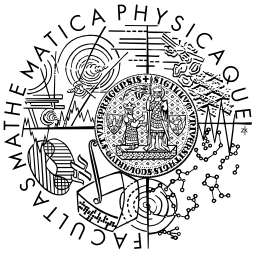


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Vít Bulín

Identifikace případů se čtyřmi top kvarky v datech z urychlovače LHC pomocí algoritmů strojového učení

Ústav částicové a jaderné fyziky

Vedoucí bakalářské práce: Mgr. Daniel Scheirich PhD.

Studijní program: Fyzika

Studijní obor: Fyzika

Praha 2024

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Rád bych poděkoval vedoucímu své bakalářské práce Mgr. Danielu Scheirichovi PhD. za jeho bezmeznou trpělivost, ochotu a vstřícný přístup. Tato práce mohla vzniknout díky jeho radám a neutuchající podpoře.

Rád bych poděkoval svým rodičům a blízkým za podporu v průběhu studia.

Název práce: Identifikace případů se čtyřmi top kvarky v datech z urychlovače LHC pomocí algoritmů strojového učení

Autor: Vít Bulín

Ústav: Ústav částicové a jaderné fyziky

Vedoucí bakalářské práce: Mgr. Daniel Scheirich PhD., Ústav částicové a jaderné fyziky

Abstrakt: Jet tagging je metoda analýzy produktů ze srážek na urychlovačích, jejíž podstatou je označování původu jednotlivých jetů na základě vůně původního kvarku, z něhož vznikly. Tato práce se zabývá využitím algoritmů strojového a hlubokého učení na vytvoření neuronové sítě schopné top-taggingu a přípravou takové sítě na využití v rámci detekce případů s rozpady na čtveřici top kvarků. Pro vybudování top-taggeru je využita neuronová síť založená na architektuře modelu Transformer. Pro výsledný model bylo vyzkoušeno několik variací architektury, několik sad hyperparametrů a pomocí dosažených výsledků na testovacích datech byl vybrán nejvýkonnější model. Výsledkem práce je otestovaný funkční top-tagger. Navíc bylo nalezeno doporučené nastavení modelu pro další využití při identifikaci rozpadů na čtyři top kvarky. Výstup z našeho modelu bude možno využít jako nadstandardní informaci pro vstup do verzatilnějších klasifikačních neuronových sítí.

Klíčová slova: Top-tagging; 4top event; Strojové učení; Hluboké učení; ATLAS

Title: Identification of events with four top quarks in data from the LHC collider using machine learning algorithms

Author: Vít Bulín

Institute: Institute of particle and nuclear physics

Supervisor: Mgr. Daniel Scheirich PhD., Institute of particle and nuclear physics

Abstract: Jet tagging is a method of analyzing accelerator collision products. It performs labeling of jets according to the flavor of quark from which they arose. The subject of the thesis is to use machine learning and deep learning algorithms to build a top-tagging neural network and to prepare this network for being used as a support mechanism for the identification of four top decay events. Our top-tagger architecture is based on the successful Transformer architecture. To achieve the final form of top-tagger several variations in architecture and several sets of hyperparameters were tried. Subsequently, the best-performing model was chosen. The result of our thesis constitutes of tested functioning top-tagger. Moreover, an optimal setting for further use in identifying four top events was found. The output of our neural network will be suitable to use as additional information to the input of more versatile classification models thus creating space for improvement of those classification models.

Keywords: Top-tagging; 4top event; Machine learning; Deep learning; ATLAS

Obsah

Úvod **3**

1 Standardní model **5**
1.1 Standardní model částic a interakcí 5
1.2 Uvěznění kvarků, vznik jetů 6
1.3 Top kvark 7

2 LHC a ATLAS **10**
2.1 Large Hadron Collider 10
2.2 Detektor ATLAS 11
2.2.1 Vnitřní detektor (Inner detector) 11
2.2.2 Kalorimetry 13
2.2.3 Mionový spektrometr 14
2.2.4 Soustava magnetů 14
2.2.5 Trigger 15

3 Data **16**
3.1 Vlastnosti 16
3.2 Analýza signálu z detektoru 16
3.3 Rekonstrukce jetů 17
3.3.1 Fixed cone algorithm 18
3.3.2 Sequential recombination algorithm 18
3.4 b-tagging, top-tagging 19
3.5 Trénovací proměnné 20

4 Strojové učení, Hluboké učení **21**
4.1 Základy strojového učení 21
4.1.1 Trénink modelu 21
4.1.2 Loss funkce 22
4.1.3 Optimalizace 24
4.1.4 Aktivační funkce 26
4.1.5 Regularizace 27
4.1.6 Evaluace 29
4.2 Cesta k architektuře Transformer 31
4.2.1 Perceptron, Fully connected NN, Feedforward NN 31
4.2.2 Multi-layer perceptron (MLP) 32
4.2.3 Rekurentní sítě (RNN) 32
4.2.4 Architektura typu seq2seq 34
4.2.5 Transformer 35

5 Výsledky **38**
5.1 Volba hyperparametrů 38
5.2 Top-tagger JetTrans 40
5.2.1 Výsledky 40
5.2.2 K-fold cross-validation 41
5.3 Využití pro identifikaci 4top eventů 42

Závěr **45**

Seznam použité literatury **46**

Seznam obrázků **49**

Úvod

Vědecké objevy učiněné na přelomu 19. a 20. století předznamenávaly otřes, který záhy zaznamenal do té doby uznávaný fyzikální pohled na svět. Pochybnosti o Newtonově představě absolutního času, objev první elementární částice, úvahy o absolutně černém tělese a mnoho dalších ve výsledku vedou k formulaci teorií, jež dnes považujeme za fundamentální součásti moderní fyziky. Vzniká částicová fyzika, kvantová mechanika a teorie relativity. Lidstvo smí konečně nahlédnout jak do nitra hmoty, tak do hlubin vesmíru. Málokdy se však zdůrazňuje, že už v dobách, kdy fyziku formují jména jako Einstein, Bohr, Dirac a Schrödinger se začal psát souběžně další příběh, a to historie neuronových sítí. Začíná roku 1943, kdy je vytvořen první model umělého neuronu, v padesátých letech přichází perceptrony, jejichž význam však po prvotním rozvoji pozvolna upadá a pomyslný hřebíček do rakve poskytne článek o neschopnosti perceptronů vypořádat se s XOR problémem. Následuje značný útlum v odvětví neuronových sítí, který prolomí až roku 1986 architektura vícevrstvého perceptronu (MLP). Kolem roku 1995 jsou neuronové sítě opět zastíněny, tentokrát algoritmem strojového učení SVM. Jak už ale dnes všichni víme, po roce 2010 nastává další renesance s příchodem hlubokých sítí a až dodnes se hluboké učení těší stále větší pozornosti a oblibě.

V současnosti hluboké sítě hrají roli efektivního nástroje v obrovském množství oborů, a to včetně částicové fyziky. Jedním z nejvýznamnějších experimentů jaderné a subjaderné fyziky současnosti je s jistotou detekce srážek částic na urychlovači LHC v CERNu. Data, která budou v naší práci využita, pochází z detektoru ATLAS, který je vybudován právě na LHC. Zaměříme se na zpracování dat s pomocí algoritmů strojového a hlubokého učení. Využijeme tedy hlubokých neuronových sítí k analýze dat, která nám ATLAS poskytuje. Zabýváme se konkrétně případy se vznikem čtyř top kvarků, jejichž podstata bude vysvětlena později. Zodpovězme nyní raději otázku, proč je pro nás rozpad na čtveřici top kvarků tak důležitý. Jedná se o velmi vzácný proces, kterého se účastní vůbec nejtěžší známé elementární částice. Takové interakce mohou mít potenciál testovat limity platnosti standardního modelu a vést k fyzice za standardním modelem. Standardní model je v dnešní době nejlepší dostupnou teorií popisující elementární částice a jejich interakce. Bohužel, ač se jedná o nejlepší dostupnou teorii, je dnes známo, že trpí jistými nedostatky, jejichž vysvětlení by mohla přinést právě teorie přesahující standardní model.

Konkrétně se v naší práci zaměříme na trénink modelu sloužícího jako takzvaný top-tagger, tedy neuronovou síť schopnou identifikovat a označovat detekované spršky částic (jety) pocházející z rozpadu top kvarku. Univerzální algoritmus či neuronová síť schopná provádět top-tagging libovolných jetů je jedním z úkolů, kterými se zabývá současná fyzika. Běžně se již však využívá klasifikačního modelu v podobě neuronové sítě na top-tagging *boosted top jetů* (jetů vznikajících z top kvarku, majících vysokou hybnost a kolinearitu), nikoliv však všech dostupných top jetů. S pomocí vytvořeného top-taggeru bude následně možné se pokusit o klasifikaci rozpadů na čtyři top kvarky, to však spíše s využitím dalších

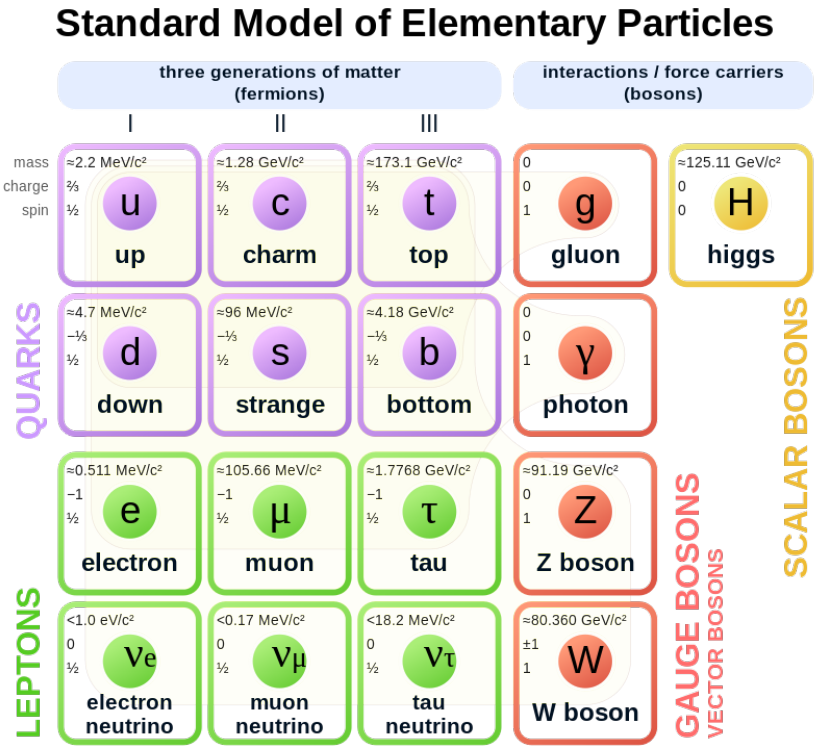
klasifikačních algoritmů, pro které by top-tagging mohl sloužit jako cenný vstup. Úkol klasifikovat rozpady na čtveřici top kvarků je ostatně velmi nelehký, protože koncový stav rozpadu čtveřice top kvarků může nabývat obrovského množství podob a je velmi obtížné jej rozpoznat.

Cílem naší práce bude vytvořit a otestovat model schopný top-taggingu jednotlivých jetů a znovu ověřit vhodnost neuronových sítí a hlubokého učení pro řešení tohoto problému.

1. Standardní model

1.1 Standardní model částic a interakcí

Standardní model je fyzikální teorie, která je v současné době nejlepším popisem základních interakcí a elementárních částic. Nejhruběji lze částice rozdělit na ty, které tvoří hmotu – *fermiony* a na ty, které zprostředkovávají interakce – *bosony*. Klasifikace částic standardním modelem je patrná na obrázku 1.1. Částice obsažené ve standardním modelu jsou všechny elementární, což znamená, že nemají vnitřní strukturu a nejsou tedy tvořeny z jiných částic.



Obrázek 1.1: Znázornění standardního modelu částic a interakcí, kde jsou barevně rozlišeny částice hmoty, neboli fermiony (kvarky a leptony) a částice zprostředkovávající interakce, neboli bosony.[1]

Bosony jsou částice mající celočíselný spin. Standardní model obsahuje čtyři takzvané vektorové bosony, zodpovídající za tři ze čtyř základních interakcí [1]:

- *Elektromagentická interakce* – Odehrává se mezi elektricky nabitými částicemi, jejím nositelem je *foton*.
- *Slabá interakce* – Jako jediná působí na všechny fermiony, jejími nositeli jsou *intermediální bosony Z a W*.

[1] Čtvrtou interakcí, která není ve standardním modelu zahrnuta, je gravitace, její nejlepší vysvětlení zatím poskytuje obecná teorie relativity.

- *Silná interakce* – Probíhá mezi kvarky, respektive mezi částicemi nesoucími *barevný náboj*, je zprostředkována *gluony* (jejím podrobným popisem se zabývá kvantová chromodynamika).

Posledním bosonem, takzvaným skalárním bosonem, je *Higgsův boson*. Jedná se o velmi významnou částici se zvláštním postavením v rámci teorie standardního modelu. Vzhledem k tomu, že není hlavním předmětem našeho zájmu, uvedeme značně zjednodušený popis. Higgsův boson je důsledkem existence skalárního *Higgsova pole*, které prostupuje celým prostorem a má nenulovou hodnotu. Díky potvrzení existence Higgsova pole je možno vysvětlit hmotnost elementárních částic v rámci standardního modelu.

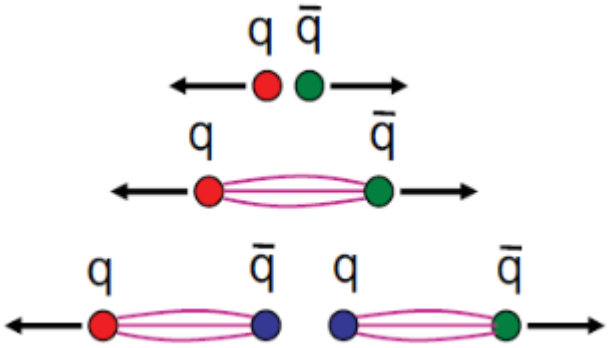
Společnou vlastností všech fermionů je jejich poločíselný spin, lze je však dále rozdělit do dvou skupin v závislosti na tom, jakým způsobem interagují. První skupinu tvoří *leptony*, konkrétně *elektron*, *mion*, *tauon* a také jim příslušná *neutrina*. Tímto způsobem tvoří tři leptonové rodiny, tedy elektron a elektronové neutrino a analogicky zbylé dvě. Důležitou vlastností leptonů je zachovávání leptonového čísla v rámci rodiny při interakcích. Elektron, mion a tauon nesou elektrický náboj a interagují tedy jak elektromagneticky, tak slabě, zatímco neutrina jsou elektricky neutrální a podléhají pouze slabé interakci.
Druhou skupinou jsou *kvarky*, kterých je taktéž šest, a jsou to *up*, *down*, *strange*, *charm*, *bottom* a *top*. Všechny nesou elektrický náboj, ale navíc mají ještě takzvaný barevný náboj (zkráceně barvu). Kvarky tedy interagují silně, slabě a elektromagneticky. Ač to není na obrázku 1.1 naznačeno, každý z dvanácti fermionů má svou antičástici, tedy částici se stejnými vlastnostmi, lišící se pouze hodnotou náboje. Kvarky se v přírodě vyskytují pouze ve vázaných stavech. Jsou k sobě poutány silnou interakcí, tedy prostřednictvím gluonů. Takové částice tvořené výhradně kvarky a gluony nazýváme *hadrony*. [2],[3]

1.2 Uvěznění kvarků, vznik jetů

Hadrony lze rozdělit do dvou kategorií. První z nich jsou *baryony*, které jsou tvořeny lichým počtem kvarků (typicky třemi kvarky) a díky tomu mají celkově poločíselnou hodnotu spinu a považujeme je za fermiony. Druhou kategorii tvoří *mesony*, neboli částice složené ze sudého počtu kvarků (typicky kvark a antikvark), a proto mají celočíselný spin a klasifikujeme je jako bosony.

Silnou interakcí a barevnými náboji se zabývá kvantová chromodynamika. Opět je ale potřeba ve zjednodušené verzi zavést pár stěžejních pojmů. *Hadronizace* je proces vzniku hadronů z kvarků a gluonů. S tímto pojmem velmi úzce souvisí také pojem *uvěznění kvarků* (někdy *barevné uvěznění*). Jedná se o pojmenování skutečnosti, že kvarky ani gluony není možné izolovat, vždy se budou vyskytovat pouze ve vázaném stavu. Hadronizace a uvěznění jsou schematicky znázorněny na obrázku 1.2. Představme si zjednodušený model dvou kvarků, které jsou k soubě poutány gluonem v podobě struny. Je-li systému dodána energie, může dojít k postupnému napínání struny ve snaze o odtržení kvarků. Jakmile je nashromážděná energie dostatečná, vzniká na jejím místě pár kvark-antikvark, které se silnou interakcí spojí s původní dvojicí za vzniku dvou nových hadronů.

Proces se může dále opakovat a dochází tak ke zmíněné hadronizaci.



