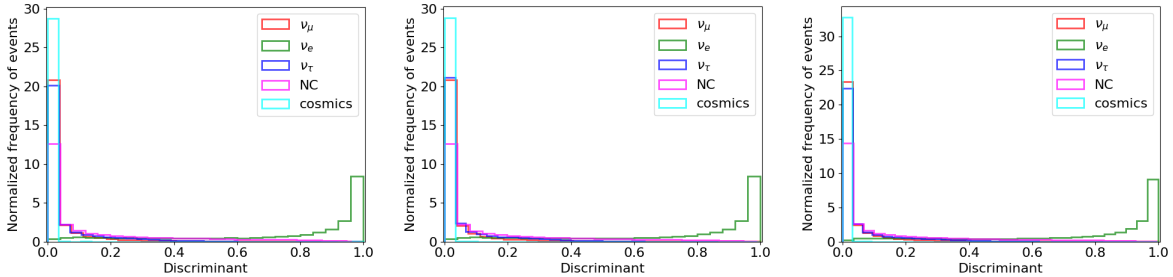


(b)

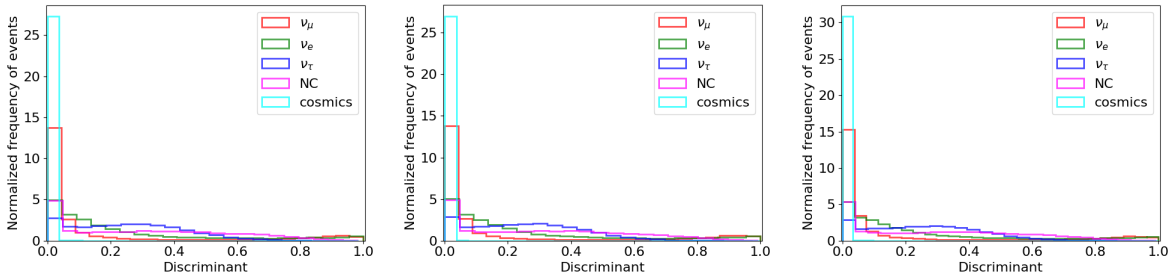
Obrázek 5.21: Ilustrace průběhu trénování pro model ResNet v rámci trénovacích cyklů (epoch).



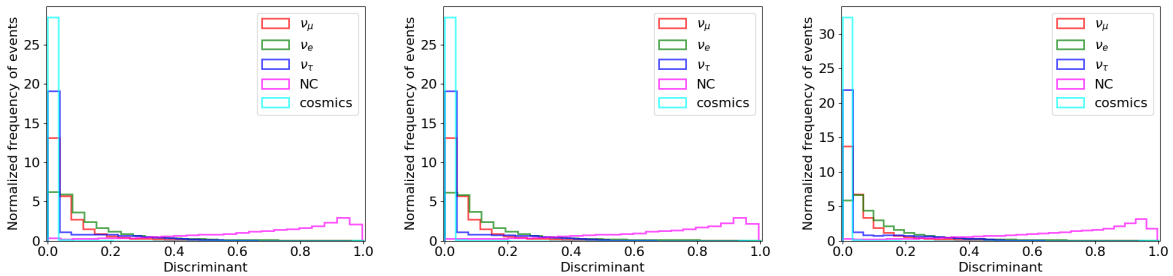
(a) test ν_μ (b) valid ν_μ (c) train ν_μ



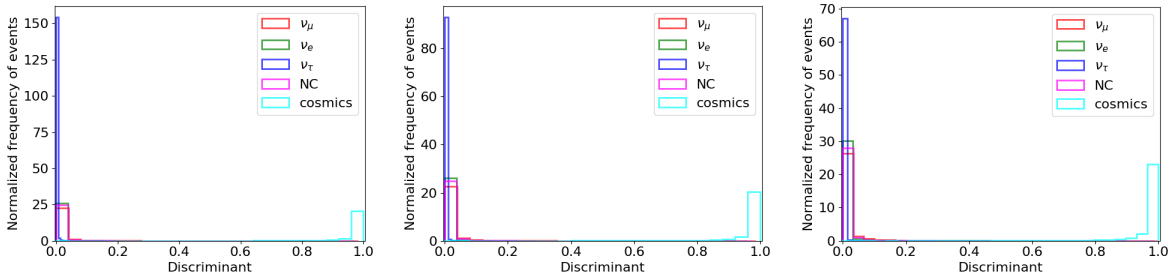
(d) test ν_e (e) valid ν_e (f) train ν_e



