Uniwersytet Warszawski
Wydział Fizyki

Agnieszka Ciepielewska
Nr albumu: 385537

Zastosowanie uczenia maszynowego do identyfikacji leptonów tau w eksperymencie CMS

Praca licencjacka
na kierunku Fizyka w ramach Międzywydziałowych Indywidualnych Studiów Matematyczno-Przyrodniczych

Praca wykonana pod kierunkiem
dr hab. Artur Kalinowski
Zakład Cząstek i Oddziaływań Fundamentalnych
Instytut Fizyki Doświadczalnej

Warszawa, <TODO: miesiąc-i-rok-złożenia-pracy>

Oświadczenie kierującego pracą

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

Streszczenie

<Krótkie (maks. 800 znaków) streszczenie pracy, na przykład:

Lorem ipsum – tekst składający się z łacińskich i quasi-łacińskich wyrazów, mający korzenie w klasycznej łacinie, wzorowany na fragmencie traktatu Cycerona „O granicach dobra i zła" (De finibus bonorum et malorum) napisanego w 45 r. p.n.e. Tekst jest stosowany do demonstracji krojów pisma (czcionek, fontów), kompozycji kolumny itp. Po raz pierwszy został użyty przez nieznanego drukarza w XVI w.

Tekst w obcym języku pozwala skoncentrować uwagę na wizualnych aspektach tekstu, a nie jego znaczeniu.

Cytat z `https://pl.wikipedia.org/wiki/Lorem_ipsum` >

Słowa kluczowe

<TODO: wykaz maksymalnie 10 słów swobodnie wybranych>

Dziedzina pracy (kody wg programu Socrates-Erasmus)

13.2 Fizyka

Tytuł pracy w języku angielskim

Application of machine learning for tau leptons identification in the CMS experiment

Spis treści

Cel pracy

Rozdział 1

Wstęp

Tutaj piszemy informacje wprowadzające w tematykę pracy, potrzebne do zrozumienia treści.

1.1. Leptony tau

Leptony tau odgrywają istotną rolę w wielu eksperymentach fizycznych. Jednym z ważniejszych jest badanie bozonu Higgsa w rozpadzie $H \rightarrow \tau\tau$. Jednak z powodu krótkiego czasu życia $\tau = 2.9 \cdot 10^{-13}$ s [5], one również nie są wykrywane bezpośrednio w eksperymentach, ale obserwuje się ich produkty rozpadu. Taony są najcięższymi z leptonów i przy swojej masie $m_\tau = 1.776$ GeV/c^2 [5] jako jedyne są w stanie rozpadać się hadronowo. Z tabeli 1.1 widać, że dzieje się tak w około 2/3 przypadków. Sprawia to problemy przy ich identyfikacji, ponieważ łatwo jest je pomylić z jetami hadronowymi (ang. *QCD jets*).

Tabela 1.1: Główne kanały rozpadu taonów wraz z prawdopodobieństwem [3, 5]. Tutaj h oznacza zarówno mezony π jak i K.

Rozpad	Prawdopodobieństwo [%]
Rozpady leptonowe	
$\tau^- \rightarrow e^- \bar{\nu}_e \nu_\tau$	17.8
$\tau^- \rightarrow \mu^- \bar{\nu}_\mu \nu_\tau$	17.4
Rozpady hadronowe	
$\tau^- \rightarrow h^- \pi^0 \nu_\tau$	25.9
$\tau^- \rightarrow h^- \nu_\tau$	11.5
$\tau^- \rightarrow h^- h^+ h^- \nu_\tau$	9.8
$\tau^- \rightarrow h^- \pi^0 \pi^0 \nu_\tau$	9.5
$\tau^- \rightarrow h^- h^+ h^- \pi^0 \nu_\tau$	4.8
Inne	3.3

1.2. Detektor CMS

Detektor CMS (*Compact Muon Solenoid*) znajduje się w Wielkim Zderzaczu Hadronów (LHC) w CERN-ie. Jego główne elementy to: nadprzewodząca cewka wytwarzająca pole magnetyczne o indukcji 3.8 T, detektory krzemowe, kalorymetr elektromagnetyczny (ECAL), kalorymetr hadronowy (HCAL) i komora jonizacyjna do wykrywania mionów.