Obrázek 1.2: Barevné uvěznění kvarků a postupná hadronizace. [4]

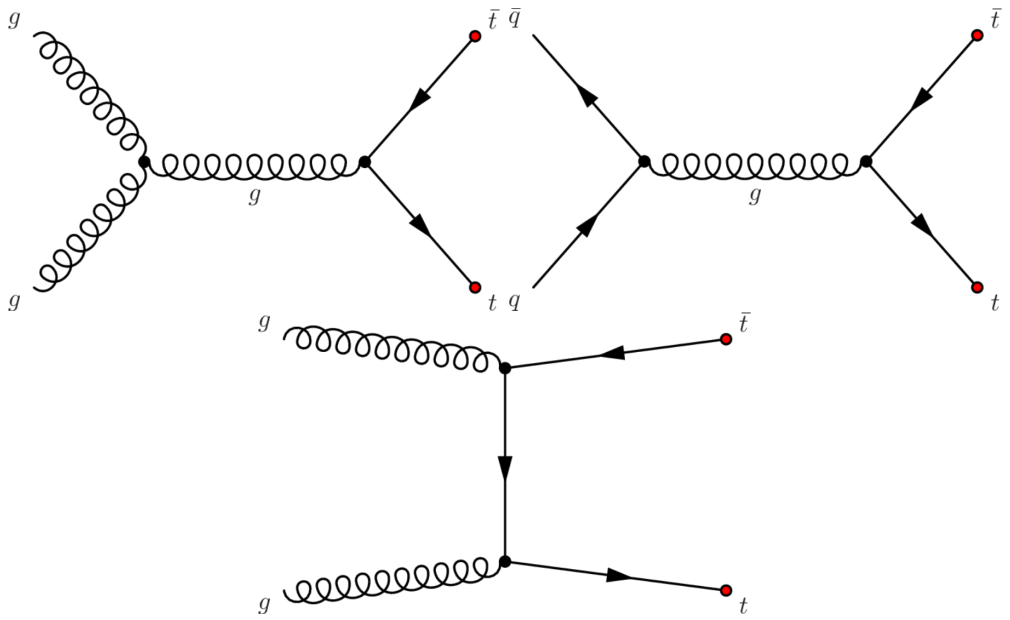
Při experimentech na urychlovačích částic proto nepozorujeme nikdy samostatné kvarky nebo gluony, nýbrž detekujeme spršky částic (mesonů a baryonů), které vznikají právě hadronizací. Tyto spršky jazýváme *jety* a lze si je představit jako úzké kužele částic, vyletujících z interakční oblasti. Typickým příkladem mohou být srážky na urychlovači v CERNu. Tam probíhají srážky protonů za velmi vysokých energií. Při srážce dochází efektivně ke srážkám a rozptylu konstituentů protonu, tedy kvarků a gluonů. Vzhledem k vysokým energiím, za kterých se interakce odehrává pak snadno dochází k hadronizaci a vzniku jetů.[2]

1.3 Top kvark

Stěžejní částicí pro naši práci bude nejtěžší elementární částice standardního modelu, tedy top kvark. Top kvark má hmotnost $m_t = (172{,}69 \pm 0{,}30)\,\mathrm{GeV}$.[5] Byl objeven roku 1995 v experimentech CDF a D0 na urychlovači Tevatron ve FNAL (Fermi National Accelerator Laboratory) ve Spojených státech. V těchto experimentech docházelo ke srážení protonů s antiprotony. Vzhledem k obrovské hmotnosti top kvarku je tento jedinou částicí, která nepodléhá hadronizaci ani barevnému uvěznění, jeho doba života je totiž přibližně $0{,}5 \cdot 10^{-24}\,\mathrm{s}$.[5] Dříve než by mohl vzniknout jakýkoliv hadronový vázaný stav se top kvark rozpadne prostřednictvím slabé interakce, přičemž ve více než 99% případů se jedná o rozpad na bottom kvark a W boson, který se poté dále rozpadá.[5],[2]

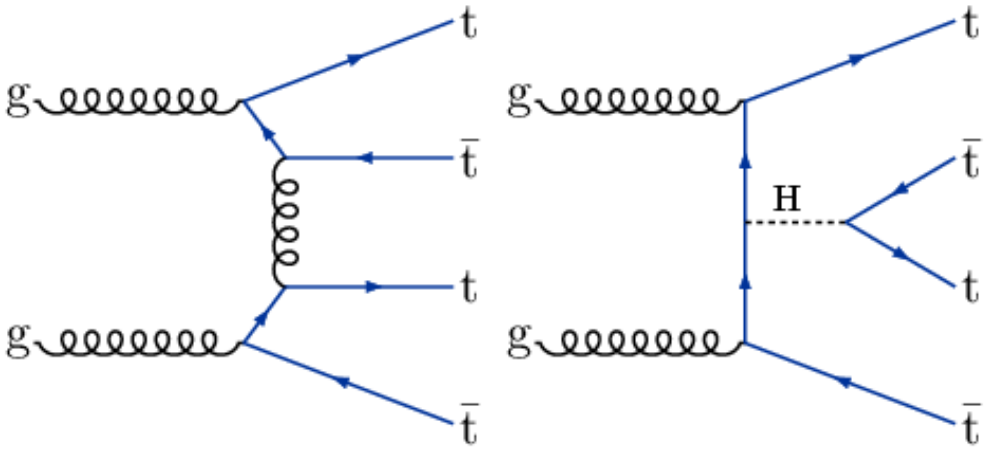
Co se týče produkce top kvarku, původně (na urychlovači Tevatron) byly pozorovány dva hlavní způsoby, a to produkce jednoho top kvarku společně s bottom kvarkem (tzv. single top production) a nebo produkce páru top kvark a top antikvark (častěji se používá výraz top-antitop). Mnohem častější je však produkce páru top-antitop, která probíhá prostřednictvím silné interakce při anihilaci kvarku a antikvarku, případně dvou gluonů. Interakce s produkcí top-antitop jsou patrné na obrázku 1.3.Single top produkce probíhá prostřednictvím slabé interakce a ač je energeticky výhodnější, má mnohem menší účinný průřez [2].[6]

[2]V klasické analogii částice a terče si lze účinný průřez představit jako plochu, kterou musí částice trefit, aby došlo k interakci.



Obrázek 1.3: Feynmanovy diagramy procesů s nejvyšším příspěvkem k produkci páru top-antitop.[7]

Procesem, který budeme v naší práci sledovat je vzácný případ produkce čtyř top kvarků. Detektory ATLAS i CMS [3] zaznamenaly produkci čtyř top kvarků s výsledky splňujícími 5σ-kritérium[4] v roce 2023.[8] Na obrázku 1.4 jsou opět k nahlédnutí Feynmanovy diagramy[5] příslušných interakcí.



Obrázek 1.4: Feynmanovy diagramy procesů s nejvyšším příspěvkem k produkci čtyř top kvarků.[9]

Rozpad top kvarku, jak již bylo zmíněno výše, probíhá velmi rychle a to prostřednictvím slabé interakce. Top kvark se rozpadá dominantně na bottom kvark

[3]Detektory na urychlovači LHC, viz kapitola 2.

[4]5σ-kritérium je obecně uznávaná hranice, kdy naměřená hodnota nabývá statistického významu. Pravděpodobnost, že naměřený signál je pouze statistickou fluktuací pozadí je poté menší něž $3 \cdot 10^{-7}$.

[5]Feynmanovy diagramy jsou grafickým vyjádřením výpočtu fyzikálních veličin v rámci poruchové teorie kvantové mechaniky.

a W boson. Podle následného rozpadu W bosonu můžeme rozlišovat takzvané rozpadové módy (kanály). Významnými produkty rozpadů jsou pro nás samozřejmě produkty z rozpadu čtyř top kvarků, ale také zde uvedeme rozpadové módy páru top-antitop ($t\bar{t}$), který je nejhůře filtrovatelným pozadím při pozorování produkce čtyř top kvarků.

Pro pár $t\bar{t}$ rozpoznáváme tři koncové stavy:

- Ve $\frac{4}{9}$ případů nalezneme v koncovém stavu dva b-jety a čtyři další jety. Někdy se tento mód nazývá all-jet či all-hadronic.
- Ve $\frac{4}{9}$ případů se v koncovém stavu nachází opět dva b-jety, dva další jety, nabitý lepton a neutrino či antineutrino (tak, aby bylo zachováno leptonové číslo). Jedná se o takzvaný semileptonic channel.
- Ve zbylé $\frac{1}{9}$ případů narazíme na leptonic channel. Pak koncový stav tvoří dva b-jety, dva nabité leptony a dvě neutrina či antineutrina.

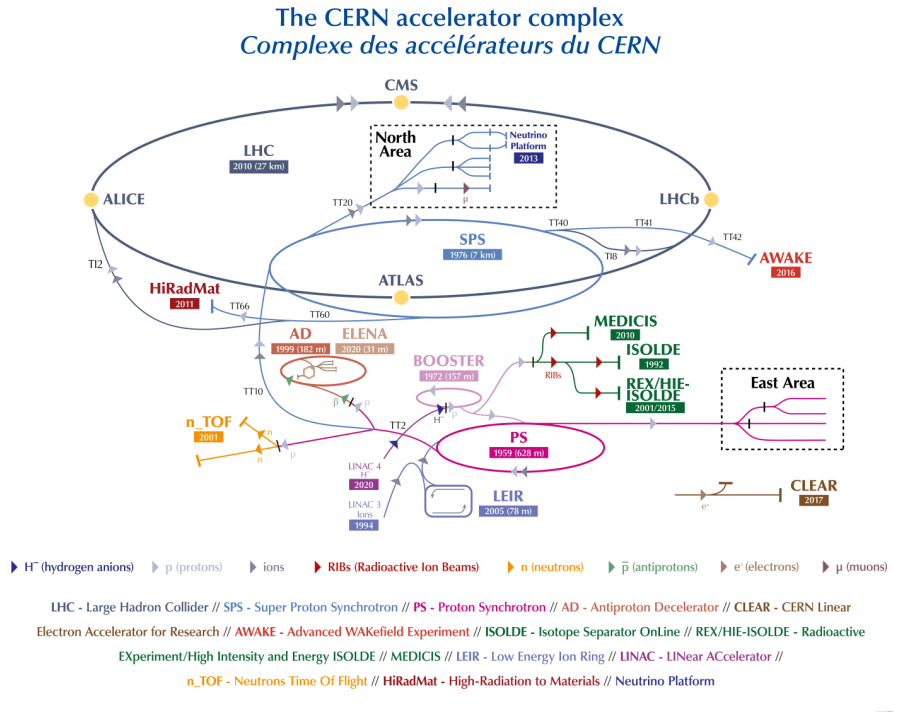
Co se týče detekce produktů koncového stavu z rozpadu čtyř top kvarků, jedná se o mnohem náročnější úkol. Každý z top kvarků se rozpadá na bottom kvark a boson W. Boson W se může poté rozpadat buďto leptonicky na nabitý lepton a příslušné neutrino či antineutrino nebo hadronicky na pár kvark-antikvark. Z toho je patrné, že koncové stavy takového rozpadu budou nabývat mnoha rozličných podob a že rozpad povede velkým množstvím kanálů. V koncovém stavu pak kromě čtveřice b-jetů nalezneme 0-4 nabité leptony a až dvanáct jetů pocházejících z kvarků. Kvůli tomu je hledání produkce čtyř top kvarků velmi obtížné. Do značné míry si však lze pomoci například metodami strojového a hlubokého učení. V naší práci se budeme zabývat top-taggingem pro all-hadronic mód.[9]

2. LHC a ATLAS

Roku 1954 byla založena Evropská organizace pro jaderný výzkum, zkráceně CERN, se sídlem v Ženevě. Jedná se o mezinárodní organizaci, která má již 23 členských států. Komplex v CERNu je dnes největší laboratoří provádějící experimenty v oblasti subjaderné fyziky. Laboratoř za sebou má mnoho úspěchů na poli fyziky vysokých energií, například objev bosonů Z a W, pozorování narušení CP-symetrie, či zřejmě vůbec nejznámější úspěch, tedy objev Higgsova bosonu. Posledně zmíněný objev mohl být učiněn až díky vybudování doposud vůbec největšího urychlovače částic na světě - *LHC* (zkratka pro *Large Hadron Collider*)[10]

2.1 Large Hadron Collider

LHC je největším urychlovačem v soustavě urychlovačů v CERNu. Částice, se kterými se pracuje na LHC, nejdříve musely projít řetězcem urychlovačů s postupně vyšší a vyšší cílovou energií, na kterou byly urychlovány, a až poté mohou být injektovány do samotného LHC, jakožto posledního článku soustavy. Znázornění komplexu v CERNu je patrné na obrázku 2.1. Samotný LHC má obvod téměř 27 km a jsou zde urychlovány svazky protonů nebo jader olova na rychlost blížící se rychlosti světla.



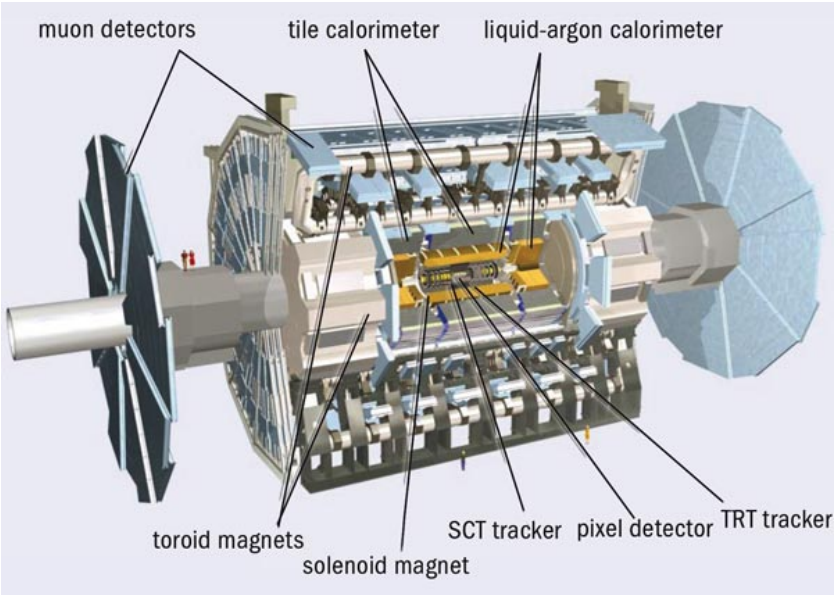
Obrázek 2.1: Schematická ilustrace soustavy urychlovačů CERN. [11]

V prostorách přibližně 100 m pod zemským povrchem se nachází dvě trubice, v nichž se pohybují urychlované shluky částic. Uvnitř trubic je vysoké vakuum a po celém jejich obvodu jsou umístěny tisíce supravodivých elektromagnetů, které

svým polem paprsky urychlují a zakřivují jejich trajektorii. Aby elektromagnety dosáhly supravodivosti, musí pracovat při nízkých teplotách. Proto jsou chlazeny na teplotu 1,9 K pomocí tekutého helia. Paprsky jsou urychlovány v protichůdném směru, aby mohlo v místech, kde se trubice kříží, dojít ke srážkám urychlených shluků částic. Na LHC jsou celkem čtyři taková místa a právě na nich jsou umístěny detektory různých typů. Celkově na LHC běží devět experimentů z nichž každý má svůj typ detektoru. Zmíníme alespoň čtyři největší z nich a těmi jsou *ATLAS, CMS, ALICE* a *LHCb*. Právě z detektoru experimentu ATLAS pochází data pro naši práci.[12],[11]

2.2 Detektor ATLAS

ATLAS (A Toroidal LHC AparatuS) je největším víceúčelovým detektorem na LHC. Má válcový tvar s úctyhodnými rozměry, průměr činí 25 m, délka 46 m a hmotnost přibližně 7000 t. Detektor se skládá z šesti vrstev, které tvoří subdetektory, schopné zaznamenávat trajektorii, hybnost a energii částic, což umožňuje jejich následnou identifikaci. Celkové schéma detektoru je k nahlédnutí jako obrázek 2.2. Uvnitř detektoru dojde každou sekundu k více než miliardě srážek a interakcí částic. Sice pouze jen jedna z milionu srážek je označena jako potenciálně hodná bližšího studia, nicméně i tak experiment produkuje a ukládá obrovské množství dat.[13],[14]

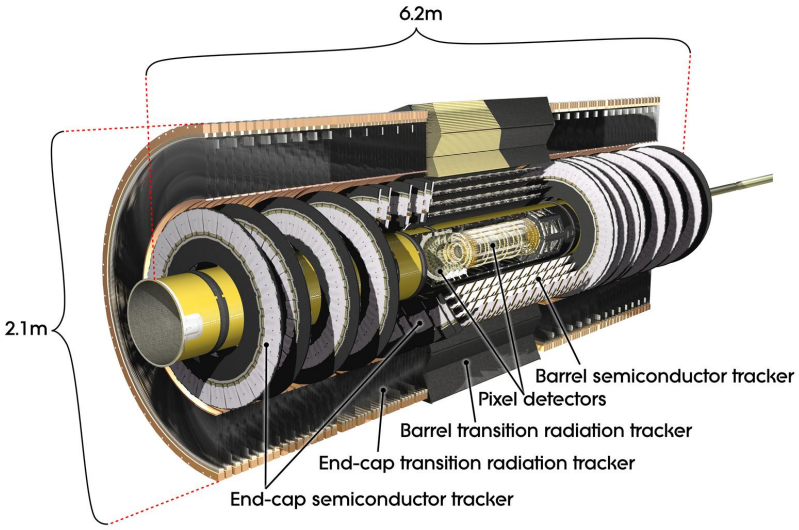


Obrázek 2.2: Schematické znázornění celkového pohledu na detektor ATLAS s popisem jeho nejdůležitějších částí.[15]

2.2.1 Vnitřní detektor (Inner detector)

První částí zasaženou bezprostředně po částicové srážce je právě vnitřní detektor. Nachází se v magnetickém poli, rovnoběžném se směrem průchodu paprsků, které zakřivuje dráhy nabitých částic. Skládá se ze tří částí: pixelového detektoru,

stripového detektoru(SCT) a TRT. Vnitřní detektor slouží k zaznamenání hybnosti, náboje a trajektorie částic produkovaných proton-protonových srážkách. Průřez vnitřním detektorem je zachycen na obrázku 2.3.



Obrázek 2.3: Detailní pohled na podélný průřez vnitřním detektorem.[16]

Pixelový detektor (Pixel detector)

Pixelový detektor je nejbližší vrstvou k interakční oblasti. Jedná se tedy o místo prvního kontaktu produktů srážky s detektorem, a proto také o místo s nejvyššími nároky na odolnost vůči ionizujícímu záření. Je složen z 92 miliónů křemíkových pixelů o rozměrech v řádech stovek mikrometrů, které jsou uspořádány do čtyř vrstev. Prolétávající částice ztrácí v pixelovém detektoru energii a pohybuje se po zakřivené trajektorii, z jejíhož poloměru zakřivení lze určit hybnost této částice a navíc i její původ s přesností na 10 μm.

Stripový detektor (Semiconductor tracker - SCT)

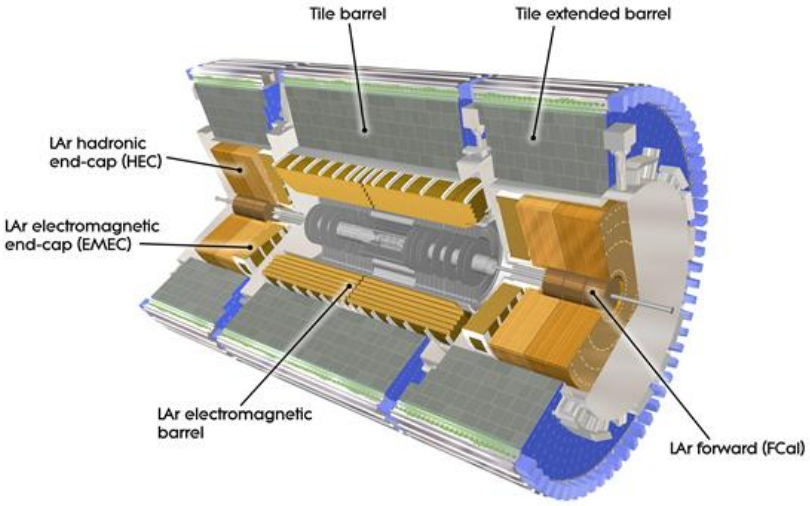
SCT tvoří vrstvu kolem pixelového detektoru. Princip pixelového detektoru a SCT je navíc velmi podobný, rozdíl je pouze v uspořádání křemíkových detektorů, které v SCT tvoří stripy (pásky). Opět je tak díky průchodu čtyřmi vrstavmi stripů možné částici trackovat (zaznamenávat její trajektorii) s přesností na 25 μm.

TRT (Transition radiation tracker)

Poslední vrstvu vnitřního detektoru tvoří TRT , který sestává z velkého počtu tenkostěnných trubiček. Uvnitř trubiček je pozlacené wolframové vlákno a plynová výplň. Prolétavající nabitá částice ionizuje plyn v trubičkách, díky čemuž vzniká detekovatelný signál. TRT je schopný poskytnout informaci o typu částice.

2.2.2 Kalorimetry

Kolem vnitřního detektoru, avšak už mimo jeho magnetické pole, se nachází další ze subdetektorů a tím je soustava *kalorimetrů*. Kalorimetr je zařízení, které je schopné pohltit téměř všechny částice vznikajících při srážkách na urychlovači. Jedná se tedy o destruktivní metodu měření. Vzhledem k tomu, že jejich funkcí není pouze měření částic, ale i jejich pohlcování a zastavování, jedná se o podstatně mohutnější vrstvu detektoru ATLAS, jak je patrné na obrázku 2.4. Strukturu kalorimetru tvoří velké množství vrstev, kde se střídá materiál s vysokou hustotou schopný pohlcovat částice s vrstvami látky, díky níž je možné měřit energii částic. V soustavě kalorimetrů na detektoru ALTAS se vyskytují dva typy kalorimetrů, a to *elektromagetický kalorimetr* a *hadronový kalorimetr*.