(g) test ν_τ (h) valid ν_τ (i) train ν_τ



(j) test NC (k) valid NC (l) train NC



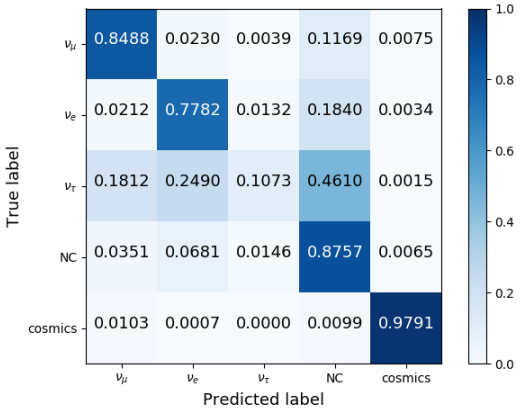
(m) test kosmika (n) valid kosmika (o) train kosmika

Obrázek 5.22: Control ploty pro ResNet.

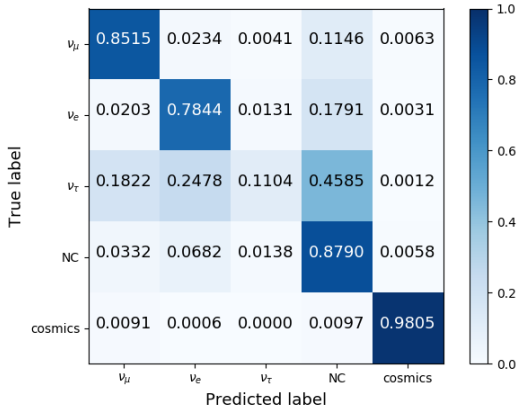
5.8 ResNet + náhodný les

	Test			Valid			Train		
Třída	prec.	recall	F1	prec.	recall	F1	prec.	recall	F1
ν_μ	0.879	0.849	0.864	0.882	0.852	0.867	0.883	0.853	0.868
ν_e	0.813	0.778	0.895	0.815	0.784	0.800	0.821	0.787	0.803
ν_τ	0.478	0.107	0.175	0.491	0.110	0.180	0.505	0.114	0.186
NC	0.701	0.876	0.779	0.705	0.879	0.782	0.706	0.882	0.784
Kosmika	0.952	0.979	0.965	0.958	0.981	0.969	0.957	0.982	0.969
Váž. průměr	0.785	0.796	0.778	0.789	0.799	0.782	0.792	0.802	0.784

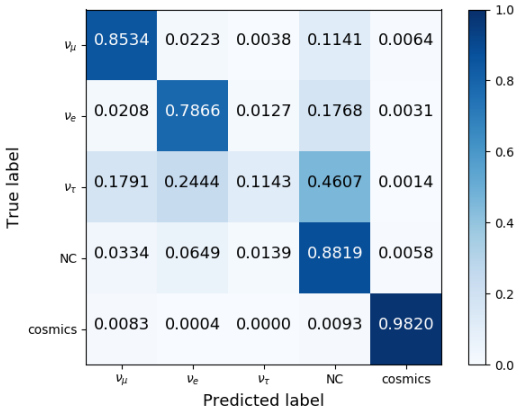
Tabulka 5.5: Výsledky pro model ResNet & náhodný les na testovací, validační a trénovací množině.



