Korištenje novih reprezentacija podataka u svrhu klasfikacije događaja pomoću strojnog učenja

Kovačić, Nino

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Nino Kovačić

Korištenje novih reprezentacija podataka u svrhu klasifikacije događaja pomoću strojnog učenja

Diplomski rad

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA; SMJER ISTRAŽIVAČKI

Nino Kovačić

Diplomski rad

Korištenje novih reprezentacija podataka u svrhu klasifikacije događaja pomoću strojnog učenja

Voditelj diplomskog rada: izv. prof. dr. sc. Nikola Poljak

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: 24.11.2022.
Zagreb, 2022.

Sažetak

Kako bi se nosili sa stalno povećavajućim količinama podataka generiranih u eksperimentima potrebno je primijeniti naprednije tehnike obrade podataka. Najbolji primjer kojih je strojno učenje, ili specifičnije, neuralne mreže. Koristeći generirane podatke o događajima s mlazovima razvijamo binarni klasifikator s ciljem razlikovanja potječe li mlaz od gluona ili kvarka. U radu proučit ćemo nekoliko reprezentacija podataka te različite arhitekture mreža i usporediti klasifikatore.

Abstract

In order to deal with the constantly increasing amount of data generated by experiments more advanced data processing techniques are required. The best example of these is machine learning, and more specifically, neural networks. Using a generated "black box" set of events describing jets, we develop a binary classifier to determine whether the origin of a particular jet is a quark or a gluon. In doing so, we'll examine several representations of events as well as different network architectures, and compare the classifiers.

Sadržaj

1 Uvod

U radu s velikim setovima podataka strojno učenje se pokazuje iznimno uspješnim u brojnim zadacima, te tako ono danas pogoni niz mehanizama u društvu, od optimizacije logističkih i trgovinskih sistema, preko samovozećih automobila sve do individualnog iskustva pretraživanja interneta. Nadalje, kako metode u znanosti napreduju, količina prikupljenih podataka se drastično povećava te se nameće potreba za razvijanjem sve moćnijih metoda analize tih podataka. Stoga, nije neobično da već dugi niz godina primjena strojnog učenja nalazi veliki uspjeh i u fizici, pa tako i u fizici visokih energija [1, 2].

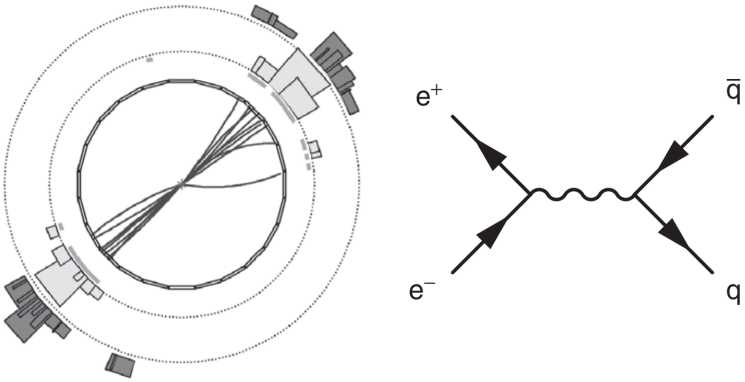
U ovom radu ćemo se fokusirati na upravo takvu jednu primjenu na simuliranom setu podataka za QCD procese s mlazovima koji bi odgovarali događajima kakve bi, primjerice, mogli očekivati kao rezultate u detektoru ALICE (*A Large Ion Collider Experiment*). Cilj nam je razviti što bolji klasifikator koji bi predviđao jesu li mlazovi porijeklom kvarkovski ili gluonski. U tu svrhu istražit ćemo performanse različitih algoritama na nekoliko reprezentacija podataka.

Sami podaci su nam dani kao tzv. crna kutija (*black box*) od približno 1,2 milijuna događaja, što znači da su za svaki simulirani događaj poznati samo podaci o izlaznim česticama (preciznije, njihovi 4-impulsi) te porijeklo događaja (kvarkovski ili gluonski, predstavljeno nulama i jedinicama), bez ikakvih detalja o tome kako se sam proces mogao odvijati.

2 Kvantna kromodinamika i procesi s mlazovima

Kao i svaki sustavni dio kvantne teorije polja, *kvantna kromodinamika* (eng. *quantum chromodynamics*, nadalje skraćeno na QCD), koja pokriva interakcije jakom silom, bazirana je na baždarnoj simetriji – SU(3) bojnoj simetriji [3]. U njoj su tri boje baždarni kvantni brojevi kvarkova, dok su hadroni bojni singleti, odnosno oni su invarijantni na SU(3) transformacije, a postoje svega 3 jednostavne kombinacije koje to zadovoljavaju: kvark-antikvark par te tri kvarka ili antikvarka[1], što odgovara mezonima, barionima i antibarionima.

Nadalje, s jedne strane, poznato je da za dovoljno veliku konstantu vezanja jaka interakcija postaje proporcionalna udaljenosti, što znači da će jedina slobodna stanja biti bojni singleti (dakle, hadroni). S druge strane, u QCD-u, za razliku od kvantne elektrodinamike, konstanta vezanja raste sa smanjenjem energetske skale [4]. Nameće se zaključak da na malim energijama[2] nikada nećemo opaziti slobodne kvarkove, kojima će pak biti energetski povoljnije „udružiti“ se u neki od hadrona.



Slika 2.1: Elektron-pozitron anihilacija koja rezultira u dva mlaza, lijevo: situacija u detektoru, desno: pripadni Feynmanov dijagram na *tree level-u*. Preuzeto iz [4].

Ako prilikom sudara u detektoru nastane visokoenergetski kvark-antikvark par njihovim udaljavanjem raste energija pohranjena u njihovoj interakciji. Pri tome su njeni nositelji, gluoni, sažeti u cjevasti oblik između para, tzv. *flux tube*. U jednom trenutku energija u takvom gluonskom polju postane dovoljna za tvrobu novog kvark-antikvark para te cijev puca. Ovaj se postupak ponavlja dokle god ima dovoljno energije za tvorbu parova, a kad nje ponestane, kvarkovi koji su ostali dovoljno blizu (tj. u interakciji) se, kao što smo prije spomenuli, udružuju u hadrone. Ovime smo

[1] Zapravo situacija je malo suptilnija, po pitanju boje, mezoni su rezultat unutarnjeg produkta kvarkova i antikvarkova, dok su (anti)barioni totalno asimetrične kombinacije tri (anti)kvarka.

[2] Malima po pitanju QCD-a, konstanta vezanja jake sile je već oko jedinice na 1 GeV.

upravo opisali proces *hadronizacije* u kojem bi nastali mlazovi (eng. *jets*), primjer kojega je prikazan na slici 2.1.

U našem slučaju, proučavat ćemo samo jedan mlaz, koji može nastati ili od kvarka, ili od gluona koji raspadom na kvark-antikvark par započne hadronizaciju.

3 Metode rada s neuralnim mrežama

3.1 Osnovni pojmovi

Strojno učenje podrazumijeva uporabu algoritama koji su u stanju učiti iz iskustva, a time želimo reći da im se uspjeh pri izvršavanju nekog zadatka generalno poboljšava s povećanjem količine podataka koje je algoritam do tada vidio (odnosno, s povećanjem podataka na kojima je treniran) [5]. Tu zapravo spada niz tehnika, od jednostavne linearne regresije metodom najmanjih kvadrata, preko nekih naprednijih kao što su SVM (eng. *support vector machines*) i stabla odluke (eng. *decision trees*), sve do neuralnih mreža (eng. *neural network*), koje ćemo mi u ovom radu ekskluzivno koristiti[3] zbog njihove iznimne moći prilagođavanja na raznolike i velike setove podataka u kojima može biti prisutan i popriličan šum.

Za početak je najbolje uvesti određene pojmove [6], usko vezane uz neuralne mreže koje ćemo redovito koristiti. Tu vrijedi napomenuti kako, obzirom na relativnu novinu ovih metoda, u hrvatskom jeziku još nisu posve ustaljeni svi nazivi, jer se većina inženjera i znanstvenika direktno koristi engleskim pojmovima. Mi ćemo se truditi koristiti nazive koji su već rašireni ili ćemo prevesti postojeće najbolje što možemo u duhu hrvatskog jezika, ukoliko je to moguće. Pri tome ćemo, kako bi izbjegli eventualne zabune uvijek pri prvom spominjanju navoditi i engleski pojam, a ako prevođenje nekog pojma nije praktično morat ćemo pribjeći direktnoj uporabi engleske nomenklature.

Treniranje je proces prilagođavanja mreže podacima. *Model* nam predstavlja jednu mrežu koja je spremna za treniranje (dakle, odabir svih detalja oko njene konstrukcije je završen). *Događaj* jest potpuni set podataka na kojemu mreža treba izvršiti zadatak. *Funkcija gubitka* (eng. *loss function*) je kvantitativna mjera koliko je dobro mreža obavila svoj zadatak na jednom događaju koji se koristi u procesu treniranja. *Značajka* (eng. *feature*) je jedan tip podatka u događaju (primjerice, geografska širina), a ovisno o kontekstu može podrazumijevati i samu vrijednost toga podatka. *Reprezentacija* znači specifičan odabir značajki kojim predstavljamo događaje, do kojega dolazimo različitim transformacijama. *Generacija* ili *epoha* (eng. *epoch*) predstavlja jedno treniranje modela (u smislu jednog prolaska kroz podatke) na setu događaja kojega smo odabrali, a ponavljanjem tog treniranja dobivamo nove epohe. *Metrika* (eng.

[3] Radit ćemo u *Keras-u* s *TensorFlow backend-om*.

metric) je veličina pomoću koje pratimo napredak mreže pri treniranju, ili ocjenjujemo njen rad na nekom setu događaja (što nazivamo *evaluacijom*). Metrika u pravilu nema utjecaja na treniranje, ali redovito ima za nas veće značenje i bolju interpretaciju od same funkcije gubitka.

Velika prednost strojnog učenja u usporedbi s tradicionalnim programiranjem je u tome što ne moramo direktno i unaprijed odrediti sve odluke koje bi naš program trebao napraviti kako bi stigao do rješenja. Primjerice, zamislimo da nam je cilj predvidjeti koja je vrsta cvijeta prikazana na slici – jasno je da bi takav problem bio praktički nerješiv ako bismo morali sami specificirati i implementirati sve kriterije koji bi vodili do konačne odluke. Namjesto toga, detalje prepuštamo programu da ih nauči, a nama, kao programeru, je zadatak „pomoći“ algoritmu da se što bolje prilagodi podacima kojima raspolažemo. Tu moramo izvršiti dvije stvari: jedna je predobrada podataka (eng. *data preprocessing*) gdje biramo i konstruiramo reprezentacije. Druga je odabir, priprema i ugađanje modela, pri kojoj prvenstveno podešavamo *hiperparametre* modela, koji su zajednički naziv za sve veličine koje diktiraju oblik i ponašanje modela te moraju biti postavljeni prije negoli što treniranje može započeti.

3.2 Neuralna mreža

Svaka neuralna mreža (nadalje skraćeno na NM) je u osnovi niz (ne nužno uzastopnih) transformacija koje se primjenjuju na ulazne podatke. Svaka transformacija je određena svojim parametrima, koji se u velikoj većini slučajeva mogu trenirati. Pojedinu transformaciju nazivamo *slojem* (eng. *layer*) te je ključna razlika NM od drugih algoritama strojnog učenja način na koji se njeni slojevi povezuju te velike dubine modela (najveći broj slojeva koje primjenjujemo) koje možemo praktično iskoristiti.

Najosnovnija vrsta sloja je tzv. *gusti* (eng. *dense layer*) gdje se vrši jednostavna transformacija ulaznih podataka iz oblika D-dim. vektora $\vec{x}$ u N-dim. vektor:

$$d_{W,\vec{b}}(\vec{x}) = \phi\left(W\vec{x} + \vec{b}\right), \tag{3.1}$$

pri čemu se elementi matrice W nazivaju *težinama* (eng. *weights*), a elementi vektora $\vec{b}$ *pristranostima* (eng. *bias*). Te dvije veličine zajedno čine parametre koji se treniraju,

dok odabir *aktivacijske funkcije*[4] (eng. *activation function*), $\phi : \mathbb{R}^D \to \mathbb{R}^N$ i dimenzije N spada u hiperparametre.

Shvatimo li sada ovakvu transformaciju kao N odvojenih operacija (sa skalarnim množenjem namjesto matričnog), onda je posve razumno promatrati gusti sloj kao uređenu N-torku *neurona*, pri čemu sada dimenziju N možemo prozvati brojem neurona. Zamislimo li sada da imamo vrlo primitivnu NM sačinjenu od više uzastopnih gustih slojeva[5], u njoj će svaki neuron imati za ulazne vrijednosti rezultate svih neurona prethodnog sloja. Zato ovakav sloj još nazivamo i *posve povezanim slojem* (eng. *fully connected layer*).

