# Rekonstrukcija mlazova pomoću metoda strojnog učenja

**Bajzek, Martin**

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Martin Bajzek

# Rekonstrukcija mlazova pomoću metoda strojnog učenja

Diplomski rad

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA; SMJER ISTRAŽIVAČKI

**Martin Bajzek**

Diplomski rad

# Rekonstrukcija mlazova pomoću metoda strojnog učenja

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

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2021.

Zahvaljujem se Marku Jerčiću, mag.phys, na uloženom trudu i pomoći kod generiranja simuliranih podataka i konstrukciji neuralne mreže. Posebno se zahvaljujem svojem mentoru, izv. prof. dr. sc Nikoli Poljaku za uložen trud kod ideje zamišljenog detektora te analizi rezultata.

# Sažetak

Rekonstrukcija cijelog mlaza u modernoj fizici problematična je zbog detektorskih efekata i nepotpunog poznavanja hadronizacije. Rekonstrukcija se temelji na simulacijama događaja i simuliranom odazivu detektora iz kojeg se onda određuje efikasnost rekonstrukcije pojedinog kanala koji se razmatra. Efikasnost se tada uračunava u mjerene podatke stvarnih događaja, za koje se pretpostavlja da prate iste distribucije kao i simulirani događaji. U ovom radu bi se napravio novi, pojednostavljeni simulator mlazova koji bi usmjeravali na $N \times N$ polje koje predstavlja detektor. U svakom elementu polja, za pojednostavljeni snop, zapisao bi se četverovektor impulsa i energije te bi cjelokupno polje predstavljalo fizikalni snop. Realna slika snopa tad bi se dobila uvodjenjem detektorskih efekata na polje. Konačno, vidjet ćemo je li pomoću metoda strojnog učenja moguće rekonstruirati fizikalni snop iz njegove realne slike, te je li efikasnost takve rekonstrukcije usporediva s efikasnostima modernih metoda.
Ključne riječi: strojno učenje, detektori, Keras

# Jet reconstruction using machine learning method


## Abstract

Jet reconstruction in modern physics poses problems due to detector effects and incomplete knowledge of hadronisation. The reconstruction is based on simulated events and simulated responses of the detector from which the efficiency of the reconstruction in individual channels is determined. Efficiency is then included in the measurement data of real events, which is assumed to follow the same distributions as the simulated events. In this paper, we generate a new simplified jet simulator that would be directed onto the $N \times N$ field represented by the detector. In each element of the field, for a simplified beam, an energy-momentum four-vector would be written and the whole field would represent a physical beam. Realistic image of the beam would then be obtained by introducing detector effects on the field. Finally, we'll see if it is possible to reconstruct the physical jet from its images by using machine learning method and whether the efficiency of such reconstruction is comparable to the efficiencies of modern methods.


Keywords: machine learning, detectors, Keras

# Sadržaj

# 1 Uvod

Kvantna kromodinamika (QCD) je baždarna teorija koja opisuje jaku interakciju [1]. Nekoliko temeljnih načela teorije nije dobro shvaćeno. Otvorena pitanja vezana su za QCD materiju na visokim temperaturama te podliježećem slamanju kiralne simetrije. Također se postavlja pitanje mase laganih kvarkova. Materija kojom dominira jaka interakcija na visokim temperaturama naziva se kvark-gluon plazma (QGP). U QGP stanju kvarkovi i gluoni slabo su vezani i njihovi stupnjevi slobode se mijenjaju. Rani svemir, par mikrosekundi nakon formacije bio je u stanju QGP-a. Navedena kozmološka QGP epoha zasjenjena je naknadnom evolucijom svemira te se ne može astronomski opažati. Jedina metoda stvaranja i opažanja QGP-a je prilikom teško-ionskih sudara na ultrarelativističkim brzinama. Nuklearna materija na trenutak postiže temperaturu i gustoću potrebnu za stvaranje QGP stanja. Ispod kritične temperature faznog prijelaza, materija se opisuje hadronskim stupnjevima slobode gdje su kvarkovi i gluoni zatočeni u bezbojne strukture (hadrone). Pri visokoj temperaturi više nisu vezani u iste strukture i taj fazni prijelaz karakterizira slamanje $Z_3$ simetrije. Teoretska predviđanja QCD na rešetki govore da na konačnim temperaturama iznad faznog prijelaza efektivni stupnjevi slobode QGP-a nisu kvarkovi i gluoni nego kompliciranije strukture. U teško-ionskim sudarima stvaraju se mlazovi (eng. *jets*) različitih čestica. Detekcijom čestica iz mlazova i preciznim mjerenjem energije i koordinata opažanja svake čestice unutar jednog događaja, moguće je rekonstruirati sam sudar. Moguće je i izračunati gubitak energije pojedinog partona prolaskom kroz QGP [2], posebno kako se ta energija disipira u mediju. Analizom kutnih korelacija teško-okusnih čestica s nabijenim česticama, moguće je karakterizirati fragmentaciju kvarkova u ultravrućem nuklearnom mediju. Nužan uvjet za potpunu rekonstrukciju događaja i mogućnost uvida u svojstva QGP-a svodi se na precizno mjerenje energija te trajektorija pojedinih fragmenata sudara. Iako se precizna kalibracija vrši za svaki detektor posebno, uvijek postoji mogućnost dekalibriranja ili kvara pojedinog kanala. Cilj ovog istraživanja je mogućnost prepoznavanja i zatim korekcije takvih sistematskih kvarova pomoću metode strojnog učenja. Drugim riječima, ako je kanal defektan, je li moguće pomoću korelacija očitanja iz drugih kanala naučiti neuralnu mrežu da sa određenom sigurnošću to utvrdi. Podaci se generiraju u programskom paketu PYTHIA koji služi za simulacije sudara čestica na vrlo visokim energijama.

# 2 Teorijska pozadina

## 2.1 Kvantna kromodinamika

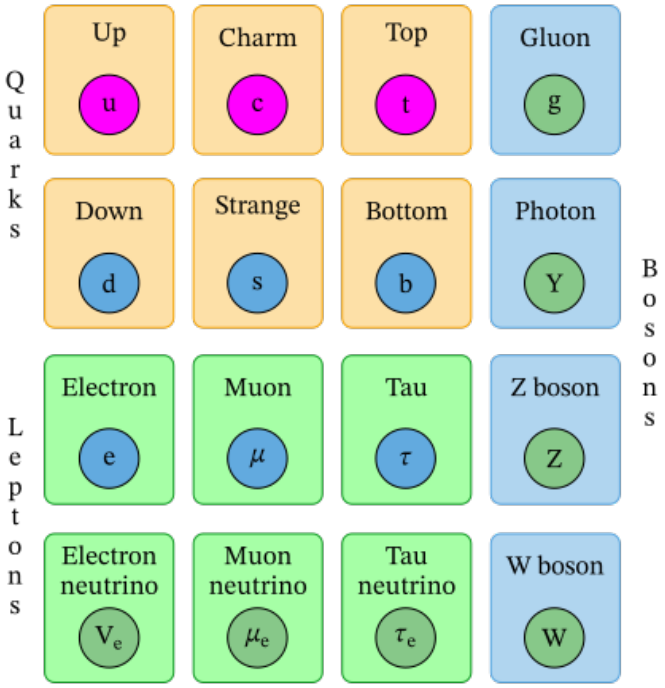
Kvantna kromodinamika dio je opsežnog standardnog modela koji sam služi kao danas prihvaćena teorija interakcije materije. Matematički gledano, QCD je Yang-Mills teorija s podliježećom baždarnom grupom $SU(3)$. Polja obuhvaćena teorijom nazivaju se *kvarkovi*. Kvarkovi su fermioni spina 1/2 i dolaze u šest *okusa* $u$ (up), $d$ (down), $c$ (charm), $s$ (strani), $t$ (top) te $b$ (bottom). Okusni kvarkovi $u,\, c,\, t$ imaju naboj $+\frac{2}{3}\, e$ dok $d,\, s,\, b$ imaju naboj $-\frac{1}{3}\, e$. Svakom je kvarku pridružen drugi stupanj slobode, *boja*. Kvarkovi dolaze u tri boje (crven, zelen, plavi). Sa kvarkovima dolaze i antikvarkovi koji imaju suprotan naboj i antiboju. Dakle, općenito polje kvarka glasi $\Psi^{\alpha,A}$ pri čemu je $\alpha = 1,\, 2,\, 3$, bojni indeks te $A = u,\, d,\, c,\, s,\, t,\, b$ okusni indeks. Osam baždarnih bozona teorije nazivaju se *gluoni*. Općeniti QCD Lagranžijan glasi [3]

$$\mathcal{L} = i\bar{\Psi}^{\alpha,A}\partial\!\!\!/\Psi^{\alpha,A} - m_A\bar{\Psi}^{\alpha,A}\Psi^{\alpha,A} - \frac{1}{4}F^a_{\mu\nu}F^{\mu\nu\ a} + gA^a_\mu\bar{\Psi}^{\alpha,A}\gamma^\mu T^a_{\alpha\beta}\Psi^{\beta,A}, \tag{2.1}$$

gdje se implicira Einsteinova sumacija, a $T^a$ su generatori grupe $SU(3)$ u fundamentalnoj reprezentaciji.

## 2.2 Usporedba s elektroslabom silom

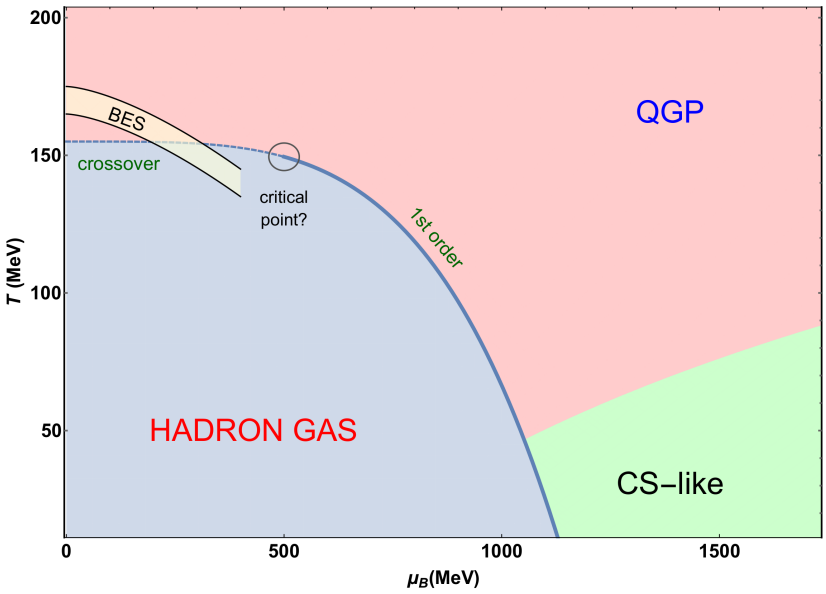
Za razliku od elektroslabe interakcije gdje fotoni ne nose naboj, u jakoj interakciji gluoni nose boju pa stoga sudjeluju u jakoj interakciji. Zato je nuklearni potencijal zanemariv na udaljenostima $r > 1$ fm, što je opet posljedica da jaka interakcija ima empirijski oblik $V(r) \sim -\frac{k}{r}e^{-\mu r}$ sa karakteristične udaljenosti $\mu \approx 1$ fm. Stoga kažemo da je jaka sila kratkodosežna, dok je elektromagnetska interakcija dugodosežna. Za razliku od elektroslabe interkacije, jaka interkacija ima karakterističnu energetsku skalu $\Lambda_{QCD} \sim 200$ MeV.




Slika 2.1: U standardnom modelu postoji šest vrsta ili okusa kvarkova. [5]

## 2.3 *Svojstva QCD-a*

Ključno je svojstvo jake interakcije da se konstanta vezanja $g(E)$ približava nuli na visokim energijama, a relativno je velika na niskim energijama. Stoga je na malim energijama teorija neperturbativna i kvarkovi su *zatočeni*. Drugim riječima, na niskim energijama kvarkovi postoje samo kao vezano stanje bojnog singleta: kvark-antikvark (mezon) ili tri kvarka ili tri antikvarka (barion). Mezoni i barioni zajedno se nazivaju hadronima. Ako bi postojala dva slobodna kvarka, onda između njih postoji gluonsko polje. Gluoni nose boju pa postoji privlačna interakcija između virtualnih gluona [6]. Time se stvara efekt konstrikcije gluonskog polja oko pravca između dva kvarka. Daljnim razdvajanjem kvarkova povećava se energija pohranjena u tom polju i dovodi do spotantnog stvaranja kvark-antikvark para koji opet vežu slobodne kvarkove u mezone. Najlakši hadron je neutralni pion $\pi^0$ čija je masa $m(\pi^0) = 135$ MeV. Fazni dijagram nuklearne materije prikazan je na slici 2.2.

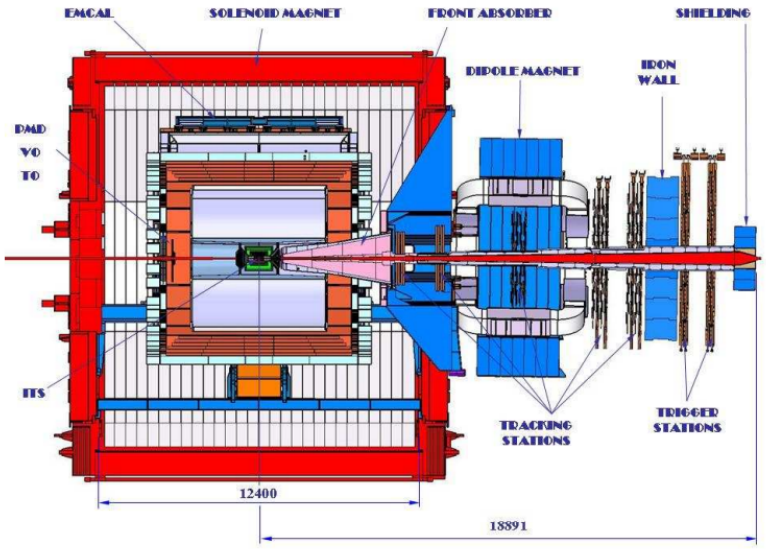



Slika 2.2: Fazni dijagram nuklearne materije [7]. X-os prestavlja kemijski potencijal, a y-os temperaturu. Rubovi dijagrama bazirani su na teoretskim predviđanjima QCD na rešetki te nisu eksperimentalno potvrđeni.

# 3 ALICE eksperiment

ALICE (A Large Ion Collider Experiment) bavi se problemom fizike jako interagirajuće materije, a posebno svojstvima kvark-gluonske plazme pri ekstremnim temperaturama i tlaku, koristeći proton-proton, proton-jezgra te jezgra-jezgra sudare na CERN LHC-u. ALICE postav sastoji se od središnjeg valjka koji pokriva puni azimutalni kut u području pseudorapiditeta $|\eta| < 0.9$. Taj dio eksperimenta mjeri hadrone, elektrone i fotone. Ostali dijelovi postava su: prednji mionski spektrometar ($2.5 < \eta < 4$) i niz manjih detektora za funkciju okidača i karakterizacije događaja [11]. Cjelokupni postav omogućava opažanje hadrona, elektrona, miona, fotona i mlazova čestica iz pp, p-Pb, Pb-Pb sudara. Najzanimljiviji su Pb-Pb sudari, ali njih prethodi niz mjerenja p-p te p-Pb sudara koji stvaraju kvalitativnu bazu za uspredbu s rezultatima Pb-Pb sudara. ALICE detektor izgrađen je kolaboracijom preko tisuću fizičara i inženjera iz preko sto sveučilišta i instituta diljem svijeta. Sveukupne dimenzije su oko $16 \times 16 \times 26$ $\mathrm{m}^3$ s težinom od preko 10000 t. Eksperiment se sastoji od 18 različitih detektorskih sustava s različitim dizajnom i svrhom. Najbitnija karakteristika je mogućnost obrade događaja velikog multipliciteta koji se očekuje u Pb-Pb sudarima. Različiti podsistemi optimizirani su za visoku rezoluciju momenata i energija, dok drugi služe za identifikaciju čestica (PID, eng. *Particle Identification*). Središnji valjak postavljen je

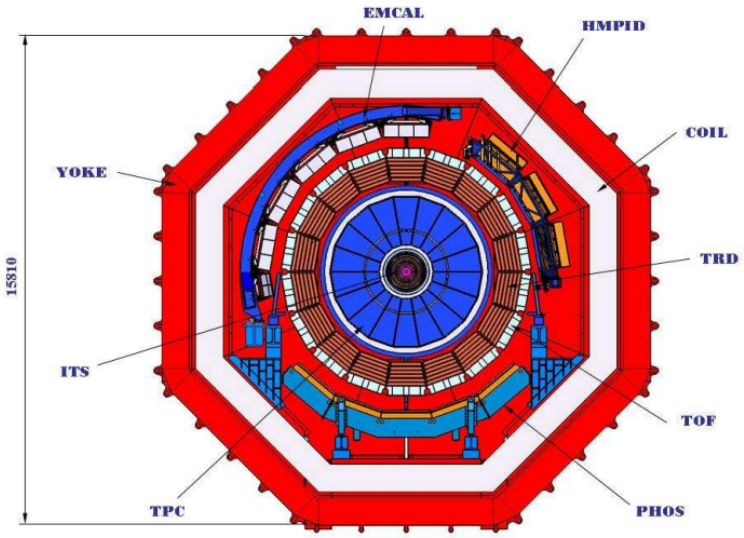
unutar velikog soleniodalnog magneta te se sastoji od Inner Tracking System (ITS) postava silicijskih detektora te Time Projection Chamber (TPC) cilindričnog sustava. Nakon toga slijede tri niza Time-of-Flight (TOF) detektora te potom prsten za snimanje Čerenkovljevog zračenja (HMPID). HMPID služi za prepoznavanje čestica vrlo visokog impulsa u intervalu 1 - 5 GeV/$c$. Čestice niskog impulsa $p \approx 600$ MeV/$c$ identificiraju se preko $\mathrm{d}E/\mathrm{d}x$ mjerenja iz ITS i TPC sustava. Osim HMPID-a, drugi manji detektor koji ne pokriva cijeli azumatalni kut jest elektromagnetski kalorimetar (eng. *Photon Spectrometer*, PHOS). Prikaz cijelog ALICE postava u dva okomita vertikalna presjeka prikazan je na slikama 3.1 i 3.2. Trodimenzionalni prikaz ALICE-a zajedno s referentnom skalom veličine čovjeka, prikazan je na slici 3.3.