(a) testovací data

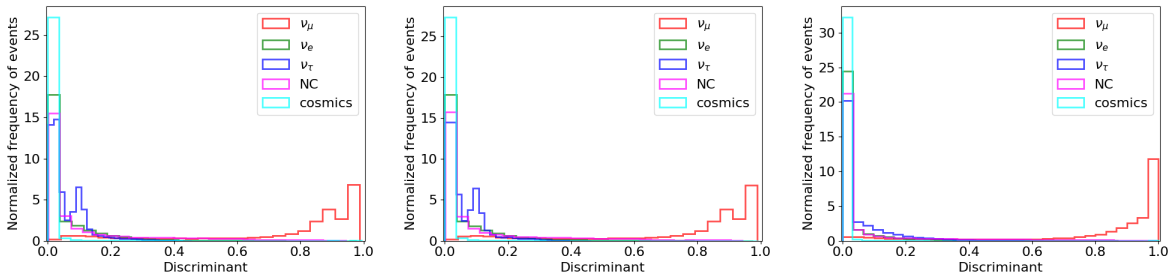


(b) validační data

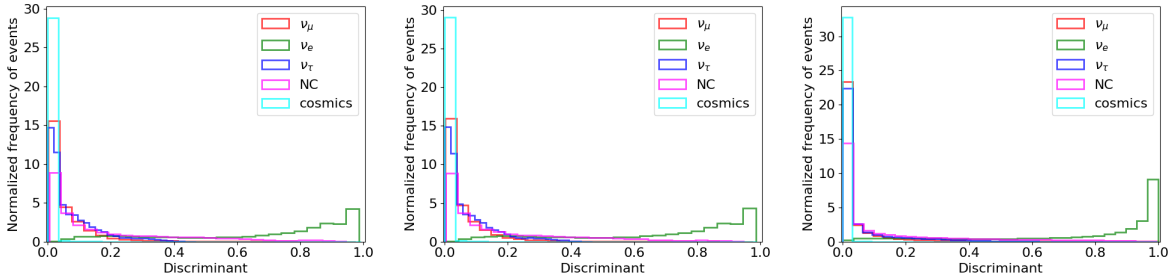


(c) trénovací data

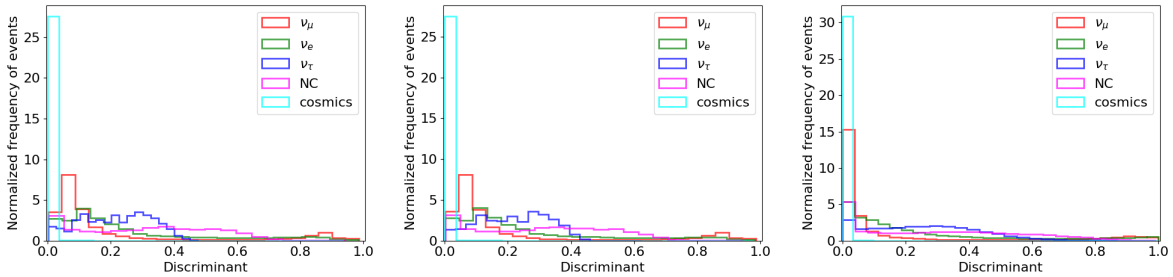
Obrázek 5.23: Normalizovaná matice záměn pro model ResNet & náhodný les. Diagonální hodnoty zvýrazněné barevnou škálou odpovídají *ACC* pro příslušnou třídu.



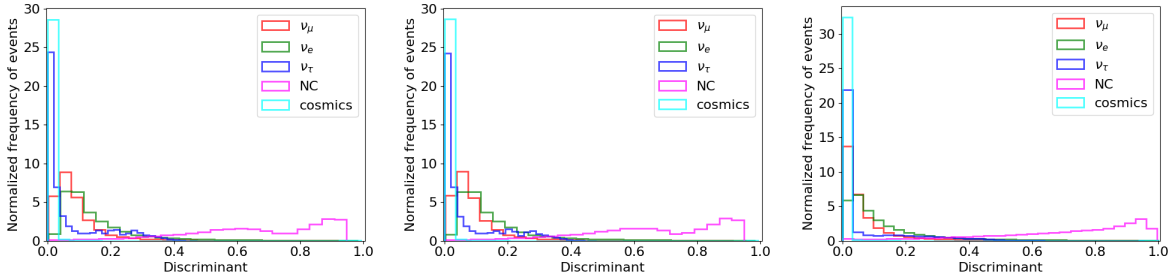
(a) test ν_μ (b) valid ν_μ (c) train ν_μ