Ključno pitanje koje je dugo priječilo razvoj ikakvih dubokih mreža je bilo kako odrediti koliko se pojedini parametri moraju podesiti pri treniranju [6]. Teškoća je sljedeća: iako je sasvim jasno da se u prostoru parametara treba kretati prema globalnom optimumu funkcije gubitka (minimumu ili maksimumu, ovisno o tome kako ju definiramo, konvencionalno minimumu), pitanje je kako, i to na efikasan način, obaviti to kretanje prema optimumu, odnosno kako pridijeliti važnost pojedinim parametrima ovisno o tome koliko su duboko u mreži (što je pogotovo nezgodno ako se sjetimo da rezultat svakog prethodnog sloja utječe na idući).

Rješenje koje se danas najčešće koristi je algoritam *širenja unazad* (eng. *back-propagation*) [7], koji sa samo dva prolaska kroz mrežu izračuna lančanim pravilom gradijent funkcije gubitka za sve parametre. Potom se težine modificiraju oduzimanjem relevantnog dijela gradijenta pomnoženog s faktorom kojega nazivamo *stopa učenja* (eng. *learning rate*). Algoritam koji nam obavlja takav račun optimizacije se, logično, naziva *optimizator* (eng. *optimizer*) te detalji oko njegovog odabira i podešavanja čine jedne od najbitnijih hiperparametara svakog modela. Vrijedi dodati kako se u pravilu pri treniranju ne računa gradijent za čitavu epohu odjednom, već se događaji raspodijele u *serije* (eng. *batch*) pa se prije opisani postupak postepeno primjenjuje iz serije u seriju.

[4] Ona se gotovo uvijek sastoji od nezavisne primjene jedne realne funkcije na sve koordinate.

[5] Takva mreža je vrlo bliska strukturom živčanom sustavu živih bića, od kuda i dolazi ne samo inspiracija za, nego i naziv *neuralna* mreža.

3.3 Česte poteškoće u radu s neuralnim mrežama

No, čak i s riješenim problemom optimizacije danas se još uvijek susrećemo s nizom poteškoća u praktičnom radu s NM, tretiranje kojih zahtjeva uporabu dobro poznatih tehnika.

Moguće je da se pri računu gradijenata zbog gomilanja derivacija u lančanom pravilu oni postepeno povećavaju ili smanjuju kako algoritam dolazi do sve dubljih slojeva. To rezultira u *nestajućim/eksplodirajućim gradijentima* (eng. *vanishing/exploding gradients*). Takva nestabilnost u ponašanju gradijenata vodi efektivnom prestanku učenja u dubljim slojevima, odnosno nenadanim promjenama u ponašanju, pri čemu iduća epoha može biti drastično lošija od prethodne. Rješenje za ovakve probleme je pažljiv odabir aktivacijske funkcije, zajedno s pripadnom distribucijom iz koje će se generirati početne težine.

Dodatno, korištenje *sloja normalizacije serije* (eng. *batch normalization*, nadalje skraćeno na BN) se pokazuje [8] veoma korisnim jer zaglađuje prostor parametara, smanjujući vjerojatnost za nestabilno ponašanje gradijenata. Način na koji BN sloj radi je vrlo jednostavan: u trenutnoj seriji svaka se značajka pomiče i reskalira tako da na kraju njena srednja vrijednost bude nula, a standardna devijacija jedan. Danas je ovo široko prihvaćena tehnika, iako se još uvijek raspravlja o tome zašto točno ona pomaže u treniranju NM i treba li dolaziti prije ili poslije aktivacijskih funkcija.

Ako treniramo mrežu kroz dovoljno epoha, i ukoliko nismo naišli na probleme s gradijentima, sigurno će nam se pojaviti tzv. *pretreniranje* (eng. *overfitting*) gdje se NM toliko dobro prilagodila na podatke na kojima trenira da više nije u stanju generalizirati na slične događaje. Kako bismo lakše ustvrdili kad počinje pretreniranje dobra je praksa pripremiti uz set za treniranje i tzv. *validacijski* (eng. *validation*) set. Dok NM treniramo na prvom setu, njene performanse iz epohe u epohu pratimo na validacijskom setu, a kad se ona počne ponašati bolje na setu za treniranje, znamo da smo došli blizu pretreniranja, iako su još uvijek moguća poboljšanja u validaciji. No, to je vrlo sklizak teren te se jako brzo performanse NM mogu početi aktivno pogoršavati u validaciji – tada znamo zasigurno da smo zašli u pretreniranje.

Pojavu pretreniranja ne možemo odstraniti (osim ako imamo toliko jednostavan model da se on ne može ni potpuno natrenirati), ali ju možemo odgoditi. Osnovni savjet je da se pokuša pribaviti još podataka, što je nažalost rijetko moguće. Stoga,

treba primijeniti metode *regularizacije*, u koje spada i već spomenuti BN.

Jedna od njih je uvođenje slojeva *ispadanja* (eng. *dropout*) [9], koji svakom neuronu u prethodnom sloju daju određenu vjerojatnost (koja je hiperparametar) da će njegova vrijednost biti fiksirana na 0 u toj seriji (dok se aktivni neuroni skaliraju kako bi ukupni izlaz bio isti). Također imamo i opciju uvesti *regularizaciju težina*, pri čemu njihove norme (ili apsolutna l_1 ili kvadratna l_2 ili kombinacija) također doprinose konačnoj vrijednosti funkcije gubitka, što forsira parametre da ostanu relativno mali. Ukoliko, nakon isprobavanja svih ovih tehnika, još uvijek imamo problema s pretreniranjem potencijalno je potrebno pojednostaviti model.

Ako imamo zadatak odrediti pripadnost događaja pojedinim klasama, što se naziva klasifikacijom, u kojemu nisu sve klase jednako zastupljene u podacima, onda ćemo vrlo vjerojatno naići na teškoće u treniranju NM, od sporog napretka do potpunog neuspjeha. Kako bi ih ublažili možemo pokušati s inicijalizacijom pristranosti zadnjeg (izlaznog) sloja na takav način da mreža očekuje nejednake raspodjele klasa, što ima tendenciju ubrzati početak treniranja. Malo drastičnija metoda je uporaba *klasnih težina* (eng. *class weights*) pri čemu ćemo dati proporcionalno veći značaj pri treniranju slabije zastupljenim klasama. Konačno, moguće je pokušati *nad(pod)uzorkovati* događaje pojedine klase (eng. *over(under)sampling*) kako bi „na silu“ balansirali ulazni set događaja.

Na samome kraju moramo spomenuti i problem dimenzije ulaznih podataka. Što je ona veća, dakle, što imamo više značajki, i NM koju koristimo će morati biti veća (barem prvi slojevi), pa će treniranje duže trajati, ali ne samo to, već je i pretreniranje izraženije. Također je vjerojatno da će u većoj reprezentaciji podaci biti rijetki (eng. *sparse*), tj. da će ih veliki broj biti jednak nuli, što dovodi do dodatne poteškoće da se neuroni koji se vežu na često „prazne“ značajke ne treniraju.

Iz svih tih razloga redukcija dimenzionalnosti je jedan od čestih fokusa u predobradi podataka, pri čemu je od velike koristi *analiza važnosti značajki* (eng. *feature importance analysis*). U toj metodi ispitujemo utjecaj svake od značajki na performanse NM, a zasigurno je njena najjednostavnija implementacija evaluiranje modela na setu događaja (ovisno što nas zanima možemo odabrati validacijski, trening ili oboje) u kojem jednu po jednu značajku postavljamo na fiksnu vrijednost (u pravilu nulu). Tako dobivene vrijednosti, od kojih oduzmemo rezultat evaluacije na nemodificiranom setu podataka, potom možemo iskoristiti da odredimo koliko koja značajka ima utjecaja na

(ne)uspjeh rada NM te da eventualno odbacimo najnebitnije ili najštetnije značajke.

3.4 Konvolucijski slojevi

Kad koristimo guste slojeve nužno je preoblikovati ulazne podatke u vektore. U takvoj transformaciji gube se potencijalno iznimno bitne informacije o prostornoj međuovisnosti i rasporedu značajki. Kako bi NM dali ekvivalent vida[6] trebat ćemo koristiti drugačiju vrstu sloja, tzv. *konvolucijske slojeve* (eng. *convolutional layer*) koji, najgrublje rečeno, rade na sljedećem principu:

u njihovom izlazu postoji dodatna dimenzija koja odgovara broju tzv. *mapa značajki* (eng. *feature maps*) konvolucijskog sloja. Kako bi se generirala svaka mapa značajki u neuronima takvog sloja se odvija transformacija donekle nalik na onu u gustim slojevima:

$$c_{W,b}(X) = \phi(W \cdot X + b) \ , \tag{3.2}$$

osim što su sada W i X tenzori, dok je ulaz X vrlo specifično odabran. Njega čine najbliži susjedi[7] iz svih mapa značajki iz prethodnog sloja, što nazivamo *perceptivnim poljem* neurona te su njegove dimenzije unaprijed određene veličinom *konvolucijske jezgre* (eng. *kernel*). Kako bi se smanjio ukupni broj parametara svi neuroni u istoj mapi značajki dijele svoje težine i pristranosti. Proces se još malo može zakomplicirati odabirom ne-jediničnog *koraka* (eng. *stride*), koji regulira koliko se, pri prethodno opisanom uzimanju najbližih susjeda, pomičemo po dimenzijama ulaza za svaki jedinični pomak po izlazu. To rezultira smanjenjem dimenzija izlaza, osim zadnje.

Očito je da konvolucijski slojevi nisu potpuno povezani (jer svaki neuron „vidi“ samo svoje vlastito perceptivno polje) tako da oni mogu imati jako velike ulaze i izlaze bez korištenja iznimno puno parametara. Dodatna je prednost, zahvaljujući načinu na koji se izlaz konstruira, da se oni mogu natrenirati da prepoznaju uzorke, neovisno o njihovom položaju (za razliku od gustih slojeva koji su po tom pitanju nefleksibilni).

U kombinaciji s konvolucijom redovno se koriste *pooling layer-i*, koji rade na vrlo sličnom principu, a glavna je razlika da namjesto transformacije oblika 3.2 njihovi neuroni imaju za izlaz ili maksimum iz perceptivnog polja (*max pooling*) ili prosjek

[6] Zaista, konvolucijski slojevi su inspirani načinom rada vida u živim bićima, točnije načinom na koji se procesiraju vizualni podražaji.

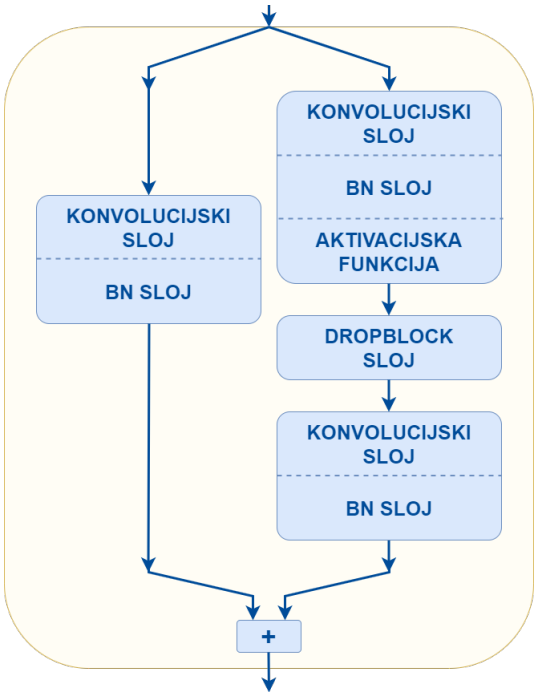
[7] Najbliži brojeći neovisno po koordinatama, tako da je ulaz uvijek oblika (hiper)kvadra.

(*avgerage pooling*). Njihova je glavna zadaća smanjiti dimenzije izlaza konvolucijskih slojeva (osim, opet, zadnje) i zato se gotovo uvijek koriste s ne-jediničnim korakom.

3.5 Rezidualne jedinice i preskočne veze

Do sada smo zapravo podrazumijevali da su nam svi modeli *uzastopni* (eng. *sequential*), gdje izlaz jednog sloja jednostavno postaje ulaz sljedećega. Treniranje takvih NM, iako uvođenje konvolucijskih slojeva omogućuje ostvarivanje veće dubine bez iznimne eskalacije broja parametara, se još uvijek može pokazati problematičnim [6].