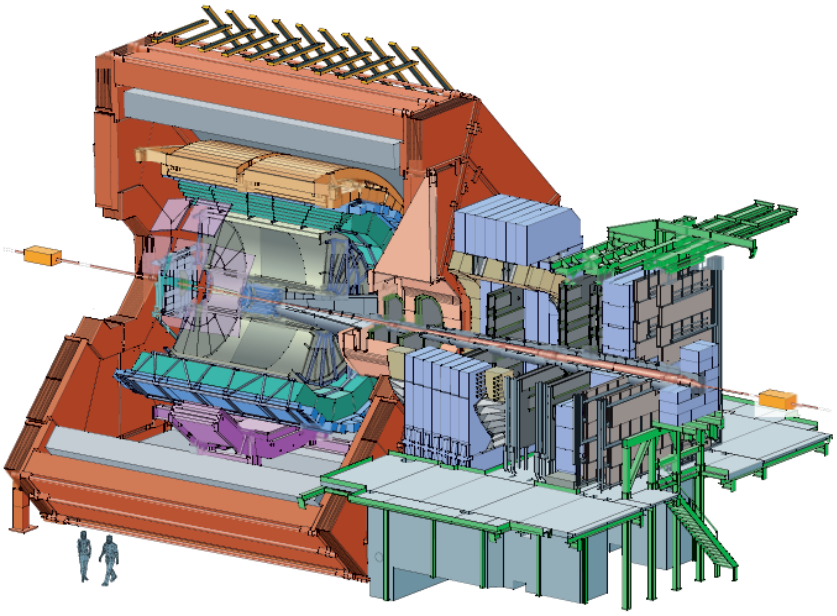

Slika 3.1: ALICE 2D presjek u yz smjeru. Pozitivna z-os gleda u smjeru snopa (prema lijevo), dok je y vertikalna os (prema gore) [8].

## *3.1 Time Projection Chamber (TPC)*

ALICE TPC je sustav zadužen prvenstveno za detektiranje i praćenje te identifikaciju čestica [1]. TPC se sastoji od cilindrične driftne komore i dvije ploče za očitanje na bazama valjka. Aktivni volumen ispunjen je Ne-$CO_2$ plinom u 90-10 smjesi i pokriva $87 < r < 247$ cm u radijalnom te $-250 < z < 250$ cm u longitudinalnom smjeru. Središnja elektroda na $z = 0$ dijeli aktivni volumen u dvije jednake polovice i stvara električno driftno polje od 400 V/cm. Maksimalno vrijeme preleta elektrona iznosi oko 100 $\mu$s. Ne-$CO_2$ smjesta koristi se zbog minimizacije višestrukih raspršenja




Slika 3.2: ALICE 2D presjek u xy smjeru. Pozitivna x-os usmjerena je prema centru akceleratora (prema desno), dok je y vertikalna os (prema gore) [8].



Slika 3.3: ALICE 3D shematski prikaz. U donjem lijevom kutu stoji prikaz čovjeka što služi kao referentna skala za procjenu dimenzije postava. [8].

čestica. Ploče za očitanje (eng. *readout planes*) podijeljene su u 18 trapeziodalnih sektora, pri čemu svaki pokriva 20° azimutalnog kuta. U radijalnom smjeru svaki je sektor još podijeljen na unutarnju i vanjsku komoru (IROC i OROC). Ploče u svakom sektoru podijeljene su na blokove dimenzija $4\times7.5$ mm$^2$ u IROC-u te $6\times10$mm$^2$, $6\times15$ mm$^2$ u OROC-u. Sveukupno ima 558,568 kanala za očitavanje s aktivnom površinom od 32 m$^2$. Sve komore imaju istu shemu rešetke žica, anodnu ploču, katodnu ploču i središnji *gating grid*. U komori kada nije prisutan signal okidač, *gating grid* je u

reverznom naponu pa time onemogućava prolazak elektrona u područje pojačanja te prolazak iona iz prethodnih događaja natrag u driftni volumen.

## *3.2 Inner Tracking System (ITS)*

Središnji plašt ITS sustava u ALICE postavu sastoji se od 6 cilindričnih slojeva silicijskih detektora postavljenih koaksijalno oko pravca snopa. Slojevi se nalaze na radijusima između 39 mm te 430 mm. Sastoje se od tri različite vrste silicijskih detektora: pikselni, drift te *strip* detektori. Glavni cilj ITS-a je lociranje primarne točke sudara (eng. *vertex*) rezolucijom boljom od 100 $\mu m$. Drugi je cilj rekonstrukcija sekundarnih vrhova B i D mezonskih raspada, rekonstrukcija trajektorije i identifikacija nabijenih čestica s tranverzalnim momentom $p_T < 200$ MeV/$c$. Sustav omogućava precizniju rezoluciju kuta i momenta čestica opaženih TPC-om. Također, dizajn je takav da se može primiti velika gustoća čestica s jako dobrom prostornom preciznošću, velikom efikasnošću i pouzdanim razdvajanjem čestica. Identifikacija čestica slijedi iz $\mathrm{d}E/\mathrm{d}x$ mjerenja u režimu niskog $p_T$.

Prva dva sloja ITS-a sastavljena su od silicijskih pikselnih detektora (SPD). Svaki takav detektor, mreža je silicijskih detektora (piksela) u reverznoj polarizaciji. Veličina piksela je 50 $\mu$m ($\mathrm{r}\phi$) $\times$ 425 $\mu$m (z). SPD radi brzo 2D očitavanje u 256 $\mu$s. Digitalni signal iz piksela dobiva se kada analogni signal prijeđe preko nametnutog praga za pojedini piksel čime se odstranjuje pozadina i bezčestični signali.

Iduća dva sloja sastavljena su od silicijskih drift detektora (SDD). SDD je podijeljen na dva driftna područja gdje se elektroni gibaju u suprotnim smjerovima pomoću centralne katode. Svako područje sadrži anode koje sakupljaju elektrone. Na njih je također nametnut napon koji usmjerava naboje neovisno o driftnom naponu. Bitno svojstvo ove vrste silicijskog detektora je mala vrijednost kapaciteta anode što omogućava veću energetsku rezoluciju na kraćim vremenskim skalama u usporedbi s tradicionalnim fotodiodama i Si detektorima. SDD koristi se primarno za mjerenja događaja vrlo visokog multipliciteta.

*Silicon Strip Detectors* (SSD) sačinjavaju zadnja dva sloja ITS-a. Svaki SSD modul sadrži 1536 traka dvostranog silicijskog senzora. Niz traka s jedne strane detektora postavljen je okomito na niz traka s druge strane. Širina trake od 95 $\mu$m omogućava da se na jednoj strani detektora odredi jedna prostorna koordinata upadne čestice, a

na drugoj strani određuje se njoj okomita koordinata. Pomoću analognog očitavanja, moguće je odrediti $\mathrm{d}E/\mathrm{d}x$ u režimu nerelativističkih brzina upada što služi kod identifikacije čestice.

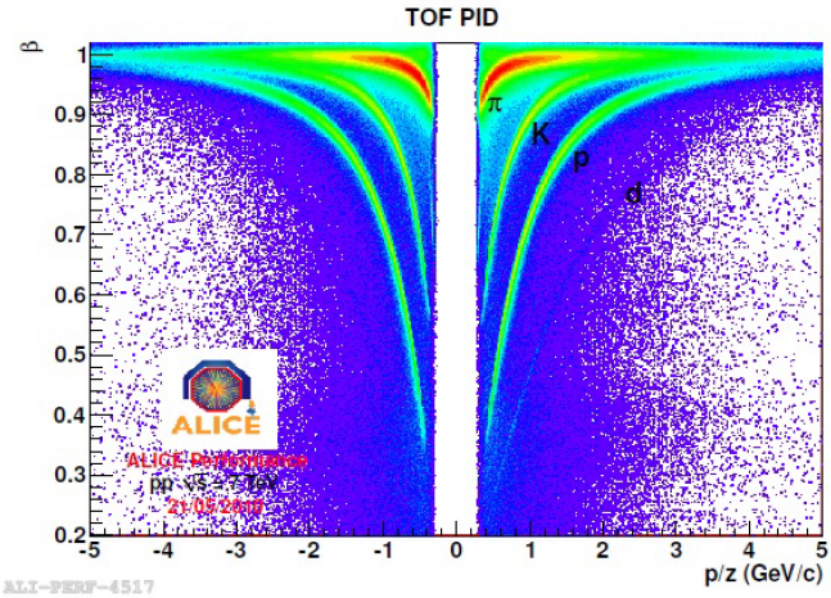
| Layer / Type | $r$ [cm] | $\pm z$ [cm] | Number of modules | Active area per module $r\phi \times z$ [mm$^2$] | Intrinsic resolution [$\mu$m] $r\phi$ | $z$ | Material budget $X/X_0$ [%] |
|---|---|---|---|---|---|---|---|
| 1 / SPD | 3.9 | 14.1 | 80 | $12.8 \times 70.7$ | 12 | 100 | 1.14 |
| 2 / SPD | 7.6 | 14.1 | 160 | $12.8 \times 70.7$ | 12 | 100 | 1.14 |
| 3 / SDD | 15.0 | 22.2 | 84 | $70.2 \times 75.3$ | 35 | 25 | 1.13 |
| 4 / SDD | 23.9 | 29.7 | 176 | $70.2 \times 75.3$ | 35 | 25 | 1.26 |
| 5 / SSD | 38.0 | 43.1 | 748 | $73 \times 40$ | 20 | 830 | 0.83 |
| 6 / SSD | 43.0 | 48.9 | 950 | $73 \times 40$ | 20 | 830 | 0.83 |

Slika 3.4: Karakteristike šest ITS slojeva [11].

## *3.3 Time of Flight (TOF)*

Pouzdana identifikacija čestica (PID) preko velikog dijela faznog prostora i za mnoge različite čestice, znatno je bitno za uspjeh ALICE eksperimenta. ALICE-ova dva detektorska sistema TOF i HMPID posvećeni su isključivo za to. TOF je optimiziran za veliku akceptanciju događaja s česticama momenta ispod 2.5 GeV/$c$. Događaj-po-događaj hadronska identifikacija omogućuje s velikom statistikom mjerenje $p_T$ distribucije piona, kaona i protona čime se može izračunati njihova pojedinačna temperatura. To nam daje uvid u termodinamiku QCD-a zasebno za različite vrste čestica, te uvid u moguće termodinamičke nestabilnosti tijekom faznih prijelaza (ponajprije prijelaza u QGP stanje). Daje se uvid u stupnjeve slobode takvoga stanja, posebno se tiče fenomena kolektivnog gibanja i dinamike širenja. Identifikacija kaona (mezona sa stranim kvarkom) daje informaciju o okusnom sastavu QGP-a i gustoći s-kvarkova, za koju se očekuje da je relativno velika zbog djelomične obnove kiralne simetrije u QGP-u. Također se kaoni stvaraju kanalom raspada $\phi \to \mathrm{K}^+\mathrm{K}^-$ pa je stoga bitno mjerenje stvaranja $\phi$-mezona čime se postrožavaju teorije podrijetla okusne kompozicije u usporedbi sa isključivo određivanjem omjera broja čestica K/$\pi$. Konačno, identifikacija kaona bitna je zbog konstrukcije sekundarnih vrhova raspada te prepoznavanja raspada šarm-mezona poput $\mathrm{D}^0 \to \mathrm{K}^-\pi^+$ ili $\mathrm{D}^- \to \mathrm{K}^-\pi^+\pi^-$. Zbog svoje velike mase, D-mezoni mogu biti stvoreni samo u ranim fazama teško-ionskih sudara i kao takvi

se raspadaju zbog male vjerojatnosti anihilacije. Šarm-mezoni su bitni za detekciju poznatog J/$\Psi$ potiskivanja čime se opet probira u formiranje QGP stanja. Dakle, navedeni ciljevi zahtjevaju pouzdan TOF detektor sa visokom akceptancijom velikog broja događaja i mogućnošću mjerenja momenta od 0.5 (gornja granica $\mathrm{d}E/\mathrm{d}x$ mjerenja iz ITS-a i TPC-a) do 2.5 GeV/$c$ (granica statistike prilikom jednog događaja). Sam detektor se nalazi na 3.7 m od osi snopa i sastoji se od 18 azimutalnih područja koja su podijeljena na 5 modula uzduž osi snopa. Energetska rezolucija TOF detektora je oko $E_\sigma = 0.5$ GeV, a vremensko razlučivanje iznosi 150 ps. U eksperimentima prije nego što snop čestica naleti na metu, prolazi kroz startni T0 detektor koji služi za mjerenje luminoziteta snopa i davanje startnog signala [14]. TOF mjeri vrijeme $t$ od dolaska startnog signala do detekcije čestica u modulima TOF detektora. Znajući vrh sudara i geometriju ALICE postava, moguće je izračunati brzinu čestica. Uz određivanje impulsa preko $\mathrm{d}E/\mathrm{d}x$ omjera iz Bethe-Blochove formule, moguće je odrediti masu čestice i identificirati je. Svaka čestica ima svoju karakterističnu liniju u TOF grafu kao sto je označeno na slici 3.5.



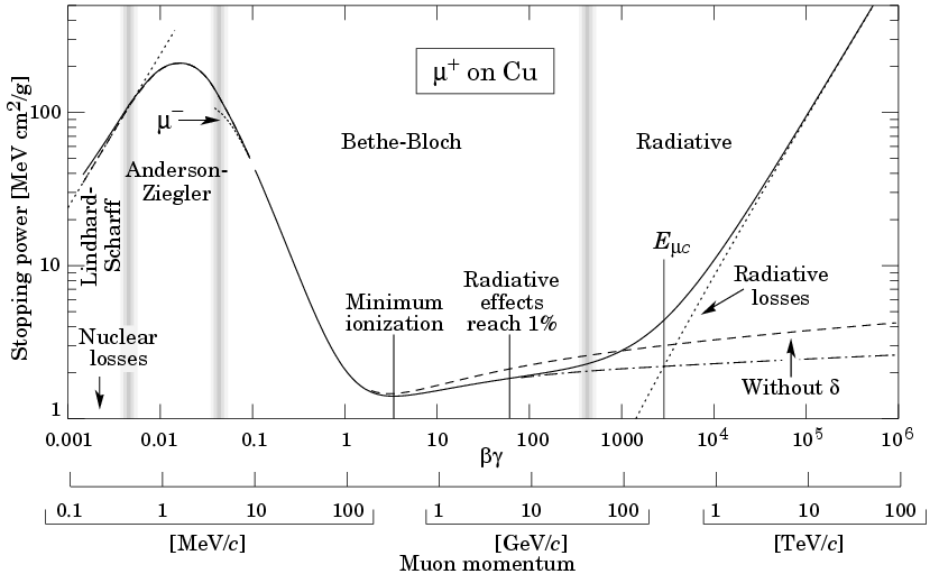

Slika 3.5: Primjer histograma brzine $\beta = v/c$ i impulsa $p$ za središnji p-p sudar na $\sqrt{s_{pp}} = 7$ TeV [13].