Obrázek 2.4: Detailní pohled na podélný průřez soustavou kalorimetrů detektoru ATLAS. [17]

Elektromagnetické kalorimetry (Liquid Argon Calorimeter)

Vnitřní vrstvu soustavy kalorimetrů tvoří elektromagnetické kalorimetry, jejichž úkolem je měřit energii elektronů, fotonů a hadronů na základě jejich schopnosti elektromagneticky interagovat. Na obrázku 2.4 jsou označeny jako LAr, což je zkratka z anglického názvu, který napovídá, že médium, díky němuž lze měřit energie částic, je tvořeno *tekutým argonem*. Proto musí být elektromagnetické kalorimetry chlazeny na teplotu 89 K. Zmíněná detekce je možná také díky vrstvám kovu, mezi nimiž je tekutý argon uzavřen. Při průletu částice kovovou vrstvou (wolfram, olovo či měď) dochází k zániku původní částice a vzniku spršky částic s nižší energií. Taková sprška poté ionizuje tekutý argon a vytváří tak detekovatelný elektrický signál. Elektromagnetické kalorimetry jsou navrženy tak, aby pohlcovaly elektrony, pozitrony a fotony.

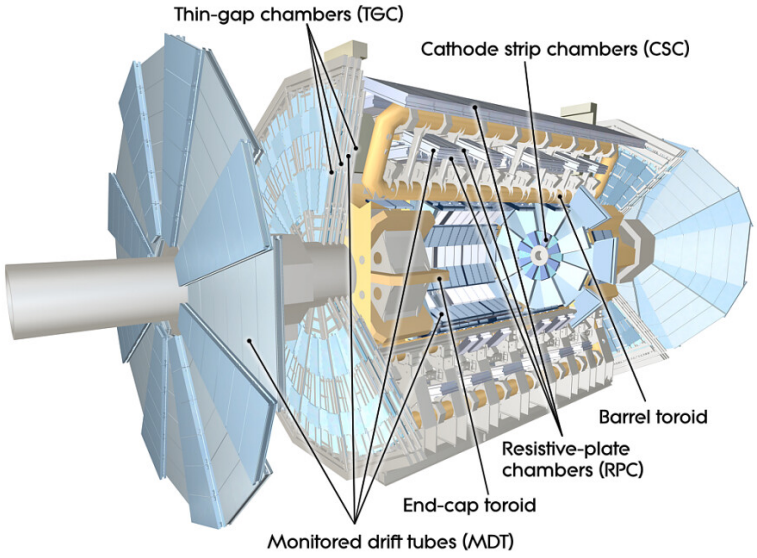
Hadronové kalorimetry (Tile Hadronic Calorimeter)

Vnější vrstvou soustavy kalorimetrů jsou kalorimetry hadronové, které mají jednak měřit energii hadronů, které již deponovaly část energie v elektromagnetic-

kých kalorimetrech a jednak hadrony absorbovat. To se odehrává prostřednictvím silné interakce, kterou hadrony (jakožto částice složené z kvarků a gluonů) interagují. Princip detekce je stejný jako u elektromagnetických kalorimetrů s tím rozdílem, že zde se střídají vrstvy oceli a scintilátorů. Na ocelových deskách vznikají spršky částic a scintilátory je přemění na signál, jehož inzenzita je úměrná energii pohlcené částice.

2.2.3 Mionový spektrometr

Skrz předchozí vrstvy detektoru se dostanou pouze neutrina a miony. Neutrina interagují pouze slabou interakcí, a proto jsou velmi obtížně detekovatelná. Miony sice interagují elektromagneticky, ale jejich hmotnost odpovídá zhruba hmotnosti 200 elektronů, pročež v kalorimetrech deponují pouze malou část své energie. Proto tvoří poslední a také největší vrstvu ATLASu mionový spektrometr, který se skládá z pěti částí (*Thin Gap Chambers, Resistive Plate Chambers, Monitored Drift Tubes, Small-Strip Thin Gap Chambers*, *Micromegas*). Miony nelze detekovat destruktivně (jejich pohlcením), pročež je potřeba zaznamenat jejich trajektorii a hybnost. Aby to bylo pro zmíněný systém detektorů možné, musí být mionový spektrometr v silném magnetickém poli.



Obrázek 2.5: Umístění detekčních systémů, tvořících mionový spektrometr. [18]

2.2.4 Soustava magnetů

Detektor ATLAS využívá silné magnetické pole vytvořené soustavou supravodivých elektromagnetů k zakřivování trajektorií částic. Díky průběhu trajektorie je pak možné určit náboj a hybnost pozorované částice. Veškeré supravodivé elektomagnety fungují až při teplotách 4,5 K, čímž se dosahuje vyjímečně silného magnetického pole. [19] Soustavu lze rozdělit na tři části:

- *Central solenoid* - Obklopuje vnitřní detektor, je tvořen niob-titaniovým supravodičem a tvoří magnetické pole o intenzitě $B = 2\,\mathrm{T}$.

- *Barrel toroid* - Je součástí vnějšího pláště detektoru a vytváří magnetické pole o intenzitě $B = 3{,}5\,\mathrm{T}$ pro mionový spektrometr.
- *End-cap toroid* - Nachází se na podstavách a slouží stejnému účelu jako barrel toroid.

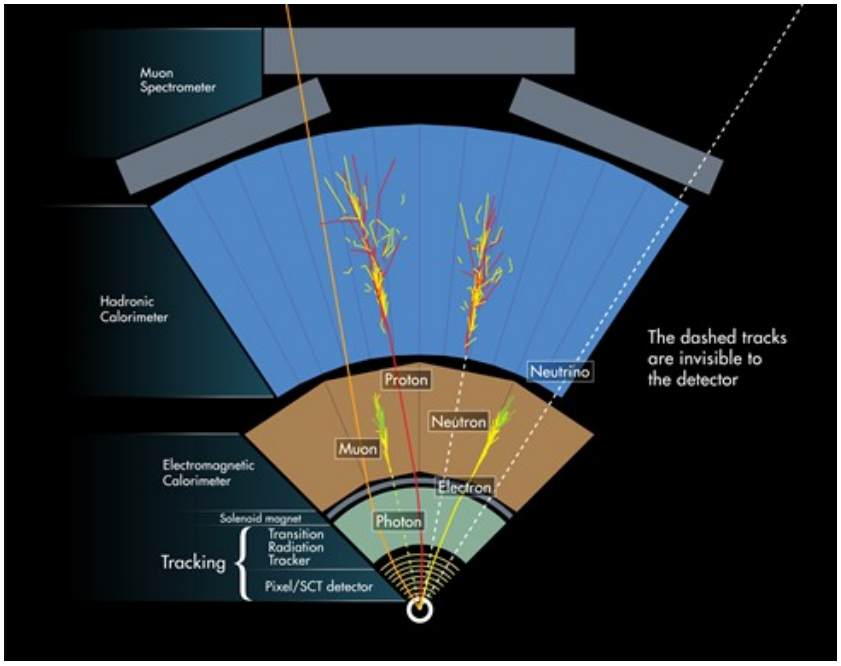
Poloha magnetů je zachycena také na obrázku 2.2.

2.2.5 Trigger

Jak již bylo výše zmíněno, v nitru detektoru se každou sekundu odehrává více než miliarda proton-protonových srážek (což odpovídá produkci zhruba 60 TB dat každou sekundu). Nicméně významné jsou pro nás z fyzikálního hlediska pouze některé interakce, a proto potřebuje ATLAS systém, který musí být schopen obrovský objem dat přefiltrovat. Tento systém se nazývá *trigger* a má dvě úrovně:

- První úroveň je hardwarová, pracuje s frakcí dat z kalorimetrů a mionového spektrometru a během $2{,}5\,\mu\mathrm{s}$[1] od eventu (události v detektoru) rozhodne zda data zachová a pošle je do druhé úrovně, která je schopna přijímat až 100 000 eventů za sekundu.
- Druhá úroveň je softwarová (má k dispozici zhruba 40 000 CPU) a během $200\,\mu\mathrm{s}$ provede podrobnou analýzu dat, přičemž vybere zhruba 1000 eventů za sekundu vhodných k uložení.[20]

Na závěr, po popisu principu všech subdetektorových vrstev uvádíme ještě obrázek 2.6, kde jsou popsané principy zachyceny.



Obrázek 2.6: Příčný průřez celým detektorem ATLAS s naznačenou formou detekce různých částic. [21]

[1] V průběhu rozhodovací doby jsou data uložena v takzvaných *buffer-storages*

3. Data

Data využívaná pro naši práci pochází z detektoru ATLAS v CERNu. Nejedná se však přímo o reálná naměřená data, ale o *Monte Carlo simulaci* (*MC*), která napodobuje srážku částic a následnou odezvu v detektoru.

3.1 Vlastnosti

Samotnou MC lze vnímat jako dvě části, první je simulace srážky, založená na fyzikálním modelu a druhá je simulace interakce produktů srážky s detektorem. Data z MC přichází ve stejném formátu jako data z reálného měření. [22] Obsahují však navíc dvě sady informací, díky kterým je umožněn efektivní trénink algoritmů strojového a hlubokého učení:

- První z nich je sada *vah* (*weights*) pro jednotivé eventy. Jednou z výhod, které MC poskytuje je možnost generovat různé typy eventů ve volitelných poměrech. Lze tedy simulovat velké množství vzácných interakcí, což je výhodné pro vyvážení *tréninkového datasetu*, který využijeme jako *vstup* při tréninku neuronové sítě. Aby však takový dataset stále dával fyzikální smysl, je potřeba jednotlivým eventům přičlenit váhy, které kompenzují změny v poměrech typů generovaných eventů.

- Druhou je takzvaná *ground-truth*. Zjednodušeně řečeno se jedná o informaci, kterou by nám k datům měl poskytnout idealizovaný 100% účinný klasifikační model. Ground-truth tedy využíváme při tréninku a snažíme se jí svou predikcí co nejvíce přiblížit.

Dataset využitý pro práci s neuronovou sítí sestává z několika typů eventů. Signálními eventy byly samozřejmě rozpady na čtyři top kvarky. Pozadí bylo tvořeno z více částí, přičemž tou první byl již zmiňovaný proces produkce top-antitop. Dále bylo využito takzvaných *multijet* eventů z kategorie *JZ3-JZ7*[1], které jsou tvořeny především procesy produkujícími lehké kvarky prostřednictvím silné interakce.

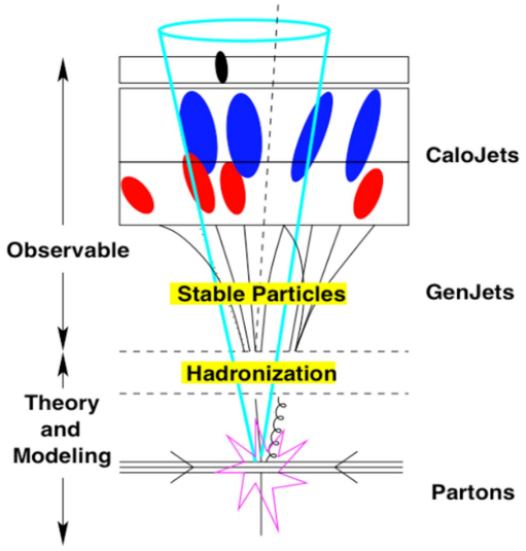
3.2 Analýza signálu z detektoru

Již jsme zavedli pojem jet, jakožto kolimovanou spršku částic a také pojem hadronizace. Jety hrají při analýze dat z detektoru klíčovou roli, neboť z nich lze získat údaje o původní podobě srážky detektoru a o částicích, které se před interakcí s detektorem již rozpadly. Schématické znázornění je k nahlédnutí jako obrázek 3.1. V nitru detektoru však dochází ke srážkám velmi početných shluků částic, takže vzniká velké množství produktů, které spolu dále interagují a snižují přesnost detektoru. Za nejvýraznější fenomén v této oblasti lze označit *Pile-up*.

Pile-up je efekt, který generuje v detektoru šum. Za šum označujeme v tomto případě značné množství srážek, které se odehrají simultánně s pozorovaným

[1] Při generaci MC eventů jsou tyto děleny do několika dílů označených JZ podle intervalů příčné hybnosti. Z nich pro nás byly jako pozadí relevantní právě části 3-7

eventem. Při srážkách na LHC kolidují dva protonové shluky, mající řádově 10^{11} protonů, proto dojde při srážce dvou shluků k několika různým srážkám proton-proton. Pile-up je tedy způsoben vysokou luminositou[2] LHC, kterou je však pro běžný provoz výhodnější udržovat. Vzácné eventy, které chceme pozorovat mají totiž velmi nízký účinný průřez a vysoká luminosita pak znamená vyšší efektivitu urychlovače. Navíc pomocí triggeru na samotném detektoru a také díky sofistikovaným algoritmům (například *Particle Flow Algorithm* [23]) je možné nechtěné eventy velmi dobře potlačovat.



Obrázek 3.1: Znázornění postupného vzniku jetů a jejich detekce - jedinou možností, jak se dobrat přesné podoby původní srážky je rekonstrukce jetu.[24]

3.3 Rekonstrukce jetů

Pro rekonstrukci jetů využíváme algoritmů, do kterých vstupují data z kalorimetrů i trackerů. Při využívání algoritmu je potřeba vzít v úvahu dvě důležité vlastnosti finální rekonstrukce:

- *Infra-red and collinear safety (IRC safety)* - O algoritmu řekneme, že je *infra-red safe*, pokud jeho výsledek není ovlivněn výskytem nízkoenergetických částic či fotonů[3] ve studovaném procesu. Analogicky považujeme algoritmus za *collinear safe*, pokud jeho výsledek neovlivní kolinearita jetů, tedy poskytuje správné výsledky i v případě, že jety zdánlivě splývají.

- *Velikost jetů* - Větší jety umožní zahrnout větší množství hadronizovaných částic, díky čemuž lze lépe dopočítat původní hmotnost a energii. Užší jety jsou úspěšné v redukci negativních vlivů, které způsobí například pile-up, díky čemuž předchází nadhodnocování hmotnosti a energie.

Nejčastěji využívanými typy algoritmů jsou *Fixed cone* a *Sequential recombination*

[2]Fyzikální veličina vyjadřující počet interakcí za jednotku času.

[3]V anglické odborné literatuře je pro tyto nízkoenergetické částice využíván pojem *soft radiation*.

3.3.1 Fixed cone algorithm

Jak název napovídá, základní myšlenkou je představa jetu jako spršky částic ohraničené pevně daným kuželem. Ve zjednodušené podobě funguje algoritmus následovně [24]:

1. Nalezneme nejtvrší objekt (částice), který využijeme jako *seed*.
2. Ve směru hybnosti zvoleného objektu konstruujeme kužel fixní velikosti.
3. Kužel označíme jako jet a vyřadíme veškeré částice, které obsahuje z další analýzy.
4. Opakujeme dokud nezachytíme veškeré částice.

Dnes se však už od fixed cone algoritmů ustoupilo, vzhledem k tomu, že většina z nich není IRC safe, ač byly původně preferovány pro menší výpočetní náročnost [24]. Příklady mohou být třeba *SIScone*[25] či *IC-SM*.

3.3.2 Sequential recombination algorithm

Tento typ algoritmu využívá pro klasifikaci metriky d_{ij} a d_{Bi}, přičemž d_{ij} je metrika popisující vzdálenost objektů i a j, definovaná jako:

$$d_{ij} = \min(p_{Ti}^a, p_{Ti}^a)\frac{R_{ij}^2}{R^2}, \tag{3.1}$$

kde p_T jsou příčné hybnosti a R_{ij} radiální vzdálenost objektů i a j, definovaná výrazem $R_{ij}^2 = (\eta_i - \eta_j)^2 + (\varphi_i - \varphi_j)^2$[4]. d_{Bi} je metrika popisující vzdálenost objektu i a svazku, definovaná jako:

$$d_{Bi} = p_{Ti}^a. \tag{3.2}$$

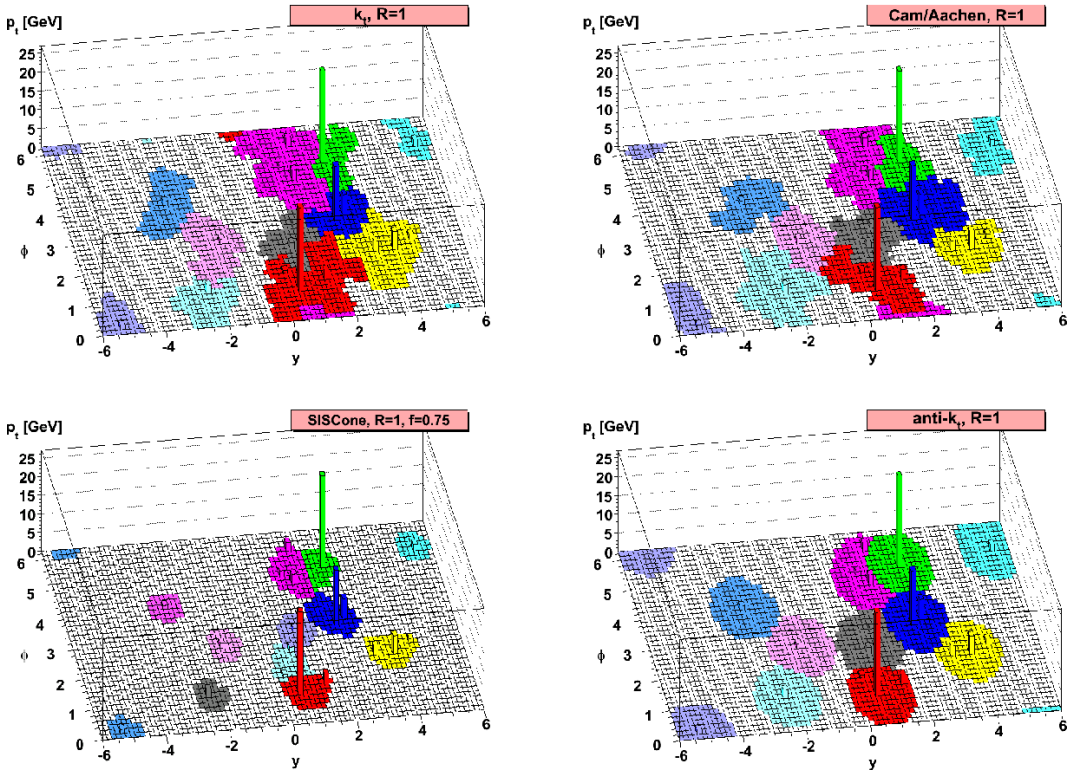
Navíc je zde zavedený parametr a, pro jehož speciální hodnoty, rozlišujeme různé algoritmy. Pro $a = 2$ se jedná o k_t algoritmus, $a = 0$ odpovídá algoritmu *Cambridge/Aachen* a pro $a = -2$ je to algoritmus *anti-k_t*. Všechny algoritmy tohoto typu však mají společnou kostru, která ve zjednodušené podobě vypadá následovně [26]:

1. Nalezneme nejtvrdší objekt, který budeme nazývat objekt i.
2. Pomocí vztahu 3.1 vypočteme vzdálenost objektu i a objektu j.
3. Pokud je $d_{ij} < d_{Bi}$ přičleníme objekt j k objektu i a vracíme se ke kroku 2, jinak objekt i označíme za jet a vyčleníme jej z další analýzy.
4. Opakujeme, dokud nejsou veškeré částice zahrnuty do jetů.

Dnes se využívá převážně algoritmus *anti-k_t*, který IRC safe. V naší analýze využíváme jetů s volbou parametru $R = 0{,}4$ a $R = 1{,}0$.