Veliko poboljšanje u radu na dubokim modelima se postiže uporabom *preskočnih veza* (eng. *skip connections*), u kojima se, očekivano, izlaz nekih slojeva koristi kao ulaz u slojevima koji nisu nužno njima susjedni, i ne samo to, već je moguće imati i posve paralelne procese koji se tek kasnije u NM kombiniraju. Jednu od poznatijih implementacija preskočnih veza nalazimo u konstrukciji *rezidualne jedinice* (eng. *residual unit*), zahvaljujući kojima i dublji slojevi mogu početi učiti, čak i ako sama rezidualna jedinica nije još ostvarila napredak.



Slika 3.1: Shematski prikaz korištene modificirane rezidualne jedinice.

Rezidualna jedinica kakvu ćemo mi koristiti sastoji se od dvije paralelne transformacije, kao što je vidljivo sa slike 3.1, koje se zbrajaju prije izlaza. Svi konvolucijski slojevi imaju identitete za aktivacijske funkcije, a glavna grana računa se sastoji od

primjene dva konvolucijska sloja, iste veličine jezgri i broja mapa, s razlikom da prvi od njih može imati korak veći od jedan. Konvolucija u paraleli ima jediničnu jezgru i služi za očuvanje dimenzionalnosti. Dropblock je sloj sličan ispadanju, s glavnom razlikom da odjednom isključuje više susjednih neurona tzv. blokova, što rezultira boljim ponašanjem kad se koristi u kombinaciji s konvolucijskim slojevima [10].

4 Naš pristup problemu

4.1 Odabir metrike

Formalno govoreći, mi ovdje razvijamo binarni klasifikator s nadziranim učenjem. Izlaz takvog klasifikatora će uvijek biti samo realni broj, a mi se odlučujemo za interval $[0, 1]$, što se prirodno može interpretirati kao vjerojatnost pripadanja klasi[8]. Jedina funkcija gubitka koja u ovom slučaju ima smisla je *binary crossentropy*, koja drastično jače penalizira promašaje što su oni dalje od točne vrijednosti, što se vidi iz njene definicije:

$$f\left(N(x), y\right) = -\frac{1}{M}\sum_{i=1}^{M}\left(y_i \log\left(N(x)_i\right) + \left(1 - y_i\right)\log\left(1 - N(x)_i\right)\right) , \tag{4.1}$$

pri čemu su y točni rezultati za klase, $N(x)$ predviđanja za klase koje daje naša mreža i M ukupni broj događaja u seriji. Ipak, do konačne odluke klasifikator tek dolazi uspoređujući svoj izlaz sa zadanom vrijednosti *praga* (eng. *treshold*), pri čemu uzimamo da svi događaji s većom vrijednosti pripadaju jednoj klasi, a svi s manjom drugoj. Naravno, pri konačnom konfiguriranju klasifikatora imamo slobodu odabira praga, pri čemu je jasno da njegovim pomicanjem mijenjamo koliko će događaja biti svrstano u koju klasu, što onda pak utječe na različite metrike kojima bismo ocijenili NM. Sad se postavlja pitanje kako različite klasifikatore sustavno usporediti.

Ukoliko nas samo zanima koji je klasifikator bolji, najmudrija je opcija uopće ni ne razmatrati metrike koje ovise o odabiru praga, već koristiti veličine koje nam omogućuju njihovu direktnu usporedbu. Mi ćemo odabrati upravo jednu takvu metriku, a to je *AUC-ROC*. ROC stoji za eng. *receiver operating characteristic*, u kojoj se uspoređuje udio točnih i lažnih pozitiva za dani prag. Ukoliko stavimo te vrijednosti na graf dobivamo *ROC krivulju*, a potom integrirajući slijedi površina ispod ROC krivulje, AUC-ROC (AUC eng. *area under curve*), veličina koja ne ovisi o pragu te opisuje sposobnost klasifikatora da uspješno identificira pozitivnu klasu. Vrijednost od 1 znači da smo razvili savršen klasifikator, dok bi nasumični klasifikator imao AUC-ROC 0,5. Do konkretne AUC-ROC vrijednosti za različite NM doći ćemo uzimanjem prosjeka

[8] Za izlaz, u osnovi, možemo odabrati bilo koji interval, koji ne mora nužno biti ni ograničen, primjer čega je *logit* koji bi obuhvaćao sve realne brojeve. U tom slučaju, pripadnost klasi se isto odlučuje odabirom praga, te jedino što zapravo gubimo je praktična interpretacija izlaza. Zanimljivo je da se u neku ruku još uvijek se vodi rasprava oko toga je li preporučljivo (u smislu sveukupnog utjecaja na performanse klasifikatora) koristiti *logit-e* ili ne.

najboljih rezultata koje ona ostvaruje pri validaciji, a treniranje zaustavljamo ukoliko dolazi do *saturacije* (koju ćemo definirati kao 3 ili više epoha bez napretka u validaciji) ili *pretreniranja* (koje definiramo ili kao opetovani pad performansi na validaciji po epohama ili kao pojavu velike nestabilnosti u metrikama validacije). Zahvaljujući ovakvom odabiru načina usporedbe nećemo imati potrebe za dodatnom evaluacijom naših modela nakon kraja treniranja.

Kako bismo uzeli u obzir i praktični dio performansi NM, zanimat će nas i dodatne veličine: brzina treniranja po epohi, vrijeme potrebno da dođemo do *konvergencije* (koju definiramo kao broj epoha potreban da se pojavi saturacija ili pretreniranje) i vrijeme potrebno za evaulaciju, te, naravno, veličina (po pitanju memorije) setova u njihovim novim reprezentacijama.

4.2 Predselekcija reprezentacija

Sasvim je jasno da postoji proizvoljno mnogo reprezentacija koje bismo mogli ispitati pa se moramo na neki način ograničiti na one koje nam se čine najobećavajućima i praktičnima. Neke zanimljive reprezentacije, poput grafova [11] morat ćemo isključiti zbog egzotičnosti NM koje zahtijevaju. Druge pak, u kojima bi se pojedine čestice predstavljale geometrijskim likovima na slici [12], neće biti pogodne zbog većeg maksimalnog broja čestica (40) u pojedinim događajima u našem slučaju te zbog nedostatka prikladnih metrika koje bi diskretizirali i koristili kao osi slike. Dodatno, ako bi htjeli dobru rezoluciju u raspodjeli po osima to bi neminovno rezultiralo u velikoj rjetkoći podataka u događaju.

U konačnici smo se odlučili za reprezentacije u kojima su dimenzije određene ili brojem čestica ili brojem njihovih kategorija ili brojem veličina (varijabli) koje se vežu uz jednu četicu. Kao što ćemo vidjeti, s porastom tih dimenzija drastično rastu i vrijeme treniranja i veličina (u memoriji) događaja te se zato dodatno ograničavamo na dvodimenzionalne reprezentacije, tj. matrice ili slike.

Na kraju se postavlja pitanje čime popuniti tako odabranu formu reprezentacije, a tu se kao logični kandidati pojavljuju veličine izvedene iz 4-impulsa čestica, od kojih ćemo isprobati većinu onih koje se redovito koriste za opis događaja. Tu spadaju transverzalne veličine: *transverzalni impuls* $\vec{p}_T$ koji odgovara projekciji 3-impulsa čestica na ravninu okomitu smjeru sudara, *transverzalna energija* E_T koja je jednaka

iznosu transverzalnog impulsa, *ukupni transverzalni impuls mlaza*:

$$\vec{p}_T^{\,tot} = -\sum_{\text{čestice}} \vec{p}_T \,, \tag{4.2}$$

ukupna transverzalna energija kao iznos ukupnog transverzalnog impulsa te *transverzalna masa* definirana kao:

$$M_T = \sqrt{\left(E_T + E_T^{miss}\right)^2 - \left\|\vec{p}_T + \vec{p}_T^{\,miss}\right\|^2} \,. \tag{4.3}$$

Zatim, kutne veličine: *azimutalni kut* ϕ kojega transverzalni impuls zatvara u ravnini okomitoj na smjer sudara te *pseudorapiditet*

$$\eta = \operatorname{Arth}\left(\frac{p_L}{\|\vec{p}\|}\right) \,, \tag{4.4}$$

pri čemu je p_L longitudinalna[9] komponenta impulsa. I konačno dvočestične veličine: razlike prije spomenutih veličina i još dodatno, *dvočestična invarijantna masa* čestica idekasa i i j:

$$m_{i,j} = \sqrt{\left(E_i + E_j\right)^2 - \left\|\vec{p_i} + \vec{p_j}\right\|^2} \,. \tag{4.5}$$

4.3 Odabir i arhitektura mreža

Osnovna arhitektura NM koju ćemo iskoristiti bit će *višeslojni perceptron* (eng. *multilayer perceptron*, nadalje skraćeno na MLP). To je iznimno jednostavna uzastopna NM, sastavljena od gustih slojeva (s dodatkom BN, DO i svih drugih tehnika koje smo do sad opisali). MLP-ovi će nam dati osnovu naspram koje možemo odraditi uspješnost složenijih modela.

Nadalje, radit ćemo s *konvolucijskim NM* (eng. *convolutional neural network*, nadalje skraćeno na CNN). One će također biti uzastopne i sastavljene su osnovnih gradivnih blokova koji sadrže jedan ili više parova konvolucijskog sloja i BN, nakon kojih dolazi pooling sloj i dropblock. Više blokova zajedno čine konvolucijski dio CNN-a, pri čemu ćemo postepeno smanjivati dimenzije slike, a broj mapa povećavati. Drugi dio CNN-a čini upravo prijeopisani MLP.

Koristeći rezidualne jedinice formirat ćemo *rezidualnu mrežu* (eng. *residual neural*

[9] Longitudinalna u smislu da je to projekcija na smjer snopa.

network, nadalje skraćeno na ResNet), arhitekturu razvijenu specifično za klasifikaciju slika, varijanta koje je osvojila poznato ILSVRC natjecanje 2015. godine [13]. ResNet[10] se sastoji [14] od niza rezidualnih jedinica koje mogu imati pojedinačno različite veličine jezgri i korake (pri čemu ćemo kao i u CNN-u postepeno smanjivati dimenziju slike i povećavati broj mapa), nakon kojih se vrši *global pooling*, sloj vrlo sličan običnom *pooling-u*, osim što se za jezgru uzimaju cijele mape značajki, tako da je njegov izlaz samo vektor dimenzije jednake broju mapa zadnje rezidualne jedinice, a na kraju dolazi gusti sloj s jednim neuronom.

Za sve prethodne NM morat ćemo svladati problem nebalansiranih klasa, no postoji način da se o njima uopće ne moramo brinuti, a to su autoenkoderi, koji se konvencionalno treniraju samo na jednom tipu klase. Autoenkoder je NM koja se sastoji od dva dijela: enkodera i dekodera, koji komprimiraju, odnosno, dekomprimiraju događaje. Nakon što je natreniran na jednoj vrsti događaja, prateći grešku rekonstrukcije, autoenkoder možemo iskoristiti za detekciju anomalija, s očekivanjem da će se tipovi podataka na kojima autoenkoder nije treniran puno gore rekonstruirati. U slučaju sa samo dvije klase, detektor anomalija se svodi upravo na binarnu klasifikaciju, za što ćemo ga mi i koristiti. Detektori anomalija su općenito vrlo korisni u stvarnim eksperimentima kao filtri za vrlo rijetke događaje (ili pak za one koji još nisu zabilježeni), primjerice ako smo u potrazi za novom fizikom.

Glavna ideja ovdje je da postepeno smanjujemo ukupnu veličinu (produkt dimenzija) izlaza slojeva u enkoderu, te da u dekoderu na točno isti način nju postepeno vraćamo nazad. Tako da su u nekom smislu ta dva dijela zrcalne slike jedna druge. Autoenkodere možemo graditi na dva osnovna načina: MLP varijanta će koristiti samo niz gustih slojeva, dok CNN varijanta koristi kombinacije *pooling-a* i konvolucije za enkoder, odnosno *upsampling* i konvoluciju za dekoder (*upsampling* sloj povećava dimenzije ulaza umećući duplikate značajki te je u vrlo grubom smislu inverz *pooling-a*).

Ukoliko smo uspješni u radu s autoenkoderima to će značiti da smo u stanju prebaciti podatke u komprimiranu reprezentaciju sa vrlo malo gubitaka bitnih informacija, pri čemu je najveći gubitak najčešće samo šum (zato danas autoenkoderi imaju najširu primjenu upravo u odstranjivanju šuma). Potom, natrenirane autoenkodere, ili samo

[10] Nerijetko se bilo koja NM koja koristi rezidualne jedinice, ili općenito koncept rezidualnog učenja, u stručnom žargonu naziva ResNet-om, iako taj pojam zapravo podrazumjeva vrlo specifičnu arhitekturu. U ovom slučaju mi ćemo ovdje koristiti ResNet u tom drugom, užem smislu.

enkoderski dio, možemo iskoristiti za prvi dio drugih NM, što će smanjiti ukupno vrijeme treniranja i razvijanja te optimiziranja modela, a može i rezultirati boljim performansama zahvaljujući ugrađenoj sposobnosti odstranjenja šuma.