## 3.4 Karakteristike i kalibracija silicijskih detektora

Većina moderno proizvedenih silicijskih detektora dolaze s već izračunatim kalibracijskim krivuljama. Stariji detektori te već korištene detektore potrebno je rekalibrirati prije eksperimenata zbog radijacijske degradacije koja mijenja karakteristiku detektora. Iako su Si detektori stabiliji od detektora na bazi tekućina ili plina, moguća su oštećenja ili promjena krivulje odgovora tijekom intenzivnih signala. Si detektori kod upada nabijene čestice daju signal sa skoro 100% efikasnošću [16]. Za razliku od scintilacijskih detektora, karakterizira ih odlična energetska i vremenska rezolucija, male dimenzije te mogućnost razlikovanja više događaja u jedinici vremena. Kada brza nabijena čestica prolazi kroz materijal, deponira dio svoje energije $\mathrm{d}E$ u jedinici puta $\mathrm{d}x$ prema Bethe-Blochovoj formuli:

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \frac{ne^4}{4\pi\epsilon_0^2 m_e c^2}\frac{Z^2}{\beta^2}\left(\ln\frac{\beta^2}{1-\beta^2} - \beta^2 + C_1\right). \tag{3.1}$$

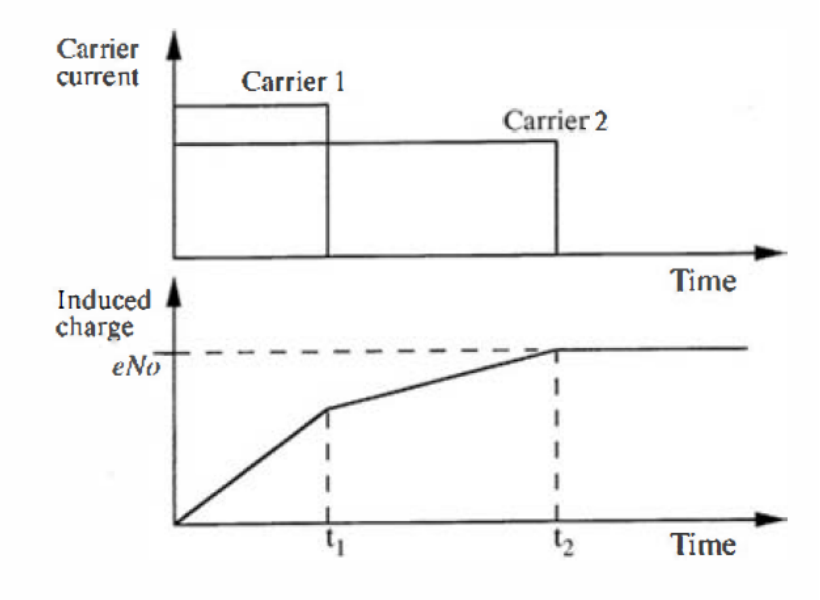
Graf Bethe-Blochove funkcije prikazan je na slici 3.6.




Slika 3.6: Deponirana energije u materijalu tijekom prolaska $\mu^+$ kroz Cu blok [18].

Kad čestica prolazi kroz poluvodički materijal, unutar par pikosekundi formiraju se elektron-šupljina parovi. Ako je $E$ deponirana energija unutar aktivnog volumena poluvodičkog detektora, tada broj elektron-šupljina parova $N_0$ glasi $N_0 = E/P$ gdje je $P$ faktor proporcionalnosti, za silicij $P[\mathrm{Si}] = 3.68\ \mathrm{eV}$. Električno polje duž aktivnog volumena omogućava nosiocima da putuju u suprotnim smjerovima. Njihovo gibanje stvara struju sve dok se stvoreni nosioci naboja ne sakupe na granicama aktivnog vo-

lumena. Struja naboja i sakupljanje na elektrodama prikazano je na slici 3.7. Omjer mobilnosti elektrona i šupljina je unutar faktora 2 - 3 za silicij. Oba pulsa, elektronski i šupljinski moraju se integrirati da bi se dobio dobar prikaz deponirane energije. Izlazni, analogni napon je reda veličine 10 $\mu$V [17].




Slika 3.7: Idealna reprezentacija gibanja $N_0$ elektrona i šupljina u poluvodiču. $t_1$ predstavlja vrijeme sakupljanja prvog, a $t_2$ drugog nosioca. Preuzeto s [16].

Svaki od tri vrsta silicijskih detektora ITS sustava ima posebni algoritam kalibracije. U ovom potpoglavlju govorit ćemo samo o kalibraciji silicijskih strip detektora (SSD). Svaki Si detektor kao izlaz daje digitalni signal iz ADC kanala (analogno-digitalnog konvertera). Deponirana energija u detektoru i izlazni ADC signal linearno su proporcionalni [15]:

$$E(N) = aN + b, \tag{3.2}$$

pri čemu je $E$ energija, $N$ izlazni signal (napon, eng. *ADC count*) iz ADC kanala, dok su $a$ i $b$ kalibracijski koeficijenti. Gubitci energije čestice ovise o ulaznoj energiji i debljini tzv. *mrtvog sloja*, područja gdje se energija deponira, ali ne stvaraju se elektron-šupljina parovi. To su, na primjer, dio silicija bez driftnog električnog polja ili metalne ploče za sakupljanje naboja. Također se uzima u obzir neodređena debljina silicija točno pored elektroda gdje je generiranje parova nosioca neefikasno [16].

Da bi se odredila debljina mrtvog sloja i kalibrirao detektor, potreban je emiter čestica poznate energije. Za čestice se uzimaju X-zrake, elektroni ili alfa čestice. Pomoću tri linija poznatih energija $E_3 > E_2 > E_1$ i izlaznog ADC signala, debljina

mrtvog sloja slijedi iz formule $d(\eta) = p_2\eta^2 + p_1\eta + p_0$ gdje je $\eta = (E_3 - E_2)/(E_3 - E_1)$ bezdimenzionalna veličina, a koeficijenti $p_2$, $p_1$, $p_0$ su tabulirani ili izvedeni u Geant4 G4EmCalculator klasi i programu LISE++, ovisno o energijama i vrsti čestice. Izračunata debljina mrtvog sloja omogućuje ispravan izračun deponirane energije čestice u aktivnom dijelu poluvodiča. Konkretnije, znajući debljinu mrtvog sloja, direktno iz Bethe-blochove formule moguć je izračun deponirane energije u mrtvom sloju, pa oduzimanjem od gubitka ukupne slijedi deponirana energija čestice u aktivnom sloju. Linearnom regresijom dobivaju se traženi koeficijenti $a$ i $b$.

## *3.5 Kinematičke varijable*

U ALICE eksperimentu, čestice koje se opažaju imaju energiju redova veličine 1-10 GeV što znači da vrijedi $E \approx |\boldsymbol{p}|c \gg mc^2$. U specijalnoj teoriji relativnosti, prilikom promjene koordinatnog sustava $L$ gdje $x^\mu = (x^0 = ct,\, x^1,\, x^2,\, x^3)$ u $L'$ gdje $x^{\mu'} = (x^{0'} = ct',\, x^{1'},\, x^{2'},\, x^{3'})$ takvih da se ishodište $L'$ sustava giba brzinom $-v$ u $z = x^3$-smjeru spram ishodišta $L$ sustava, tada linearna Lorentzova transformacija glasi $x^{\mu'} = \Lambda^\mu_\nu x^\nu$ gdje je

$$\Lambda = \begin{pmatrix} \gamma & 0 & 0 & \beta\gamma \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \beta\gamma & 0 & 0 & \gamma \end{pmatrix}. \tag{3.3}$$

$\beta = v/c$, dok je $\gamma = (1 - \beta^2)^{-1/2}$. Pojam rapiditeta dolazi iz činjenice da višestruke Lorentzove transformacije u istom smjeru nisu linearne u $\beta$ nego vrijedi $\beta'' = \frac{\beta+\beta'}{1+\beta\beta'}$. Ako se pojam brzine čestice redefinira u *rapiditet* $y$ kao $\beta = \tanh y$ ili $y = \frac{1}{2}\ln\frac{1+\beta}{1-\beta}$ tada vrijedi $y'' = y + y'$. Problem mjerenja rapiditeta u eksperimentima jest da je potrebno znati i energiju i ukupni moment čestice. Eksperimentalno utvrđivanje ukupnog momenta često je puno teže od geometrijskih izračuna. U režimu ultra-relativističkih čestica gdje invarijantna energija iznosi $E \approx |\boldsymbol{p}|c$ moguće je definirati drugu varijablu, *pseudorapiditet* $\eta$

$$\eta = -\ln \mathrm{tg}\frac{\theta}{2}, \tag{3.4}$$

pri čemu je $\theta$ polarni kut oko $z$-osi koja je smjer gibanja snopa. Za ultra-relativističke čestice vrijedi $y \approx \eta$ što je bitno kod eksperimenata na LHC-u gdje često sustav

centra mase sudarajućih čestica ne miruje spram referentnog sustava mirovanja detektora. Ovakav prijelaz rapiditeta na pseudorapiditet eksperimentalno je opravdan jer najveći dio stvorenih hadronskih pljuskova čine pioni ($m_\pi \approx 140$ MeV) sa energijom $\sim 1$ GeV. Ako se sustav centra mase poklapa s laboratorijskim sustavom, vrijednost $\eta$ približava se beskonačnosti blizu linije snopa ($\theta = 0°$) te je jednaka nuli za čestice okomite na liniju snopa ($\theta = 90°$). Invarijantna energija u sustavu centra mase $E_{cm} = \sqrt{s}$ karakterizira energetsku skalu sudara dviju čestica. Nakon sudara, fragmenti se karakteriziraju tranverzalnim momentom $p_T$ koji je komponenta ukupnog momenta okomita na upadni snop. Takva informacija je bitna jer u ultrarelativističkim sudarima, paralelna komponenta $p_{||}$ ne može se razlikovati od momenata čestica iz snopa koje nisu bile dio sudara. Čestice snopa koje se samo elastično raspršе imaju znatno manji $p_T$ od onih koje su sudjelovale u sudaru - velik $p_T$ daje indikaciju da je opažen fragment sudara. Treća komponenta jest azimutalni kut $\phi$ oko smjera gibanja snopa. Nehomogenosti u $\phi$ raspodjeli čestica ukazuju na nejednoliku distribucija spina (polarizaciju) sudarajućih čestica snopa. Zadnja bitna veličina je udaljenost točaka (1) i (2) u $\phi - \eta$ ravnini dana s:

$$R_{1,2} = \sqrt{(\phi_1 - \phi_2)^2 + (\eta_1 - \eta_2)^2}. \tag{3.5}$$

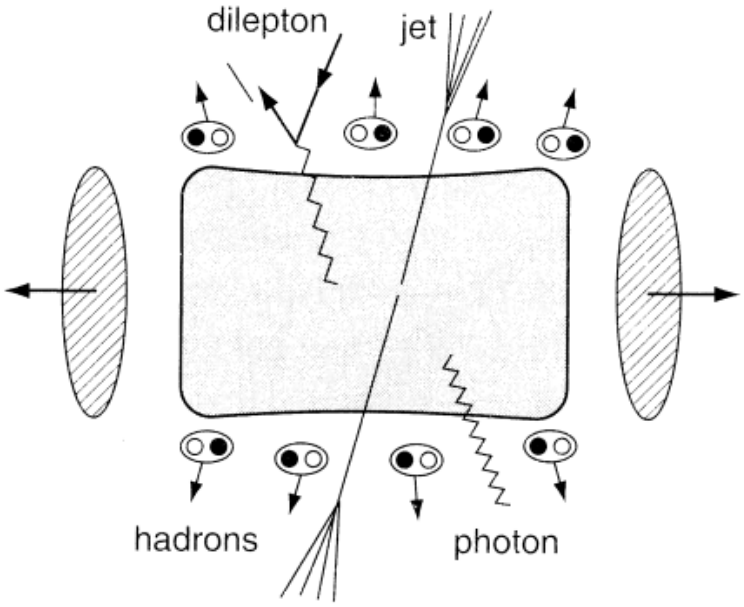
Dakle iz detektora moguće je dobiti informaciju u obliku $(p_T,\, \eta,\, \phi)$ iz čega se rekonstruira događaj. S takvim definiranim kinematičkim i geometrijskim veličinama moguće je prikazati rezultat mjerenja sudara čestica u obliku *čestičnog spektra*. Razmatraju se veličine poput $\mathrm{d}N/\mathrm{d}y$, $\mathrm{d}N/\mathrm{d}\eta$, $\mathrm{d}N/\mathrm{d}p_T$, ali većinom se reprezentira kao *invarijantni udarni presjek* [20]

$$E\frac{\mathrm{d}^3\sigma}{\mathrm{d}p^3} = \frac{\mathrm{d}^3\sigma}{p_T\,\mathrm{d}\phi\mathrm{d}\eta\mathrm{d}p_T}. \tag{3.6}$$

Budući da samo neelastična raspršenja (sudari) dovode do detekcija čestica na velikim polarnim kutevima ($|\eta| \lesssim 2$), time se može mjeriti neelastični udarni presjek. *Luminozitet* snopa definira se kao $L = nf\frac{N_1 N_2}{A}$ gdje je $n$ broj buncheva jednog snopa, $f$ frekvencija križanja snopa, $N_{1,2}$ broj čestica u svakom snopu po jednom bunchu te $A$ površina preklapanja dva snopa.

## *3.6 Rekonstrukcija mlazova*

Prilikom sudara visokoenergetskih čestica $\sqrt{s} > 10$ TeV, iz samih se čestica izbijaju visokoenergetski partoni (kvarkovi, gluoni, mezoni) koji se fragmentiraju u mlaz hadrona (eng. *jet*). U sustavu središta mase partona, fragmentacija je skoro izotropna, ali zbog velikog Lorentzovog potiska, u laboratorijskom sustavu fragmenti su orijentirani oko osi vektora momenta partona. Parton prolaskom kroz juhu kvarkova i gluona stvorenih u sudarima jako interagira i gubi mjerljiv dio energije što se naziva *gašenje mlaza* (eng. *jet quenching*).




Slika 3.8: Sudar teških iona na kratko vrijeme stvara vruće i gusto područje nevezanih kvarkova i gluona [19].

Sam parton nemoguće je direktno mjeriti, a njegova se svojstva deduciraju mjerenjem svojstava cijelog mlaza, to jest svih fragmenata. Dakle potrebno je pozbrajati četveromomente čestica i time se dobije četveromoment prvobitnog partona. Glavni problem u tome je pitanje kako razlikovati čestice koje pripadaju mlazu od bogate pozadine zbog velikog multipliciteta događaja prilikom teško-ionskih sudara. Postoji mnoštvo algoritama za rekonstrukciju mlaza (eng. *jet clustering*) sa svojom vlastitom shemom dizajniranom za određene primjene (na primjer, detekcija isključivo mlazova nastalih raspadom b-kvarka). Algoritmi se temelje na klasterizaciji bliskih čestica u određenom prostornom području. Konusni algoritmi gledaju grube djelove faznog prostora, sumiraju četveromomente energetskog toka tog područja te

provjeravaju podudaranje osi konusa i vektora momenta sume. Najrazvijeniji algoritmi danas su takozvani $\mathrm{k_t}$ [21] i anti-$\mathrm{k_t}$ [22] algoritmi. Oni iterativno klasteriraju čestice na temelju njihove blizine u $\phi$-$\eta$ dijagramu. Razmatraju se hadron-hadron sudari u sustavu centra mase gdje je z-os uobičajeno postavljena u smjeru snopa. Konačno stanje sudara karakterizira se nizom prvobitnih protomlazova (eng. *proto-jets*), $i$, sa četveromomentima $p_i^\mu$. Pretpostavka je da je masa $[p_i^\mu\, p_{i\mu}]^{1/2}$ mala spram tranverzalnog momenta $p_{i,T}$ tako da se protomlazovi mogu opisati azimutalnim kutom $\phi_i$, pseudorapiditetom $\eta_i$ i tranverzalnom impulsom $p_{T,i}$. Počevši od prvobitnog niza protomlazova, algoritam ih rekurzivno grupira u novu listu protomlazova. Mlazovi sa skoro paralelnim momentom trebaju se grupirati. Algoritam također odlučuje kad grupiranje prestaje te u tom slučaju protomlaz postaje reprezentacija fizikalnog mlaza čestica. Konvergencija algoritma ovisi o faktoru $R$ reda veličine 1, koji ima sličnu ulogu kao parametar veličine konusa u konusnim algoritmima. Početni uvjet algoritma jest da je lista protomlazova jednaka listi čestica $(p_{T,i},\, \eta_{T,i},\, \phi_{T,i})$. Iterativni koraci algoritma su sljedeći:

1. Izračunati sve udaljenosti $d_{ij}$

$$d_{ij} = \min(p_{T,i}^2,\, p_{T,j}^2)\frac{R_{i,j}^2}{R^2}, \tag{3.7}$$

   te udaljenosti $d_i = p_{T,i}^2$.