Detektor krzemowy, pokrywający przedział pseudopospieszności $|\eta| < 2.5$, składa się z dwóch rodzajów detektorów: pikselowych i paskowych. Korpus centralny składa się z trzech warstw detektorów pikselowych oraz jedenastu warstw detektorów paskowych, natomiast końcówki z jedenastu dysków, z czego dwa zawierają detektory pikselowe, a pozostałe detektory paskowe [1]. Tor lotu hadronów jest rekonstruowany ze skutecznością 80-90% w zależności od pędu poprzecznego i pseudopospieszności η. Detektory krzemowe mają grubość od 0.4 do 2.0 długości drogi radiacyjnej (X_0), także fotony z dużym prawdopodobieństwem konwertują wewnątrz detektora na pary e^+e^- [3].

Kalorymetr ECAL jest zrobiony ze scyntylacyjnych kryształów stolzytu ($PbWO_4$). Posiada ponad 61000 kryształów w korpusie centralnym, pokrywającym $|\eta| < 1.48$ oraz ponad 7300 kryształów na końcach detektora, pokrywających $|\eta| < 3.0$. Stolzyt posiada krótką drogę radiacyjną ($X_0 = 0.89$ cm), krótki promień Molièra (2.2 cm) oraz szybko wyświeca światło (80% światła jest wyświecane w 25 ns). Dzięki temu dobrze rozdziela kaskady, więc jest często wykorzystywany do budowy kalorymetrów [1].

Na zewnątrz ECAL znajduje się kalorymetr hadronowy HCAL zawierający mosiądz i plastik. Mosiądz został wybrany ze względu na krótką drogę swobodną (*interaction length*) hadronów oraz słabe właściwości magnetyczne. Tak samo jak ECAL pokrywa obszar $|\eta| < 3.0$. Jego grubość to od siedmiu do jedenastu dróg swobodnych w zależności od η. Natomiast dla $3.0 < \eta < 5.0$ używany jest stalowo kwarcowy kalorymetr przedni (ang. *Hadron Forward*) [1].

Detektor mionowy składa się z trzech rodzajów komór gazowych. W części centralnej ($|\eta| < 1.2$) używane są komory *drift tube* (DT), w części końcowej ($|\eta| < 2.4$), pole magnetyczne nadal jest duże, ale niejednolite, używa się *cathode strip chambers* (CSC). Dodatkowo we wszystkich miejscach są zastosowane *resistive plate chambers* (RPC). RPC zapewnia dobrą dokładność czasową, natomiast DT i CSC dobrą dokładność pozycyjną [1].

1.3. Sieci neuronowe

Sieci neuronowe to klasa modeli uczenia maszynowego stosowana w uczeniu nadzorowanym [6]. Na podstawie danych treningowych zawierający zmienne objaśniające i objaśniane model uczy się zależności występujących między tymi zmiennymi. Dzieki temu możliwe jest stosowanie modelu do predykcji zmiennych objaśnianych na podstawie nowych obserwacji.

Sieci neuronowe składają się z warstw. Podstawową warstwą używaną w sieciach neuronowych jest warstwa gęsta (*dense layer* lub *fully connected layer*). Warstwa gęsta przyjmuje na wejściu wektor, a na wyjściu zwraca wektor ustalonej wielkości, niekoniecznie tego samego rozmiaru. Jest to warstwa bardzo generyczna, jednak jej minusem jest duża liczba parametrów (sieć długo się trenuje, zajmuje dużo miejsca i może gorzej generalizować [6]). Każda warstwa gęsta składa się z dwóch podstawowych elementów: transformacji liniowej oraz funkcji nieliniowej, zwanej funkcją aktywacji. Każda regresja liniowa posiada wagi w_i oraz przesunięcia b_i (*bias*). Wyjście z warstwy definiowane jest jako:

$$y = f(x^T W + b),$$

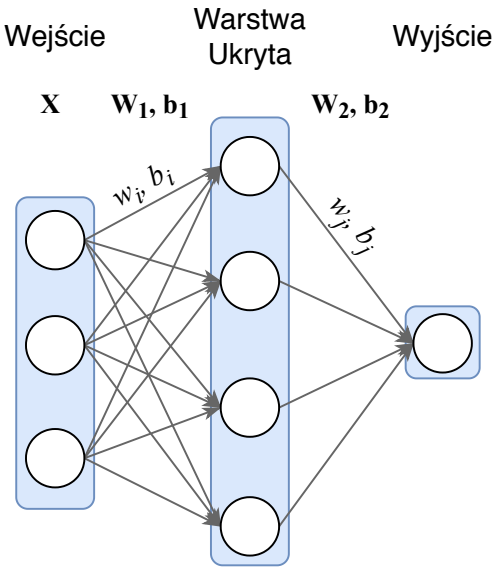
gdzie x to wektor wejściowy, f - funkcja aktywacji, nie zmieniająca wymiaru wektora, W - macierz wag warstwy, b - wektor przesunięcia. Dodanie kolejnych warstw to złożenie kolejnych takich funkcji.

Przykładowo na rysunku 1.1 widoczne jest wejście do sieci x, które jest wektorem 3-elementowym. Następnie jest on transponowany i przemnażany przez macierz wag W_1, która jest wymiaru 3×4 oraz dodane jest przesunięcie b_1 o wymiarze 1×4. Na tym etapie nakłada się również funkcję aktywacji f_1. Kółka w warstwie ukrytej reprezentują wyjścia transformacji

liniowych już po zastosowaniu funkcji aktywacji. Kolejny krok jest analogiczny, tym razem jest to jedna regresja liniowa, czyli macierz wag W_2 ma wymiar 4×1, a przesunięcie b_2 1×1. Zatem wyjście y:

$$y = f_2(f_1(x^T W_1 + b_1)W_2 + b_2),$$

gdzie f_1, f_2 to funkcje aktywacji. Zauważmy, że nieliniowe funkcje aktywacji są konieczne, gdyż inaczej wielowarstwową sieć neuronową zawsze dałoby się sprowadzić do przypadku zwykłej transformacji liniowej odpowiednio przemnażając wagi [6].



Rysunek 1.1: Przykładowa sieć neuronowa z jedną ukrytą warstwą gęstą

Następnie aby otrzymać optymalny model poszukiwane jest minimum pewnej funkcji straty L (czyli miejsce, gdzie model najmniej się myli). Przez funkcję straty rozumiana jest funkcja przybliżająca błąd popełniany przez model. Minimum jest znajdowane przy pomocy gradienty funkcji L względem parametrów modelu. Niestety nie da się wyznaczyć gradientu L w każdym miejscu, ale można go znaleźć w punkcie, gdzie dokonywana jest predykcja. Także, aby poprawić wagi oraz przesunięcie w modelu używa się iteracyjnego algorytmu *Gradient Descent*, który polega na obliczeniu gradientu funkcji straty po parametrach modelu $\nabla_{w_i} L$, a następnie aktualizowane są wagi o pewien krok w kierunku ujemnego gradientu, także dla przykładowej wagi w przy $n+1$-szej iteracji (górny indeks oznacza numer iteracji):

$$w^{(n+1)} \leftarrow w^{(n)} - \gamma \frac{\partial L^{(n)}}{\partial w^{(n)}},$$

gdzie γ to tzw. stała uczenia (*learning rate*) [6]. Widać, że istotne jest aby wszystkie funkcje użyte w modelu były różniczkowalne prawie wszędzie (w punktach nieciągłości pochodnej można przyjąć dowolną wartość brzegową).

W praktyce sieci neuronowe uczone są algorytmem *Stochastic Gradient Descent* (SGD). Polega on na liczeniu funkcji straty dla pewnej liczby obserwacji (zwaną później *batchem*), zamiast dla wszystkich obserwacji. Jednorazowe przejście przez wszystkie obserwacje nazywane jest epoką, a zazwyczaj trening modelu składa się z wielu epok.

Algorytm SGD często okazuje się mało wydajny, aby otrzymać dobre rezultaty. Dlatego często stosuje się bardziej skomplikowany algorytm *Adam* (*Adaptive momentum*) [8].