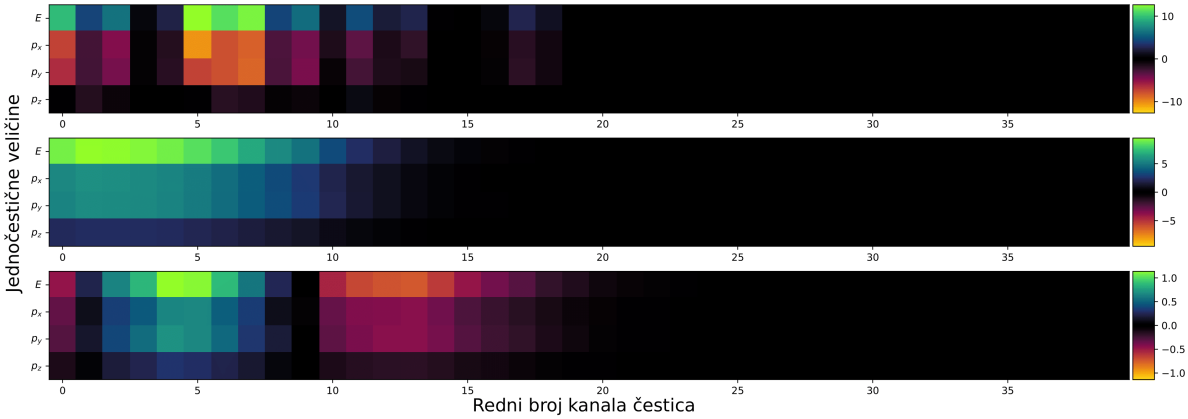
4.4 Proces treniranja modela

Kako bi ubrzali rad s modelima, za koje očekujemo da će ih biti mnogo, implementirat ćemo funkcije koje generiraju prethodno spomenute tipove arhitektura (MLP, CNN, ResNet, MLP autoenkoder, CNN autoenkoder). Njihovi argumenti, koji će će uključivati i one kojima kontroliramo oblik modela, ćemo razmatrati kao hiperparametre te ćemo veliku pažnju posvetiti tome da oni budu što bolje namješteni. No, obzirom da je broj njihovih kombinacija koje bismo trebali isprobati golem, nećemo koristiti tehnike poput *grid/random search cross-validaion* (pri čemu se ili sustavno ili nasumično pretražuje prostor hiperparametara trenirajući modele *cross-validacijom*) koje bi iziskivale nevjerojatno puno vremena, nego ćemo pretragu obaviti ručno, koristeći povratne informacije o performansama modela kako bi odabrali ispravku hiperparametara koja bi, nadamo se, rezultirala boljim modelom.

Opet iz razlga praktičnosti, prva ugađanja modela obavljat ćemo na malom setu za treniranje i validaciju (tzv. *toy dataset* – set-igračka, kod nas veličine 50 i 10 tisuća), sve dok nismo zadovoljni s postavkama hiperparametara modela koji bi se trebao lagano pretrenirati na tako malenim podacima. Prelazimo potom na pune setove od 250 (trening) i 50 (validacija) tisuća, na kojima obavljamo konačne popravke hiperparametara i treniramo finalnu verziju modela za dani tip arhitekture i danu reprezentaciju. Prelaskom na novu arhitekturu ili reprezentaciju ponovit ćemo ovaj proces.

5 Osnovni oblik podataka

Kako bi dobili osnovnu ideju koliko dobro možemo prilagoditi NM na dane podatke i koliki smo (ako upoće) napredak ostvarili u odabiru reprezentacija, započet ćemo s treniranjem MLP-a direktno na podacima kako su nam dani, primjer kojih je dan na slici 5.1. Ta reprezentacija je matričnog oblika s 40 čestičnih kanala (što je maksimalni broj čestica kojega imamo u jednom događaju) koji sačinjavaju stupce i vrijednostima 4-vektora koje popunjavaju retke.



Slika 5.1: Prikaz događaja u osnovnoj reprezentaciji, od gore prema dolje: jedan događaj s kvarkovskim mlazom; uprosječenje (apsolutnih vrijednosti) 25 tisuća događaja s gluonskim mlazovima; razlika apsolutnih uprosječenja gluonskih i kvarkovskih događaja prisutnih u validacijskom setu, u relativnoj skali od nula do jedan. Sve veličine koje imaju dimenziju su u GeV.

Možemo dodati kako bi se najbolji rezultati na ovakvom obliku podataka vjerojatno mogli postići korištenjem tzv. *deep set* mreže [15] koja je vrlo robusna[11] na promjenjiv ukupni broj ulaznih vektora (kao što bi ovdje bili varirajući brojevi detektiranih čestica u događajima, svaka sa svojim 4-impulsom). No, kako mi želimo istražiti reprezentacije u kojima smo upravo te promjene ukupnog broja čestica stavili pod kontrolu nećemo se u taj tip NM upuštati u ovom radu.

5.1 Odabir hiperparametara i skaliranja

Kako se bavimo s nejednakim udjelima klasama (imamo otprilike 13,5% udjela gluonskih mlazova), naišli smo na velike poteškoće u treniranju MLP-a bez uporabe

[11]Ta otpornost se postiže propuštanjem svih 4-impulsa kroz NM-ove koji ih prevode u D-dimenzionalne vektore, koji se potom svi sumiraju i propuštaju kroz drugu NM koja dolazi do konačne odluke po pitanju pripadnosti klasama. U neku ruku, kroz prvi korak u deep set mrežama mi prepuštamo algoritmu težak zadatak odabira i konstrukcije reprezentacija.

ikakvih dodatnih tehnika. Korištenje inicijalizacije pristranosti zadnjeg sloja t.d. NM očekuje nebalansirane klase imalo je malo efekta. Uključivanje klasnih težina, zahvaljujući kojima pri optimizaciji dajemo veći utjecaj manje zastupljenoj klasi je pomoglo, ali daleko je najbolji rezultat postignut ako smo unaprijed uravnotežili klase.

Obzirom da će vrijeme treniranja za najveću reprezentaciju na najvećoj mreži zasigurno biti vrlo veliko (i kao takvo predstavljati usko grlo) odlučujemo se za poduzorkvanje kvarkovskih mlazova ispuštanjem nekih njihovih događaja, jer tako iskorištavamo najveći broj događaja od slabije zastupljene klase dok istovremeno ukupna veličina setova ne raste. Zbog konzistencije ćemo fiksirati *seed* za generator permutacija kojima mješamo događaje tako da je svaka mreža trenirana i ocijenjena na točno istim događajima.

Glede odabira optimizatora, fino ugođeno *stohastičko spuštanje po gradijentu* (eng. *stochastic gradient descent*, nadalje skraćeno na SGD) pokazalo se efektivnijim i stabilnijim od danas preferiranog *ADAM* i *NADAM* optimizatora. Pod finim ugađanjem mislimo na pažljiv odabir rate učenja pomoću estimatora [16], optimizacije momenta SGD-a te *zakazivanja stope učenja* (eng. *learning rate scheduling*), dok je Nestorov trik bio kontraproduktivan i tako ostao isključen. Matematičke detalje ovih hiperparametara i njihove implikacije na treniranje NM zbog dužine nećemo raspraviti, ali oni se lako mogu naći u literaturi [5, 6].

Kao što možemo i očekivati [17] ELU aktivacijska funkcija s He distribucijom za težine (bilo uniformnom ili normalnom) je bila daleko najbolja opcija pružajući bržu konvergenciju i stabilnije ponašanje u treniranju. Kako bi smanjili utjecaj šuma na našu NM, iskušali smo i dvostruko-temperiranu modifikaciju *binarne crossentropy-je* [18] za funkciju gubitka, ali nismo imali pretjerano uspjeha s njom. Od regularizacijskih tehnika najuspješnija je bila kombinacija BN i ispadanja, dok je uvođenje bilo koje regularizacije težina slojeva bilo štetno po performanse NM.

Skaliranje podataka, pod čime mislimo na modificiranje distribucija vrijednosti značajki, što je inače iznimno bitno za dobar rad NM, se pokazalo problematičnim. Stvar je u tome što ova reprezentacija nema nikakve strukture u rasporedu čestica po drugoj dimenziji, (koja se jednostavno puni redom), niti su veličine omeđene (kao što bi primjerice, kutne veličine bile). Zato su pokušaji reskaliranja individualnih značajki, bilo *min-max* ili *standardno* skaliranje – što je i najčešća praksa [6], rezultirali daleko lošijim ishodima treniranja, ali to nije ni začuđujuće jer smo tu u osnovi „na silu“

istovremeno mijenjali vrijednosti 4-impulsa fundamentalno različitih čestica. Jedina varijanta skaliranja koja je pružila ikakvo poboljšanje bilo je globalno skaliranje, pri čemu smo svaku komponentu 4-vektora dijelili s njezinim maksimumom iz čitavog seta. Upravo iz ovog razloga manjka strukture i prostornog rasporeda značajki u događajima, su nam se CNN-ovi u najboljem slučaju ponašali kao i MLP-ovi na ovoj reprezentaciji te ih zato dodatno ne navodimo u raspravi o osnovnom obliku podataka.

Prelazak u sustav centra momenta i korištenje u sfernog sustava, koje svakom fizičaru prirodno dolaze, nisu pružili nikakvo primjetno poboljšanje u radu NM.

Od generalnih oblika MLP-a, pod čime mislimo na to kako se širine slojeva, točnije brojevi neurona, mijenjaju ovisno o njihovoj dubini u mreži, koristili smo „piramidalni“ oblik (u kojem slojevi kontinuirano postaju uži od ulaza do izlaza) i „bačvasti“ (u kojem slojevi do pola puta prema izlazu postaju širi, a potom se sužavaju), od kojih je drugi donio bolje rezultate.

5.2 Rezultati za MLP koristeći 4-vektore

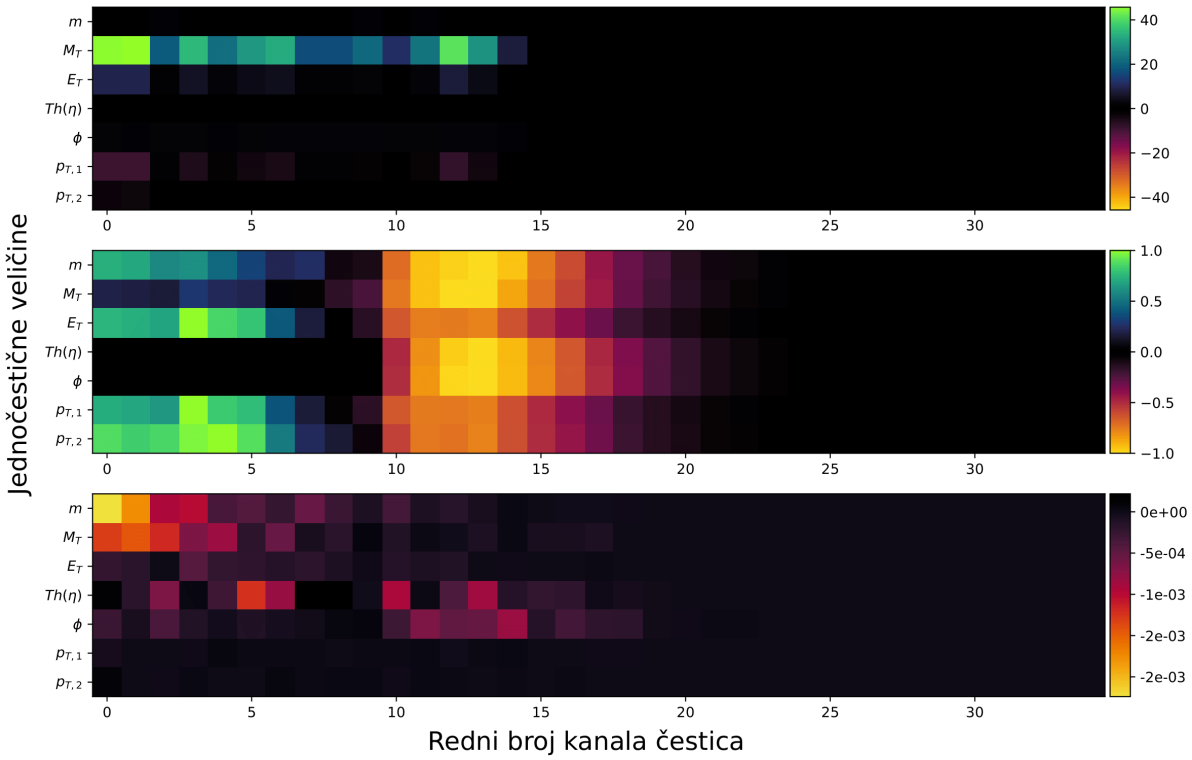
U konačnici, s prije opisanim odabirima različitih hiperparametara i skaliranja, najuspješniji MLP imao je približno 2,6 milijuna parametara i bio je sastavljen od pet blokova te izlaznog gustog sloja s jednim neuronom i sigmoidalnom aktivacijskom funkcijom. Svaki blok građen je redom od gustog sloja (s aktivacijom), BN i sloja ispadanja. Broj neurona (jedinica) gustih slojeva bio je redom: 576, 768, 1024, 768, 576, 1. Treniranje po epohi trajalo je 115 sekundi u prosjeku, a do saturacije se stizalo oko 25. epohe. AUC-ROC pri saturaciji bio je 0,661.