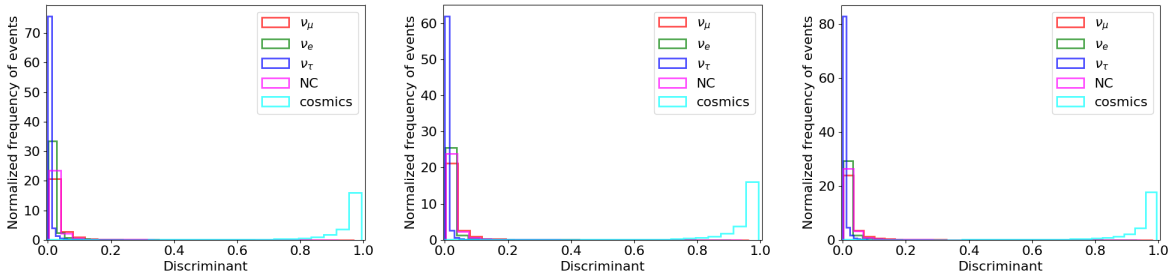
(d) test ν_e (e) valid ν_e (f) train ν_e



(g) test ν_τ (h) valid ν_τ (i) train ν_τ



(j) test NC (k) valid NC (l) train NC



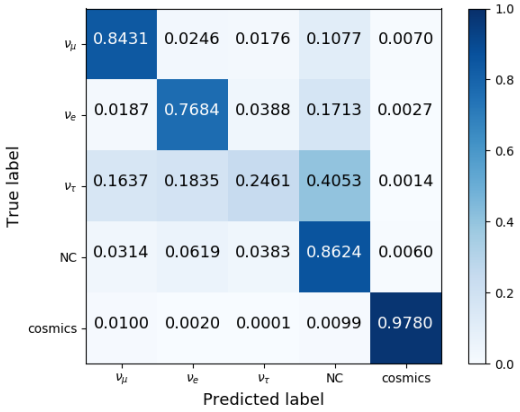
(m) test kosmika (n) valid kosmika (o) train kosmika

Obrázek 5.24: Control ploty pro ResNet & náhodný les.

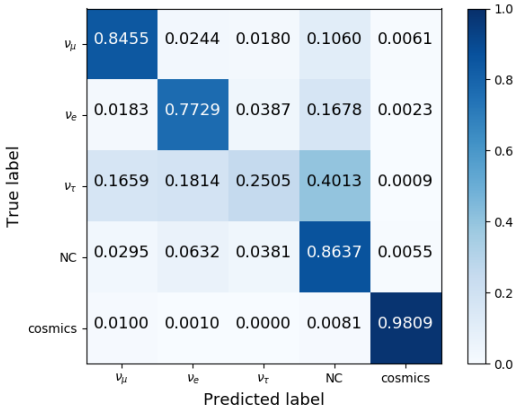
5.9 ResNet + AdaBoost

	Test			Valid			Train		
Třída	prec.	recall	F1	prec.	recall	F1	prec.	recall	F1
ν_μ	0.889	0.843	0.866	0.891	0.845	0.868	0.894	0.848	0.870
ν_e	0.833	0.768	0.799	0.834	0.773	0.803	0.839	0.776	0.806
ν_τ	0.415	0.246	0.309	0.419	0.251	0.314	0.427	0.255	0.320
NC	0.717	0.862	0.783	0.719	0.864	0.785	0.722	0.867	0.788
Kosmika	0.956	0.978	0.967	0.961	0.981	0.971	0.962	0.983	0.972
Váž. průměr	0.793	0.798	0.791	0.796	0.801	0.794	0.799	0.804	0.797