Souhrnně se uvedené algoritmy nazývají *jet clustering algoritmy*. Přílady využití vybraných typů jsou k nahlédnutí jako obrázek 3.2.

[4] η označuje pseudorapiditu částice a φ úhel jejího směru v rovině kolmé na rovinu urychlovače a rovnoběžné se směrem paprsku



Obrázek 3.2: Ukázka využití různých jet clustering algoritmů. Nahoře zleva k_t a *Cambridge/Aachen*, dole zleva *SIScone* a *anti-k_t*.[26]

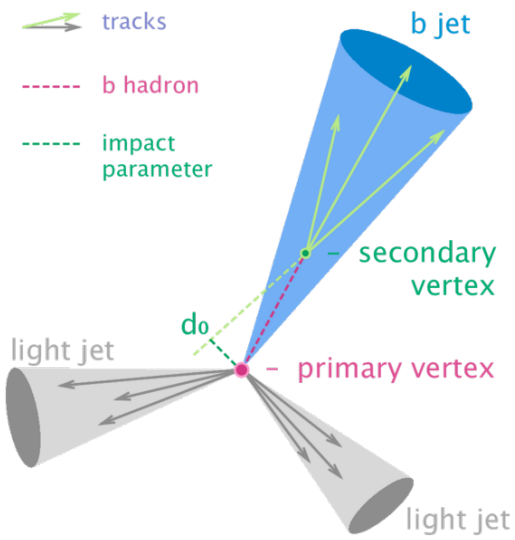
3.4 b-tagging, top-tagging

Tagging znamená označování jetů podle jejich původu, konkrétně podle vůně kvarku, ze kterého pochází. Metoda *b-tagging* se zaměřuje na jety pocházející z bottom kvarku. Tyto je možné idetnifikovat díky dvěma vlastnostem bottom kvarku.

- Po vzniku spoluvytváří v detektoru hadron, který před svým rozpadem urazí nenulovou vzdálenost, která je dostatečně dlouhá na to, aby z rozpadových produktů bylo možné poznat, že nevylétávají z primární kolize, ale z druhotného vrcholu, kde došlo k rozpadu bottom kvarku.
- Bottom kvark se rozpadá na mnohem lehčí částice, a proto vzniká širší jet, tvořený částicemi s vysokou hybností.

Tyto skutečnosti tvoří základní myšlenku b-taggingu, schematicky jsou navíc znázorněny jako obrázek 3.3.

Hlavní ideou top-taggingu je identifikace top kvarků s vysokou hybností (*boosted top*). Vysoká hybnost způsobuje silnou kolinearitu rozpadových produktů (bottom kvarku a bosonů W), přičemž při využití algoritmu *anti-k_t* s vysokou hodnotou poloměru lze pak tyto produkty sloučit do jednoho *large jetu*. Large jetu navíc přiřazujeme vlastnost *N-subjettines*, která udává počet jetů, obsažených v large jetu.[27]



Obrázek 3.3: Vizualizace základních principů b-taggingu.[28]

3.5 Trénovací proměnné

Jako trénovací proměnné označujeme veličiny, které přiřazujeme jednotlivým jetům jako vlastnosti a využíváme je jako input pro klasifikační model. V našem případě byl každý jet charakterizován 11 fyzikálními veličinami:

- První skupinu tvoří popis kinematiky samotného jetu či large jetu, jedná se o:
  - Příčnou hybnost jetu - p_T (respektive pro každou hybnost uvažujeme její logaritmus, čímž se sníží hodnoty inputu a díky tomu síť nastavuje nižší hodnoty parametrů)
  - Celkovou rekonstruovanou energii (taktéž uvažujeme logaritmus)
  - Celkovou rekonstruovanou hmotnost
  - Pseudorapiditu - η
  - Úhel směru jetu ve vertikální rovině ve směru paprsku - φ (Tento je však přepočten tak, že jetu s nejvyšší hybností přiřadíme hodnotu $\varphi = 0$ a směr ostatních vyjádříme relativně vzhledem k tomuto jetu. Díky tomu jsou data odpoutána od absolutní geometrie dané urychlovačem, což síti usnadní trénink, neboť se nemusí učit preferovat relativní polohu jetů oproti absolutní.)
- Druhou skupinu tvoří trojice veličin související s chybějící příčnou energií:
  - Chybějící příčná energie jetu
  - Úhel směru chybějící energie jetu
  - Celková energie všech objektů v eventu
- Výsledek z b-taggingu jetu
- Dvojice proměnných jakožto výsledek z top-taggingu jetu (pro jiné než large jety automaticky 0)

4. Strojové učení, Hluboké učení

Strojové učení (*Machine learning - ML*) je studium algoritmů založených na metodách statistické matematiky. Tyto algoritmy jsou schopny učit se, respektive generalizovat na základě poskytnutých dat a naučenou generalizaci aplikovat i na dosud netrénovaná data. *Hluboké učení* (*Deep learning - DL*) je podobor strojového učení zabývající se učením modelů tvořených hlubokými neuronovými sítěmi. Takové modely jsou schopny nejen generalizovat, ale vytvářet si na základě vstupních dat sady abstraktních charakteristických rysů, pomocí kterých se data učí. Hluboké učení vyniká při využití na vysokodimenzionální data (jako například obrázky) a při učení na mohutných datasetech. Algoritmy ML naopak dodnes slaví úspěch při tréninku na malých datasetech, které nejsou dostačující pro kvalitní trénink hlubokých sítí.

4.1 Základy strojového učení

Zaměřme se nyní na základní pojmy a principy, které tvoří základy ML i DL. Ve vší obecnosti nastiňme cíle tréninku modelů a metody, kterými jich lze dosahovat. Následovat bude jejich podrobný popis.

4.1.1 Trénink modelu

Pro trénink algoritmů ML či hlubokých neuronových sítí je nutné mít *dataset* dostačující kvality a mohutnosti. Naše práce je zaměřena čistě na takzvaný *supervised learning*, neboli učení s učitelem. Pro takový druh učení využíváme dataset, obsahující nejen samotná data mířící na vstup modelu, ale k nim i spárovaný očekávaný správný výstup z modelu (*target output*, tedy zmiňovaná ground-truth).

Dataset je ještě před tréninkem rozdělen na tři části, a to *tréninkový set*, *validační set* a *testovací set*.

Z tohoto rozdělení vyplývá struktura tréninku. Ten je rozdělen na několik menších částí, nazývaných *epochy*. V průběhu jedné epochy provede model výpočet na celém tréninkovém datasetu a proces opakuje po zadaný počet epoch. Epochy lze dále dělit na jednotlivé kroky. V každém kroku provede model *forward pass*, tedy výpočet na menší části vstupních dat (*batch*) a vypočtený output porovná s ground-truth. Na výsledek srovnání pak model reaguje algoritmem zpětného šíření chyby (*backpropagation*). Jedná se o optimalizační algoritmus, který spočívá ve výpočtu parciálních derivací *loss funkce* podle trénovatelných parametrů modelu (formálně tedy výpočtu gradientu) a následné úpravě těchto parametrů.

Pro natrénovaný model využijeme validační dataset jako zcela nový vstup. Poté, co model poskytne output lze na základě různých *metrik* zlepšovat výkon modelu úpravou *hyperparametrů* (například počet vrstev, velikost batchů, *aktivační funkce*, počet epoch, *regularizace*,...).

V momentě, kdy považujeme model za hotový, poskytneme mu na vstup testovací dataset a na základě metrik zjistíme finální úspěšnost modelu.[29],[30]

V průběhu tréninku budeme čelit dvěma zásadním problémům, a to:

- *Underfitting* – Nedostatečné využití kapacity modelu, způsobené nedokonalým učením a nedostatečným přizpůsobením tréninkovým datům. Proti underfittingu bojujeme optimalizačními metodami.
- *Overfitting* – Přílišné přizpůsobení tréninkovým datům, ztráta schopnosti modelu produkovat relevantní output pro jiné datasety než tréninkové. Proti overfittingu bojujeme regularizačními technikami.

Za dvě hlavní úlohy strojového učení jsou běžně považovány:

- *Regrese* – Cílem regrese je předpovědět pro daný input jeho příslušný target output.
- *Klasifikace* – Cílem klasifikace je pro fixní počet tříd buďto přiřadit vstupu příslušnost do některé ze tříd a nebo přiřadit ke vstupu celou distribuci hustoty pravděpodobnosti, nesoucí informaci o příslušnosti k jednotlivým třídám. To znamená, že pro každé vstupní dato (v našem případě jet) je výstupem rozdělení, které vyjadřuje pravděpodobnost s níž toto dato spadá do každé z uvažovaných tříd. Pro případ binární klasifikace se rozdělení zjednoduší na dvě hodnoty, tedy na pravděpodobnost P, že se jedná o signál a pravděpodobnost, že se jedná o pozadí $(1 - P)$. Přiřazovaní dat ke třídě poté provedeme jednoduše nastavením pravděpodobnostního prahu P_p. Každé dato, které bude splňovat $P > P_p$ pak označíme jako signál.

4.1.2 Loss funkce

Loss funkce je error funkce, které umožňuje trénink modelů strojového učení. V prvním přiblížení loss funkce popisuje neshodu výstupu modelu s target outputem. Podstatou tréninku je tedy zpravidla minimalizace této funkce.

Pravděpodobnost a teorie informace

Pravděpodobnost hraje v ML obecně velmi důležitou roli. Zadefinujme si nyní (podle [31]) několik stěžejních pojmů potřebných pro výstavbu základů ML:

- *Střední hodnota* – Střední hodnota funkce *f(x)* vzhledem k pravděpodobnostní distribuci *P(x)* je průměrnou hodnotou, které nabude funkce *f(x)* při náhodném výběru x podle *P(x)*. Pro diskrétní proměnné je dána výrazem:

$$\mathbb{E}_{x\sim P}[f(x)] = \sum_x P(x)f(x), \tag{4.1}$$

  zatímco pro spojité proměnné je určena jako:

$$\mathbb{E}_{x\sim P}[f(x)] = \int p(x)f(x)\mathrm{d}x \tag{4.2}$$

- *Rozptyl (variance)* – Variance je střední hodnota kvadrátu odchylky od střední hodnoty funkce *f(x)*. Určuje tedy míru, kterou se liší hodnoty funkce *f(x)* od její střední hodnoty, vybíráme- li x náhodně podle ditribuce *P(x)*:

$$\mathrm{Var}(f(x)) = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2] \tag{4.3}$$

Teorie informace je odvětví matematiky, jež se zabývá kvantifikací množství informace v signálech. Jako jeden ze zcela základních pojmů zavádí takzvanou *self-information*[1] (česky self-informace či informanční obsah):

$$I(x) \coloneqq -\ln P(x). \tag{4.4}$$

- *Entropie* – Entropii lze neformálně chápat jako průměrné *překvapení* při výběru x z *P(x)* či alternativně jako veličinu kvantifikující *nejistotu* v celé distribuci *P(x)*. Je definována jako střední hodnota self-informace pro pravděpobnostní distribuci *P(x)*:

$$H(P) \coloneqq \mathbb{E}_{x\sim P}[I(x)] = -\mathbb{E}_{x\sim P}[\ln P(x)]. \tag{4.5}$$

  Entropie distribucí, které jsou téměř deterministické se blíží nule, naopak, maximální entropie dosáhnou distribuce *rovnoměrné*.

- *Křížová entropie (cross-entropy)* – Jak název napovídá, jedná se o veličinu velmi blízkou entropii. Uvažujme dvě pravděpodobnostní distribuce pro stejnou náhodnou proměnnou *P(x)* a *Q(x)*. *Q(x)* je *očekávaná* distribuce, ale ve skutečnosti se informace řídí podle distribuce *P(x)*. Neformálně ji tedy lze chápat jako průměrné překvapení z očekávané distibuce *Q(x)*, když skutečnou distribucí je *P(x)*. Cross-entropy definujeme jako:

$$H(P,Q) \coloneqq -\mathbb{E}_{x\sim P}[\ln Q(x)] \tag{4.6}$$

- *KL divergence* – *Kullback-Leibnerova divergence*, někdy také *relativní entropie*, je veličina, kterou lze *porovnat* dvě různé pravděpodobnostní distribuce. Definujeme ji jako:

$$D_{KL}(P||Q) \coloneqq H(P,Q) - H(P) = \mathbb{E}_{x\sim P}[\ln P(x) - \ln Q(x)] \tag{4.7}$$

Metoda maximální věrohodnosti (Maximum likelihood estimation)

Nejčastějším principem, o který se opíráme při tvorbě loss funkcí je právě princip maximální věrohodnosti (MLE). Jeho podstatou je maximalizace takzvané *věrohodnostní funkce (likelihood function)*.

[1] Byla zavedena tak, aby splňovala trojici základních požadavků:

1. Měla by být nulová pro události s pravděpodobností 1.
2. Méně pravděpodobné události nesou větší informačnín obsah.
3. Informace nesené nezávislými jevy jsou aditivní.

Mějme datový soubor $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, ...)$, řídící se pravděpodobnostní distribucí $P(\mathbf{x})$. Dále uvažme pravděpodobnostní distribuci $Q(\mathbf{x}; \mathbf{w})$, kterou poskytuje trénovaný model a s jejíž pomocí modeluje trénovaná data. Nyní lze zavést věrohodnostní funkci ve tvaru:

$$L(\mathbf{w}) = Q(\mathbf{X}; \mathbf{w}) = \prod_{i=1}^{N} Q(\mathbf{x_i}, \mathbf{w})$$

Zobecněme nyní úlohu pro případ, že cílem tréninku je, aby náš model predikoval target oputput $\mathbf{t}$. Poté je pravděpodobnostní distribuce vystupující z modelu $Q(\mathbf{t}|\mathbf{X}; \mathbf{w})$. Maximální věrohodný odhad poté určíme následovně (maximalizujeme či minimalizujeme vždy vzhledem k $\mathbf{w}$):

$$\begin{aligned}
\mathbf{w_{MLE}} &= \text{argmax} L(\mathbf{w}) = \text{argmax} \prod_{i=1}^{N} Q(t_i|\mathbf{x_i}; \mathbf{w}) \\
&= \text{argmin} \sum_{i=1}^{N} -\text{log} Q(t_i|\mathbf{x_i}; \mathbf{w}) = \text{argmin} \mathbb{E}_{(\mathbf{x},t)\sim P} - [\text{log} Q(t|\mathbf{x}; \mathbf{w})] \\
&= \text{argmin} H(P(\mathbf{x},\mathbf{t})|Q(t|\mathbf{x}; \mathbf{w})) = \text{argmin} D_{KL}(P(\mathbf{x},\mathbf{t})||Q(t|\mathbf{x}; \mathbf{w}))
\end{aligned}$$

Výslednou loss funkci nazýváme *negative log likelihood*, někdy také cross-entropy či KL divergence. Za povšimnutí stojí, že z MLE úpravami snadno plyne KL divergence, která je vhodným nástrojem pro porovnávání pravděpodobnostních distribucí. Podstatou loss funkce je hodnotit odlišnost distribuce poskytnuté modelem od reálné distribuce dat a díky minimalizaci tohoto rozdílu se model zlepšuje – probíhá trénink.[31],[32]

4.1.3 Optimalizace

Optimalizací rozumíme proces minimalizace loss funkce prostřednicvím různých algoritmů, které upravují parametry modelu. Tyto algoritmy jsou nejčastěji založeny na metodě *gradient descent*.

Gradient descent

Uvažujme model s libovolnou loss funkcí $L(\mathbf{w})$, přičemž je naším cílem tuto funkci minimalizovat. Minimalizace je prováděna iterativně postupnou aktualizací parametru modelu. Algoritmus gradient descent definuje aktualizaci parametrů jako:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} L(\mathbf{w}), \tag{4.8}$$

kde koeficient α nazveme *learning rate*. Learning rate je hyperparametrem, který ovlivňuje velikost kroku algoritmu při každé iteraci a ovlivňuje tedy míru aktualizace parametrů modelu. Obecně rozlišujeme tři přístupy k výpočtu gradientu loss funkce:

- *Standard gradient descent* – K výpočtu gradientu jsou využívána všechna data v dostupném vzorku. Jedná se o výpočetně náročný způsob.
- *Stochastic gradient descent (SGD)* – Gradient je v podstatě odhadnut výpočtem z jednoho náhodně vybraného data. Takový postup je výpočetně velmi nenáročný, ale mnohem pomaleji konverguje.

- *Batch SGD* – Střední cesta mezi zmíněnými přístupy. Ze vzorku je náhodně vybrána podskupina prvků (batch) z nichž je následně gradient určen.

Momentum

Velmi často je možné gradient descent zefektivnit modifikací aktualizace parametrů 4.8 do podoby 4.9. Oproti původní verzi má nová rovnice navíc *momentum* neboli hybnost či setrvačnost. Klasický gradient descent totiž neefektivně hledá minimum loss funkcí, které mají tvar podlouhlého údolí. Představíme-li si jednotlivé kroky algoritmu hledání minima, budou značně ovlivněny tvarem údolí. V jednotlivých krocích bude algoritmus skákat ze stěny na stěnu, ale samotný postup dolů údolím bude pomalý. Momentum však umožňuje rozpoznat i pozvolný trend klesání na dno pomyslného údolí, jeho vliv umocnit a naopak omezit oscilace.

$$\begin{aligned} &1) \qquad \mathbf{z} \leftarrow \beta\mathbf{z} + \alpha\nabla_{\mathbf{w}} L(\mathbf{w}) \\ &2) \qquad \mathbf{w} \leftarrow \mathbf{w} - \alpha\mathbf{z} \end{aligned} \tag{4.9}$$

Zde hyperparametr β určuje míru vlivu momenta na trénink. Efektivně tedy momentum gradient descent vylepšuje o krátkodobou paměť a algoritmus je schopen do aktualizace zohlednit velikost předchozího kroku.[33]

Learning rate

Learning rate byl zadefinován jako pojem v rovnici 4.8. Vůbec nejméně efektivní způsob jakým tento hyperparametr nastavit je volbou konstanty. Naopak variabilní learning rate se prokazuje jako úspěšnější a efektivnější při hledání minima loss funkce. Typicky variabilní learning rate nastavujeme dvěma způsoby:

- *Learning rate scheduling* – Jedná se o proces, kdy dochází ke změně learning rate v průběhu tréninku, tedy skutečně jakési rozvržení tohoto hyperparametru. Typicky se jedná o *decay*, tedy postupné ubývání learning rate. Označme nyní celkový počet iterací v tréninku jako N a písmenem i indexujme pořadové číslo iterace. Nejčastěji se pak využívá následujících metod:
  - Linear decay: $\alpha_i = \alpha_0 \left(1 - \frac{i}{N}\right)$
  - Inversed square root decay: $\alpha_i = \alpha_0 \frac{1}{\sqrt{i}}$
  - Exponencial decay: $\alpha_i = \alpha_0 C^i$
  - Cosine decay: $\alpha_i = \frac{\alpha_0}{2} \left(1 + \cos(\frac{\pi i}{N})\right)$

  Přičemž α_0 je počáteční hodnota learning rate a C volitelná konstanta
- *Adaptivní learning rate* – Předchozí metoda opět závisí na volbě hyperparametru uživatelem, naproti tomu adaptivní learning rate je variován implicitně optimalizačním algoritmem. S adaptivním learning rate pracuje například algoritmus *Adam*.