5.3 Uvođenje izvedenih jednočestičnih veličina i selekcija značajki

Kako bismo ispitali smislenost uvođenja jednočestičnih veličina izvedenih iz 4-impulsa čestica, uvest ćemo malu modifikaciju u trenutnu reprezentaciju te preći na iduće varijable: masu (m), transverzalnu masu (M_T), transverzalnu energiju (E_T), omjer longitudinalnog i ukupnog 3-impulsa, što upravo odgovara uzimanju tangensa hiperbolnog pseudorapiditata ($\mathrm{Th}(\eta)$), azimutalni kut (ϕ), te prve i druge komponente transverzalnog impulsa ($p_{T,1}$ i $p_{T,2}$).

Tako dobivenu reprezentaciju smo iskoristili u treniranju MLP-a (pri čemu se generalno isti odabir hiperparametara kao i kod prethodnog pokazao optimalnim), i

obavili analizu važnosti značajki, koja je, zajedno s primjerom nove reprezentacije prikazana[12] na slici 5.2.



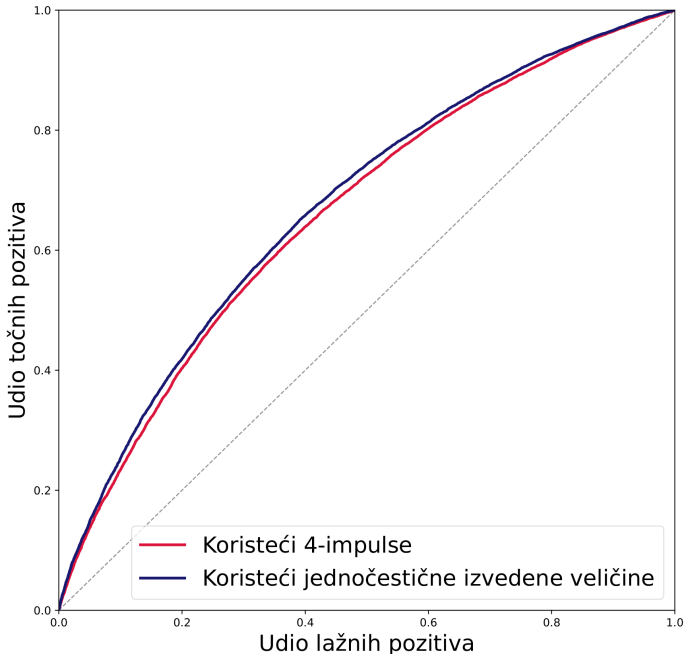
Slika 5.2: Prikaz reprezentacije koja koristi izvedene jednočestične veličine i analiza važnosti značajki: od gore prema dolje: jedan događaj s kvarkovskim mlazom (bez skaliranja, sve ne-kutne veličine su u GeV); razlika apsolutnih uprosječenja gluonskih i kvarkovskih događaja prisutnih u validacijskom setu, u relativnoj skali od nula do jedan; rezultati analize važnosti značajki za AUC-ROC metriku.

Kao što je vidljivo sa zadnjeg grafa na slici 5.2, nemaju sve značajke isti utjecaj na sposobnost kategorizacije MLP-a te se to samo donekle prostorno poklapa s razlikama uprosječenja (te razlike naravno ne daju cijelu priču, ali su dobar indikator kako generalno klase događaja izgledaju). Daleko je najutjecajnija masa čestica, što možemo objasniti činjenicom da je ona, u nedostatku informacije o bilo kakvim kvantnim brojevima, jedini način kako identificirati čestice. Zanimljivo je kako, iako imamo jasno vidljive razlike u uprosječenjima komponenata transverzalnog impulsa, one iznimno slabo, ako i uopće, doprinose sposobnosti NM da donese točnu odluku. Čini se kao da vektorske veličine nisu vrlo korisne MLP-u te da bi uporaba samo skalarnih veličina koje su tu već prisutne trebala dati bolje rezultate.

[12]Isti je događaj korišten kao primjer u svim grafovima. Također, kao što je već bilo spomenuto, sve NM koriste isti odabir događaja za validaciju i trening, tako da i grafovi razlike uprosječenja prikazuju iste podatke, samo u drugim reprezentacijama.

5.4 *Rezultati za MLP koristeći izvedene veličine*

Dodatno smo potvrdili zaključke iz prethodne analize treniranjem nekoliko MLP-a na različitim odabirima izvedenih varijabli, od kojih je zaista najbolje ponašanje imala reprezentacija sastavljena od samo skalarnih veličina (dakle, ona od $m, M_T, E_T, \mathrm{Th}(\eta), \phi$). Najbolji MLP na toj reprezentaciji bio je vrlo sličan prethodno opisanom, ali s malo lakšom arhitekturom (pri čemu su brojevi neurona bili: 256, 512, 1024, 512, 256, 1). Stoga je palo i vrijeme treniranja po epohi na ≈69 sekundi, a do saturacije se stizalo isto oko 25. epohe. AUC-ROC vrijednost je bila 0,674 te predstavlja malo poboljšanje, od oko 2%. Usporedba ROC krivulja dva MLP-a dana je na slici 5.3.



Slika 5.3: Graf ROC krivulja MLP-ova treniranih na osnovnom obliku podataka. Savršeni klasifikator imao bi krivulju što bližu gornjem lijevom uglu, dok je krivulja za posve nasumičan klasifikator ravni pravac, prikazan iscrtkano na grafu.

6 Reprezentacija sa svrstavanjem po masama

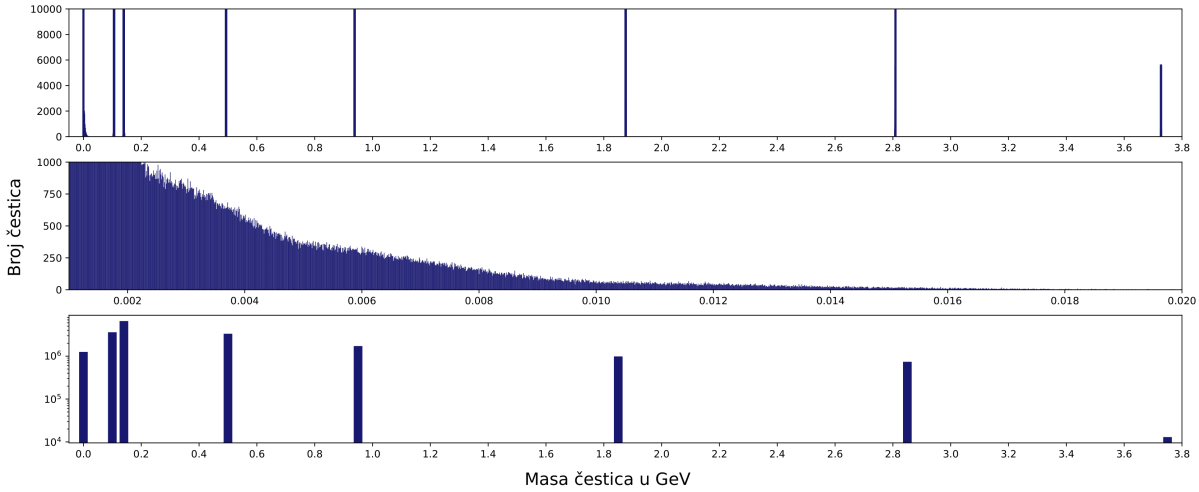
6.1 Mase čestica i dobivanje nove reprezentacije

Prethodno smo vidjeli kako je masa čestica najbitnija veličina pri klasifikaciji. Dodatno, kako očekujemo samo konačni (i relativno mali) broj različitih masa u svim događajima, ona bi trebala biti iznimno pogodna za njihovo svrstavanje u kategorije (*binning*), što se potom može iskoristiti za definiranje jedne ili obje dimenzije reprezentacije.

Računajući normu 4-impulsa dobili smo mase svih čestica, rezultati čega su prikazani na grafu na slici 6.1. Zahvaljujući tome da raspodjela masa ima dobro definirane šiljke centrirane na samo 8 vrijednosti lako smo ih smjestitili u kategorije. Ipak, ne možemo nikako tvrditi da smo identificirali individualne vrste čestica, ako ništa drugo onda zbog šuma oko nule, pri čemu veliki broj različitih čestica može biti obuhvaćen. Dodatno, kako nemamo nikakve kvantne brojeve, ne postoji ni način kojim bismo razlikovali čestice vrlo slične mase kojima je neki naboj jedina razlika (uključujući razlikovanje čestica od antičestica). Konačne kategorije su sljedeće:

Približna pozicija vrha	Najvjerojatnije čestice u kategoriji	Udio u ukupnim česticama iz svih događaja
$<$20 Mev	različite bezmasene i lagane čestice	6,9%
100 MeV	mioni	19,1%
140 MeV	pioni	36,1%
0,5 GeV	kaoni	18,3%
0,94 GeV	neutroni i protoni	9,4%
1,85 GeV	D-mezoni	5,4%
2,8 GeV	J/Ψ	4,0%
3,75 GeV	J/Ψ rezonanca	0,1%

U sljedećem koraku prema novoj reprezentaciji prošli smo kroz sve događaje i identificirali maksimalni broj puta koliko se pojedina kategorija pojavljuje u jednom događaju. Zbrojimo li sve maksimume dobivamo 94, što će biti jedna dimenzija te reprezentacije, u kojoj će dijelovi matrice biti rezervirani za pojedine kategorije. Duljina svakoga dijela će odgovarati najvećem broju puta koliko se ta kategorija čestice može pojaviti u jednom događaju. Tako čestice (odnosno, veličine vezane uz njih) pripadne istoj kategoriji uvijek popunjavaju, sortirane po energiji, iste lokacije u reprezentaciji. Konačno preostaje samo odabrati jednočestične veličine koje će



Slika 6.1: Graf raspodjele masa čestica i njihovo združivanje u kategorije. Od gore prema dolje: histogram rezultata za razdiobu masa; isti histogram, fokusiran na šum oko nule; konačne kategorije i njihove zastupljenosti.

odrediti drugu dimenziju matrice. Primjer jedne ovakve reprezentacije je dan na slici 6.2.

U ovom novom obliku reprezentacije uvodimo vlastito skaliranje individualnih značajki. Ono može biti ili *min-max* (sve značajke su preslikane na interval od 0 do 1) ili *standardno* (koje transformira značajke tako da je srednja vrijednost njihove raspodjele 0 i standardna devijacija 1). Naravno, veličine kojima skaliramo određuju se samo na trening setu te ih potom primjenjujemo i na njega i na validaciju. Ključna razlika od gotovih funkcija za skaliranje, kakve bi primjerice mogli naći u *Scikit* paketu, jest u tome da ne mijenjamo vrijednosti značajki ukoliko su one jednake nuli, preciznije, ako nema zabilježene čestice u kanalu kojemu ta značajaka pripada. Tako garantiramo da je oblik događaja očuvan kroz skaliranje, što je značajno pomoglo performansama NM.

Slika 6.2: Primjer reprezentacije s 94 čestična kanala baziranih na njihovim masama. Od gore prema dolje: jedan događaj s kvarkovskim mlazom (bez skaliranja, M_T i E_T su u GeV, a ϕ u radijanima); isti događaj na kojega je primjenjeno naše modificirano min-max skaliranje; razlika apsolutnih uprosječenja reskaliranih gluonskih i kvarkovskih događaja prisutnih u validacijskom setu.

6.2 Rezultati za MLP

Najuspješniji MLP-ovi koji su korisitli novi oblik događaja su općenito slični prethodno korištenima: ELU aktivacijske funkcije s He distribucijom, BN i ispadanje za regularizaciju (regularizacija težina se pokazala detrimentalnom), SGD optimizator s optimizacijom momenta i stopom raspada stope učenja. Sastavljeni su od istih osnovnih blokova od kojih svaki uključuje jedan gusti sloj, BN i sloj ispadanja, a na kraju imamo jedan gusti sloj s jednim neuronom i sigmoidalnom aktivacijskom funkcijom.