2. Naći najmanju vrijednost od svih $d_i$ te $d_{ij}$ te je prozvati $d_{\min}$.

3. Ako je $d_{\min} = d_{ij}$ tada se protomlazovi $i$ i $j$ spajaju u protomlaz $k$ sa

$$E_{T,k} = E_{T,i} + E_{T,j}, \tag{3.8}$$
$$\eta_k = \frac{E_{T,i}\eta_i + E_{T,j}\eta_j}{E_{T,k}}, \tag{3.9}$$
$$\phi_k = \frac{E_{T,i}\phi_i + E_{T,j}\phi_j}{E_{T,k}}. \tag{3.10}$$

4. Ako je $d_{\min} = d_i$ tada se protomlaz $i$ ne može više spajati i miče se s liste protomlazova te se prozove pravim mlazom.

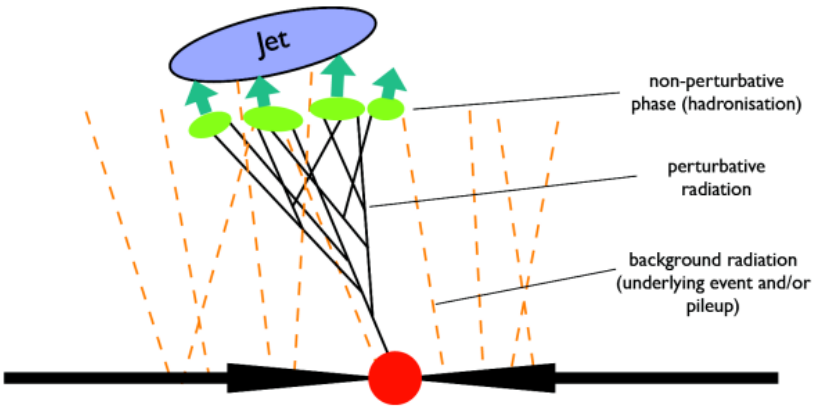
Iteracija se vrši sve dok se ne isprazni lista protomlazova. Navedena shema efektivno klasterira čestice do nekog praga $R$ počevši od najmekših čestica (čestice malog $p_T$).

U listi mlazova, samo mlazovi velikog $p_T$ su fizikalno zanimljivi jer mlazovi malog $p_T$ su ili minimlazovi ili raspršene krhotine iz snopa. Udarni presjek za sudar partona eksponencijalno pada s povećanjem parton-parton energije centra mase $\sqrt{s}$. Tako da ako postoji mlaz s velikim $p_T$, mala je vjerojatnost da u događaju postoji više od ukupno dva mlaza velikog $p_T$ koja su potrebna da se ispuni očuvanje tranverzalne komponente količine gibanja. Za razliku od $k_t$ algorithma, mjenjanjem definicije

$$\hat{d}_{ij} = \min\left(\frac{1}{p_{T,i}^2}, \frac{1}{p_{T,j}^2}\right)\frac{R_{i,j}^2}{R^2}, \tag{3.11}$$

$$\hat{d}_i = \frac{1}{p_{T,i}^2}, \tag{3.12}$$

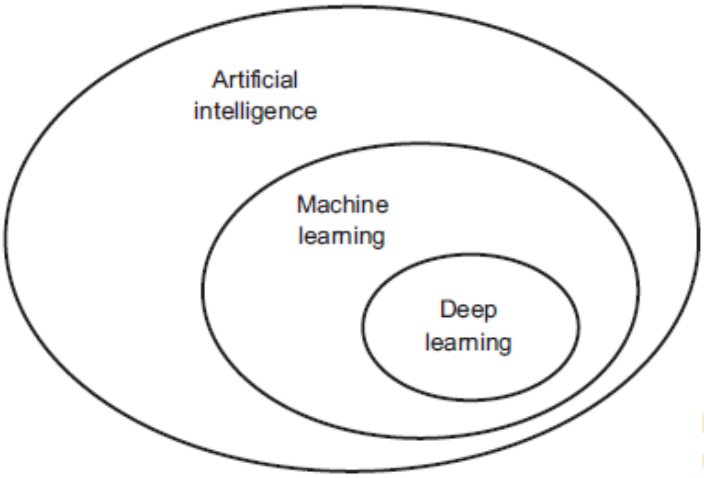
dobiva se anti-$k_t$ algoritam koji počinje klasteriranje oko tvrde čestice (čestice velikog $p_T$). Bez prisustva druge tvrde čestice unutar polumjera $2R$, anti-$k_t$ algoritam sakuplja sve čestice unutar radijusa $R$, dakle daje konusne mlazove poprečne površine $\pi R^2$. Meke čestice konvergirat će u klaster s tvrdim česticama puno prije nego što će se međusobno klasterirati. Dok $k_t$ algoritam služi za procjenu broja mlazova temeljnog događaja, anti-$k_t$ služi za pronalazak *signalnog* mlaza - mlaza preko kojeg će se probirati u gusto nuklearno stanje.




Slika 3.9: Klasteriranje mlazova [23].

# 4 Strojno učenje

Prošlih par godina, umjetna inteligencija (AI, eng. *artificial intelligence*) česta je tema u medijima i popularnoj znanosti. Strojno učenje, duboko učenje i AI termini su prisutni u brojnim člancima, čak i često izvan znanstvenih i tehnoloških publikacija. Cilj umjetne inteligencije je budućnost inteligentnih robota, samo-vozećih automobila i virtualnih pomoćnika u svakodnevnom životu. Sažeta definicija područja umjetne inteligencije jest *nastojanje da se automatiziraju intelektualni zadaci koje inače izvode ljudi*.

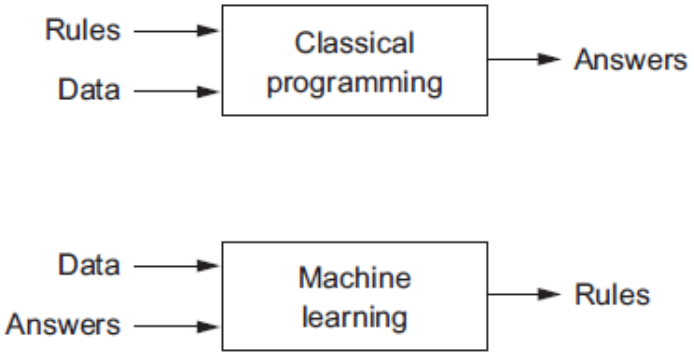



Slika 4.1: Umjetna inteligencija, strojno učenje i duboko učenje [24].

Strojno učenje proizlazi iz pitanja može li se računalo samo naučiti kako izvesti određeni zadatak. Umjesto da programeri sami izvode pravila obrade podataka, može li samo računalo odrediti pravila gledajući i uspoređujući podatke? U klasičnom programiranju, ljudi unose pravila (program) i podatke koji se obrađuju prema tim pravilima i odgovor računala je rješenje koje je primjena pravila na unešene podatke. Kod strojnog učenja, ljudi unose (input) podatke i pripadna rješenja (output) koja se očekuju iz unešenih podataka, a računalo traži pravilo koje veže input i output. Dobivena pravila se zatim klasično mogu primijeniti na nove unosne podatke s nepoznatim rješenjima te se time stvara traženi odgovor. Usporedba shemi strojnog i klasičnog učenja prikazana je na slici 4.2.

Općenito za primjenu strojnog učenja potrebni su sljedeći podaci:

- Točke ulaznih podataka - općenita lista jedne klase podataka (u slučaju ovog rada, lista tenzora dimenzije 2).




Slika 4.2: Strojno učenje: nova paradigma programiranja [24].

- Primjeri očekivanog rezultata - za svaku točku ulaznih podataka, postoji točka rezultata koja je očekivani odgovor.

- Način mjerenja preciznosti algoritma - potrebno je definirati neku vrstu *udaljenosti* između trenutnog odgovora algoritma i očekivanog odgovora. Cilj programa je iterativno minimizirati takvu funkciju. Ovaj korak naziva se *učenje*.

Tijekom učenja, model transformira ulazne podatke u neku drugu vrstu (reprezentaciju). To jest, središnji problem strojnog učenja je smisleno transformirati podatke; naučiti korisne reprezentacije ulaznih podataka koje se onda karakteriziraju i povezuju s očekivanim izlaznim podacima.

## *4.1 Primjer jednoslojne neuralne mreže*

Najjednostavniji problem binarne klasifikacije može se postaviti na sljedeći način: zamisle se dvije klase 1 (pozitivna klasa) i -1 (negativna klasa). Zatim se definira *aktivacijska funkcija* $\phi(z) = \phi(\boldsymbol{w}^\intercal \boldsymbol{x})$ koja uzima linearnu kombinaciju ulaznih podataka $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^\intercal$ i pripadajući *težinski vektor* $\boldsymbol{w} = [w_1, w_2, \ldots, w_n]^\intercal$. Ako je aktivacija (rezultat funkcije) pojedinog uzorka $\boldsymbol{x}^{(i)}$ veća od definiranog praga $\theta$, tada

se predviđa klasa 1, inače klasa -1. U takvom algoritmu je $\phi$ obična step funkcija

$$\phi(z) = \begin{cases} 1, & z \geq \theta \\ -1, & z < \theta \end{cases} \tag{4.1}$$

Zbog jednostavnosti, dodaje se pristrana jedinica (eng. *bias unit*) $x_0 = 1$ i $w_0 = \theta$ pa je $z = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$. Početni uvjet za $\boldsymbol{w}$ postavlja se ili nula ili nasumično odabrani mali brojevi.

Prolazak kroz cijeli set podataka za treniranje naziva se *epoha*. Prilikom jedne epohe za svaki set unosnih podataka $\boldsymbol{x}^{(i)}$ izračuna se izlaz aktivacijske funkcije $\phi(\boldsymbol{w}^\intercal \boldsymbol{x}^{(i)}) \coloneqq \hat{y}^{(i)} = a^{(i)}$ koji se naziva *aktivacija neurona* te je različit od traženog izlaza (oznake) $y^{(i)}$. Da bi se optimizirao težinski vektor $\boldsymbol{w}$, potrebno je definirati *funkciju cijene* (eng. *cost function*) $J(\boldsymbol{w})$. Na primjer, prilikom optimizacije preko gradijentnog spusta, sve težinske vrijednosti ažuriraju se nakon svake epohe prema pravilu

$$\frac{\partial}{\partial w_j} J(\boldsymbol{w}) = \sum_i \left( y^{(i)} - a^{(i)} \right) x_j^{(i)} \tag{4.2}$$

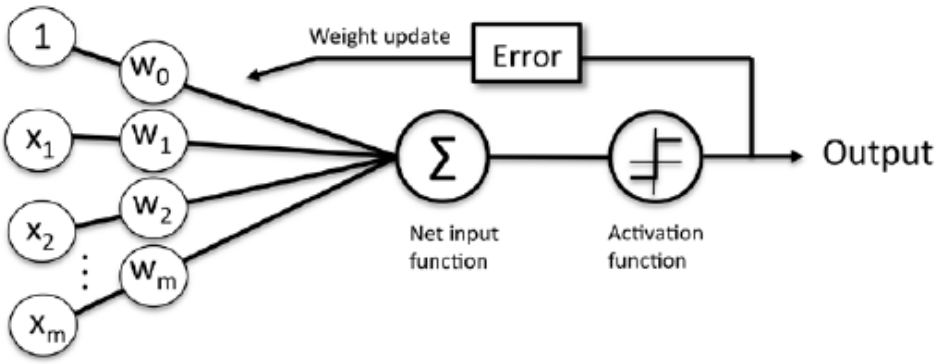
te se težinski vektor $\boldsymbol{w}$ mijenja prema

$$\boldsymbol{w} = \boldsymbol{w} - \eta \nabla J(\boldsymbol{w}) \tag{4.3}$$

pri čemu je $\eta$ stopa učenja (eng. *learning rate*). Stopa se pažljivo bira tako da se uravnoteži brzina učenja s rizikom da se preleti preko globalnog minimuma funkcije cijene. Shematski je jedna epoha predstavljena na slici 4.3.

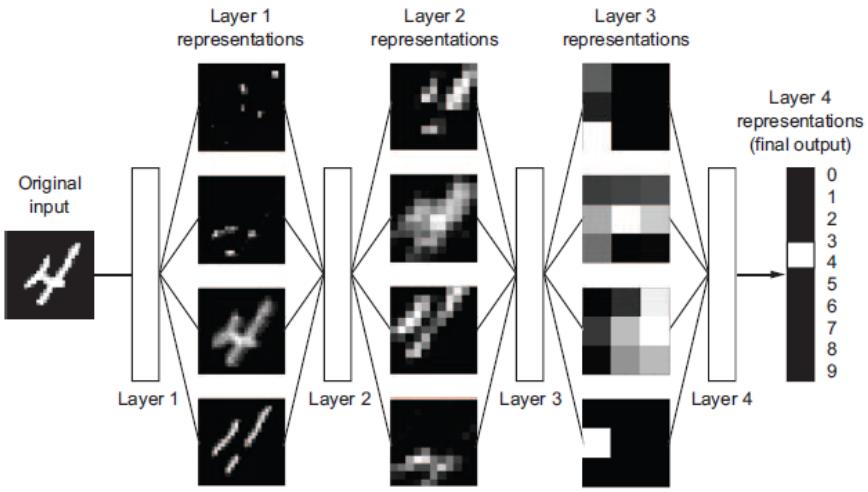
## *4.2 Duboko učenje*

Duboko učenje (eng. *deep learning*) polje je strojnog učenja gdje se uče uzastopni *slojevi* sve više smislenije reprezentacije unosnih podataka. Koliko slojeva doprinosi modelu podataka naziva se dubina modela. Današnji modeli dubokog učenja koriste desetak pa do čak i sto slojeva reprezentacije - i svi se slojevi automatski uče na podacima za trening. Kod dubokog učenja, takve slojevite reprezentacije uče se pomoću modela *neuralnih mreža* koje su strukturirane da se slojevi nastavljaju jedan na drugi. Iako neki koncepti neuralnih mreža su inspirirani razumijevanjem mozga, modeli dubokog učenja nisu nikako referentni modeli rada ljudskog mozga. Duboko




Slika 4.3: Shema rada jednoslojne neuralne mreže. Mreža prima ulazni podatak $\boldsymbol{x}$ i kombinira ga s težinama $\boldsymbol{w}$ te izračuna vrijednost koja prolazi kroz aktivacijsku funkciju gdje se generiraju predviđene oznake koje se zatim uspoređuju sa stvarnim oznakama i ažuriraju se težine [25].

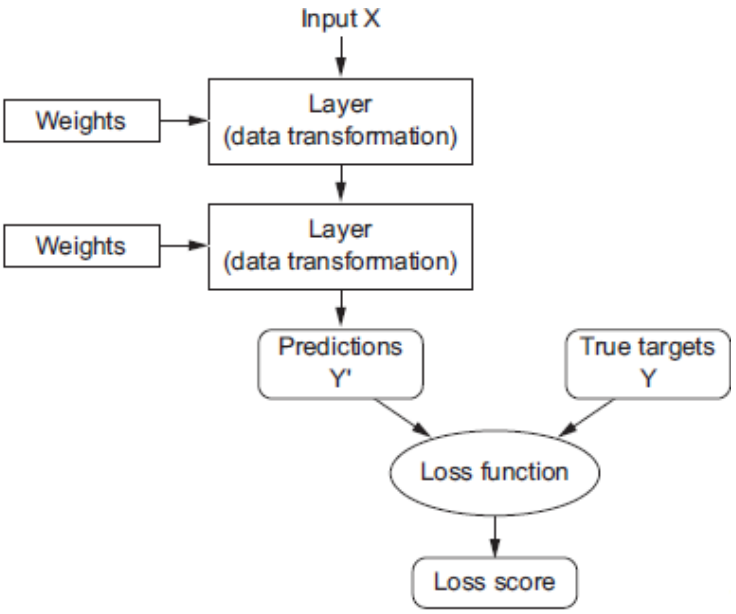
učenje samo je matematički okvir za određivanje smislenih reprezentacija podataka. Jedna od standarnih primjena neuralnih mreža i dubokog učenja je klasifikacija podataka. Shematski prikaz rada problema klasifikacije slike prema tome koji je broj prisutan na slici, prikazan je na slici 4.4.




Slika 4.4: Neuralna mreža transformira digitalnu sliku u reprezentacije koje su svakim korakom sve drugačije od originalne slike i daju informaciju potrebnu za odluku [24].

Slično kao i kod jednoslojnih neuralnih mreža, specifikacija što točno sloj radi s ulaznim podacima, spremljena je u težinskom vektoru sloja, također nazvanom *parametru sloja*. U ovom kontekstu, učenje znači pronalazak optimalnih parametara koji će ispravno preslikati ulazne podatke u tražene (ispravne) izlazne podatke. Problem je u tome što duboka neuralna mreža ima tisuće, pa čak i preko milijun parametara.

Zbog toga se definira funkcija gubitka koja uzima predviđanje mreže i traženi izlazni podatak te se izračunava udaljenost koju je potrebno minimizirati (slika 4.5).