Adam (Adaptive Moment Estimation)

Jedním ze v současnosti nejpoužívanějších optimalizačních algoritmů je Adam. Je založen na batch SGD, využívá momentum prvního i druhého řádu a má adaptivní learning rate. Kombinuje tedy většinu výše uvedených pokročilých metod. Níže uvádíme ve zjednodušené formě jeho průběh. [34] Zde využíváme dříve zavedeného značení, tedy α je learning rate, β_1,β_2 jsou koeficienty pro momentum, i je pořadové číslo iterace v epoše, m je počet dat v batchi a navíc zde je konstanta ε[2], která zajišťuje dělení nenulovým číslem a také λ neboli penalizační koeficient L^2 regularizace (podrobněji v 4.1.5). Na počátku uvažujeme $i \leftarrow 0; s \leftarrow 0; r \leftarrow 0$ a dále:

$$
\begin{aligned}
&1) \quad \mathbf{g} \leftarrow \frac{1}{m} \sum^{batch} \nabla_{\mathbf{w}} L(\mathbf{w}) \\
&2) \quad i \leftarrow i + 1 \\
&3) \quad \mathbf{s} \leftarrow \beta_1 \mathbf{s} + (1 - \beta_1)\mathbf{g} \\
&4) \quad \mathbf{r} \leftarrow \beta_2 \mathbf{r} + (1 - \beta_1)\mathbf{g}^2 \\
&5) \quad \mathbf{s} \leftarrow \frac{\mathbf{s}}{1 - \beta_1^i} \\
&6) \quad \mathbf{r} \leftarrow \frac{\mathbf{r}}{1 - \beta_2^i} \\
&7) \quad \mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{\sqrt{\mathbf{r}} + \varepsilon}\mathbf{s}
\end{aligned}
\tag{4.10}
$$

Přesto, že Adam využívá adaptivní learning rate, ukazuje se, že je výhodné využít i metody learning rate scheduling. [30] Takto bude implementována optimalizace i pro náš model.

Gradient clipping

Gradient clipping je metoda eliminující *exploding gradient*. Tento problém se vyskytuje hlavně při tréninku hlubokých neuronových sítí. V případě, že gradient loss funkce podle parametrů modelu nabude vysokých hodnot (často právě u hlubokých sítí s parametry vyššími než 1, což zapříčiní exponenciální růst gradientu) dochází ke značné komplikaci numerického řešení, které následkem dokonce často vůbec nezkonverguje. Metoda Gradient clipping omezuje velikost gradientu $\mathbf{g}$ na nastavitelnou hodnotu c:

$$
\mathbf{g} \leftarrow \begin{cases} \mathbf{g} & \text{pro} \quad |\mathbf{g}| \leq c \\ c\frac{\mathbf{g}}{|\mathbf{g}|} & \text{pro} \quad |\mathbf{g}| > c \end{cases}
\tag{4.11}
$$

4.1.4 Aktivační funkce

Jak již bylo naznačeno, forward pass je označení pro proces, kdy se z dat na vstupu do sítě vypočítává postupně přes jednotlivé vrstvy kýžený výstup. Obecně neuronové sítě s daty provádí dvojici operací. Tou první je maticové násobení a jedná se tedy o lineární operaci, tou druhou jsou pak operace nelineární a jsou zprostředkovány *aktivačními funkcemi*[35]. Každý neuron má přiřazenu aktivační funkci, která v závislosti na jeho parametrech a konkrétním vstupu určí výstup z daného neuronu, který pak postupuje dál sítí. Právě díky nelineárním aktivačním

[2]Defaultní hodnoty parametrů jsou: $\alpha = 0{,}001; \beta_1 = 0{,}9; \beta_2 = 0{,}999; \varepsilon = 10^{-8}$

funkcím bylo umožněno neuronovým sítím řešit složité nelineární problémy. Podle *Universal approximation theorem* [36], tedy podle věty o univerzální aproximaci, je možno aproximovat libovolnou spojitou funkci na kompaktní podmnožině $\mathbb{R}^N$ s libovolnou přesností pomocí standartní neuronové sítě *MLP*(4.2.2) s jednou skrytou vrstvou o konečném počtu neuronů.

- *Sigmoid* – Nelineární aktivační funkce s oborem hodnot (0;1). Typicky se využívá na poslední vrstvě při binární klasifikaci (vstupem je skalár a výstupem je příslušnost k jedné z dvou tříd). Při tréninku hlubokých sítí a nadměrném využití sigmoidy dochází k patologickému jevu jménem *vanishing gradient*, který může trénink výrazně zpomalit, či úplně paralyzovat optimalizační algoritmus.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.12}$$

- *Hyperbolický tangens* – Jedná se o jednu z prvních využívaných aktivačních funkcí. Taktéž se typicky využívá pro binární klasifikaci a podléhá vanishing gradient problému.

$$\text{tgh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4.13}$$

- *ReLU* – Dodnes jednou z nejvyužívanějších aktivační funkcí je ReLU (Rectified Linear Unit). Je velmi rychlá jak při forward pass výpočtu tak na při derivaci a následném zpětném šíření chyby. Pro velmi jednoduché regrese může být efektivnější volbou hyperbolický tangens, ale pro hluboké architektury je jednou z nejefektivnějších aktivací.

$$\text{ReLU}(x) = \max(0,x) \tag{4.14}$$

- *Softmax* – Jedná se v podstatě o zobecnění sigmoidy. Typicky se využívá jako aktivace na výstupní vrstvě při klasifikaci do více než dvou tříd (při binární se redukuje právě na sigmoid). Na vstupu je vektor a na výstupu pravděpodobnostní distribuce pro příslušnost do jednotlivých tříd.

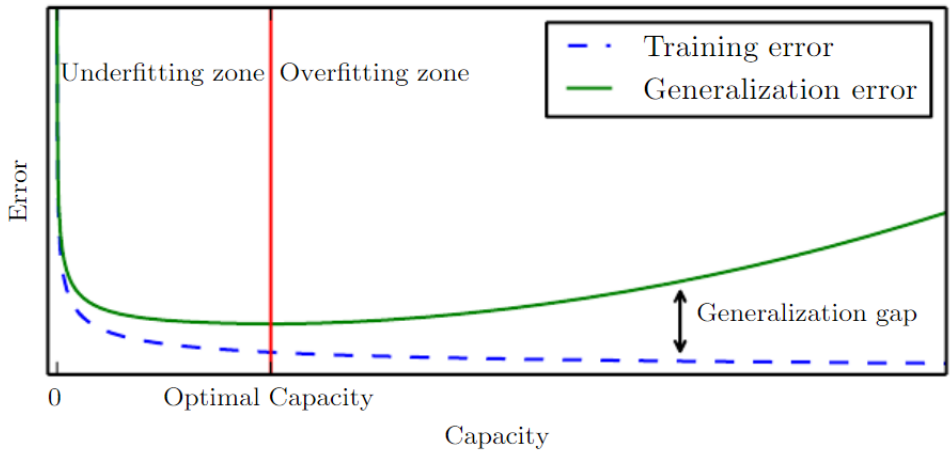
$$\text{softmax}(\mathbf{x}) = \frac{e^{\mathbf{x}}}{\sum_{j=1}^{n} e^{x_j}} \tag{4.15}$$

4.1.5 Regularizace

Již v úvodu byly zavedeny pojmy overfitting a underfitting. Vyhnout se oběma těmto nechtěným jevům představuje skutečnou výzvu při tréninku neuronových sítí. Hlavním cílem při tréninku je, aby byl model úspěšný v *generalizaci*, tedy aby úspěšně pracoval na nových datech. Tuto schopnost ověřujeme pomocí testovacího datasetu, který využíváme pouze k hodnocení. V průběhu tréninku na trénovacím datasetu snižujeme tréninkovou chybu. Při následném běhu na testovacím datasetu lze očekávat hodnotu generalizační chyby přinejlepším stejnou, spíše vyšší. Na obrázku 4.1 je znázorněna závislost trénovací i generalizační chyby na kapacitě modelu. Za úspěšný trénink považujeme takový, který jednak dostatečně minimalizuje trénikovou chybu a jednak minimalizuje rozdíl mezi tréninkovou a

generalizační chybou. Při underfittingu se tedy nepodaří splnit první požadavek a model nedokáže dosáhnout dostatečně nízké trénovací chyby. Naopak při overfittingu je tréninková chyba minimalizovaná, ale generalizační chyba roste. Velkou měrou lze overfitting a underfitting ovlivnit kapacitou modelu, tedy jeho schopností aproximovat širokou paletu funkcí. Kapacita však nezáleží pouze na architektuře, nýbrž například i na průběhu optimalizace.

Výkonnost a úspěšnost modelů významně vzrůstá s jejich rostoucí kapacitou. Abychom se však u mohutných modelů ubránili overfittingu, který by byl za normálních okolností nevyhnutelný, využíváme různé druhy regularizace [31],[32]:



Obrázek 4.1: Typický průběh závislosti chybovosti modelu na jeho kapacitě. [31]

- *Early stopping* – Nejjednodušší z metod regularizace je nasnadě. V překladu se jedná o včasné zastavení. Podstatou je tedy ukončit trénink modelu v okamžiku, kdy je nejmenší rozdíl mezi trénovací a generalizační chybou.
- L^2 *regularizace* – Mechanismus zabudovaný do loss funkce regularizovaného modelu, který při tréninku preferuje modely s menšími hodnotami parametrů (vah). Dosahuje toho penalizací modelů s vysokými vahami. Upravená loss funkce[3] vypadá například následovně:

$$\tilde{L}(\mathbf{w}) = L(\mathbf{w}) + \frac{\lambda}{2}||\mathbf{w}||_2^2,$$

  kde λ je míra regularizace (nastavitelný hyperparametr). Ve zjednodušeném pohledu tato forma regularizace potlačuje aktualizace parametrů a snaží se je udržet v původní podobě. Efektivně tím brání modelu příliš se přizpůsobit trénovacímu datasetu a jediné aktualizace, které přetrvají jsou ty založené na opakujících se motivech (obecná pravidla).

[3]Spodní index u normy $||\mathbf{w}||_2$ naznačuje, že se jedná o Euklidovskou, neboli L^2 normu. Občas se využívá také L^1 regularizace, zavedená analogicky (tato je však nevhodná pro neuronovvé sítě).

- *Dataset augmentation* – Pro určité typy dat lze aplikovat umělé rozšíření datasetu. Rozšíření probíhá jako drobná úprava stávajících dat (např. pro obrázky se jedná o oříznutí, maskování, otáčení...) a jejich následné vmíchání do původního datasetu. Díky zvětšení mohutnosti datasetu efektivně snižujeme kapacitu modelu, což je prevence overfittingu.

- *Ensembling* – Další z jednoduchých technik, která je založena na tréninku většího množství modelů a následném kombinování (typicky průměrování) jejich výsledků. Výpočetně se však jedná o jednu z náročnějších metod.

- *Droupout* – Metoda dropout pro každý neuron v síti nezávisle rozhodne o tom, zda má být ponechán, či má být hodnota jeho výstupu nastavena na 0. Rozhodování probíhá pokaždé s pravděpodobností p, kterou nazýváme *dropout rate* a je nastavitelným hyperparametrem. Opět díky tomuto mechanismu dojde k potlačení naučených nuancí trénovacího datasetu a zachování obecně platných principů.

- *Label smoothing* – Při klasifikaci typicky nazýváme náš target output slovem label. Takový target output je však typu one-hot, tedy pro třídu do které dato opravdu náleží má hodnotu 1 a pro všechny ostatní třídy 0. Při dlouhém tréninku totiž loss funkce typu negative log likelihood nutí model k overfittingu, protože vynucuje aktualizaci parametrů dokud není klasifikační distribuce na výstupu přesně ve tvaru target outputu (tedy one-hot), čehož však téměř nikdy nedosáhneme. Tento negativní vliv lze zmírnit, když nastavíme parametr α a target output modifikujeme do podoby, kdy správná třída bude mít hodnotu $(1 - \alpha)$ a zbylé třídy $\frac{\alpha}{K-1}$, kde K je počet tříd.

4.1.6 Evaluace

Evaluace modelů strojového učení proces hodnocení výkonnosti modelu. K tomuto účelu využíváme druhý druh error funkcí běžně zaváděných v ML, tedy *metriky*. Mezi loss funkcemi a metrikami lze spatřit dva hlavní rozdíly, a to, že metriky nemusí být diferencovatelné a standardně jejich výpočet probíhá přes vysoké množství batchů.

	Target positive	Target negative
Predicted positive	True positive (TP)	False positive (FP)
Predicted negative	False negative (FN)	True negative (TN)

Obrázek 4.2: Obecná podoba 2×2 confusion matrix (tedy chybové matice pro binární klasifikaci).

V naší práci se budeme zabývat pouze binární klasifikací (na top a non-top jety), a proto uvedeme metriky pouze pro tuto problematiku. Zavedení konkrétních metrik nám usnadní pochopení pojmu *confusion matrix* neboli chybová matice. Její podoba pro binární klasifikací je přiložena jako obrázek 4.2.

Ve sloupcích chybové matice jsou uspořádány hodnoty podle své skutečné příslušnosti, tedy zda jsou opravdu top (positive) či non-top (negative) v našem případě binární klasifikace. V řádcích jsou poté seskupeny podle predikce, kterou přiřadil model. V každém ze čtyř políček je dvouslovné označení. První slovo hodnotí, zda byla predikce modelu úspěšná, pokud je *True*, shodl se output modelu s target outputem a naopak. Druhé slovo vyjadřuje predikci modelu. Konkrétně tedy na příkladu *False positive* – jedná se o dato označené modelem za positive (top jet), což se však neshoduje s target outputem a jedná se tedy o špatně klasifikovaný negative (non-top jet).

Definujme nyní několik metrik:

- *Accuracy* – Vyjadřuje poměr úspěšných predikcí (True) ku všem učiněným predikcím. Lze na ni také nahlížet jako na pravděpodobnost správné predikce.
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.16}$$

- *Precision* – Vyjadřuje poměr úspěšných pozitivních predikcí ku všem datům označeným predikcí jako positive. Lze ji tedy chápat také jako čistotu skupiny eventů označených top.
$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.17}$$

- *Recall* – Má význam podílu, který tvoří úspěšné pozitivní predikce ze skupiny všech dat, které jsou skutečně positive. Jedná se tedy o kvantifikaci záchytu top jetů sítí. Někdy se také nazývá účinnost (efficiency).
$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.18}$$

- *Precision at Recall* - Metriky precision a recall mají v jistém slova smyslu antagonistický vztah, jinými slovy, potlačením jedné lze dosáhnout maxima druhé. Při evaluaci je nutné hledat vzájemnou rovnováhu, což zprostředkovává třeba tato metrika. Nejprve je zvolena hodnota recall. Model si pak upraví pravděpodobnostní práh (pro zbylé metriky defaultně 50%, zaveden v 4.1.1) pro přijetí data jako positive tak, aby celkově dosáhl požadovaného recall. Pro takto získanou predikci je následně vypočtena hodnota precision.

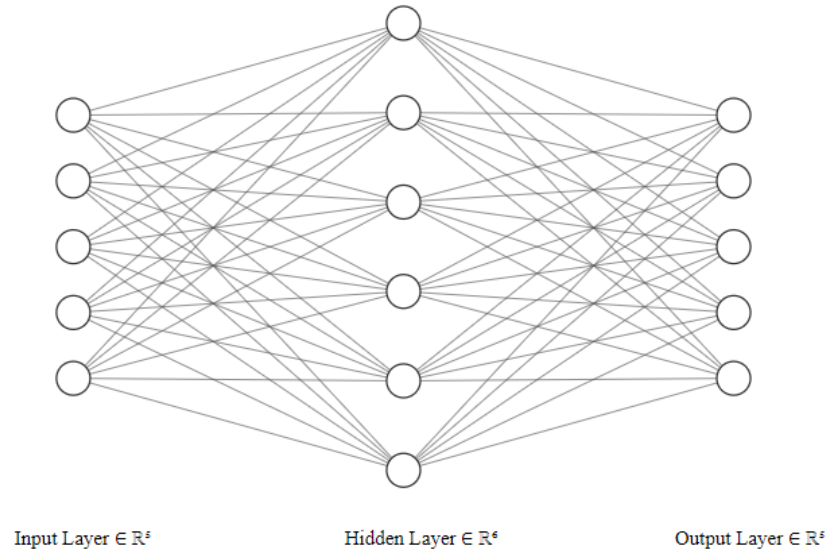
- *AUC* – Poslední z metrik, které zmíníme je Area Under Curve. Jedná se skutečně o velikost plochy pod ROC křivkou, kterou zjistíme integrací. ROC (Reciever Operating Charakteristic) křivka je vynesená závislost metriky recall na metrice *False positive rate*, zavedené jako: $\frac{FP}{FP+FN}$. Analogicky jako u precision at recall je zde upravován pravděpodobnostní práh pro označení data za positive pro dosažení hodnot false positive rate a následně pro každou takovou predikci dopočten recall.

4.2 Cesta k architektuře Transformer

Se znalostí fundamentálních principů tréninku neuronových sítí (NN) lze nyní popsat různé typy architektur neuronových sítí od těch nejjednodušších až k síti *Transformer*, kterou budeme využívat v naší práci.

4.2.1 Perceptron, Fully connected NN, Feedforward NN

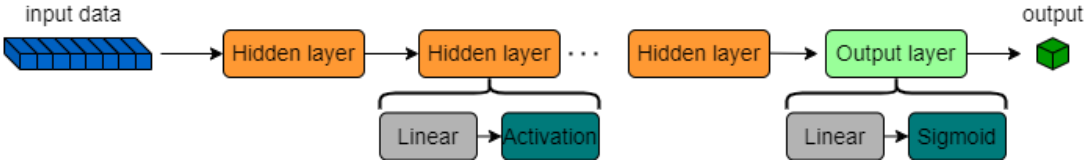
Vůbec nejjednodušším předchůdcem neuronových sítí je perceptron, který je tvořen v podstatě jediným neuronem. Jednalo se o lineární binární klasifikátor, který byl často uspořádán do takzvaného single layer perceptronu, tedy jediné vrstvy neuronů. Perceptron je však velmi triviální prostředek ML a pojí se s ním řada nedostatků. Kupříkladu není schopen aproximovat XOR funkci, pro neseparovatelná data nekonverguje a v momentě, kdy najde řešení, není schopen dále aktualizovat své parametry. Dalším významným krokem je příchod *Fully connected NN*(FC), respektive fully connected vrstvy (někdy *dense vrstva*). Jedná se o dvě vrstvy neuronů, přičemž každý neuron první vrstvy je propojen s každým neuronem druhé vrstvy. Nyní lze již snadno zadefinovat pojem *Feedforward NN* (někdy česky *dopředná síť)*. To je taková síť, která obsahuje jednu nebo více *skrytých vrstev* a informace v ní plynou pouze dopředně (narozdíl od například rekurentních sítí). Skrytá vrstva (*hidden layer*) je typicky dense vrstva, která je hustě spojena s předchozí a následující vrstvou, avšak z vnějšku do ní nevstupuje žádná informace. Podoba takové sítě je k nahlédnutí na obrázku 4.3. Díky architektuře typu feedforward se dostáváme k prvním modelům hlubokého učení. Zároveň je dobré poznamenat, že pro veškeré vrstvy je typicky uvažována nelineární aktivační funkce, popřípadě regularizační metoda. [31]



Obrázek 4.3: Zjednodušené vyobrazení fully connected sítě, která zároveň spadá do kategorie feedforward NN.