Za odabir 4-vektora kao jednočestičnih varijabli koje popunjavaju reprezentaciju najbolji MLP imao je približno 2,3 milijuna parametara sa sljedećim brojevima neurona: 512, 1024, 1024, 512, 1. Treniranje je trajalo oko 96 sekundi po epohi, a saturacija se postizala do 15. epohe. AUC-ROC je bio 0,726, što predstavlja poboljšanje od $\approx$10% u odnosu na osnovu[13].

Iz smanjenja broja epoha potrebnih za konvergenciju te povećanja kvalitete klasifikatora možemo zaključiti kako je već samo promjena oblika podataka (što je omogućilo i uvođenje boljeg skaliranja) dovoljna za veliki napredak u uporabljivosti događaja pri treniranju NM.

Odabir izvedenih varijabli koji je dao najveći uspjeh pokazao se istim kao i u prethodnoj raspravi: $M_T, E_T, \mathrm{Th}(\eta), \phi$. Najbolji MLP natreniran na takvoj reprezentaciji imao je približno 2,7 milijuna parametara sa sljedećim brojevima neurona: 576, 768, 1024, 768, 576, 1. Treniranje je trajalo oko 118 sekundi po epohi, a saturacija se isto postizala do 15. epohe. AUC-ROC je bio 0,731 što je $\approx$11% bolje od osnove, ali unutar 1% od odabira 4-vektora kao varijabli. Stoga, možemo tvrditi kako oblik reprezentacije ima daleko najviše utjecaja na uspjeh NM, ili barem MLP-a.

6.3 Rezultati za autoenkodere

Iduće smo se posvetili treniranju autoenkodera, ovaj puta koristeći samo uspješniji set varijabli sastavljen od izvedenih veličina ($M_T, E_T, \mathrm{Th}(\eta), \phi$). Kako bi definirali grešku rekonstrukcije iskušali smo srednju kvadratnu grešku (eng. *mean square error*), što koristimo i kao funkciju gubitka u treniranju autoenkodera, te *kosinusnu sličnost* (eng. *cosine similarity*), koja računa kut među tenzorima te eventualnu kombinaciju njih

[13] Pod osnovom ćemo uvijek podrazumijevati najbolji MLP treniran na osnovnom obliku podataka popunjenog 4-impulsima čestica.

dvoje. Ispalo je da je samo uporaba srednje kvadratne greške najbolja, a kako bi od nje došli do metrika za autoenkoder-klasifikator koristiti ćemo funkcije za analizu iz *Scikit-learn* paketa.

Vrlo je brzo postalo jasno kako imamo istovremeno iznimno jaku pojavu pretreniranja, zajedno s poprilično lošim rezultatima klasifikacije. Kako bi smanjili pretreniranje uveli smo regularizacijske tehnike, od koji se ponovo najboljom pokazala kombinacija BN i ispadanja. Također smo probali trenirati autoenkoder i na manje zastupljenoj gluonskoj klasi, što je neuobičajeno, ali smo se nadali da ona možda ima ujednačenije događaje. Moguće je da smo u pravu, jer su rezultati za autoenkodere trenirane na gluonskim događajima su bili malo bolji, ali su još uvijek po pitanju klasifikacije daleko od čak i osnovnog MLP-a.

Najbolji MLP autoenkoder bio je relativno plitak, sa svega tri sloja u enkoderu i isto tri u dekoderu, a brojevi neurona su sljedeći: 300, 225, 168, 225, 300, 376, sa sveukupno $\approx$440 tisuća parametara. Faktor kompresije ulazne reprezentacije je bio blizu 3. Vrijeme treniranja po epohi bilo je oko 40 sekundi, a pretreniranje je nastupalo jako rano, oko pete epohe. AUC-ROC vrijednost bila je jedva bolja od nasumičnog klasifikatora s 0,566, tako predstavljajući pogoršanje od 14% u odnosu na osnovu.

Prelaskom na CNN autoenkoder problemi su se ponovili, a najbolje rezultate smo postigli preoblikovanjem reprezentacije u kvadratnu matricu dimenzija 20 (dodatne značajke su popunjene nulama), što je omogućilo uporabu većih konvolucijskih jezgri. Enkoder je bio građen od dva bloka koji su se sastojali svaki od dva uzastopna konvolucijska sloja i jednog *max pooling* sloja. Dekoder je slično bio sastavljen od dva bloka sačinjena od *upsampling* sloja kojega je slijedio par konvolucijskih slojeva. Nakon tih blokova u dekoderu dodan je još jedan konvolucijski sloj bez aktivacije koji osigurava pravilnu dimenziju.

Konvolucijski slojevi imali su ELU aktivaciju, jedinični korak i 3×3 jezgru, a *max pooling* sloj imao je 2×2 jezgru i korak od 2, tako da se dimenzija slike prepolovila nakon svake primjene *pooling* sloja. Broj mapa konvolucijskih slojeva kroz čitav autoenkoder bio je sljedeći: 64, 32, 8, 4, 4, 8, 32, 64, 1. Tako dobiveni model je vrlo malen, sa svega 44 tisuće parametara, što je bilo nužno zbog pretreniranja. Komprimirana reprezentacija bila je oblika $5\times5\times4$, 4 puta manja od ulazne.

Vrijeme treniranja po epohi bilo je 44 sekunde, a saturacija se postizala nakon 6.

epohe. Rezultat za AUC-ROC je 0,595 – bolje od MLP autoenkodera, ali još uvijek daleko od osnove, u usporedbi s kojom imamo pogoršanje od skoro 10%.

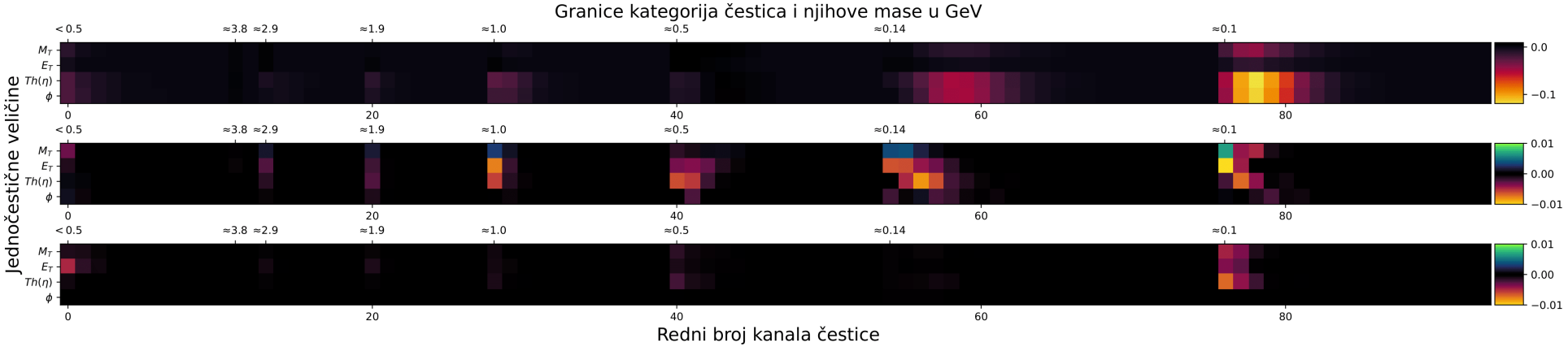
Nakon ovako loših rezultata s relativno jednostavnim autoenkoderima, kod kojih je problem pretreniranja bio persistentan, odlučili smo se da nećemo dalje pokušavati sa složenijim arhitekturama autoenkodera te smo odustali od pokušaja eventualne implementacije autoenkodera kao prvih slojeva drugih mreža. Vjerojatno objašnjenje koje možemo dati za ovakvo ponašanje je to da svaku klasu čini više složenih procesa koji mogu rezultirati veoma različitim izlaznim česticama. To pak za posljedicu ima veću raznolikost u podacima iste klase, koja je možda i usporediva s razlikama među klasama, što neminovno izazove poteškoće u radu autoenkodera. No takve bi tvrdnje trebalo, naravno, dodatno ispitati.

6.4 Rezultati za CNN

Kao što smo već naglasili, nova reprezenatcija s 94 kanala ima puno veću prostornu uređenost u položaju značajki te se nadamo da će CNN to uspješno iskoristiti. Trenirat ćemo ga koristeći izvedene varijable ($M_T, E_T, \mathrm{Th}(\eta), \phi$), na 94×4 formatu, kao i na preoblikovanju na 19×20, pri čemu ćemo moći koristiti veće jezgre i korake. U tom preoblikovanju namjerno je odabrana dimenzija 20 kako bi se iste varijable uvijek pojavljivale u istim recima, a dodane značajke su popunjene nulama. Individualne značajke su skalirane prije opisanim vlastitim skaliranjem.

Konvolucijski dio najuspješnijeg CNN-a (koji je treniran na preoblikovanim podacima) bio je sastavljen od tri bloka, svaki od kojih je građen redom od: konvolucijskog sloja, BN, konvolucijskog sloja, BN, *max pooling* sloja i *dropblock* sloja. Drugi, MLP dio, isto čine blokovi, njih pet, sastavljenih od po jednog gustog sloja, BN i sloja ispadanja. Zadnji, izlazni sloj, je gusti s jednim neuronom i sigmiodalnom aktivacijskom funkcijom. Veličine jezgri za konvoluciju i *pooling* te korak *pooling-a* su 2×2. Svi slojevi koriste ELU aktivacijsku funkciju.

Dimenzije izlaza pojedinih konvolucijskih blokova (zadnja vrijednost je broj mapa) su: 9×10×64; 4×5×128; 2×2×256. Broj neurona gustih slojeva su: 128, 256, 512, 256, 128, 1. Ukupno imamo 980 tisuća parametara, a vrijeme treniranja po epohi iznosilo je 305 sekundi.



Slika 6.3: Rezultati analize važnosti značajki za reprezentaciju s 94 kanala. Od gore prema dolje: razlika apsolutnih uprosječenja reskaliranih gluonskih i kvarkovskih događaja prisutnih u validacijskom setu; rezultati analize važnosti značajki za AUC-ROC metriku za MLP; rezultati analize važnosti značajki za AUC-ROC metriku za CNN.

CNN-ovi su nam se brže i jače prilagođavali na podatke, pogotovo na preoblikovanim, tako da smo imali pojavu pretreniranja već između 10. i 15. epohe. Vrijednosti za AUC-ROC bile su 0,741 (obični oblik) i za 0,747 (preoblikovani), što predstavlja 13% poboljšanja u odnosu na osnovu, i oko 3% naspram MLP-a.

Konačno, sproveli smo analizu važnosti značajki za MLP i CNN, rezultati koje su dani na slici 6.2. Tu jasno vidimo koliko su konvolucijske mreže robusnije, jer mijenjanje jedne značajke je rijetko imalo katastrofalnog utjecaja. Zanimljivo je kako ispada da su čestice mase $\approx$0,1GeV, što ugrubo odgovara μ-leptonima, najbitnije za razlučivanje klasa. Očito, do njih dolazimo raspadima piona, kojih u mlazovima ima napretek, no nije jasno zašto bi ti raspadi bili puno jači/češći u kvarkovskim mlazovima.

6.5 Rezultati za ResNet

Najbolji ResNet smo natrenirali na preoblikovanim podacima i bio je sastavljen od 8 modificiranih rezidualnih jedinica, čiji su izlazi bili redom: $20\times19\times64$ (triput ponovljeno); $10\times10\times128$ (triput ponovljeno); $5\times5\times128$; $3\times3\times256$. Sveukupno je imao 2,3 milijuna parametara, s približno 300 sekundi vremena treniranja po epohi. Vrijedi napomenuti kako je, zbog manje veličine jezgri koje smo mogli primijeniti, što je usporilo redukciju dimenzije slike u mreži, treniranje na neoblikovanim podacima trajalo dosta duže, oko 455 sekundi po epohi.

Ovo je zaista vrlo moćna arhitektura, što se očituje da za već ovako mali ResNet[14] imamo nevjerojatno brzu pojavu vrlo jakog pretreniranja već prije 10. epohe, te u tome da mreža može nastaviti s dobrim napretkom u treniranju kroz još puno epoha, daleko od drugih (iako je sav taj napredak beskoristan zbog pretreniranja). Jedini nedostatak je veća nestabilnost koju smo primijetili prateći validaciju, pri čemu se pojavljuju nezanemarivi jedno-generacijski skokovi u funkciji gubitka.