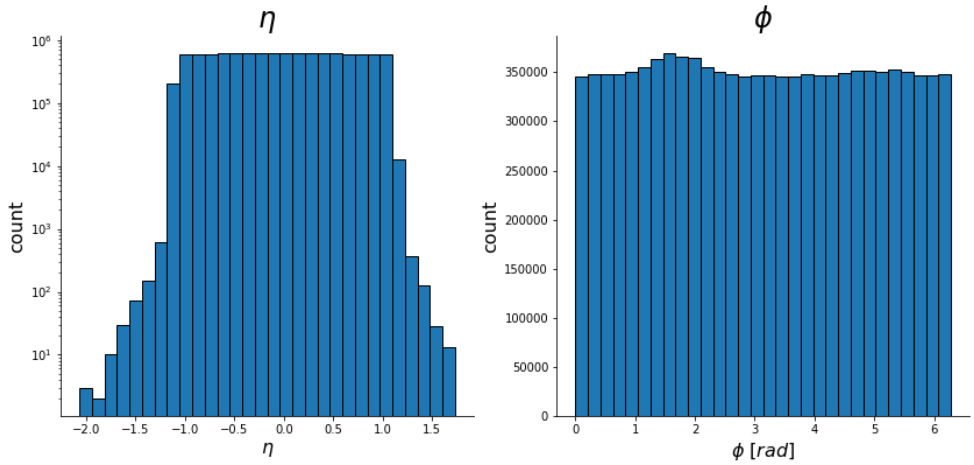

Slika 4.5: Funkcija gubitka daje podatak koliko je trenutno stanje mreže blizu traženog rezultata [24].

## *4.3 TensorFlow*

TensorFlow je open-source platforma za stvaranje i korištenje raznih modela strojnog učenja. Implementira već gotove algoritme potrebne za učenje računala. TensorFlow podržava stvaranje modela i učenje u više programskih jezika, uključujući Python, Java, C/C++. U principu postoje dvije vrste implementacije TensorFlow-a; mogu se koristiti ugrađeni estimatori [26] koji su gotove neuralne mreže s mogućnošću prilagođavanja, ili se koristi Keras. Keras je API koji omogućava implementaciju popularnih načina strojnog učenja u vlastiti kôd. U ovom istraživanju većinom koristimo Keras API pri izboru slojeva i funkcija prilikom definiranja modela. Svi programi napisani su u programskom jeziku Python 3.7.4 te se koristi TensorFlow 2.6.0. Svi sistemi strojnog učenja unutar TensorFlow-a koriste podatke spremljene u višedimenzionalne nizove (NumPy array) - tenzore. U jeziku strojnog učenja, tenzor je općenita struktura u koju se spremaju podaci. Prilikom dubokog učenja, većinom se manipuliraju tenzori dimenzija od 0 do 4. Prva os tenzora je uvijek redni broj tenzora u listi uzoraka. Modeli dubokog učenja ne mogu obraditi cijelu listu podataka od jednom, nego se dijele u manje serije (eng. *batches*).

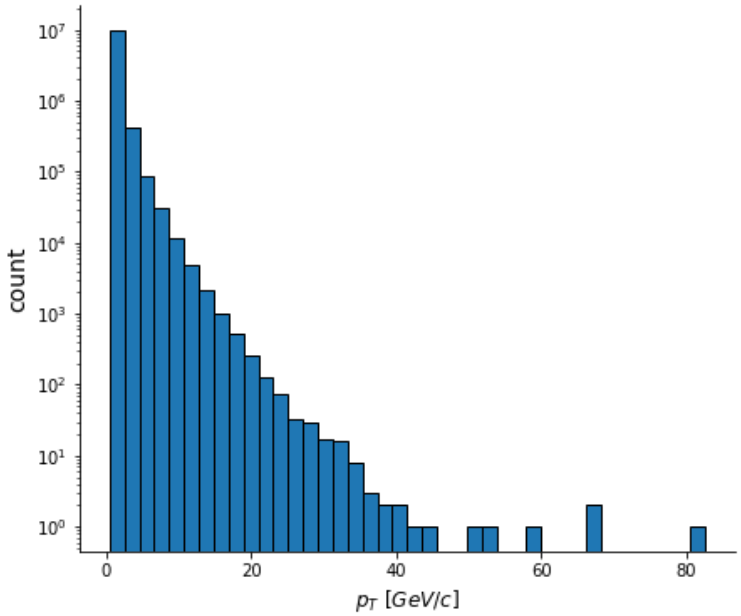
# 5 Simulacije i rezultati

Generirani su podaci za 2,000,000 p-p sudara pri $\sqrt{s_{\mathrm{pp}}} = 7$ TeV energiji u programskom paketu PYTHIA. PYTHIA je standardizirani program za generiranje događaja u hadron-hadron, lepton-lepton ili hadron-lepton sudarima. Pouzdane simulacije za teško-ionske sudare u ultrarelativističkom režimu trenutno ne postoje. Zbog jednostavnosti uzeto je da je laboratorijski sustav ekvivalentan sustavu središta mase p-p sudara. Od 2,000,000 događaja uzeto je samo oko 600,000 događaja visokog $(\geq 20)$ multipliciteta. Detektor se postavlja kao $\mathcal{N} \times \mathcal{N}$ mreža koja pokriva azimutalne kutove $(0, \phi_{\max})$ te pseudorapiditet $(-\eta_0, \eta_0)$. Kada je $\phi_{\max} = 2\pi$, simulira se detektor nalik jednom sloju ITS-a. Prostorna raspodjela čestica prikazana je na slici 5.1 dok je impulsna raspodjela prikazana na slici 5.2. U svim eksperimentima uzeto je $\mathcal{N} = 32$ kao broj kanala detektora u jednom smjeru.




Slika 5.1: Lijeva slika predstavlja čitavu $\eta$ distribuciju signala. Na desnoj slici je prikazana $\Phi$ distribucija signala za sve azimutalne kuteve $0 < \phi < 2\pi$. Y-osi predstavljaju prikazane gustoće brojnosti čestica, lijevo $\mathrm{d}N/\mathrm{d}\eta$ te desno $\mathrm{d}N/\mathrm{d}\phi$. Uzete su u obzir samo čestice s $p_T > 0.5$ GeV/c te samo događaji multipliceta većeg od 20.

Simulirani su podaci spremljeni u polju oblika $(p_{T,j}^{(i)}, \eta_j^{(i)}, \phi_j^{(i)})$ gdje je $i$ redni broj događaja, a $j$ redni broj čestice u jednom događaju. Iz svakog događaja detektor sakuplja podatke u diskretnom obliku slike $\epsilon^{(i)}[\hat{x}, \hat{y}]$ gdje su $\hat{x}$ i $\hat{y}$ pikseli slike $\hat{x}, \hat{y} \in \{0, \ldots, N-1\}$ a $\epsilon$ očitani signal. U eksperimentima se očitavanje $\epsilon^{(i)}[\hat{x}, \hat{y}]$ kalibrira u deponiranu energiju. Vektor impulsa ili transverzalna komponenta impulsa

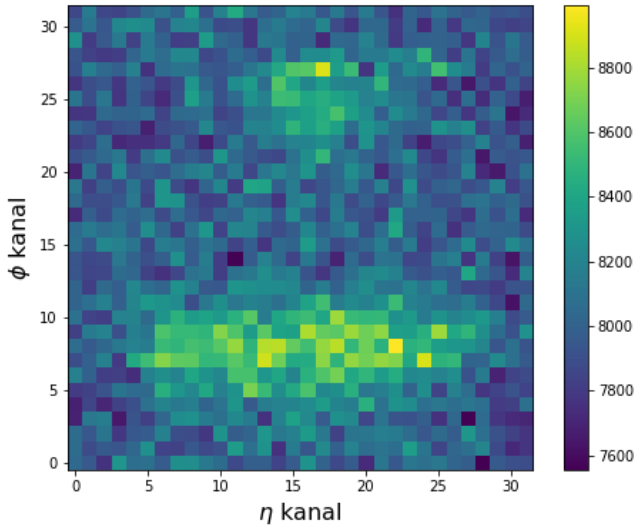



Slika 5.2: Distribucija transverzalnog momenta fragmenata sudara.

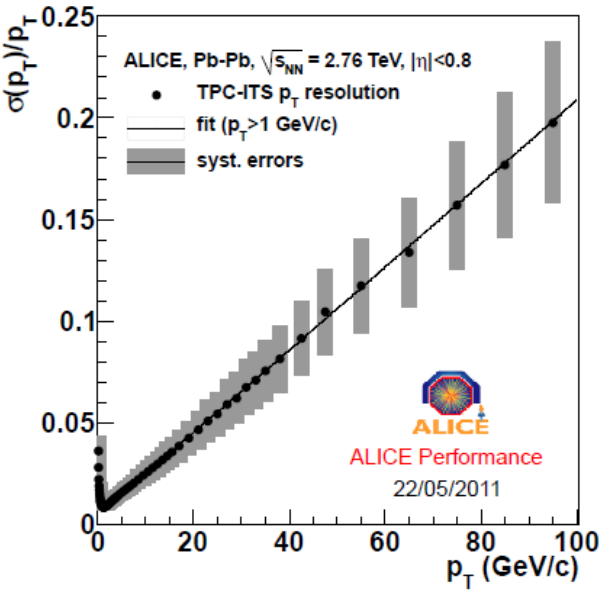
dobiva se tijekom akvizicije podataka ili prije analize pomoću informacija iz ostalih detektorskih sustava putem praćenja i $\frac{\mathrm{d}E}{\mathrm{d}x}$ mjerenja. Zato je uzeta pretpostavka da intenzitet piksela u ovom eksperimentu predstavlja samo očitani $p_T$ čestice koja je prošla kroz detektor. Na slici 5.3 prikazano je kumulativno očitanje svih događaja uz pretpostavku idealnog detektora. Primjer kako izgleda pikselizacija jednog događaja prikazano je na slici 5.5.

U realnom detektoru, distribucija istog signala nije savršeno oštra krivulja nego je distribuirana prema slici 5.4. Također deponirana energija ima prostornu raspodjelu takvu da postoji mogućnost da vrijednost napona u susjednim ćelijama prijeđe preko praga za okidanje. Taj efekt modeliran je gausijanskom raspodjelom deponirane energije oko točke upada čestice.

U ultrarelativističkim sudarima, produkcija fragmenata je stohastički proces. Kao takav, ako detektor pokriva samo dio faznog prostora, nije moguće točno rekonstruirati čestice koje upadaju na preostali dio faznog prostora. Neuralna mreza neće moći pronaći ni lokalne ni globalne uzorke kojima bi mogla predvidjeti nepoznati dio faznog prostora. Pitanje koje se postavlja jest, iako nijedna neuralna mreža ne može točno utvrditi da postoje čestice u nepoznatom dijelu, može li jednostavna mreža

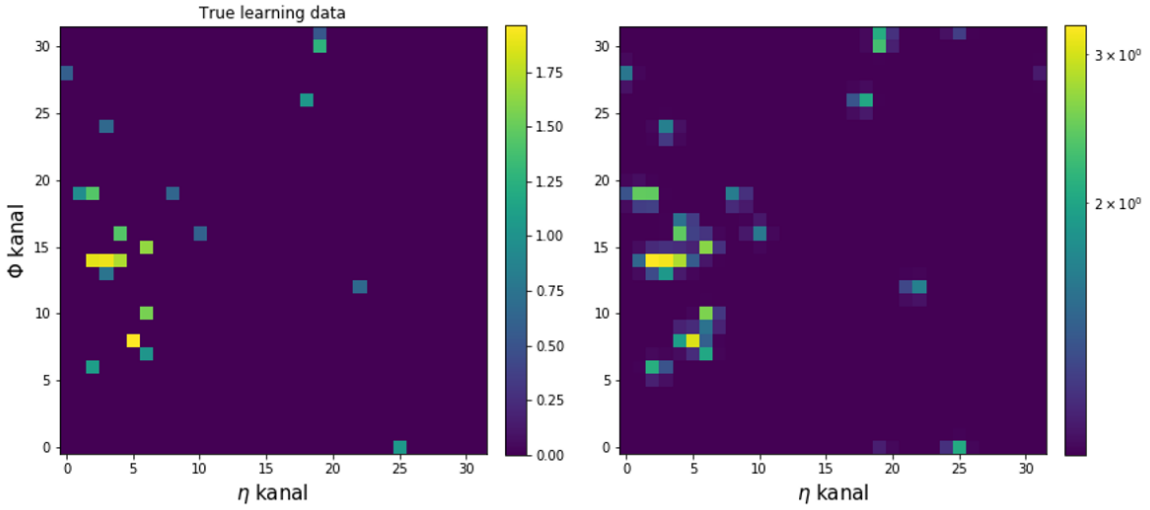



Slika 5.3: Raspodjela ukupnog očitanja piksela za 300,000 događaja. Zamišljeni detektor pokriva $|\eta| < 0.9$ te cijeli azimut $\phi \in [0, 2\pi\rangle$. Grbe u y-smjeru predstavljaju malu anomaliju u uniformoj distribuciji $\phi$ koordinate (slika 5.1).




Slika 5.4: Ovisnost fluktuacije očitavanja $p_T$ u ITS+TPC sustava kombiniranog praćenja čestica [11].

barem rekonstruirati globalne distribucije signala koje se vide u podacima? Drugim riječima, ako u detektoru postoje šupljine (tzv. *mrtve ćelije*) i naivno se postavlja neuralna mreža koja će pokušati naučiti uzorke i predvidjeti signal praznine, hoće li rezultat mreže za te mrtve kanale biti distribuiran prema grafu 5.2 ?




Slika 5.5: Za detektor koji pokriva $|\eta| < 0.9$ u x-smjeru te cijeli azimut $\phi \in [0, 2\pi\rangle$, prikazan je primjer jednog događaja. Lijeva slika prestavlja idealni detektor gdje je izračunati $p_T$ egzaktno jednak fizikalnoj veličini. Desna slika predstavlja realni detektor gdje je deponirana energija distribuirana i u susjedne ćelije te sam signal ima inherentnu gausijansku grešku prema grafu 5.4. Efekt je prikazan logaritamski.

## 5.1 Predobrada podataka

Kao što se vidi na grafu 5.2, $p_T$-distribucija je otprilike eksponencijalna i čuva oblik kad se podaci pikseliziraju na $32 \times 32$ mrežu detektora. Podaci se prvo transformiraju u oblik

$$X'(p_T) = \ln(p_T + c) - \ln c \tag{5.1}$$

pri čemu je $c$ neka mala konstanta. Ovakva tranformacija je ekvivalentna skaliranju $p_T \to \frac{p_T}{c}$ te uzimanju $c = 1$. Ponovno skaliranje $p_T$ ne mijenja izgled krivulje pa je transformacija pojednostavljena uzimanjem $c = 1$. Jedan razlog odabira transformacije 5.1 jest da u događaj-po-događaj slikama, gausijanska mrlja nije toliko značajna da kvantificira razliku između idealne i realne slike. Neuralna mreža teško prepoznaje prostornu širinu pojedinog signala i prilikom regresije zatrne na takvim težinama da se rekonstruira identična slika. Normalna raspodjela trne puno sporije u logaritamskoj nego linearnoj skali. Drugi razlog je što signali visokog intenziteta dominiraju male signale u linearnim slikama. Pošto za stvarnu eksponencijalnu $p_T$ distribuciju visoki signali razlikuju se za veliki faktor od malih, to u logaritamskoj skali znači razlika za konstantu. Time se guše doprinosi visokih anomalnih signala. Zadnje je da je proizvoljni normalizacijski faktor između 3-5 umjesto reda veličine

100. Uočeno je da neuralna mreža puno bolje konvergira kad su vrijednosti slike razmazane jednoličnije u intervalu $\langle 0,\ 1\rangle$.

## 5.2 Izgled neuralne mreže

Cilj je mrežu postaviti što jednostavnije. Pošto je bitan samo globalni izgled događaja gdje ne postoje konzistentni lokalni uzorci ili korelacije, definira se jednostavna gusto povezana neuralna mreža s četiri sloja.