4.2.2 Multi-layer perceptron (MLP)

MLP je neuronová síť typu feedforward, která obsahuje alespoň jednu skrytou vrstvu. Pokud obsahuje více než jednu skrytou vrstvu, jedná se o nejjednodušší architekturu schopnou hlubokého učení. Input vrstva záleží na povaze vstupních dat a output vrstva na target outputu. Skryté vrstvy jsou dále schopné extrahovat charakteristické rysy dat nezávisle na rysech, které datům přiřazujeme my. Čím více skrytých neuronů je v architektuře zařazených, tím abstraktnější rysy je síť schopna v rámci hlubokého učení extrahovat. Díky těmto abstraktním rysům, vstupujícím do output vrstvy, dosahují neuronové sítě vyjímečných výsledků při porovnání s ostatními mechanismy strojového učení.



Obrázek 4.4: Schéma architektury MLP. Blok fully connected vrstav zakončených výstupní vrstvou s aktivací sigmoid.

Schopnost univerzální aproximace a s ní související zdatnost v řešení komplexních úloh pramení jednak z využití skrytých vrstev, ale hlavně z využití nelineárních aktivací na těchto vrstvách. Historicky byly pro MLP nejpopulárnějšími hyperbolický tangens a sigmoid. Dnes už jsou využívány novější aktivace (například ReLU a její modifikakce).

Uvažme nyní příklad nejjednoduššího MLP jako například na obrázku 4.3 a nastiňme alespoň pro tento nejjednoduší model ideu výpočtu výstupu.
Mějme na vstupu vektory $\mathbf{x}$, které přichází v batchi, tedy na vstupu je matice $\mathbb{X}$. Každá vrstva má trénovatelné parametry, a to matici vah $\mathbb{W}$ a vektor $\mathbf{b}$, tedy takzvaný *bias*. Prvky spjaté se skrytou vrstvou značíme písmenem h a prvky output vrstvy analogicky písmenem y. Výpočet probíhající v MLP má pak podobu:

$$\mathbb{H} = a_1(\mathbb{X}\mathbb{W}_h + \mathbf{b}_h) \rightarrow \mathbb{Y} = a_2(\mathbb{H}\mathbb{W}_y + \mathbf{b}_y), \tag{4.19}$$

kde a_1,a_2 jsou obecně různé aktivační funkce, přiřazené daným vrstvám.

4.2.3 Rekurentní sítě (RNN)

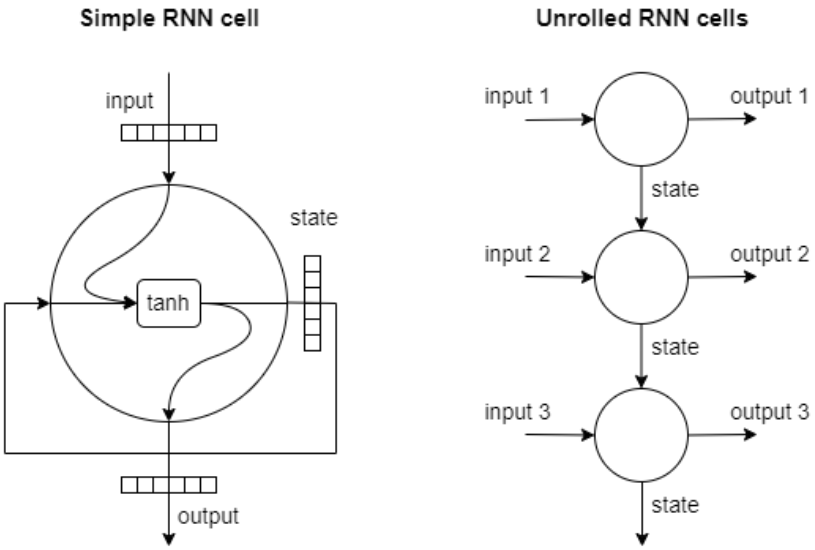
Rekurentní neuronová síť je obecně typ architektury, který je určen ke zpracovávání datových sekvencí a vymyká se třídě feedforward, neboť output z jednotlivých neuronů může ovlivňovat výpočty na sousedních neuronech v téže vrstvě. Výpočty a plynutí dat nejsou čistě dopředné, což je předpokladem k úspěšnému zpracování datových sekvencí. Za nejjednodušší typ rekurentní sítě považujeme takzvanou RNN buňku, popřípadě uspořádání takových buněk do vrstvy, viz obrázek 4.5.

RNN buňka sestává z fully connected vrstvy s aktivací typicky v podobě hyperbolické tangenty. Tato aktivace se využívá pro svůj omezený obor hodnot. RNN buňky jsou totiž při tréninku velmi citlivé na hodnoty gradientu a s neomezenou aktivací okamžitě nastává problém s exploding gradientem. Uvažujme nyní

nad libovolnou buňkou zapojenou v RNN řetězci[4]. Jako vstup dostává jednak samotné dato ze sekvence, ale také *stav* poskytnutý předcházející buňkou. Tato dvojice vstupů je konkatenována, projde FC a vystupuje z buňky jednak jako stav pro následující buňku a jednak jako samostatný output. Chceme-li postup popsat matematicky za pomoci zavedeného značení, probíhá výpočet:

$$\mathbf{h}^{(n)} = \tanh(\mathbb{U}\mathbf{h}^{(n-1)} + \mathbb{V}\mathbf{x}^{(n)} + \mathbf{b}),$$

kde $\mathbb{U}$,$\mathbb{V}$ jsou trénovatelné matice parametrů.



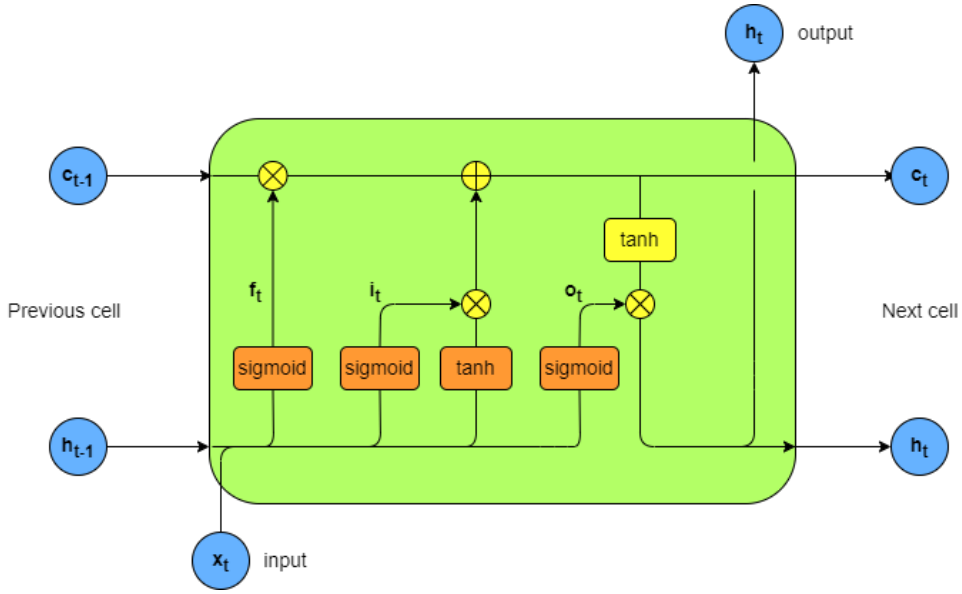
Obrázek 4.5: Vlevo jednoduchá RNN buňka a vpravo uspořádání takových buněk do vrstvy, která poté zpracovává sekvenci vstupních dat.

Bohužel rekurentní sítě v této podobě nefungují dobře, neboť nezvládají reflektovat informace o vzdálenějších prvcích frekvence. Při tréninku totiž trpí na vanishing gradient, kvůli omezenému oboru hodnot hyperbolické tangenty, která nás chrání před exploding gradientem. Narážíme na takzvaný *challenge of long-term dependencies*, který vyřeší až architektura *LSTM*.

LSTM (Long short-term memory)

Pro odstranění problému s vanishing gradientem byla navržena LSTM buňka, pro níž byl navržen *konstantní error flow*, tedy paměťová linka c, po níž bude derivace při zpětném šíření chyby rovna jedné. Navíc byla do architektury přidána dvojice bran, a to *input gate*, jejíž funkcí je rozhodovat o datech, která si síť zapamatuje a *output gate*, která rozhoduje, jaká část výstupu je relevantní a bude postoupena jako output a stav vedlejší buňce. Jedinou nedokonalostí tohoto uspořádání byla neschopnost modelu zapomínat, pročež byla přidána třetí brána – *forget gate*. Schéma buňky LSTM je přiloženo jako obrázek 4.6.

[4]Analogicky lze uvažovat i o samostatné buňce, do níž postupně vstupují data ze sekvence.



Obrázek 4.6: Schematické znázornění principu LSTM buňky. Žlutě zvýrazněné operátory a funkce jsou aplikovány prvek po prvku, zatímco oranžově zvýrazněné implementujeme jako klasické vrstvy s maticovým násobením.

Chceme-li popsat princip LSTM matematicky [31], držíme se zavedeného značení, přičemž výrazy náležící k jednotlivým branám indexujeme prvním písmenem jejich anglického názvu:

$$\begin{aligned}
\mathbf{i}_t &\leftarrow \sigma(\mathbb{W}^i \mathbf{x}_t + \mathbb{V}^i \mathbf{h}_{t-1} + \mathbf{b}^i) \\
\mathbf{f}_t &\leftarrow \sigma(\mathbb{W}^f \mathbf{x}_t + \mathbb{V}^f \mathbf{h}_{t-1} + \mathbf{b}^f) \\
\mathbf{o}_t &\leftarrow \sigma(\mathbb{W}^o \mathbf{x}_t + \mathbb{V}^o \mathbf{h}_{t-1} + \mathbf{b}^o) \\
\mathbf{c}_t &\leftarrow \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh(\mathbb{W}^c \mathbf{x}_t + \mathbb{V}^c \mathbf{h}_{t-1} + \mathbf{b}^c) \\
\mathbf{h}_t &\leftarrow \mathbf{o}_t \circ \tanh(\mathbf{c}_t),
\end{aligned} \tag{4.20}$$

přičemž $\circ$ značí operaci násobení prvek po prvku.

4.2.4 Architektura typu seq2seq

Nejatraktivnějším využitím při vývoji rekurentních sítí byla práce se sekvencemi tvořenými lidskou řečí, ať už mluvenou, či psanou. *Seq2seq* je větev neuronových sítí, které jsou na takový úkol připravené. Typicky se takový model skládá ze dvou velkých částí, kterými jsou *encoder* a *decoder*. Encoder přijímá vstup a vytváří jeho abstraktní reprezentaci. Tato reprezentace putuje do decoderu, který z abstraktní reprezentace generuje novou datovou sekvenci. Jako stavební kameny encoderu a decoderu se dříve využívaly právě rekurentní sítě. Dnes se používájí převážně *Transformery*. Díky práci s lidským jazykem vznikly dva důležité prvky neuronových sítí. Vzhledem k potřebě pracovat s rozsáhlým slovníkem vznikla *embedding vrstva*. Pro zkvalitnění práce se sekvencemi v závislosti na významu samostatných dat vznikla také *attention vrstva*, která opravdu napodobuje schopnost upírat pozornost (ve finálním modelu budeme pracovat s modernější *self-attention* viz 4.2.5).

Embedding

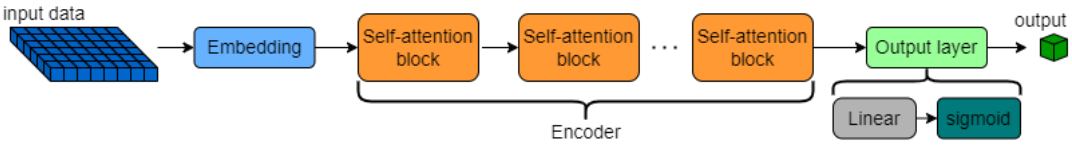
Embedding vrstva převádí diskrétní data do spojitých vektorů a zároveň zachycuje jejich vzájemné souvislosti. Pro diskrétní data je ve velkém měřítku nevýhodná reprezentace one-hot, neboť se poté potýkáme s obrovskými vektory plnými nul, s vyjímkou jediné pozice. Proto takové vektory embedding vrsta reprezentuje jako vektory fixní (volitelné) velikosti s nenulovými složkami. Embedding tedy zajišťuje reprezentaci, obsahující vzájemné vztahy jednotlivých dat a také redukci dimenze vstupu, což jsou ideální vlastnosti budoucího vstupu do neuronové sítě.

Toto je zavedení embeddingu je motivované potřebou práce s textem, tedy diskrétními daty s vysokou dimenzionalitou. Pro naši práci jsou však data spojitá s velmi nízkou dimenzionalitou. Embedding v našem případě využíváme k opačnému účelu, tedy ke zvyšování dimenzionality na úroveň vhodnou pro architekturu transformer.

4.2.5 Transformer

Architektura hluboké neuronové sítě Transformer byla představena článkem *Attention Is All You Need* [37] a záhy se stala nejvyužívanější architekturou pro jazykové modely. Těžištěm článku je však úprava a rozšíření attention vrstvy na *self-attention* (SA), respektive *multi-head self-attention* (MHSA). Self-attention bloky poté často nahrazovaly RNN v seq2seq architekturách, neboť pro určité úkoly je takové sekvenční zpracování nevhodné a dostatečně nevyužívá potenciálu výpočetní techniky. Self-attention umožnila propojovat informace z dat nezávisle na jejich pořadí v sekvenci.

Zaměřme se nyní již na úpravu Transformeru, kterou využijeme pro top-tagging. Náš model sestává pouze z encoderu (ač původní obsahuje i decoder v návaznosti na starší seq2seq modely), který bude vytvářet abstraktní reprezentace dat, na nichž bude prováděna klasifikace. Strukura modelu je schematicky znázorněna na obrázku 4.7



Obrázek 4.7: Schéma architektury Transformer. Prvním procesem je embedding, aby encoder mohl pracovat s reprezentacemi dat uniformní délky.

Self-Attention

Mějme vstupní datovou sekvenci o n prvcích, reprezentovanou maticí $\mathbb{X}^{n\times d}$, kde d je dimenze embeddingu. Při vstupu do SA bloku je každé jedné reprezentaci přiřazena trojice vektorů, a to:

- *Query* – Pro každé vstupní dato reprezentuje jeho query vektor otázku, na kterou se snaží najít nejlepší odpověď mezi zbylými členy sekvence. V

praxi je určována matice všech queries $\mathbb{Q} \in \mathbb{R}^{n \times d_k}$ jako výsledek maticového násobení vstupu s trénovatelnou maticí, tedy

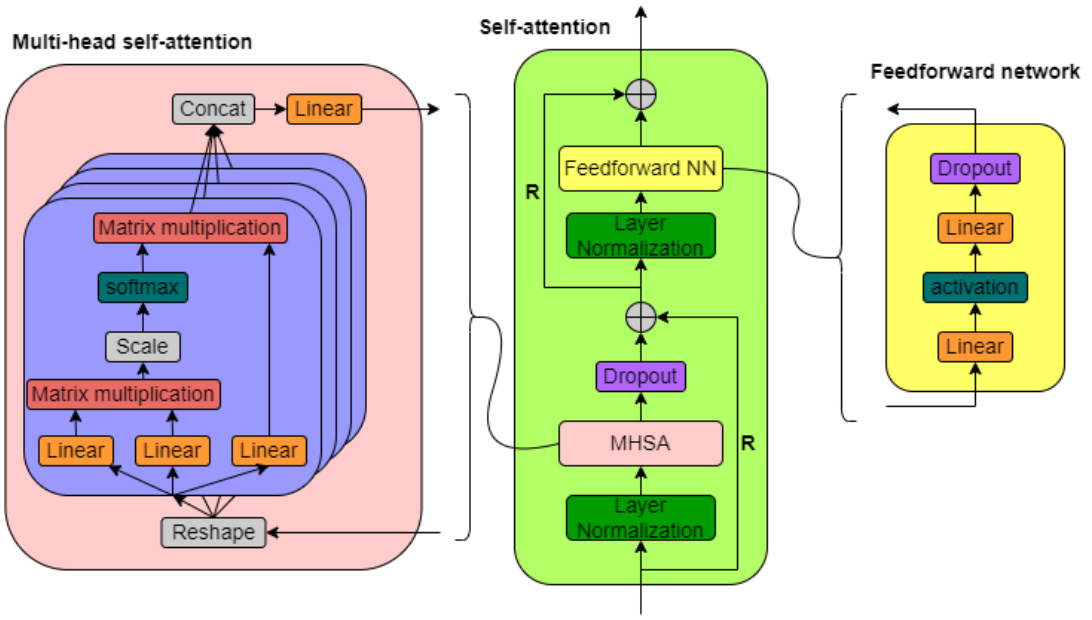
$$\mathbb{Q} = \mathbb{X}\mathbb{W}^Q.$$

- *Key* – Keys tvoří doplněk ke queries, tedy pro každé dato formulují charakteristiku relevantní pro queries. Jedná se tedy o odpovědi. Prakticky je opět určována matice $\mathbb{K} \in \mathbb{R}^{n \times d_k}$, pomocí maticového násobení s další trénovatelnou maticí:
$$\mathbb{K} = \mathbb{X}\mathbb{W}^K.$$

- *Value* – Posledním přiřazovaným vektorem jsou values, které uchovávají původní konkrétní informaci, kterou dato ze sekvence obsahovalo. Analogicky je počítána matice $\mathbb{V} \in \mathbb{R}^{n \times d_v}$ s pomocí poslední ze tří trénovatelných matic:
$$\mathbb{V} = \mathbb{X}\mathbb{W}^V.$$

Mechanismus self-attention vrstvy spočívá ve výpočtu attention-score, což je proces ve kterém se vzájemně párují queries a keys, následně je pomocí aktivace softmax z attention-score vytvořena distribuce, podle níž je pak rozdělen význam dat z matice $\mathbb{V}$. Matematická podoba takového procesu je následující:

$$\text{Attention}(\mathbb{Q},\mathbb{K},\mathbb{V}) = \text{softmax}\left(\frac{\mathbb{Q}\mathbb{K}^T}{\sqrt{d_k}}\right)\mathbb{V}. \tag{4.21}$$



Obrázek 4.8: Jednotka self-attention s detailem multi-head self-attention bloku a feedforward bloku. Písmenem R jsou označeny residuální spoje.