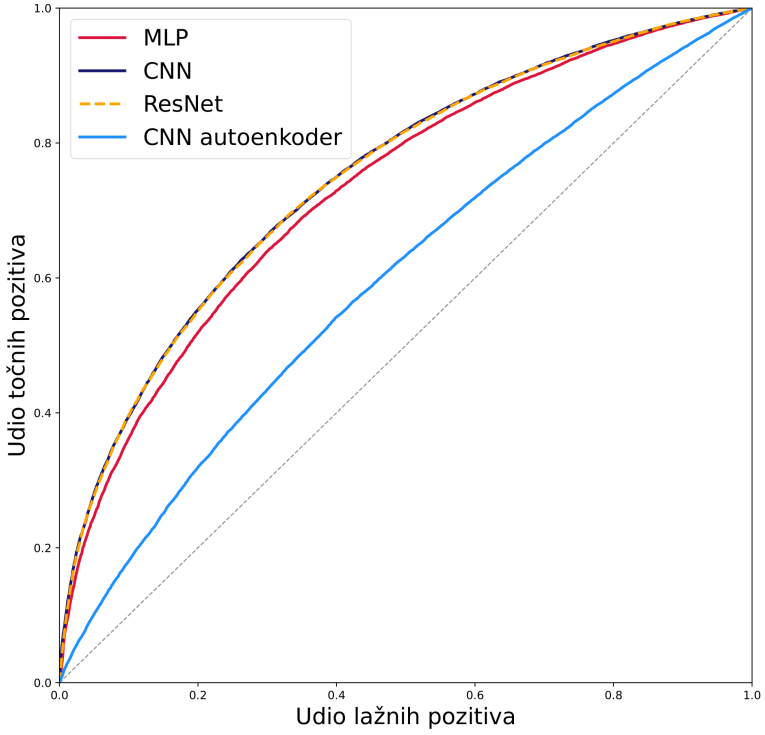
AUC-ROC na običnim podacima bio je 0,737, dok je na preoblikovanim bio 0,749. To je 13% bolje od osnove i 3% bolje od MLP-a na istoj reprezentaciji. Ti rezultati su iznimno blizu CNN-a te možemo tvrditi da smo „iskoristili“ sve korisne relacije među značajkama na koje se mreža mogla natrenirati, odnosno kako smo možda došli do najboljih mogućih performansi na reprezentaciji na kojoj smo radili. No, takve tvrdnje za sada ostaju samo hipotetske te bi ih trebalo dodatno provjeriti.

[14] ResNet-i u pravilu imaju stotine slojeva.

6.6 *Rezime rezultata*

Rekapitulirajmo dosadašnje rezultate. Već samo prelaskom na novi oblik s 94 kanala postigli smo poboljšanje od 10%, dok smo korištenjem boljih NM uspjeli drastično smanjiti broj epoha potrebnih za konvergenciju i stigli do ≈0,75 vrijednosti za AUC-ROC. ROC krivulje za najbolje tipove NM dane su na slici 6.4, a pripadne AUC-ROC vrijednosti su:

Mreža	AUC-ROC	Promjena u odnosu na osnovu
MLP	0,731	+11%
Autoenkoder CNN	0,595	-10%
CNN	0,747	+13%
ResNet	0,749	+13%



Slika 6.4: Graf ROC krivulja različitih NM treniranih na reprezentaciji s 94 kanala: ovisnost udjela točnih pozitiva o udjelu lažnih. Savršeni klasifikator imao bi krivulju što bližu gornjem lijevom uglu, dok je krivulja za posve nasumičan klasifikator ravni pravac, prikazan iscrtkano na grafu.

7 Matrica rapiditeta i masa

7.1 Definicija i generiranje matrice rapiditeta i masa

S jedne strane, želimo li proširiti reprezentaciju iz prethodnog poglavlja, prirodno se nameće korištenje 94 kanala čestica kategoriziranih po masama za obje dimenzije slike. S druge strane, vrlo je slična reprezentacija već dobro poznata u fizici visokih energija, te je nedavno ostvarila dosta dobre rezultate na vrlo sličnom problemu [19]. Tako da je sasvim logično proučiti tzv. *matricu masa i rapiditeta* (eng. *rapidity-mass matrix*, nadalje skraćeno na RMM), iako ju iz praktičnih razloga za naše potrebe treba malo modificirati.

$$\begin{bmatrix} e_T^{miss} & m_{T,1} & m_{T,2} & \dots & m_{T,n} & m_{T,n+1} & \dots \\ \mathrm{Th}(\eta_1) & e_{T,1} & m_{1,2} & \dots & m_{1,n} & m_{1,n+1} & \dots \\ \mathrm{Th}(\eta_2) & \delta\mathrm{Th}(\eta_{1,2}) & \delta e_{T,2} & \dots & m_{2,n} & m_{2,n+1} & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ \mathrm{Th}(\eta_n) & \delta\mathrm{Th}(\eta_{1,n}) & \delta\mathrm{Th}(\eta_{2,n}) & \dots & \delta e_{T,n} & m_{n,n+1} & \dots \\ \mathrm{Th}(\eta_{n+1}) & \delta\mathrm{Th}(\eta_{1,n+1}) & \delta\mathrm{Th}(\eta_{2,n+1}) & \dots & \delta\mathrm{Th}(\eta_{n,n+1}) & e_{T,n+1} & \dots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \ddots \end{bmatrix} \tag{7.1}$$

Naša RMM, prikazana s 7.1, ima 95 redaka i stupaca, pri čemu su prvi redak i stupac umetnuti kako bi mogli navesti jednočestične veličine. Ostatak matrice prati poredak čestica iz prethodne reprezentacije s jedinom razlikom da su sad čestice unutar kategorija sortirane po transverzalnoj energiji E_T. Prvo mjesto u matrici zauzima ukupna transverzalna energija podijeljena s energijom u sustavu centra momenta $\sqrt{s}$ (zato je označena malim slovom). Nakon toga, prvi redak popunjavaju transverzalne mase individualnih čestica, ponovo dijeljene sa $\sqrt{s}$, a prvi stupac tangensi hiperbolni pseudorapiditeta, što predstavlja glavnu razliku od konvencionalne RMM [20]. Ako bi htjeli dobiti varijantu rapiditeta koja se inače koristi trebali bi primjeniti areatangens hiperbolni pa kosinus hiperbolni, no najviše što bi ta transformacija napravila bilo bi povećati rezoluciju blizu jedinice i preslikati vrijednosti na sve realne brojeve. Nama su zapravo obje te promjene nepoželjne, prva jer nam većina vrijednosti leži u neposrednoj okolini nule i radi toga nemamo problema s vrlo gustom razdiobom

oko ± 1, a druga jer bismo morali uvesti dodatno skaliranje (nešto što konstrukcijom RMM upravo i želimo izbjeći).

Dijagonala, ukoliko smo na mjestu prve čestice u kategoriji, je sačinjena od skaliranih transverzalnih energija $e_T = E_T/\sqrt{s}$, ili, u narednim mjestima u kategoriji, skaliranih razlika transverzalnih energija:

$$\delta e_{T,n} = \frac{E_T(n-1) - E_T(n)}{E_T(n-1) + E_T(n)}\,. \tag{7.2}$$

Ostakak gornjeg trokuta popunjen je dvočestičnim invarijantnim masama, isto podijeljenima s $\sqrt{s}$, a donji trokut čine sljedeće razlike:

$$\delta\mathrm{Th}(\eta_{n,n+1}) = \mathrm{Th}(\eta_n) - \mathrm{Th}(\eta_{n+1})\,. \tag{7.3}$$

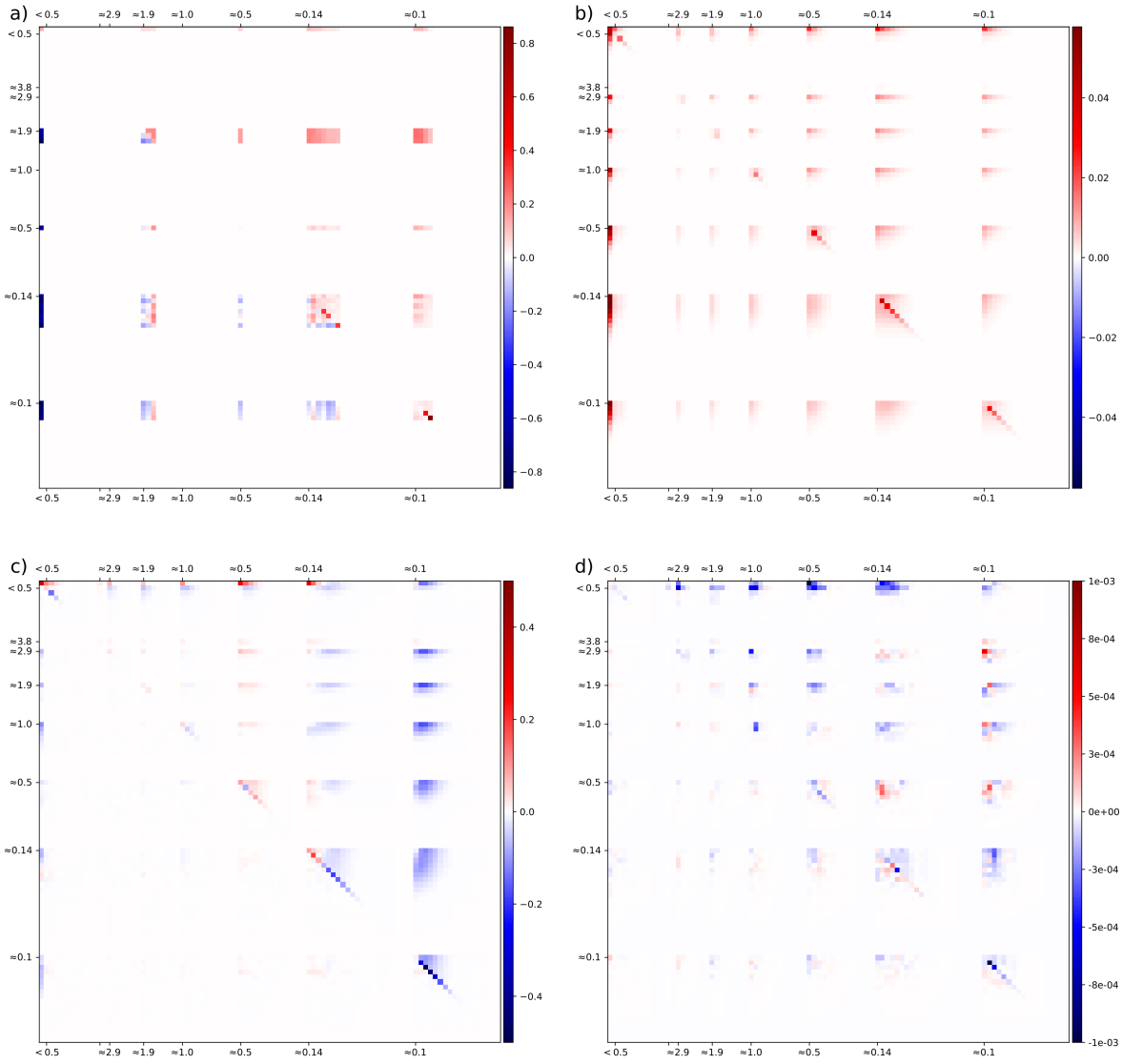
Ovako konstruirana RMM, na primjeru događaja koji je dan na slici 7.1, ima dobra statistička svojstva (poput nezavisnosti veličina), sastavljena je od teorijski vrlo bitnih veličina [20], te je unaprijed skalirana, što sve ide u prilog tomu da bi ona trebala biti vrlo korisna reprezentacija, kao i činjenica da su u spomenutim radovima postignuti vrlo dobri rezultati s njom.

No, RMM donosi i niz praktičnih problema. Prvo, mi imamo značajno veći broj kanala čestica i stoga veću matricu koja će najčešće biti iznimno rijetka (očekujemo popunjenost od samo 3%!), što znamo da se negativno odražava na rad NM. Dodatno, veličina u memoriji jednog događaja se povećala preko 20 puta u usporedbi s prethodno korištenom reprezentacijom, što će značajno povećati vrijeme treniranja. Preostaje pitanje hoće li se te poteškoće pokazati jačima od prednosti.

7.2 Rezultati na RMM

Praktično govreći, na RMM nema smisla trenirati MLP, iz jednostavnog razloga što bi već prvi gusti sloj, ukoliko mu damo svega 200 neurona, imao preko 1,8 milijuna parametara pa bi treniranje takve NM trajalo iznimno dugo ili bi morali drastično smanjiti složenost arhitekture MLP-a. Dodatno, RMM je vrlo uređena po pitanju rasporeda značajki što bi MLP bio daleko manje u stanju iskoristiti negoli mreže koje koriste konvolucijske slojeve.