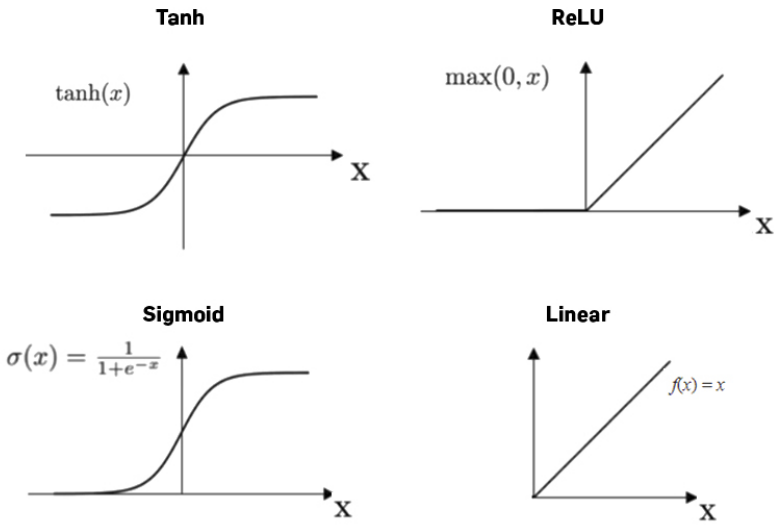
```
img_shape = (32,32)
model = Sequential()
model.add(Flatten(input_shape=img_shape))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(32*32, activation = 'relu'))
model.add(Reshape((32,32)))
```

U enkoderskom dijelu mreže, podatak tenzora ulaza, slika dimenzije $32 \times 32$, preslikava se u vektor dimenzije $32 \cdot 32 = 1024$. Takav vektor zatim prolazi kroz sloj od 128 neurona (čvorova). Svaki čvor sadrži 1024+1 težinskih parametara za svaki piksel ulaznog vektora iz funkcije `flatten()`. Iz tog sloja izlazi vektor dimenzije 1024 koji ulazi u drugi sloj od 64 neurona, te izlaz iz tog sloja postaje ulaz za idući sloj neurona. Kaže se da su ti slojevi skriveni jer njihov izlaz nije izlaz čitave mreže. Cijeli model ima 279,872 slobodnih parametara koje pokušava optimizirati. Aktivacijska funkcija ReLU koristi se jer popularne sigmoidna i tanh aktivacijske funkcije imaju iščezavajući gradijent na velikim vrijednostima. Također su te funkcije samo osjetljive na promjene oko svojih srednjih vrijednosti pa za aktivacijske vrijednosti daleko od nule vrlo sporo konvergiraju.

Mreža je postavljena kao funkcija koja uzima sliku $X^{(i)}$ iz liste *pokvarenih* slika $X$ i pokušava rekonstruirati sliku $Y^{(i)}$ iz pripadne liste *idealnih* slika, $Y$.

## 5.3 Analiza podataka

Nepoznati dio faznog prostora modelira se $n_x \times n_y$ dijelom detektora koji predstavlja mrtve ćelije. Zbog bolje konvergencije stavljeno je da mrtve ćelije daju izlaz pozadine

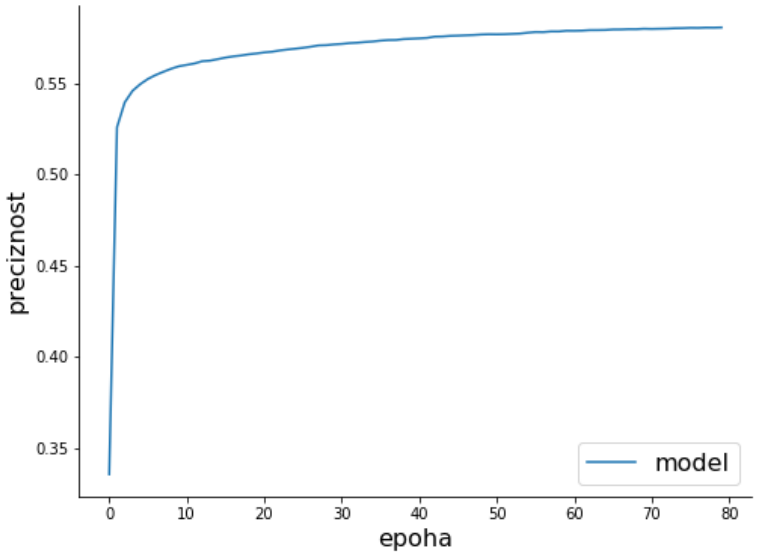



Slika 5.6: Najčešće korištene aktivacijske funkcije [27].

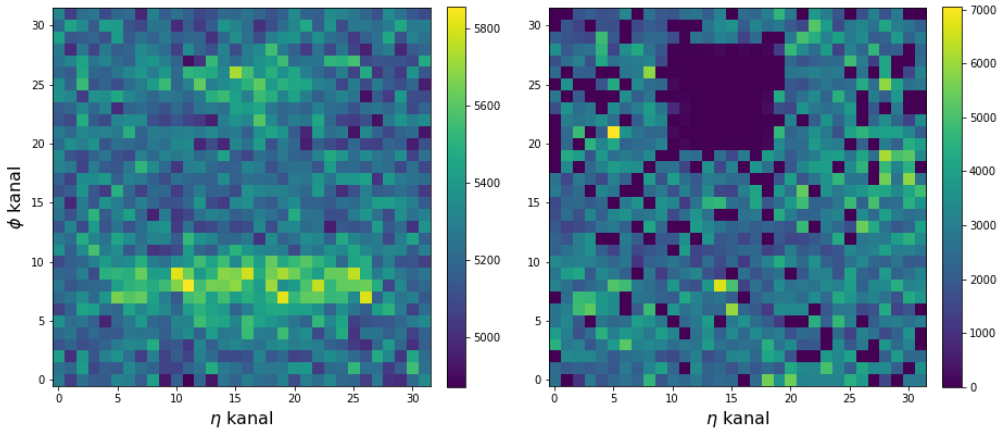
koji je normalno distribuiran signal sa sredinom $10^{-5}$ i standardnom devijacijom $10^{-6}$ čime se modelira šum. Treniranje modela vrši se na serijama podataka (eng. *batch*) veličine 100. Kroz neuralnu mrežu prođe 100 slika prije nego što se težinske vrijednosti ažuriraju. Takvo je uprosječivanje opravdano jer pojedina slika ima relativno malo punih piksela pa bi ažuriranje nakon svake pojedine slike bilo nestabilno. Ako u prosjeku postoji $N_{\text{avg}} = 25$ čestica po događaju onda u seriji od 100 događaja očekivani je broj čestica u kanalu $\Delta N_{\text{avg}}/1\,\text{ch} = 2.4$. To jest svaki piksel ulaznog tenzora sudjeluje u ažuriranju težinskih vrijednosti mreže. Ukupno za trening se koristi 400,000 parova slika $(X^{(i)}, Y^{(i)})$ gdje su $X$ pokvarene slike generirane dodavanjem detektorskih efekata na $Y$: male gausijanske pozadine reda veličine $10^{-6}$ , distribucije signala upadne čestice u susjedne ćelije te dodavanje efekta mrtvih ćelija. Lista idealnih slika, $Y$ nastaje direktnom događaj-po-događaj diskretizacijom egzaktnih podataka iz simulacije u zadni detektor. Broj epoha optimizacije uzima se između 50 i 150. Iako se preciznost mreže stabilizira nakon desetak epoha, i dalje je prisutan vrlo spor rast na $\geq 40$ epoha što je vidljivo na slici 5.7.

Intenziteti piksela pojedine slike $X^{(i)}$ ili $Y^{(i)}$ dani su logaritamski prema izrazu 5.1. Za testiranje modela koristi se lista drugih 200,000 pokvarenih slika $Z^{(i)}$ u log-skali, pomoću kojih mreža daje listu izlaznih slika $P$. Potom se izlazni tenzor $P$ neuralne mreže skalira se natrag u linearnu skalu inverzijom jednadžbe 5.1.
Rezultat takve analize prikazan je na slici 5.8.

U slučaju kada nema mrtvih piksela, razlika između realnih i idealnih slika bila




Slika 5.7: Preciznost optimizacije mreže ovisno o epohi.



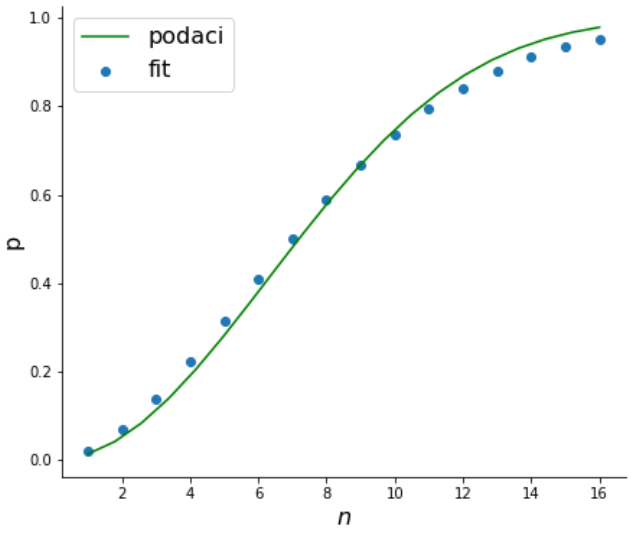

Slika 5.8: Desni graf predstavlja sumu svih slika rezultata neuralne mreže prilikom testiranja modela. Lijevi graf je očekivana suma svih slika. Događaj-po-događaj analiza i usporedba slika ne daje uvid u rezultat mreže pa se radi preglednosti slike sumiraju.

bi efektivno samo oduzimanje gausijanske disperzije oko središnjeg signala. Dakle idealna bi mreža ulaznu sliku otprilike preslikala u samu istu, za sve slike.

Kao što je napomenuto, stohastički proces stvaranja čestica je nepredvidljiv, ali čestice se svejedno međusobno grupiraju. Ako pretpostavimo da su čestice nezavisne, tada za $N_{\text{avg}}$ čestica kao prosječni multiplicitet slike, vjerojatnost da se barem jedna

nađe u nekom potprostoru detektora veličine $n \times n$ glasi

$$p(n) = 1 - \left(1 - \frac{n^2}{32^2}\right)^{N_{\text{avg}}}. \tag{5.2}$$




Slika 5.9: Os y predstavlja omjer broja slika gdje postoji signali u nekom kvadratnom području piksela dimenzije $n \times n$ te ukupnog broja slika. X-os je broj piksela kvadrata u jednom smjeru. Veličina slika je $32 \times 32$.

Prosječni multiplicitet $N_{avg}$ iznosi 19.4 u detektiranim podacima za $|\eta| < 0.9$. Nelinearnom regresijom jednadžbe 5.2 na podatke 5.9 dobiva se $N_{\text{avg}}^{\text{fit}} = 13.4 \pm 0.1$. Omjer $N_{avg}/N_{\text{avg}}^{\text{fit}} \approx 0.7$ ukazuje na činjenicu da čestice nisu potpuno nezavisne, nego u detektor često ulaze u parovima. Naime, prilikom visoko-energetskih p-p sudara pojedini nezavisni fragmenti koji se izbijaju iz protona raspadaju se na par čestica. Neovisno o međusobnoj orijentaciji impulsa svake od dviju čestica u sustavu središta mase fragmenta, u laboratorijskom sustavu dvije su čestice usmjerene u vrlo bliske kutove zbog velikog Lorentzovog potiska.

Širina mrtvog područja varira se od $n = 6$ do $n = 12$ kanala. Za pojedinu širinu, neuralna mreža se resetira te se ponovno uči na nizu 400,000 slika događaja oblika 5.5 gdje je na $X$ slike nametnut uvjet da u kvadratu slike dimenzije $n \times n$ signal uvijek isključivo pozadina. Zatim se na temlju preostalih 200,000 slika $Z^{(i)}$ vrši testiranje modela. Širina manja od $n = 6$ ne uzima se jer mala mrtva područja imaju vrlo male kumulativne signale u testnim podacima pa je statistika izlaznih slika iz neuralne mreže nepouzdana u tom području. Prilikom učenja, pokvarene i idealne slike u

prosjeku se vrlo malo razlikuju.

Slično kao i kod ostalih slika, $p_T$-distribucija unutar mrtvog područja za rezultate testiranja neuralne mreže gleda se samo za *cutoff* $p_T > 0.5$ GeV/c. Uočeno je da neovisno o širini mrtvog područja, pripadajuća distribucija slijedi očekivanu eksponencijalnu raspodjelu (dodatak A). Da bi se usporedile dvije raspodjele iz dobivenih slika i idealnih slika, normirane su na vrijednost 1 u prvom binu log-histograma. Drugim riječima, za distribucije $N^{\text{real}}(p)$ te $N^{\text{neural}}(p)$ definira se

$$\nu^{\text{real}}(p_T) = \frac{1}{\ln\left|\frac{\mathrm{d}N^{\text{real}}}{\mathrm{d}p_T}\right|_{p_T=0.5}} \ln\left|\frac{\mathrm{d}N^{\text{real}}}{\mathrm{d}p_T}\right|, \tag{5.3}$$

$$\nu^{\text{neural}}(p_T) = \frac{1}{\ln\left|\frac{\mathrm{d}N^{\text{neural}}}{\mathrm{d}p_T}\right|_{p_T=0.5}} \ln\left|\frac{\mathrm{d}N^{\text{neural}}}{\mathrm{d}p_T}\right|. \tag{5.4}$$

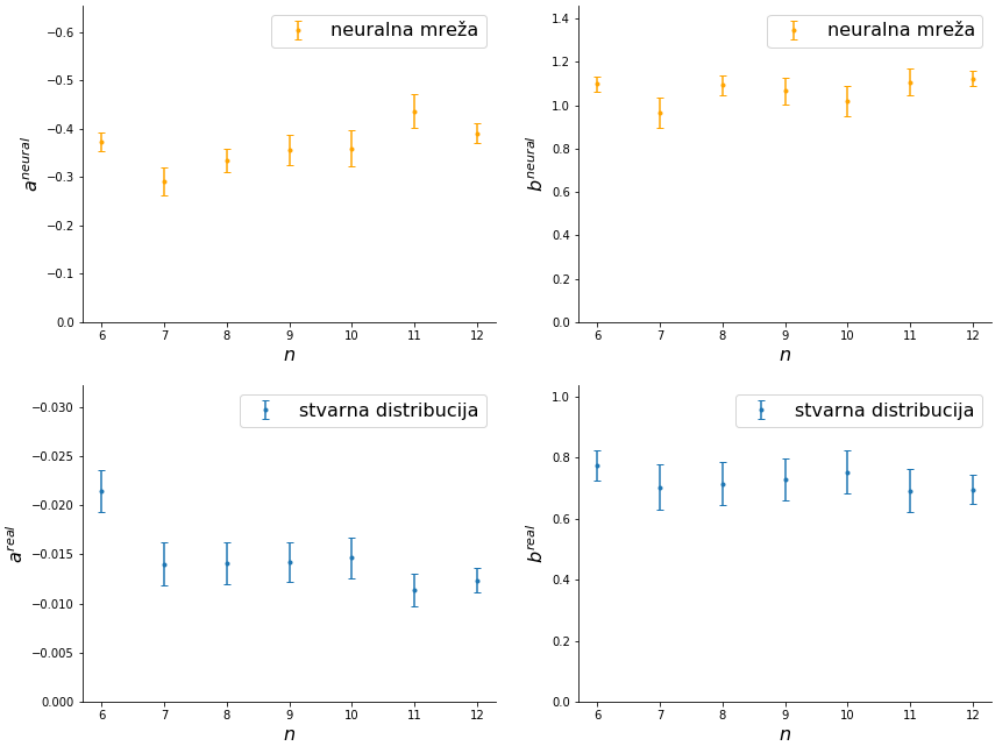
Pripadajući histogrami distribucije signala prikazani su u dodatku A.2. Skalirane distribucije modeliraju se linearnom regresijom $f(n; a, b) = an + b$ te su pripadajući koeficijenti $a^{\text{neural}}, b^{\text{neural}}$ te $a^{\text{real}}, b^{\text{real}}$ prikazani ovisno o širini $n$ mrtvog područja na slici 5.10. U realnim (idealnim) slikama distribucija se mijenja jer povećanjem širine $n$ obuhvaća se više faznog prostora konačnog broja čestica. Također zbog konačnog broja događaja uvodi se mala slučajna pogreška na oblik raspodjele.

Grafovi 5.10 i A.3 pokazuju da ako se impulsna skala te broj registriranih signala u jedinici impulsa promijene za faktore

$$p_T - p_{T,0} \longrightarrow \frac{a^{\text{real}}}{a^{\text{neural}}}(p_T - p_{T,0}) \equiv K_0(p_T - p_{T,0}) \tag{5.5}$$

$$\left|\frac{\mathrm{d}N}{\mathrm{d}p_T}\right| \longrightarrow \exp(b^{\text{real}} - b^{\text{neural}}) \frac{\ln\left|\frac{\mathrm{d}N^{\text{real}}}{\mathrm{d}p_T}\right|_{p_T=0.5}}{\ln\left|\frac{\mathrm{d}N^{\text{neural}}}{\mathrm{d}p_T}\right|_{p_T=0.5}} \left|\frac{\mathrm{d}N}{\mathrm{d}p_T}\right| \equiv K_1 \left|\frac{\mathrm{d}N}{\mathrm{d}p_T}\right| \tag{5.6}$$

gdje je $p_{T,0} = 0.5$ cutoff signal, tada se krivulje poklapaju. Neuralna mreža pogodi oblik krivulje distribucije, samo što intenzitet signala, $p_T$, promaši otprilike za faktor $K_0$, a gustoću čestica po jedinici korigiranog signala za $K_1$. Uočeno je da za pokvarene slike $X^{(i)}$ i $Z^{(i)}$ iznos pozadinog signala u mora biti reda veličine $10^{-3}$ do $10^{-5}$ s pripadnim gausijanskim šumom reda veličine manje, da bi jednadžbe 5.5 i 5.6 vrijedile. Za pozadinu egzaktno nula, mreža preslikava piksele mrtvih ćelija u signale reda veličine pozadine preostalih kanala. To jest, nema čestica iznad cutoff impulsa $p_T > 0.5$ GeV/c. Za pozadine reda veličine $10^{-2}$ i više, pozadina mrtvih ćelija postaje




Slika 5.10: Koeficijenti $a$ i $b$ proizlaze iz linearne regresije na log-distribuciju gustoće $p_T$ signala u mrtvom dijelu detektora. Gornji grafovi predstavljaju ovisnost koeficijenata o širini za rezultate neuralne mreže. Donji grafovi predstavljaju ovisnost koeficijenata $a$ i $b$ za idealne slike.

usporediva s veličinom signala i mreža više ne predviđa eksponencijalnu distribuciju nego se u svakom događaju očituje visoka pozadina bez značajnog broja čestica iznad cutoff impulsa.