SA je velice silná architektura, která při práci s velkým množtvím dat překoná jiné užívané architektury, pro malé datové soubory však velmi rychle přetrénovává. Běžně se SA neimplementuje ve tvaru jednoho obrovského SA výpočtu,

nýbrž jako multi-head self attention, tedy několik menších paralelních výpočtů SA. Schéma MHSA ve formě, ve které je vybudována v našem modelu je přiloženo jako obrázek 4.8

Ve výsledné implementaci podle schématu 4.8 jsou využity ještě dva dosud nezmíněné prvky, těmi jsou:

- *Residuální spoje* – Někdy také *skip connections* byly představeny jako součást architektury *ResNet*. Podstatou je zobrazení identitou, které překročí blok vrstev a je pak znovu spojeno s výstupem. Residuálnín spoje díky tomu zefektivňují trénink hlubokých sítí, protože zmírňují vanishing gradient problém a ukázalo se, že značně zlepšují schopnost sítě generalizovat. [38]

- *Layer normalization* – Jedná se o techniku vyvinutou pro RNN, pro které nefungovala do té doby používaná *batch normalization*, tedy normalizace přes dimenzi batche. Proto se zavedla normalizace přes dimenzi rysů datového vstupu. Normalizace obecně stabilizuje trénink modelu, může mít mírně regularizační efekt a urychluje konvergenci tréninku. [39]

5. Výsledky

Cílem naší práce je vytvořit top-tagger, tedy model schopný identifikovat a označit jety pocházející z rozpadu top kvarku. Výstupem z modelu jsou tedy jety označené buďto jako *top* nebo jako *non-top*. Následně zhodnotíme schopnost našeho modelu identifikovat top jety pocházející z rozpadů na čtveřici top kvarků (4top eventů). Top jety z 4top eventů poté považujeme za *signál* a zbylé jety za *pozadí*. Naše neuronová síť byla vybudována v programovacím jazyce *Python* za využití knihoven *Tensorflow*[40] a *Keras*[41]. Jako výchozí architektura byl zvolen model Transformer, který byl implementován v úpravě popsané v sekci 4.7. Architektura typu Transofmer je momentálně jedním z nejvyužívanějších typů hlubokých neuronových sítí, zaznamenávající úspěch v oblasti práce s lidským jazykem, ale vyniká nad ostatními architekturami i v oblasti klasifikace dat z částicových experimentů. [42]

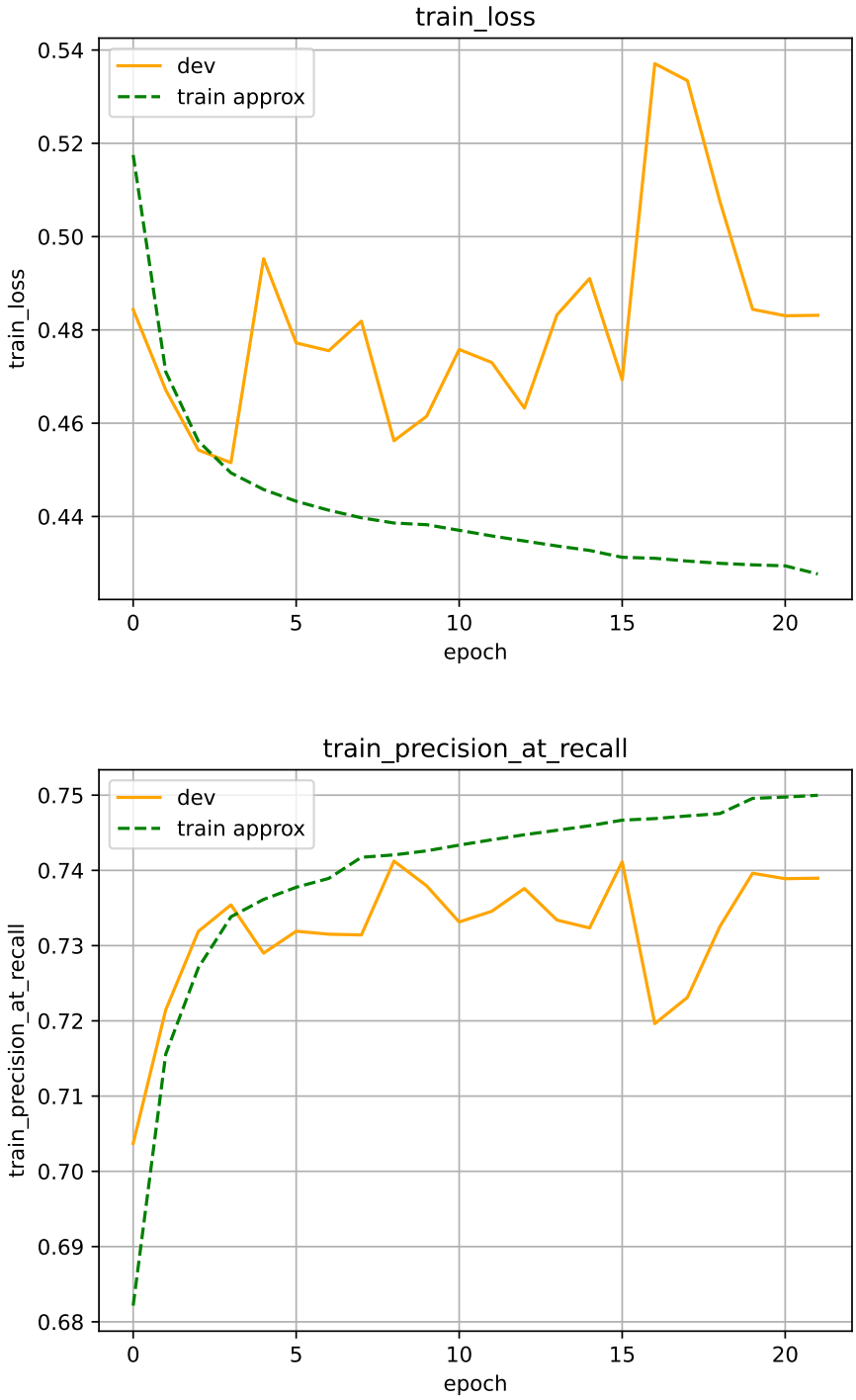
5.1 Volba hyperparametrů

Jak bylo popsáno v sekci 4.1, z vnějšku lze ovlivnit výkon a trénink neuronové sítě pomocí hyperparametrů. Prvním uvažovaným hyperparametrem je délka tréninku, tedy počet epoch. Nejdříve byly modely testovány na 10 epochách, v porovnání však dosahovaly nejlepších výkonů pro 20 epoch. Dalším hyperparametrem byly learning rate, nastavený na 0,001. Dále byl využit CosineDecay 4.1.3 a optimalizační algoritmus Adam 4.1.3, jehož parametry byly ponechány v defaultním nastavení. Z regularizačních metod byl využíván hlavně dropout, přičemž dropout rate byl nastaven na 0,5. Počet hlav bloku MHSA byl nastaven na 8, což je empiricky osvědčený počet [30]. Aktivační funkcí v SA blocích byla ReLU.

Dalším alternujícím faktorem byla dimenze embeddingu a celkový počet SA bloků v síti. Srovnání modelů pro relevantní uvažované hodnoty je vyneseno do tabulky 5.1. Každý model byl hodnocen pomocí tří metrik, a to precision at recall, AUC a accuracy. Za nejrelevantnější považujeme metriku precision at recall. Ve všech třech metrikách však dosáhl nejlepších hodnot model JetTrans_2 s dimenzí embeddingu 256 a čtyřmi SA bloky. Na základě tohoto modelu byl vytvořen finální top-tagger. Za povšimnutí však stojí i model JetTrans_1, jehož průběh tréninku je zachycen na obrázku 5.1.

Tabulka 5.1: Srovnání výkonu modelů na testovacích datastech vzhledem k různým dimenzemím embeddingu a počtům SA bloků pomocí vybraných metrik. Nejlepší výsledek je zvýrazněn tučně. Výsledky ukazují na nejlepší výkon modelu JetTrans_2

Model	Embedding	SA bloky	Precision at recall	Accuracy	AUC
JetTrans_1	256	8	0,7389	0,7617	0,8699
JetTrans_2	256	4	**0,7516**	**0,7930**	**0,8787**
JetTrans_3	128	8	0,7493	0,7911	0,8770
JetTrans_4	128	4	0,7478	0,7900	0,8758



Obrázek 5.1: Průběh tréninku modelu JetTrans_1 s dimenzí embeddingu 256 a 8 SA bloky. Na horním grafu je vynesen průběh metriky loss a na spodním grafu metriky precision at recall. Na grafech *train approx* označuje výsledky na tréninkových datech a *dev* výsledky na validačním datasetu. U obou grafů je patrné postupné zhoršování krátce po začátku tréninku. V případě loss se otvírá obrovská generalizační chyba mezi výkonem na tréninkových a validačních datech. Model ztratil schopnost úspěšně pracovat i na nových datech.

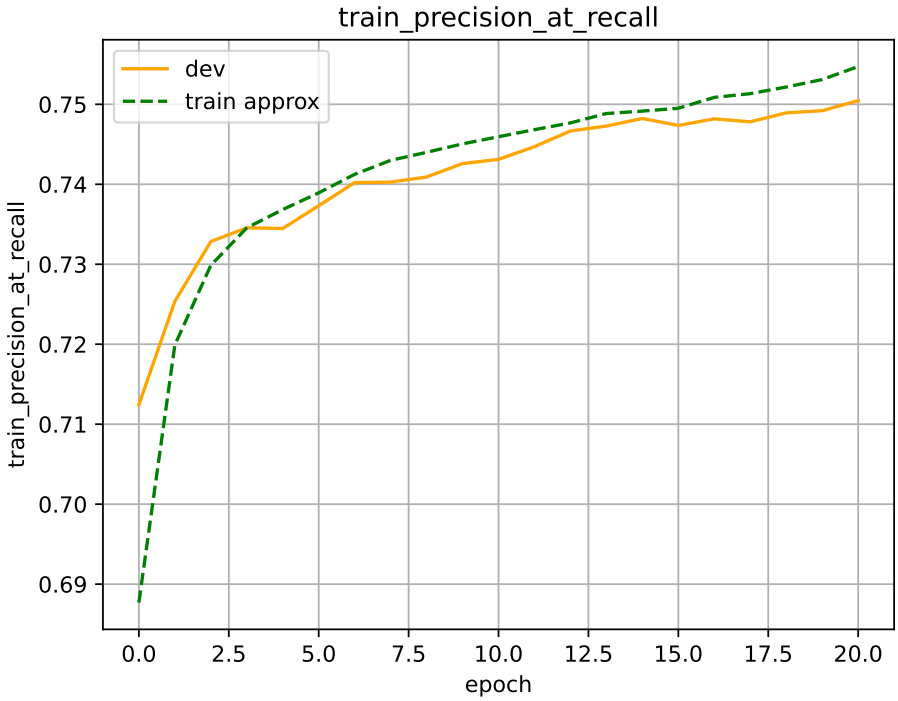
JetTrans_1 byl vůbec nejmohutnějším z uvážených modelů. Při zběžném pohledu na průběh jeho tréninku je patrné velmi silné přetrénování, neboli overfitting. Konkrétně křivky vývoje loss funkce dokonale odpovídají obrázku 4.1, kde je demonstrován nárůst generalizační chyby pro model s nadměrnou kapacitou vzhledem k množství tréninkových dat. Přesně to se stalo modelu JetTrans_1. I v tabulce 5.1 je patrný pokles hodnot všech uvážených metrik, a to dokonce na nejhorší ze všech uvedených modelů. Pokud bychom chtěli pro top-tagging využívat silnějších modelů s vysokou kapacitou, museli bychom uvážit zapojení regularizačních technik v mnohem větším měřítku, aby jejich kapacita nebyla při tréninku na škodu. Navíc jsou na grafech patrné výrazné výkyvy obou metrik v průběhu tréninku. Pro mohutnější modely tedy není současně nastavený trénink stabilní a bylo by nutné zintenzivnit i normalizaci.

5.2 Top-tagger JetTrans

Výsledný top-tagger, který budeme označovat pouze JetTrans je založen na modelu JetTrans_2.

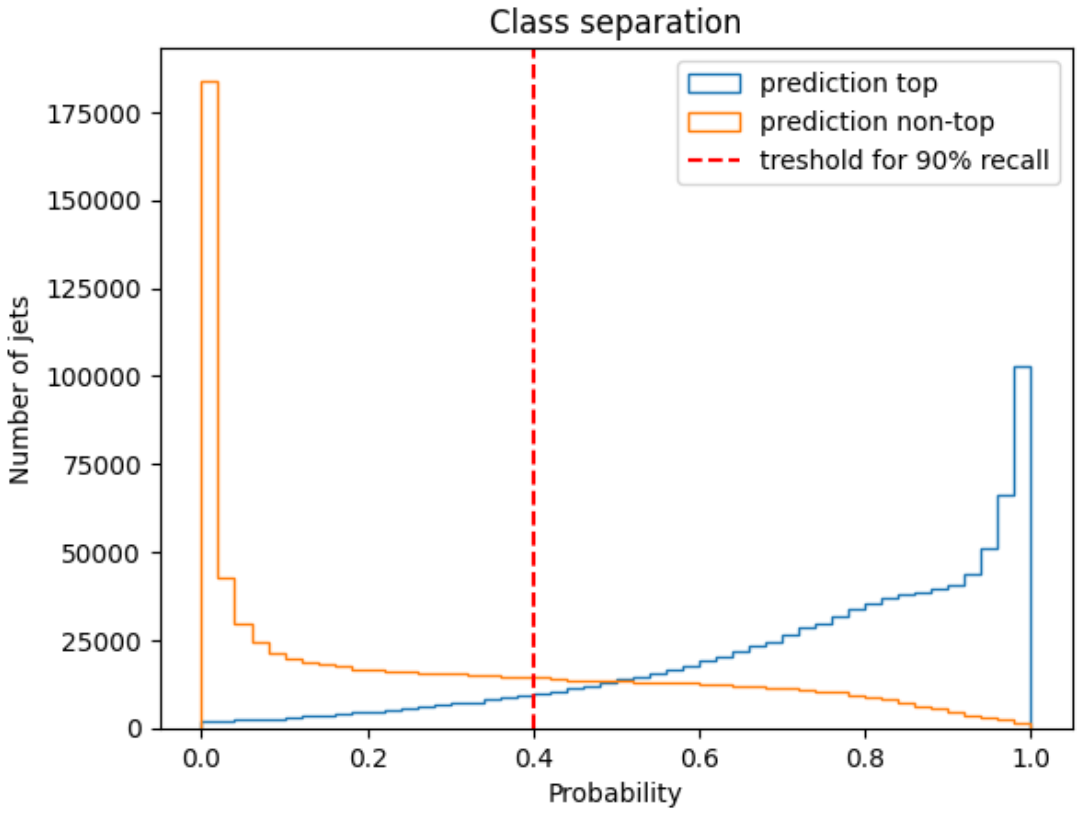
5.2.1 Výsledky

Průběh tréninku top-taggeru je ilustrován pro metriku precision at recall na obrázku 5.2. Hranice požadovaného recall je nastavena na 90%. Z grafu lze vyčíst, že trénink byl jednak stabilní a jednak úspěšný a model má předpoklady fungovat dobře i na nových datech. Precision at recall v takovém případě opět přesahuje 75%.



Obrázek 5.2: Průběh tréninku z pohledu metriky precision at recall.

Jak již bylo řečeno, top-tagger separuje top a non-top jety. Každému jednomu jetu je přiřazena pravděpodobnost, že se jedná o top jet. Pomocí pravděpodobnosti se pak jety rozvrství, jak je znázorněno na obrázku 5.3. Zde je vynesena taktéž čárkovaná úsečka, která rozděluje graf na dvě části. Vlevo jsou jety, o kterých (podle predikce) uvažujeme jako o non-top a vpravo jsou jety, které za top považujeme. Je však patrné, že se doprava přimísily i non-top jety. Proto pro jety obsažené v pravé části histogramů vypočteme metriku precision. Výsledné číslo je hodnota metriky precision at recall (4.1.6)



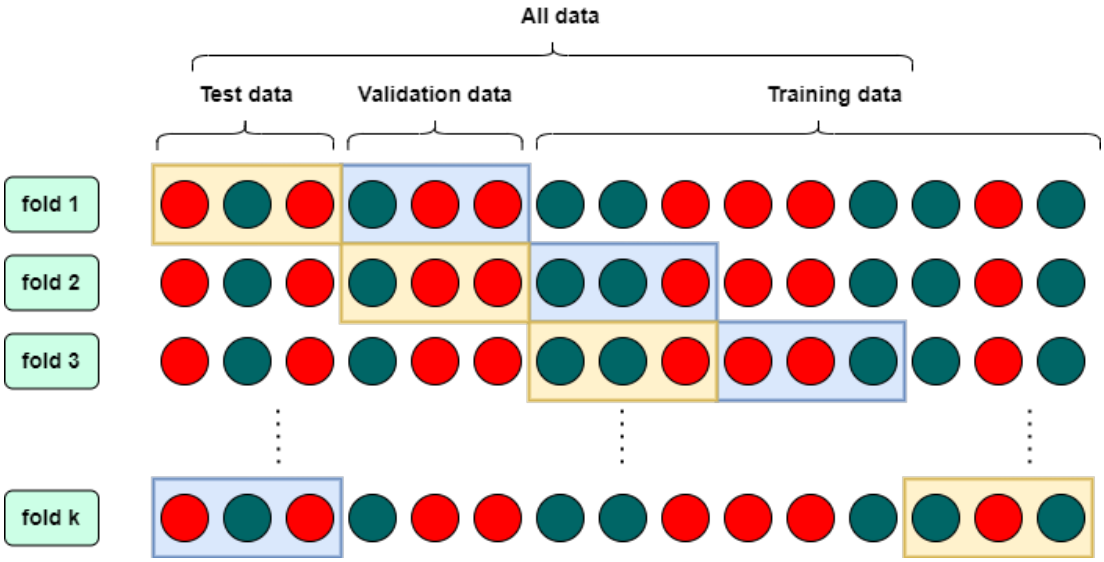
Obrázek 5.3: Predikce top-taggeru JetTrans. Oranžovou barvou jsou vyneseny do hitogramu non-top jety, podle jejich pravděpodobnostního označení top-taggerem. Vidíme klesající trend histogramu směrem k vyšším pravděpodobnostem. Analogicky poté vysvětlujeme modrý histogram top jetů s opačným trendem. Je patrné, že dochází k separaci, která však ještě poskytuje prostor pro zlepšení. V idealizovaném nedosažitelném případě by se histogramy neměly vůbec překrývat.

5.2.2 K-fold cross-validation

Při tréninku výsledného modelu, byla oproti předchozím využita navíc ještě jedna technika a to *křížová validace*, neboli *cross-validation*. Ilustrace průběhu cross-validation je přiložena jako obrázek 5.4. Princip této techniky spočívá v rozdělení dat na k stejných částí, z nichž některé hrají roli tréninkového datasetu, některé roli validačního datasetu a některé testovacího datasetu. Využívá se tedy v případech, kdy potřebujeme natrénovat mohutný model na velmi omezeném datasetu a spolehlivě otestovat jeho výkon na nových datech. Celkem je trénováno

k modelů, přičemž v momentě, kdy dochází k evaluaci na testovacím datasetu, lze pro každý z těchto modelů najít data, která nebyla využita k jeho tréninku a dostaneme tak nezaujaté hodnocení výkonu modelu.

V našem případě se jednalo o 5-fold cross-validation a naše data z MC simulace byla rozdělena na pětiny. Pětina tvořila testovací dataset, pětina validační dataset a zbylé tři pětiny zůstaly tréninkovými daty. Celkem bylo také natrénováno 5 modelů.



Obrázek 5.4: Ilustrace principu techniky k-fold cross-validation.

5.3 Využití pro identifikaci 4top eventů

Dalším z cílů je připravit top-tagger na využití pro klasifikaci 4top eventů. Je očekávatelné, že 4top eventy budou v průměru obsahovat více top jetů než $t\bar{t}$ či multijet eventy. V ideálním případě bychom pro 4top eventy předpokládali 12 top jetů, pro $t\bar{t}$ eventy 6 top jetů a pro multijet eventy žádný top jet. Bohužel, vzhledem k nedokonalosti top-taggeru, takový výsledek nepozorujeme. Počet otagovaných jetů lze však využít jako proměnnou, kterou využijeme při klasifikaci 4top eventů. Model na tuto klasifikaci připravíme optimalizací hodnoty pravděpodobnostního prahu, které musí jet na výstupu dosáhnout, aby byl otagován jako top.

Zavedeme novou proměnnou N_{topjet}, která vyjadřuje počet jetů v eventu, označených jako top jet. Tato proměnná je tedy závislá na pravděpodobnostním prahu a konstruujeme ji pro každý event. Pro všechny typy eventů následně vyneseme závislost počtu eventů na počtu označených top jetů v podobě histogramu 5.6. Při pohledu na přiložený histogram 5.6 je patrné, že závislosti pozadí ($t\bar{t}$ a multijet) a signálu (4top) mají různé tvary, což potvrzuje očekávání vyššího průměrného počtu top jetů pro 4top eventy. Vzhledem k obrovské převaze eventů pozadí však nedochází v této fázi k výraznější separaci signálu od pozadí.