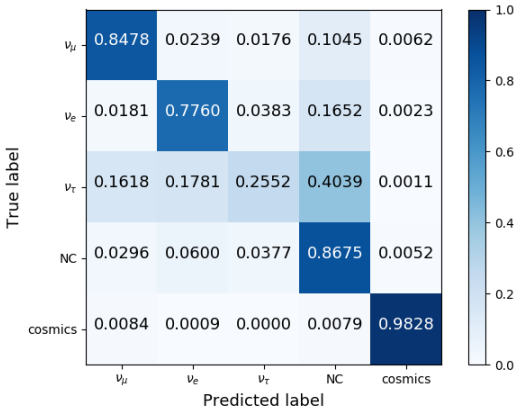
Tabulka 5.6: Výsledky pro model ResNet & AdaBoost na testovací, validační a trénovací množině.



(a) testovací data

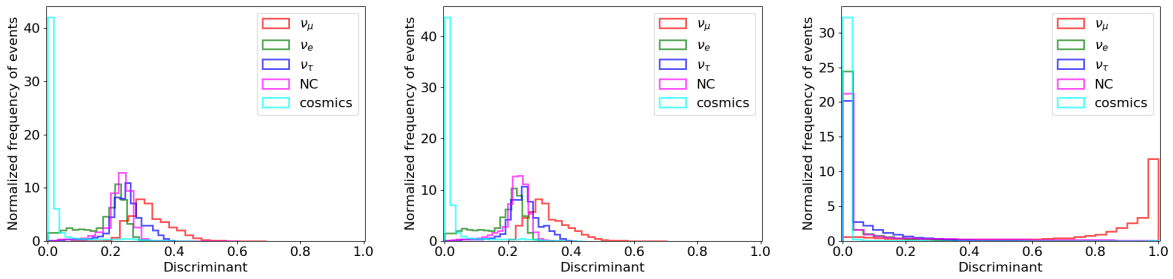


(b) validační data

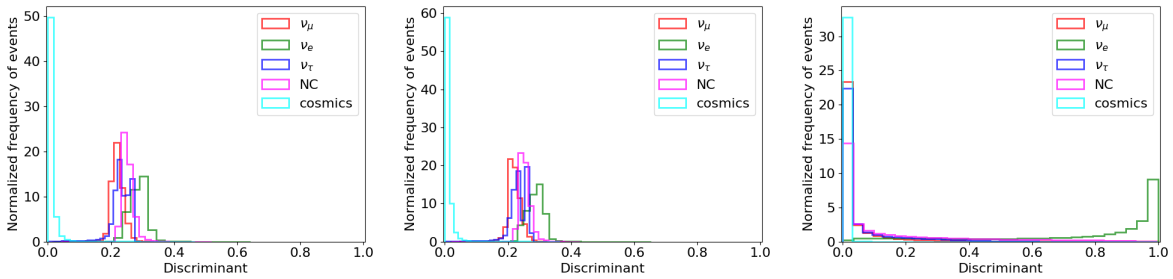


(c) trénovací data

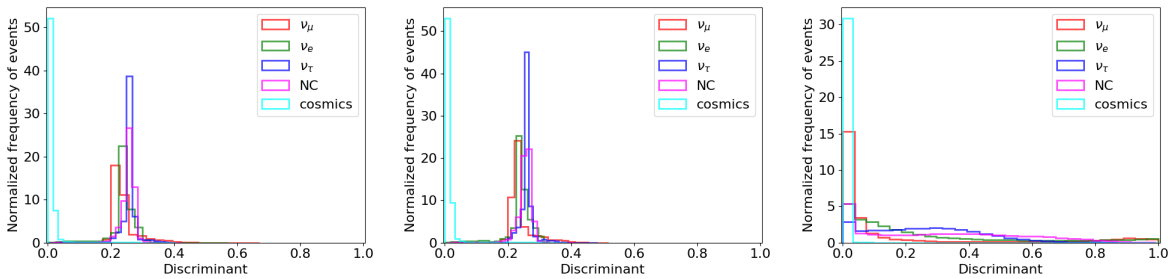
Obrázek 5.25: Normalizovaná matice záměn pro model ResNet & AdaBoost. Diagonální hodnoty zvýrazněné barevnou škálou odpovídají *ACC* pro příslušnou třídu.