Mi ćemo se odlučiti testirati ovu reprezentaciju na CNN, a razlog za taj odabir je dvostruk: pri radu na *toy dataset-u* obje su NM (CNN i ResNet) imale podjednako



Slika 7.1: Grafovi s RMM: a) prikaz jednog kvarkovkog događaja; b) uprosječenje apsolutnih vrijednosti gluonskih događaja iz validacijskog seta; c) razlika uprosječenja kvarkovskih i gluonskih događaja iz validacijskog seta; d) rezultat analize važnosti značajki za AUC-ROC metriku. Na osima su istaknute mase u GeV kod prvog mjesta u kategorijama čestica.

uspjeha, dok je CNN ponovo malo stabilniji, a glavna prednost ResNet-a, brzina konvergencije po epohama, bijedi u usporedbi s predviđenim vremenom njegova treniranja po epohi koje bi bilo oko 2 sata.

Problem veličine podataka, kojeg smo već spomenuli, ovdje dolazi do izražaja. Samo naš standardni set za trening od 250 tisuća događaja ima preko 17 GB, što zajedno s velikom memorijskom zahtjevnošću treniranja konvolucijskih mreža, znači da nije praktično, pa ni izvedivo, držati taj set u radnoj memoriji. Srećom, veliki setovi nisu ništa novo u radu s NM te Keras pruža elegantno rješenje u obliku generatora, pri čemu se pripremljeni podaci pri treniranju postepeno (ali uvijek par serija ispred treniranja) čitaju s diska.

Najbolja CNN arhitektura sastojala se od pet konvolucijskih blokova: konvolucijski sloj, BN, *dropblock* sloj, *max pooling* i tri gusta bloka: gusti sloj, BN, sloj ispadanja. Nakon njih je kao i inače slijedio jedan gusti sloj s jednim neuronom i sigmoidalnom aktivacijskom funkcijom. Sve druge aktivacijske funkcije su bile ELU. Izlazi konvolucijskih blokova bili su formata: $47\times47\times64$, $23\times23\times96$, $11\times11\times144$, $5\times5\times216$, $2\times2\times324$, dok su brojevi neurona gustih slojeva bili: 128, 256, 128, 1. Sveukupno model je imao 2,2 milijuna parametara. Konvolucijske jezgre bile su 4×4 s jediničnim korakom, dok su korak i jezgra *pooling-a* bili 2×2.

Saturacija se postizala nakon svega 10 epoha, a vrijeme treniranja po epohi je približno bilo 1720 sekundi, ili malo manje od pola sata. AUC-ROC vrijednost bila je 0,742, što je $\approx$12% bolje od osnove. Vidimo da su mane ove reprezenatacije u našem slučaju bile utjecajnije od njenih teoretskih prednosti, tako da ukupni rezultat nije ništa bolji od dosadašnjih, iako je ona pružila najbržu konvergenciju (po epohama) do sada.

Na kraju možemo pogledati rezultate analize važnosti značajki sa slike 7.1, gdje vidimo velik broj različitih značajki koje su „zbunjivale“ našu mrežu (u crvenome) tako da je njihovo gašenje imalo u prosjeku pozitivan utjecaj na performanse. Možemo uočiti kako su ponovo mioni (kategorija mase 0,1 GeV) bili veoma bitni, ali zajedno s njima su ovaj puta veliku ulogu igrale dvočestične veličine koje su uključivale bezmasene čestice. Činjenice da se pojavila tako jaka i neobična ovisnost, da postoje značajke koje „zbunjuju“ mrežu te velike nepodudarnosti u obliku razlike prosjeka događaja i važnosti značajki impliciraju da je mreža imala nezanemarivih poteškoća pri učenju. Sve u svemu, ova reprezentacija se pokazala daleko od idealne.

8 Zaključak

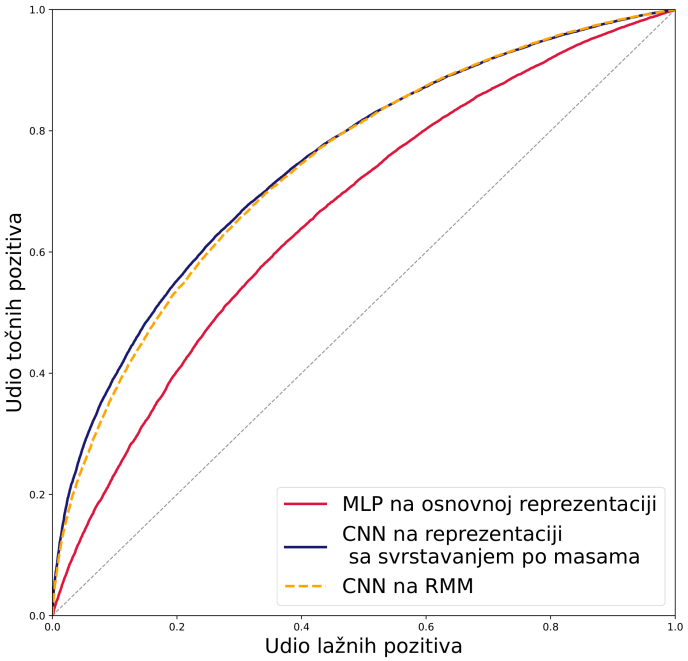
U ovom radu istražili smo performanse različitih NM na nekoliko reprezentacija događaja s mlazovima s ciljem razvijanja što boljeg binarnog klasifikatora (s nadziranim učenjem) koji bi odredio je li događaj porijeklom kvarkovski ili gluonski. ROC krivulje najboljih mreža u pojedinim kategorijama su dane na slici 8.1, a raščlamba rezultata je sljedeća[15]:

Oblik reprezentacije	Varijable	Mreža	AUC ROC	Promjena u odnosu na osnovu
Osnovna: 40 čestičnih kanala, 4 varijable po čestici	4-impulsi	MLP	0,661	n/a
	M_T, E_T, Th(η), ϕ	MLP	0,674	+2%
Svrstavanje po masama: 94 čestičnih kanala, 4 varijable po čestici	M_T, E_T, Th(η), ϕ	MLP	0,726	+10%
		CNN autoenkoder	0,595	-10%
		CNN	0,747	+13%
		ResNet	0,749	+13%
RMM: 95×95 matrica	v. podpoglavlje 7.1	CNN	0,742	+12%

Odabir oblika podataka s pažljivim rasporedom značajki u prostoru se pokazao najbitnijim korakom koji je sam po sebi rezultirao poboljšanjem od 10%. Korištenjem prikladnijih mreža bili smo u stanju dobiti još malo bolje performanse i drastično bržu konvergenciju. Ipak, tu postoje sasvim realna ograničenja te ne smijemo pretjerati s razvojem značajki i povećavanjem veličine reprezentacije, jer, kao što smo vidjeli na RMM, to drastično povećava vrijeme treniranja, a može biti i štetno za performanse NM.

Kako smo uveli reda u prostornu raspodjelu značajki u kasnijim preprezentacijama, nije začuđujuće da su najbolje su rezultate dale mreže koje su koristile konvolucijske slojeve, CNN i ResNet, a činjenica da su ti rezultati toliko blizu nas upućuje na mogući zaključak kako smo „iskoristili“ sve korisne relacije među značajkama na koje se mreža mogla natrenirati, odnosno kako smo možda došli do najboljih mogućih performansi na reprezentaciji, ako ne i na podacima uopće, na kojoj smo trenirali NM. No, takve

[15] Pri čemu treba napomenuti kako su NM s konvolucijskim slojevima u drugoj reprezentaciji dali bolje rezultate na preoblikovanju dimenzija na 19×20, koji su i navedeni u tablici.



Slika 8.1: Graf ROC krivulja najboljih mreža u pojedinim reprezentacijama.

tvrdnje za sada ostaju samo hipotetske te bi ih trebalo dodatno provjeriti.

Kao što se jasno vidi, nažalost, nismo bili uspješni u razvijanju autoenkoderskog detektora anomalija i klasifikatora, a kao moguće objašnjenje smo dali sljedeće: svaku klasu čini više složenih procesa koji mogu rezultirati veoma različitim izlaznim česticama. To pak, za posljedicu ima veću raznolikost u podacima iste klase, koja je možda i usporediva s razlikama među klasama.

Rezultati se vrlo vjerojatno mogu poboljšati povećanjem seta za treniranje zamjenom poduzorkovanja više zastupljene klase naduzorkovanjem manje zastupljene klase i potencijalno uporabom metoda augmentacije podataka (kojima bi generirali još događaja baziranih na onima koje imamo). Dodatno, moguće je da naš *one-size-fits-all* pristup problemu, u kojem želimo razviti klasifikator koji radi na svim ulaznim podacima, nije najbolji. Tako bi se moglo isprobati stratificiranje događaja po nekoj metrici i treniranje više NM, svaka od kojih bi se bavila pojedinim tipom događaja.

9 Dodatak

9.1 Popis kratica

AUC – površina ispod krivulje (eng. *area under curve*)

BN – sloj normalizacije serije (eng. *batch normalization*)

CNN – konvolucijska neuralna mreža (eng. *convolutional neural network*)

MLP – višeslojni perceptron (eng. *multi-layer perceptron*)

NM – neuralna mreža (eng. *neural network*)

QCD – kvantna kromodinamika (eng. *quantum chromodynamics*)

ResNet – rezidualna mreža (eng. *residual neural network*)

RMM – matrica masa i rapiditeta (eng. *rapidity-mass matrix*)

ROC – eng. *receiver operating characteristic*

SGD – stohastičko spuštanje po gradijentu (eng. *stochastic gradient descent*)

9.2 Specifikacije računala

Tensorflow verzija: 2.9.1

OS: Windows 10.0.19044.1806

Procesor: Intel i7-9750HF

Ram: 16 GB DDR4 2667MHz (2x8 GB)

Korišteni disk: 256GB SSD – Samsung MZVBL256HBHQ-00L2

Grafička kartica: NVIDIA GeForce GTX 1660 Ti

Literatura

[1] Guest D., Cranmer K., Whiteson D., Deep Learning and its Application to LHC Physics // Ann.Rev.Nucl.Part.Sci. 68 (2018), str. 161-181

[2] Radovic A, Williams M, Rousseau D, Kagan M, Bonacorsi D, Himmel A, Aurisano A, Terao K, Wongjirad T. Machine learning at the energy and intensity frontiers of particle physics. // Nature. Vol. 560(2018), str. 41-48.

[3] Peskin M.E., Schroeder D.V., An Introduction To Quantum Field Theory. Avalon Publishing, 1995.

[4] Thomson M. Modern Particle Physics. Cambridge University Press, 2013.

[5] Goodfellow I., Bengio Y., Courville A. Deep Learning. The MIT Press, 2016.

[6] Géron A. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. Second edition. O'Reily, 2019.

[7] David E. R., Geoffrey E. H., Ronald J. W. Learning representations by back-propagating errors. // Nature, Vol. 323 (1986), str. 533,-536

[8] Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // Proceedings of Machine Learning Research. Vol. 37 (2015), str. 448-456

[9] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting // Journal of Machine Learning Research. Vol. 15, 56 (2014), str. 1929-1958

[10] Ghiasi G., Lin T., Le Q.V., DropBlock: A regularization method for convolutional networks // 32nd International Conference on Neural Information Processing Systems 2018. / NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, Curran Associates Inc, 2018. Str. 10750-10760.

[11] Shlomi J., Battaglia P., Vilmant J. Graph neural networks in particle physics // Machine Learning: Science and Technology. Vol. 2, 2 (2020)

[12] Madrazo C.F., Heredia I., Lloret L.,Marco de Lucas J. Application of a Convolutional Neural Network for image classification for the analysis of collisions in High Energy Physics // 23rd International Conference on Computing in High Energy and Nuclear Physics (CHEP 2018) / EPJ Web Conf. 214 (2019) 06017

[13] Russakovsky O., Deng J. et al. ImageNet Large Scale Visual Recognition Challenge (ILSVRC2015). // IJCV, 2015.

[14] He K., Yhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016) / IEEE, 2016. Str. 770-778.

[15] Komiske P.T., Metodiev E.M., Thaler J. Energy flow networks: deep sets for particle jets. // J. High Energ. Phys. 2019

[16] Github repozitorij: https://github.com/surmenok/keras_lr_finder, zadnje pristupanje web adresi: 8.11.2022.

[17] Clevert D-A, Unterthiner T., Hochreiter S., Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). // arXiv: Learning (2015) & ICLR (Poster) (2016)

[18] Amid E., Warmuth M., Anil R., Koren T. Robust Bi-Tempered Logistic Loss Based on Bregman Divergences // Advances in Neural Information Processing Systems. Vol. 32 (2019)

[19] Chekanov S.V. Machine Learning Using Rapidity-Mass Matrices for Event Classification Problems in HEP. // Universe. Vol. 7, 1 (2021)

[20] Chekanov S.V. Imaging particle collision data for event classification using machine learning. // Nuclear Inst. and Methods in Physics Research A. Vol. 931 (2019), Str. 92-99