Abychom optimalizovali hodnotu pravděpodobnostního prahu, musíme kvantifikovat míru překrytí histogramů signálu a pozadí v závislosti na N_{topjet}. Optimalizace byla provedena pro prahové hodnoty: 0,3; 0,4; 0,5; 0,6 a 0,7. Jako vhodná metrika pro kvantifikaci překrytí byla zvolena *signifikance*.

Signifikanci lze pro poissonovská data standardně odhadnout pomocí vztahu [43]:

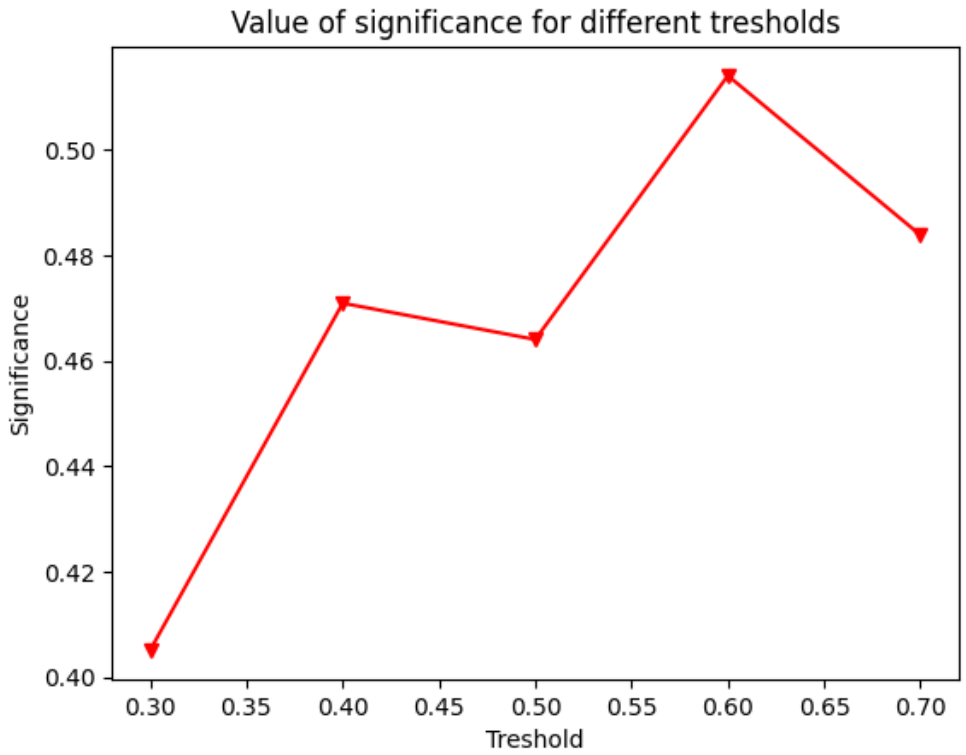
$$\text{signifikance} = Z = \sqrt{(n_{sig} + n_{bkg})\ln\left(1 + \frac{n_{sig}}{n_{bkg}}\right) - n_{sig}}, \tag{5.1}$$

kde n_{sig} a n_{bkg} jsou obecně počty objektů klasifikovaných jako signál respektive pozadí. Za splnění podmínky $n_{sig} << n_{bkg}$ se však vzorec 5.1 redukuje do tvaru:

$$\text{signifikance} = Z = \frac{n_{sig}}{\sqrt{n_{bkg}}}, \tag{5.2}$$

s jehož pomocí spočteme Z pro každý sloupec histogramu. Celková signifikance je pak určena jako suma čtverců jednotlivých Z, přičemž sloupce, které nesplňují podmínku $n_{sig} << n_{bkg}$ nezahrneme do výpočtu signifikance, čímž získáme její dobrý konzervativní odhad.

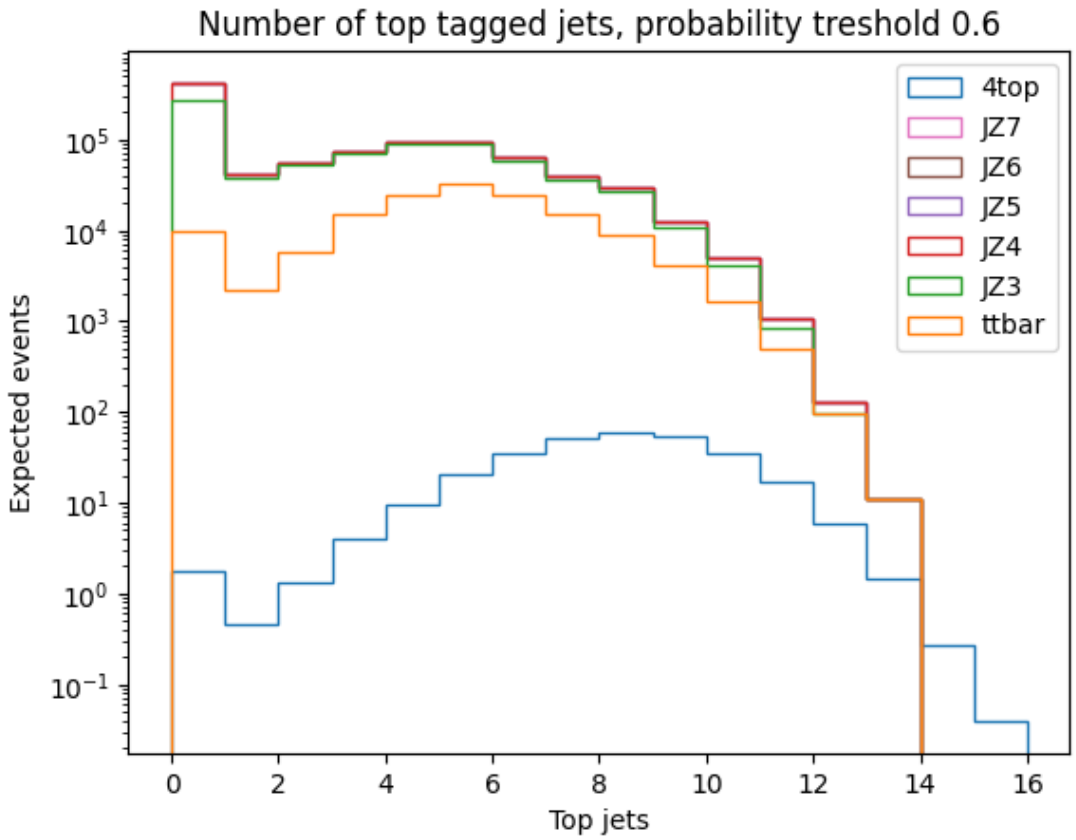
Výsledky srovnání jsou přiloženy v grafické podobě jako obrázek 5.5.



Obrázek 5.5: Srovnání klasifikací pro různé pravděpodobnostní prahy. Nejlepší nastavení pravděpodobnostního prahu top taggeru pro klasifikaci 4top eventů je 0,6 neboli 60%.

Z grafu 5.5 je patrné, že nejmenší překrytí pozorujeme pro pravděpodobnostní práh 0,6. Budeme-li využívat top-tagger JetTrans jako součást klasifikačního procesu zaměřeného na 4top eventy, označíme za top jet takový jet, kterému predikce

modelu přiřkne pravděpodobnost vyšší nežli 0,6. V grafu na obrázku 5.6 je vynesený histogram vypočtené závislosti právě pro nejvhodnější prahovou hodnotu 0,6.



Obrázek 5.6: Histogram s logaritmickou vertikální osou, popisující zastoupení očekávaných eventů pro různé počty top jetů.

Závěr

První část práce tvoří fyzikální úvod, druhou rozsáhlá rešerše na téma strojové učení a učení hlubokých neuronových sítí, kde uvádíme informace, díky nimž bylo možné v poslední části naplnit cíle naší práce a prezentovat výsledky.

S využitím programovacího jazyka Python byla vybudována a otestována hluboká neuronová síť architektury Transformer, která byla natrénována, aby sloužila jako top-tagger. Výsledný top-tagger byl založen na nejlepším modelu z několika testovaných modelů vzájemně se odlišujících stavbou či volbou hyperparametrů při tréninku.
Následně bylo možné zaměřit se na klasifikaci případů s rozpadem na čtveřici top kvarků, tedy takzvané 4top eventy. Obecně je velmi komplikované takové eventy klasifikovat vzhledem k tomu, že koncový stav může nabývat velkého množství podob. Významně úspěšná separace 4top eventů od pozadí není v možnostech samostatného top-taggeru. Top-tagger JetTrans však může poskytnout cenné údaje pro vstup do mohutnějších klasifikačních modelů (jako například DeParT [42]), což bylo cílem naší práce.
Top-tagger byl na tento úkol připraven v rámci naší práce určením pravděpodobnostního prahu predikce pro označení jetu za top jet. Volbou prahu můžeme ovlivnit poměr očekávaných eventů v závislosti na pozitivně klasifikovaných jetech. Bylo otestováno několik variant, přičemž výsledná varianta prahové hodnoty byla zvolena tak, aby top tagger přijmul co možná nejvyšší množství top jetů pocházejících z 4top eventů.

Výhledově může model JetTrans poskytovat stále jistý prostor pro zlepšení výkonu. Taková práce může připadnout v úvahu po otestování implementace kombinace JetTrans top taggeru a již využívaných klasifikačních modelů pro identifikaci 4top eventů, což bylo hlavní z jeho zamýšlených využití.

Seznam použité literatury

[1] Cush. Standard_model_of_elementary_particles.svg. "https://commons.wikimedia.org/w/index.php?curid=4286964. Fermilab, Office of Science United States Department of Energy Particle Data Group Public Domain.

[2] Rupert Leitner Tomáš Davídek. *Elementární částice od prvních objevů po současné experimenty*. Matfyzpress, 2012.

[3] Jiří Hořejší. *Historie standardního modelu mikrosvěta*. MatfyzPress, 2 edition, 2017/2018.

[4] Stephanie Hansmann-Menzemer. Mkep 1.2: Particle physics ws 2012/13. https://www.physi.uni-heidelberg.de/~menzemer/PP_WS1213/Lecture-XV.pdf, December 2012.

[5] R. L. Workman and Others. Review of Particle Physics. *PTEP*, 2022:083C01, 2022.

[6] Richard Hawkings. The last quark. https://atlas.cern/updates/feature/top-quark, July 2021.

[7] Nazar Bartosik. Ttbar_production. http://bartosik.pp.ua/feynman_diagrams/ttbar_qq,CCBY4.0,https://commons.wikimedia.org/w/index.php?curid=49748619.

[8] Naomi Dinmore. Atlas and cms observe simultaneous production of four top quarks. https://home.cern/news/news/physics/atlas-and-cms-observe-simultaneous-production-four-top-quarks, march 2023.

[9] CMS Collaboration. Evidence for four-top quark production in proton-proton collisions at $\sqrt{s} = 13$ tev. *Phys. Lett. B 844 (2023) 138076*, 844:138076, September 2023.

[10] The large hadron collider. https://home.cern/science/accelerators/large-hadron-collider, April 2024.

[11] Facts about lhc. https://home.cern/resources/faqs/facts-and-figures-about-lhc, April 2024.

[12] Lyndon Evans and Philip Bryant. Lhc machine. *Journal of Instrumentation*, 3(08):S08001–S08001, August 2008.

[13] The atlas detector. https://atlas.cern/Discover/Detector, April 2024.

[14] G. Aad et al. Collaboration, The ATLAS. The atlas experiment at the cern large hadron collider. *Journal of Instrumentation*, 3(08):S08003–S08003, August 2008.

[15] The atlas detector walks another mile. https://cerncourier.com/a/the-atlas-detector-walks-another-mile/, August 2007. CERN Courier.

[16] Inner detector. https://atlas.cern/Discover/Detector/Inner-Detector, April 2024.

[17] Calorimeter. https://atlas.cern/Discover/Detector/Calorimeter, April 2024.

[18] Muon-spectrometer. https://atlas.cern/Discover/Detector/Muon-Spectrometer, April 2024.

[19] Magnet system. https://atlas.cern/Discover/Detector/Magnet-System, April 2024.

[20] Trigger and data aquisition. https://atlas.cern/Discover/Detector/Trigger-DAQ.

[21] Joao Pequenao; Paul Schaffner. How atlas detects particles: diagram of particle paths in the detector. https://cds.cern.ch/record/1505342, January 2013.

[22] Introduction to monte carlo simulations. https://atlasopendata.docs.cern.ch/docs/documentation/monte_carlo/introduction_MC/, April 2024. ATLAS Open Data.

[23] Florian Beaudette. The cms particle flow algorithm. *Proceedings of the CHEF2013 Conference - Eds. J.C. Brient, R. Salerno, and Y. Sirois - p295 (2013), ISBN 978-2-7302-1624-1*, January 2014.

[24] Ryan Atkin. Review of jet reconstruction algorithms. *Journal of Physics: Conference Series*, 645:012008, October 2015.

[25] Stefan Weinzierl. The siscone jet algorithm optimised for low particle multiplicities. *Computer Physics Communications*, 183(3):813–820, March 2011.

[26] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The anti-k_t jet clustering algorithm. *JHEP 0804:063,2008*, 2008(04):063–063, April 2008.

[27] Jana Šafránková, editor. *22nd Annual Conference of Doctoral Students, WDS'13 "Week of Doctoral Students 2013", Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic, June 4, 2013 - June 7, 2013*, volume Pt. 3, Praha, 2013. Matfyzpress.

[28] Nazar Bartosik. B-tagging_diagram.png. http://bartosik.pp.ua/hep_sketches/btagging,CCBY4.0,https://commons.wikimedia.org/w/index.php?curid=49738737.

[29] Milan Straka. Machine learning for greenhorns. https://ufal.mff.cuni.cz/courses/npfl129/2223-winter, Winter semester 2022/2023.

[30] Milan Straka. Deep learning. https://ufal.mff.cuni.cz/courses/npfl114/2223-summer, Summer semester 2022/2023.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[32] Christopher M. Bishop. *Pattern recognition and machine learning*. Information Science and Statistics. Springer Science+Business Media, LLC, New York, NY, 2019.

[33] Gabriel Goh. Why momentum really works. *Distill*, 2017.

[34] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. December 2014.

[35] Tomasz Szandała. Review and comparison of commonly used activation functions for deep neural networks. *Studies in Computational Intelligence*, October 2020.

[36] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. June 2017.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. December 2015.

[39] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. July 2016.

[40] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org/`, 2015. Software available from tensorflow.org.

[41] François Chollet et al. Keras. `https://keras.io`, 2015.

[42] Samuel JANKOVÝCH. Quark/gluon jet tagging. Praha: Univerzita Karlova Matematicko-fyzikální fakulta Ústav částicové a jaderné fyziky, 2023. Bakalářská práce, vedoucí Pleskot Vojtěch.

[43] M. J. Basso. Generalized asymptotic formulae for estimating statistical significance in high energy physics analyses. *J. Phys. G: Nucl. Part. Phys. 49 025001 (2022)*, 49(2):025001, December 2021.

Seznam obrázků

1.1 Znázornění standardního modelu částic a interakcí, kde jsou barevně rozlišeny částice hmoty, neboli fermiony (kvarky a leptony) a částice zprostředkovávající interakce, neboli bosony.[1] 5
1.2 Barevné uvěznění kvarků a postupná hadronizace. [4] 7
1.3 Feynmanovy diagramy procesů s nejvyšším příspěvkem k produkci páru top-antitop.[7] 8
1.4 Feynmanovy diagramy procesů s nejvyšším příspěvkem k produkci čtyř top kvarků.[9] 8

2.1 Schematická ilustrace soustavy urychlovačů CERN. [11] 10
2.2 Schematické znázornění celkového pohledu na detektor ATLAS s popisem jeho nejdůležitějších částí.[15] 11
2.3 Detailní pohled na podélný průřez vnitřním detektorem.[16] . . . 12
2.4 Detailní pohled na podélný průřez soustavou kalorimetrů detektoru ATLAS. [17] 13
2.5 Umístění detekčních systémů, tvořících mionový spektrometr. [18] 14
2.6 Příčný průřez celým detektorem ATLAS s naznačenou formou detekce různých částic. [21] 15

3.1 Znázornění postupného vzniku jetů a jejich detekce - jedinou možností, jak se dobrat přesné podoby původní srážky je rekonstrukce jetu.[24] 17
3.2 Ukázka využití různých jet clustering algoritmů. Nahoře zleva k_t a *Cambridge/Aachen*, dole zleva *SIScone* a *anti-*k_t.[26] 19
3.3 Vizualizace základních principů b-taggingu.[28] 20

4.1 Typický průběh závislosti chybovosti modelu na jeho kapacitě. [31] 28
4.2 Obecná podoba 2×2 confusion matrix (tedy chybové matice pro binární klasifikaci). 29
4.3 Zjednodušené vyobrazení fully connected sítě, která zároveň spadá do kategorie feedforward NN. 31
4.4 Schéma architektury MLP. Blok fully connected vrstav zakončených výstupní vrstvou s aktivací sigmoid. 32
4.5 Vlevo jednoduchá RNN buňka a vpravo uspořádání takových buněk do vrstvy, která poté zpracovává sekvenci vstupních dat. . . . 33
4.6 Schematické znázornění principu LSTM buňky. Žlutě zvýrazněné operátory a funkce jsou aplikovány prvek po prvku, zatímco oranžově zvýrazněné implementujeme jako klasické vrstvy s maticovým násobením. 34
4.7 Schéma architektury Transformer. Prvním procesem je embedding, aby encoder mohl pracovat s reprezentacemi dat uniformní délky. 35
4.8 Jednotka self-attention s detailem multi-head self-attention bloku a feedforward bloku. Písmenem R jsou označeny residuální spoje. 36

5.1 Průběh tréninku modelu JetTrans_1 s dimenzí embeddingu 256 a 8 SA bloky. Na horním grafu je vynesen průběh metriky loss a na spodním grafu metriky precision at recall. Na grafech *train approx* označuje výsledky na tréninkových datech a *dev* výsledky na validačním datasetu. U obou grafů je patrné postupné zhoršování krátce po začátku tréninku. V případě loss se otvírá obrovská generalizační chyba mezi výkonem na tréninkových a validačních datech. Model ztratil schopnost úspěšně pracovat i na nových datech. 39
5.2 Průběh tréninku z pohledu metriky precision at recall. 40
5.3 Predikce top-taggeru JetTrans. Oranžovou barvou jsou vyneseny do hitogramu non-top jety, podle jejich pravděpodobnostního označení top-taggerem. Vidíme klesající trend histogramu směrem k vyšším pravděpodobnostem. Analogicky poté vysvětlujeme modrý histogram top jetů s opačným trendem. Je patrné, že dochází k separaci, která však ještě poskytuje prostor pro zlepšení. V idealizovaném nedosažitelném případě by se histogramy neměly vůbec překrývat. 41
5.4 Ilustrace principu techniky k-fold cross-validation. 42
5.5 Srovnání klasifikací pro různé pravděpodobnostní prahy. Nejlepší nastavení pravděpodobnostního prahu top taggeru pro klasifikaci 4top eventů je 0,6 neboli 60%. 43
5.6 Histogram s logaritmickou vertikální osou, popisující zastoupení očekávaných eventů pro různé počty top jetů. 44