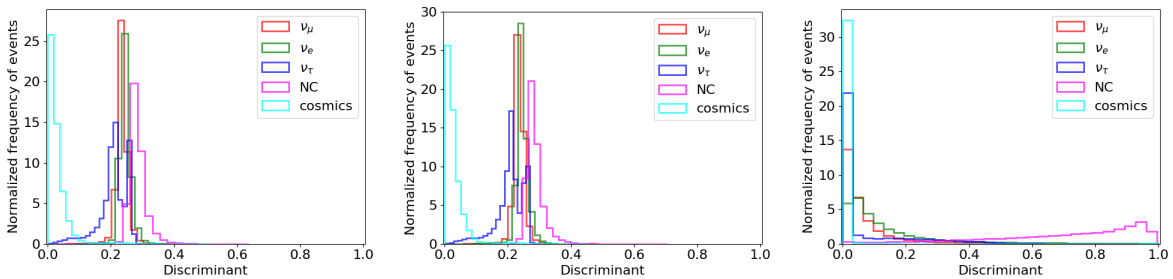
(a) test ν_μ (b) valid ν_μ (c) train ν_μ



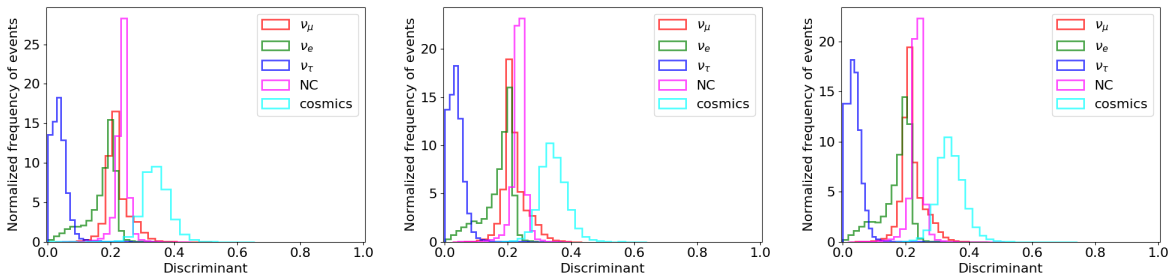
(d) test ν_e (e) valid ν_e (f) train ν_e



(g) test ν_τ (h) valid ν_τ (i) train ν_τ



(j) test NC (k) valid NC (l) train NC



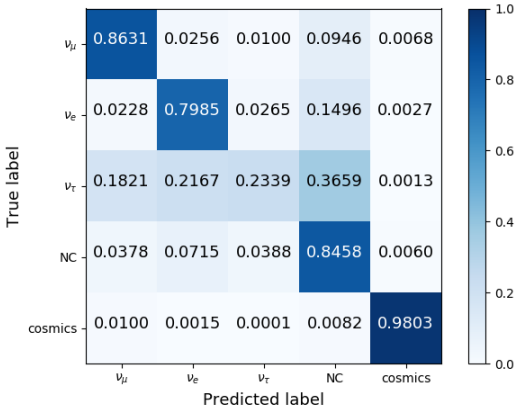
(m) test kosmika (n) valid kosmika (o) train kosmika

Obrázek 5.26: Control ploty pro ResNet & AdaBoost.