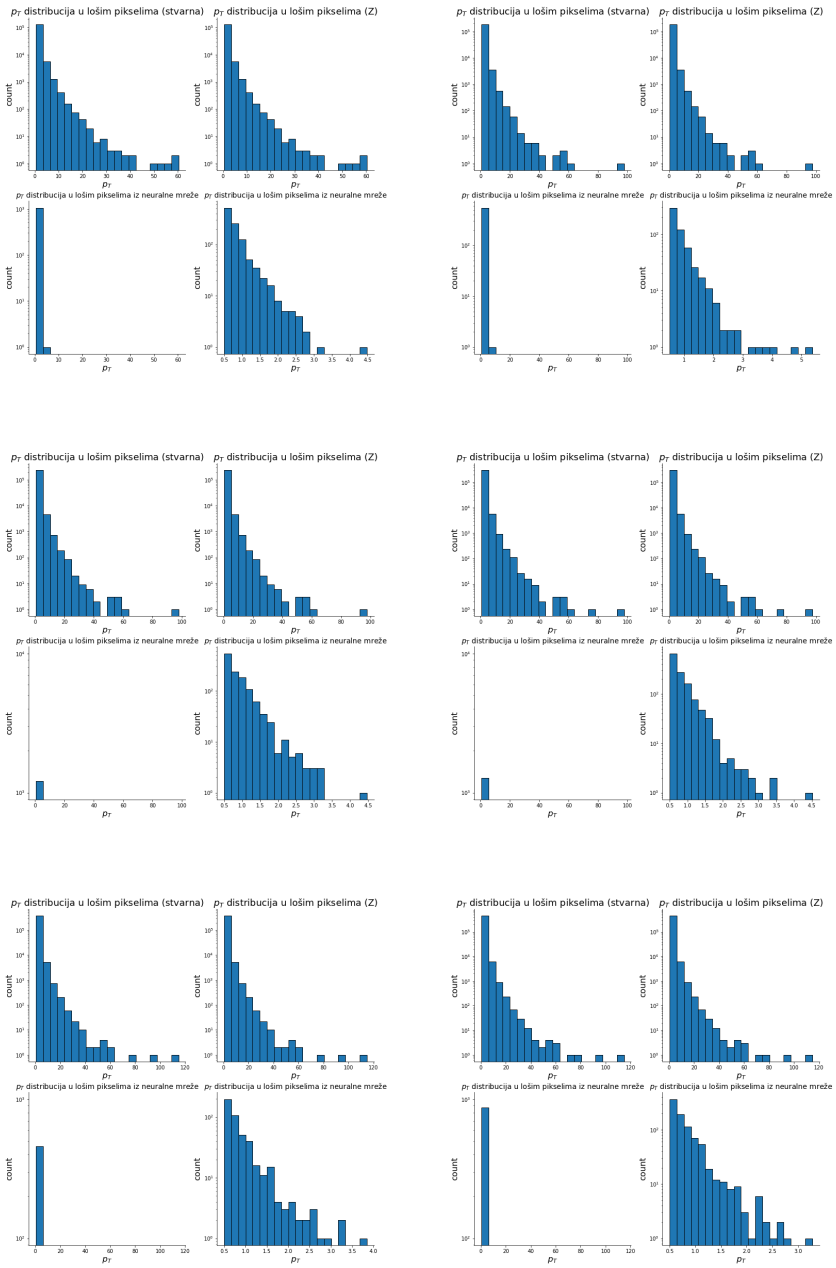
# 6 Zaključak

U ovom istraživanju simulirani su proton-proton sudari na $\sqrt{s_{pp}} = 7$ TeV u sustavu centra mase. Osmišljen je cilindrični detektor kao mreža $32 \times 32$ detektora koji registriraju transverzalni impuls čestica nalik na jedan sloj ITS sustava. Detektor pokriva cijeli azimut i pseudorapiditete u intervalu $|\eta| < 0.9$ gdje se gustoća upadnih $\mathrm{d}^2N/(\mathrm{d}\eta\,\mathrm{d}\phi)$ čestica znatno ne mijenja. Takav idealni detektor diskretizira egzaktne čestice oblika $(p_T, \eta, \phi)$ i zapisuje impulse čestica kao vrijednost pripadnog piksela (signal) slike gdje jedna slika predstavlja jedan događaj u detektoru. Zamišljen je i drugi detektor koji nameće varijabilnu gausijansku disperziju signala svake čestice u susjedne kanale te se na detektor može dodati područje mrtvih ćelija - kvadratni dio $n \times n$ mreže detektora gdje je signal nula. Točna rekonstrukcija signala u mrtvim ćelijama nije moguća zbog stohastične prirode stvaranja i distribucije čestica u faznom prostoru. Pokazano je da značajan dio upadnih čestica u prosjeku korelira s nekom drugom česticom te da se takav par mjeri u istim ili bliskim ćelijama detektora. Problem rekonstrukcije slike pokušali smo analizirati korištenjem metode strojnog učenja. Pošto u slici pojedinih događaja nema mjerljivih uzoraka, model strojnog učenja zadaje se kao jednostavna mreža četiri sloja gusto povezanih neurona. Problem nije rekonstruirati signale u mrtvom području događaj-po-događaj, nego predvidjeti distribuciju signala kroz cijeli niz slika za testiranje. U slučaju nepostojanja mrtvih ćelija, neuralna mreža pokušava klasterirati signale stvorene gausijanskom disperzijom. Intenzitet u obližnjim pikselima znatno je manji od intenziteta središnje ćelije kroz koju čestica prolazi. Zbog problema saturacije neuralne mreže oko točke oko funkcije identitete na polju slika, slike se razmatraju u logaritamskoj skali. Korištenjem neuralne mreže rekonstruiran je oblik distribucije signala u mrtvim ćelijama. Mijenjanjem veličine mrtvog kvadrata ustanovljeno je da faktori koji transformiraju naučenu distribuciju u realnu (očekivanu) ostaju unutar reda veličine neovisno o širini mrtvog područja ako kvadrat čini između 3% do 15% površine detektora.

# Dodaci

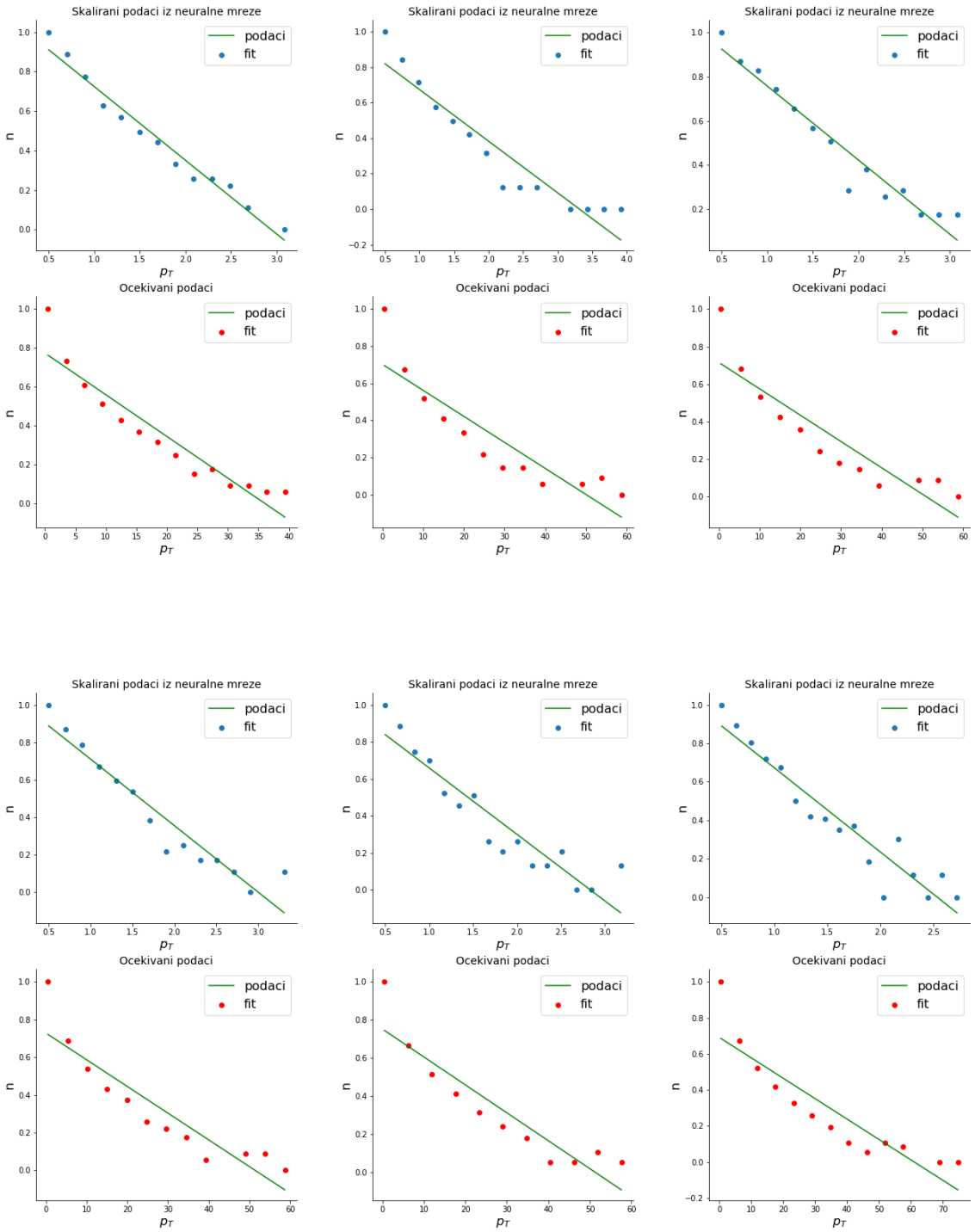
## Dodatak A Podaci

### *A.1* $p_T$ *distribucija unutar mrtvog područja*



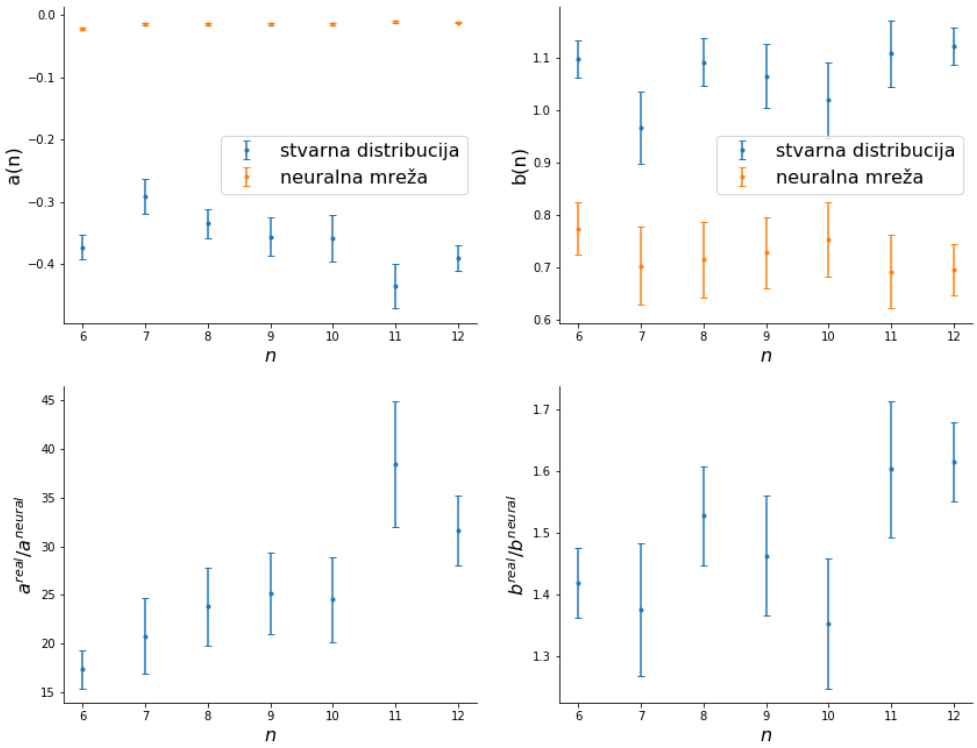

Slika A.1: Histogrami $p_T$ distribucije u mrtvom području za 6 različitih širina $n$. Za svaki $n$, gornja dva histograma su očekivana $p_T$ distribucija dok su donja dva naučena $p_T$ distribucija u očekivanoj $p_T$ skali (lijevo) te do maksimalnog $p_T$ (desno).

## A.2 $p_T$ distribucija unutar mrtvog područja - regresija




Slika A.2: Izračun regresijskih koeficijenata $a^{\text{neural}}$, $b^{\text{neural}}$ (plave točke podataka) te $a^{\text{real}}$, $b^{\text{real}}$ (crvene točke podataka).

## A.3 $p_T$ distribucija unutar mrtvog područja - regresija




Slika A.3: Ovisnost regresijskih koeficijenata o širini mrtvog područja, $n$.