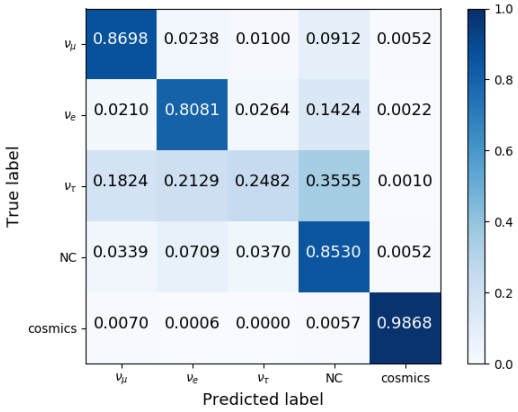
5.10 ResNet + Gradient Boosting

	Test			Valid			Train		
Třída	prec.	recall	F1	prec.	recall	F1	prec.	recall	F1
ν_μ	0.877	0.863	0.870	0.884	0.870	0.877	0.885	0.872	0.879
ν_e	0.819	0.798	0.809	0.825	0.808	0.817	0.831	0.821	0.812
ν_τ	0.454	0.234	0.309	0.476	0.248	0.326	0.493	0.257	0.338
NC	0.738	0.846	0.788	0.746	0.853	0.796	0.749	0.857	0.799
Kosmika	0.957	0.980	0.968	0.965	0.987	0.976	0.966	0.988	0.977
Váž. průměr	0.796	0.805	0.797	0.804	0.813	0.805	0.808	0.817	0.809

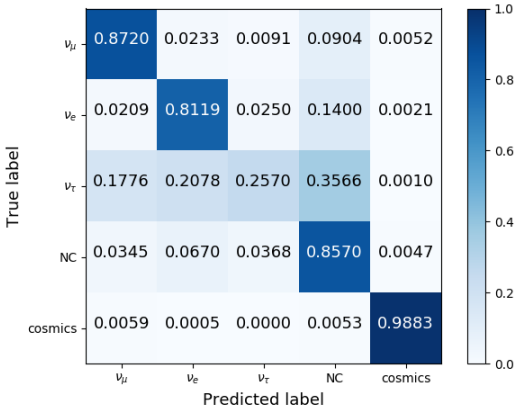
Tabulka 5.7: Výsledky pro model ResNet & Gradient Boosting na testovací, validační a trénovací množině.