## A.4 Kôd za definicije funkcija

```
import numpy as np
import matplotlib.pyplot as plt
import random as rnd
import math

from math import pi as pi
from numpy.random import normal as norm
from numpy.random import standard_exponential as nexp
from numpy.random import exponential as exp
from matplotlib.colors import LogNorm
```

```
from scipy.optimize import curve_fit

twopi = 2*pi

import os

def LimitPtSubevents(events, ptmin=0.0, ptmax = 100):
    newevents = events.copy()
    for event in newevents:
        for subevent in event:
            if (subevent[0] > 0 and subevent[0] < ptmin) or
               subevent[0] > ptmax:
                subevent[0] = 0.
                subevent[1] = 0.
                subevent[2] = 0.
    return newevents

def FindMultiplicity(x):
    xshape = np.shape(x)
    if (len(xshape)==2):   #jedan dogadjaj
            return np.count_nonzero(x[:,0])
    elif (len(xshape)==3): #vise dogadjaja
        listnum = []
        for i in range(len(x)):
            listnum.append(np.count_nonzero(x[i][:,0]))
        return listnum
    else:
        return False

def eta_to_theta(eta):
    theta = 2*math.atan(math.exp(-eta))
    if theta<0:
        theta = math.pi + th
```

```
    return theta

def ptError(pt):
    if pt<1e-2:
        return pt
    sigma = 0.167/80. * pt*pt
    pt1 = pt + rnd.gauss(0,sigma)
    if pt1<0.0:
        return 0.0
    return pt1

def etaError(eta):
    if eta==0.0:
        return eta
    sigma_deg = 0.1
    sigma = math.pi/180 * sigma_deg
    th = eta_to_theta(eta)
    eta1 = eta + rnd.gauss(0, abs(sigma/math.sin(th)))
    return eta1

def phiError(phi):
    if phi==0.0:
        return phi
    sigma_deg = 0.1
    sigma = math.pi/180 * sigma_deg
    phi1 = phi + rnd.gauss(0, sigma)
    if phi1 < 0.0:
        phi1 = math.pi+phi1
    elif phi1 > 2*math.pi:
        phi1 = phi1 - 2*math.pi
    return phi1

def AddErrorEtaPhi(events):
```

```
    newEvents = np.empty([len(events),30,3])
    for i in range(len(events)):
        for j in range(30):
            newEvents[i][j][1] = etaError(events[i][j][1])
            newEvents[i][j][2] = phiError(events[i][j][2])
    return newEvents

def AddPixelError(images, pixels=[], const=1e-12, sigma=1e
   -15): #pixels = N x 2 array, dodaje se NA SLIKE
    if len(pixels)==0:
        return images

    newImages = images.copy()
    for pixel in pixels:
        for image in newImages:
            image[pixel[0], pixel[1]] = const + rnd.gauss(0,
               sigma)
    return newImages

def AddBackground(images, const = 1e-6, sigma=1e-9):
    return images + const + norm(loc=0., scale=sigma, size=np
       .shape(images))

def AddPtError(images):
    vfunc = np.vectorize(ptError)
    return vfunc(images)

def FindImagesWithPixels(images, pixels): #returns list of
   images which have pixel different than 0.0
    listnum = []
    for i in range(len(images)):
        for pixel in pixels:
            if images[i][pixel[0], pixel[1]] > 0.0:
```

```
                listnum.append(i)
                break
    return listnum

def FindPixelValues(images, pixel):
    listnum = []
    for image in images:
        listnum.append(images[pixel[0], pixel[1]])
    return listnum

def FindMaxValues(events):
    return [np.max(events[:,:,0]), np.min(events[:,:,1]), np.
      max(events[:,:,1])]

def FindMaxPt(x):
    eventsize = (30,3)
    xshape = np.shape(x)
    if (len(xshape)==2):
        if (xshape==eventsize): #one event
            return np.max(x[:,0])
        else: #one image
            return np.max(x)
    elif (len(xshape)==3):
        if (xshape[1:3] == eventsize): #multiple events
            return np.max(x[:,:,0])
        else: #multiple images
            return np.max(x)
    else:
        return false

def FindMultiplicity(x):
    xshape = np.shape(x)
    if (len(xshape)==2):
```

```
            return np.count_nonzero(x[:,0])
    elif (len(xshape)==3):
        listnum = []
        for i in range(len(x)):
            listnum.append(np.count_nonzero(x[i][:,0]))
        return listnum
    else:
        return false

def FindImageMultiplicity(images):
    listnum = []
    for image in images:
        listnum.append(np.count_nonzero(image))
    return np.array(listnum)

def LimitPt(events, ptmin=0, ptmax=500):
    newEvents = np.zeros([len(events),30,3])
    for i in range(len(events)):
        for j in range(30):
            if events[i][j][0] > ptmin and events[i][j][0] <
               ptmax:
                newEvents[i][j] = events[i][j]
            elif events[i][j][0] > ptmax:
                newEvents[i][j][0] = ptmax
    return newEvents

def LimitEta(events, etamin=-100, etamax=100):
    newEvents = np.zeros([len(events),30,3])
    for i in range(len(events)):
        for j in range(30):
            if events[i][j][1] > etamin and events[i][j][1] <
               etamax:
                newEvents[i][j] = events[i][j]
```

```
    return newEvents

def LimitPhi(events, phimin=0, phimax=3.15):
    newEvents = np.zeros([len(events),30,3])
    for i in range(len(events)):
        for j in range(30):
            if events[i][j][2] > phimin and events[i][j][2] <
                phimax:
                newEvents[i][j] = events[i][j]
    return newEvents

def NeighboursXY(coordinate, img_size=32, IsFullAzimuth =
   True, IsFullPolar = True):
    lst, edge = [],[[-3,0], [3,0],[0,-3],[0,3]]
    (x,y) = coordinate

    if IsFullAzimuth and not IsFullPolar:
        for dy in range(-2,3):
            for dx in range(-2,3):
                if x+dx < 0 or x+dx >= img_size:
                    continue
                else:
                    lst.append([x+dx, y+dy])
        for (dx,dy) in edge:
            if x+dx < 0 or x+dx >= img_size:
                continue
            else:
                lst.append([x+dx, y+dy])

    elif not IsFullAzimuth and not IsFullPolar:
        for dy in range(-2,3):
            for dx in range(-2,3):
                if x+dx < 0 or x+dx >= img_size:
```

```
                continue
            elif y+dy < 0 or y+dy >= img_size:
                continue
            else:
                lst.append([x+dx, y+dy])
    for (dx,dy) in edge:
        if x+dx < 0 or x+dx >= img_size:
            continue
        elif y+dy < 0 or y+dy >= img_size:
            continue
        else:
            lst.append([x+dx, y+dy])

elif IsFullAzimuth and IsFullPolar:
    for dy in range(-2,3):
        for dx in range(-2,3):
            lst.append([x+dx, y+dy])
    for (dx,dy) in edge:
        lst.append([x+dx, y+dy])

elif not IsFullAzimuth and IsFullPolar:
    for dy in range(-2,3):
        for dx in range(-2,3):
            if y+dy < 0 or y+dy >= img_size:
                continue
            else:
                lst.append([x+dx, y+dy])
    #dodati [-3,0], [3,0],[0,-3],[0,3]
    for (dx,dy) in edge:
        if y+dy < 0 or y+dy >= img_size:
            continue
        else:
            lst.append([x+dx, y+dy])
```

```
    lst.remove(coordinate)
    return lst

def ArrayToImages(events, img_size=32, etamin=-0.9, etamax
   =0.9, phimin=0.0, phimax=twopi, sigmaX=0.38, sigmaY=0.38):
    #generiranje Nx(size x size) slika
    images = np.zeros([len(events), img_size, img_size])
    etamin = etamin - 0.0000000000000001
    etamax = etamax + 0.0000000000000001
    phimax = phimax + 0.0000000000000001
    phimin = phimin - 0.0000000000000001
    if abs(phimax - twopi) < 0.01:
        phi_bool = True #pokriva puni azimut
    else:
        phi_bool = False #NE pokriva puni azimut

    #sigme od 0.424661 (1/2 na rubu) do 0.35353 (1/e na rubu)
       , default 0.44
    sX = 2*sigmaX**2
    sY = 2*sigmaY**2

    for i in range(len(events)):
        image,event = images[i], events[i]
        for subevent in event: #[pt, eta, phi]
            if subevent[0]<0.001:
                continue
            x = (subevent[1] - etamin)/(etamax - etamin) *
               img_size
            y = subevent[2] / phimax * img_size

            xindex = math.floor(x)
            yindex = math.floor(y)
```

```
            image[xindex,yindex] += subevent[0]
            neighbours = NeighboursXY([xindex,yindex],
               img_size=img_size, IsFullAzimuth = phi_bool)
            for [x0,y0] in neighbours:
                image[x0 \% img_size, y0 \% img_size] +=
                   subevent[0] * math.exp( -(x0+0.5 -x)**2/sX
                   ) * math.exp( -(y0+0.5 -y)**2/sY)

    return images

def ArrayToImages_Exact(events, img_size=32, etamin=-0.9,
   etamax=0.9, phimin=0.0, phimax=twopi): #generiranje Nx(
   size x size) slika
    images = np.zeros([len(events), img_size, img_size])
    etamin = etamin - 0.0000000000000001
    etamax = etamax + 0.0000000000000001
    phimax = phimax + 0.0000000000000001
    phimin = phimin - 0.0000000000000001

    for i in range(len(events)):
        image,event = images[i], events[i]
        for subevent in event:
            if subevent[0]<0.001:
                continue
            x = (subevent[1] - etamin)/(etamax - etamin) *
               img_size
            y = subevent[2] / phimax * img_size

            xindex = math.floor(x)
            yindex = math.floor(y)
            image[xindex,yindex] += subevent[0]
    return images
```

```
def SubImage(image, p, xlen, ylen, img_size=32):
    minx = np.max( (p[0], 0) )
    maxx = np.min( (p[0]+xlen, img_size))
    miny = np.max( (p[1], 0) )
    maxy = np.min( (p[1]+ylen, img_size))
    return image[minx:maxx , miny:maxy]

def StackImages(images):
    res = np.zeros(shape = np.shape(images[0]))
    for image in images:
        res += image
    return res

def ImagesToPtData(images, cutoff = 1e-3):
    pTs = []
    lenx = np.shape(images)[1]
    leny = np.shape(images)[2]
    for image in images:
        for x in range(lenx):
            for y in range(leny):
                if image[x,y] > cutoff:
                    pTs.append(image[x,y])
    return np.array(pTs)

def AddMaskBackground(images, badpixels):
    newImages = images.copy()
    for pixel in pixels:
        for image in newImages:
            image[pixel[0], pixel[1]] = const
    return newImages

def CreatePixels(p, xlen, ylen, img_size = 32):
```

```
    #stvara piksele oko p (donji lijevi piksel) sa sirinom
       xlen i visinom ylen
    lst = []
    for dx in range(xlen):
        for dy in range(ylen):
            (x,y) = (p[0] + dx, p[1] + dy)
            if x >= 0 and x < img_size and y >= 0 and y <
               img_size:
                lst.append([x,y])
    return lst

def Hist_to_xy(hist): #hist is tuple ( (N), (N+1) )
    return (np.array(hist[1][:-1]), hist[0])

def FilterZero(xs, ys): #izfiltrirava nule iz histograma, da
   se moze logaritmirati
    if len(xs) != len(ys):
        return False
    x,y = [],[]
    ind = 0
    for i in range(len(ys)):
        if ys[i] > 0 and i-ind < 3:
            ind = i
            x.append(xs[i]); y.append(ys[i])
    return (np.array(x), np.array(y))
```

## A.5 Kôd za učenje i izračun koeficijenata $a$ i $b$

```
from tensorflow import Variable
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Input, Dense, Reshape,
   Flatten, Lambda, Add, Subtract, Layer
```

```
from tensorflow.keras.layers import BatchNormalization,
    Activation, ZeroPadding2D, Masking, LSTM, Dot, Multiply,
    RNN, Conv2D, MaxPooling2D, UpSampling2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import load_model

data = np.load('mc_events.npy')
data = LimitPtSubevents(data, ptmin=0.5)
multp = FindMultiplicity(data)
newdata = []
for i in range(len(data)):
    if multp[i] > 20:
        newdata.append(data[i])
data = np.array(newdata)

(ptmin, ptmax) = (0.5000, 80.)
(etamin, etamax) = (-0.9, 0.9)
(phimin, phimax) = (0.0, twopi)
img_size=32
train = LimitEta(train, etamin, etamax)
train = LimitPhi(train, phimin, phimax)
train = LimitPt(train, ptmin, ptmax)
predict = LimitEta(predict, etamin, etamax)
predict = LimitPhi(predict, phimin, phimax)
predict = LimitPt(predict, ptmin, ptmax)

(sigmaX, sigmaY) = (0.44, 0.44)
const_background = 1e-5
sigma_background = 1e-6

Y = ArrayToImages_Exact(train, img_size, etamin, etamax,
    phimin, phimax) #(N, 32, 32)
X = ArrayToImages(train, img_size, etamin, etamax, phimin,
```

```
    phimax, sigmaX, sigmaY) #(N, 32, 32)
X = AddBackground(X, const = const_background, sigma=
    sigma_background)
X = AddPtError(X)

W = ArrayToImages_Exact(predict, img_size, etamin, etamax,
    phimin, phimax) #(M, 32, 32)
Z= ArrayToImages(W, img_size, etamin, etamax, phimin, phimax,
     sigmaX, sigmaY) #(M, 32, 32)
Z = AddBackground(Z, const = const_background, sigma=
    sigma_background)
Z = AddPtError(Z)

X = np.log(trainReal_Images + 1.)
del trainReal_Images

Y = np.log(train_Images + 1.)
del train_Images

Z = np.log(predictReal_Images + 1.)
del predictReal_Images

W = np.log(predict_Images + 1.)
del predict_Images

scale=4.
X = X/scale
Y = Y/scale
Z = Z/scale
W = W/scale

img_shape = (32,32)
model = Sequential()
```

```
model.add(Flatten(input_shape=img_shape))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(32*32, activation = 'relu')) #(N, 1024)
model.add(Reshape((32,32))) #(N, 32,32)

model.compile(optimizer='adam', loss='mse', metrics = ['
    accuracy'])
P = model.predict(Z)
del Z; del X; del Y
P0 = np.exp(P) - 1. ; del P
W0 = np.exp(W) - 1. ; del W

P_bad = np.array([SubImage(image, pix0, pixlenx, pixleny) for
     image in P0])
W_bad = np.array([SubImage(image, pix0, pixlenx, pixleny) for
     image in W0])
Z_bad = np.array([SubImage(image, pix0, pixlenx, pixleny) for
     image in Z0])

pTs_Pall = ImagesToPtData(P_bad, cutoff = 0.0001)
pTs_P = ImagesToPtData(P_bad, cutoff = 0.49999)
pTs_W = ImagesToPtData(W_bad, cutoff = 0.49999)
pTs_Z = ImagesToPtData(W_bad, cutoff = 0.49999)

nbins_pT = 20
histW = plt.hist(pTs_W, range = (0.5, maxmax), bins =
    nbins_pT, ec='black', log = True)
histP = plt.hist(pTs_P, range = (0.5, np.max(pTs_P)), bins =
    nbins_pT, ec='black', log = True)

(histP_x, histP_y) = Hist_to_xy(histP)
```

```
(histP_x, histP_y) = FilterZero(histP_x, histP_y)
(histW_x, histW_y) = Hist_to_xy(histW)
(histW_x, histW_y) = FilterZero(histW_x, histW_y)
(histP_y, histW_y) = np.log(histP_y), np.log(histW_y)
(histP_y, histW_y) = histP_y / histP_y[0], histW_y / histW_y
    [0] #normiranje

paramsP, paramsP_cov = curve_fit(lambda x,a,b: return a*x + b
    , histP_x, histP_y)
paramsW, paramsW_cov = curve_fit(lambda x,a,b: return a*x + b
    , histW_x, histW_y)
```

# Bibliography


[1] Abelev B. *et al* and The ALICE Collaboration 2014 J. Phys. G: Nucl. Part. Phys. 41 087001

[2] Kvapil J. : Heavy-flavour jet production and correlations with ALICE // Nuclear Physics A, vol. 1005, 2021, 121921

[3] Maggiore, M. : A modern introduction to quantum field theory, Oxford University Press, 2005

[4] Standard model of elementary particles, (16.12.2019), Physics Central, `https://www.physicscentral.com/explore/action/particle-model.cfm`

[5] Lessons Explainer: Quarks, Nagwa, `https://www.nagwa.com/en/explainers/912143010859/`

[6] Thomson, M. : Modern Particle Physics : 1st ed. Cambridge University Press, 2013

[7] Gómez, N. A. : Aspects on Effective Theories and the QCD Transition // *Symmetry* **2020**, 12, 945. `https://doi.org/10.3390/sym12060945`

[8] ALICE Collaboration : The ALICE experiment at the CERN LHC // 2008 JINST 3 S08002

[9] ALICE Collaboration : Technical Design Report of the High Momentum Particle Identification Detector // CERN / LHCC 98–19 ALICE TDR 1, 14.8 1998

[10] ALICE Collaboration : Technical Design Report for the Upgrade of the ALICE Inner Tracking System // J. Phys. G 41 (2014) 087002

[11] Contin G. : Performance of the present ALICE Inner Tracking System and studies for the upgrade // JINST 7 C06007

[12] ALICE Collaboration : ALICE Time-Of-Flight System: Techical Design Report. //Technical Report CERN-LHCC-2000-012, CERN, 2001

[13] TOF measured particle beta vs. signed momentum in pp collisions, 28.5.2016. `http://aliceinfo.cern.ch/Figure/node/919/`

[14] Grigoriev, V *et al.* : ALICE T0 detector // IEEE Transactions on Nuclear Science // IEEE TRANS NUCL SCI. 52. 605- 608 Vol. 1. 10.1109/NSSMIC.2004.1462267.

[15] Silicon strip detector calibration, 2021, JINR. http://er.jinr.ru/si_detector_calibration.html

[16] Knoll G.F. : Radiation Detection and Measurement, 4th ed. ISBN: 978-0-470-13148-0, str. 365-378.

[17] Anghinolfi F. : Silicon strip detectors and their readout electronics // 1.12.2009, CERN/PH/ESE Seminar

[18] Meroli S. : The Straggling function. Energy Loss Distribution of charged particles in silicon layers // Home Cern, https://meroli.web.cern.ch/lecture_StragglingFunction.html

[19] Hatsuda T. *et al.* : Quark-Gluon Plasma // Cambridge University Press, 2005.

[20] Poppenberg H. : Charged Jet Properties Measured with the ALICE Experiment, Master Thesis, Institut für Kernphysik Westfälische Wilhelms-Universität Münster, 2015

[21] Ellis S. D., Soper D. E. : Successive Combination Jet Algorithm For Hadron Collisions // Phys.Rev.D 48:3160-3166, 1993

[22] Cacciari M., Salam G. P., Soyez G. : The anti-kt jet clustering algorithm // JHEP 0804:063, 2008

[23] Cacciari M., Talk at 25th Indian-Summer School of Physics, 2013. http://rafael.ujf.cas.cz/school13/presentations/Cacciari2.pdf

[24] Chollet F. : Deep Learning with Python, ISBN 9781617294433, str. 4,5

[25] Raschka S. : Python Machine Learning, Packt Publishing (2015), ISBN 978-1-78355-513-0, str. 19-24, 343 - 350

[26] Moroney L. : AI and Machine Learning for Coders - A Programmer's Guide to Artificial Intelligence, O'Reilley Media 2020, ISBN 978-1-492-07819-7, str. 7-19

[27] Activation Function, Machine Learning, Gradient Paperspace https://docs.paperspace.com/machine-learning/wiki/activation-function