(a) testovací data

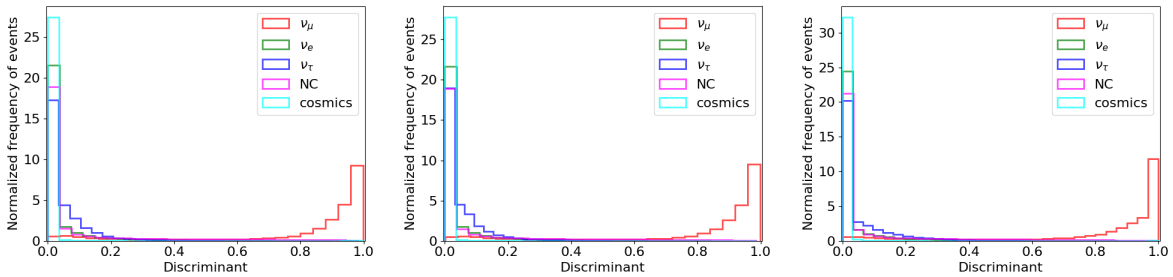


(b) validační data

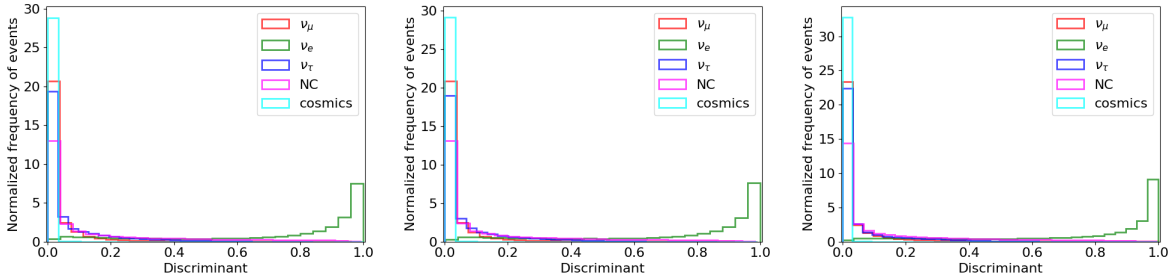


(c) trénovací data

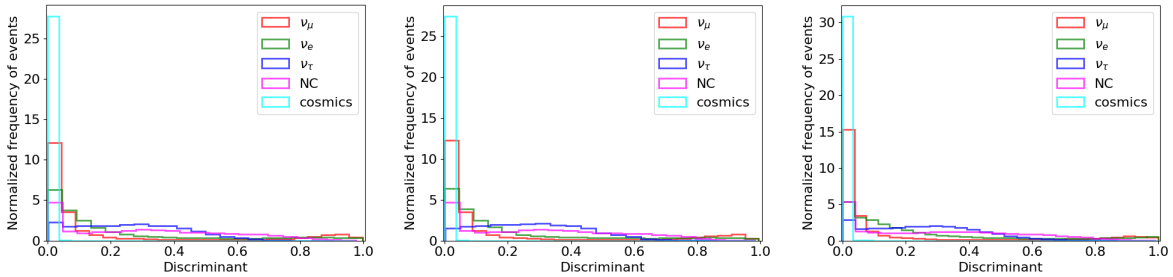
Obrázek 5.27: Normalizovaná matice záměn pro model ResNet & Gradient Boosting. Diagonální hodnoty zvýrazněné barevnou škálou odpovídají *ACC* pro příslušnou třídu.



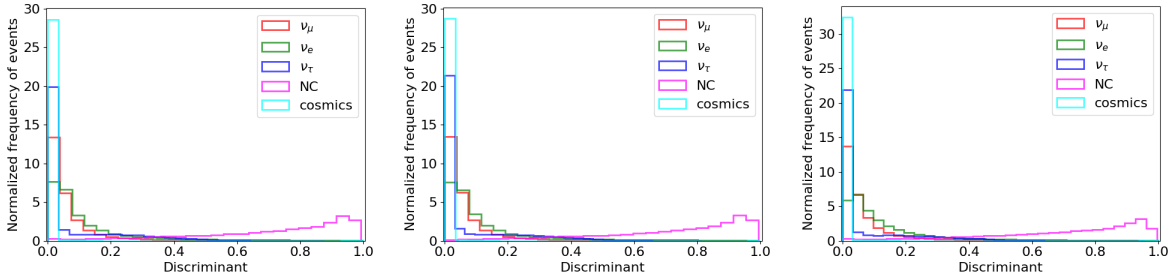
(a) test ν_μ (b) valid ν_μ (c) train ν_μ



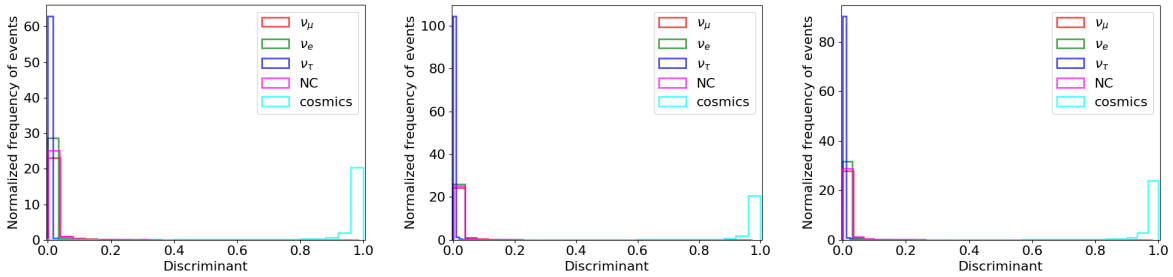
(d) test ν_e (e) valid ν_e (f) train ν_e



(g) test ν_τ (h) valid ν_τ (i) train ν_τ



(j) test NC (k) valid NC (l) train NC



(m) test kosmika (n) valid kosmika (o) train kosmika

Obrázek 5.28: Control ploty pro ResNet & Gradient Boosting.