1.3.1. Funkcje aktywacji

Stosowane później funkcje aktywacji wraz ze wzorami:

- Sigmoid:
$$S(x) = \frac{1}{1 + e^{-x}},$$
- ReLU[9]
$$ReLU(x) = \max(0, x).$$

1.3.2. Warstwa *batch normalization*

Wraz z postępem w treningu sieci wyjścia z warstw sieci mogą się zmieniać. W szczególności rozkład wyjścia może się zmieniać, do czego każda następna warstwa sieci musi się na nowo dostosowywać. Przeciwdziała temu warstwa *batch normalization*, która przeskalowuje każdy batch tak aby miał tą samą średnią i odchylenie standardowe [7]. Dzięki zastosowaniu *batch normalization* trening sieci trwa krócej, a sieć może osiągnąć lepszą skuteczność.

1.3.3. Harmonogram stałej uczenia

Poza dopasowywaniem wielkości kroku aktualizacji wag zastosowanym w algorytmie *Adam*, stosuje się harmonogram stałej uczenia. Polega on na zmniejszaniu stałej uczenia, gdy występuje wypłaszczenie krzywej uczenia (wartości funkcji straty w kolejnych iteracjach). Jeśli funkcja straty nie zmniejsza się przez określoną liczbę epok (przejść przez cały zbiór treningowy), stała uczenia zmniejszana jest o pewien czynnik aż osiągnie zadaną wcześniej wartość minimalną.

1.3.4. Miary skuteczności modeli

Jako, że identyfikacja taonów to problem klasyfikacji binarnej, dobrą metryką do mierzenia skuteczności modelu jest ROC AUC (*Area Under Receiver Operating Characteristic Curve*) oraz sama krzywa ROC. Analizowany model przewiduje prawdopodobieństwo, czy dany rozpad zawiera taon. Aby dokonać klasyfikacji musimy ustalić próg odcięcia ponad którym zawsze zwracane jest 1 (wystąpił taon), a poniżej 0 (nie wystąpił). Wówczas można policzyć *true positive rate* (TPR) oraz *false positive rate* (FPR). Krzywa ROC to wykres FPR od TPR dla różnych progów odcięcie.

ROC AUC to pole pod krzywą ROC. W związku z tym przyjmuje ona wartości pomiędzy 0 a 1, gdzie 0.5 to wartość osiągana przez model losowy, zaś 1 przez bezbłędny klasyfikator. ROC AUC nie jest różniczkowalna, więc nie można jej bezpośrednio wykorzystać przy treningu sieci neuronowej jako funkcji straty. Jest jednak jest przydatna przy ewaluacji oraz porównywaniu wytrenowanych modeli.

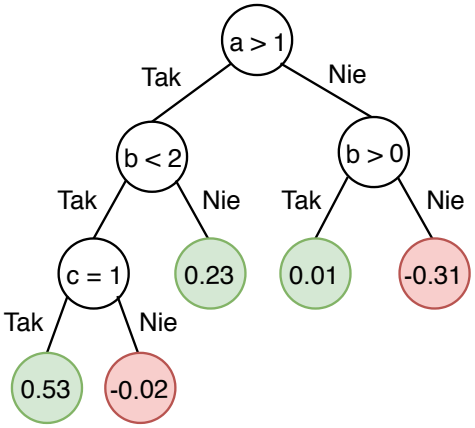
Do treningu jako funkcje straty wykorzystuje się zazwyczaj binarną entropię krzyżową (*binary cross entropy*) [6], daną wzorem:

$$L(y, p) = -(y \log(p) + (1 - y) \log(1 - p)),$$

gdzie y to prawdziwe wartości, a p to predykcje modelu.

1.4. XGBoost

XGBoost (*eXtreme Gradient Boosting*) [2] to implementacja algorytmu *Gradient Boosting*, polegającym na budowaniu kolejnych drzew decyzyjnych (rys. 1.2). Każde kolejne drzewo jest tworzone aby jak najlepiej przewidywać błąd popełniany przez poprzedni zestaw drzew. Końcowa predykcja to suma wszystkich odpowiedzi drzew.



Rysunek 1.2: Przykładowe drzewo decyzyjne, a, b, c to zmienne występujące w danych

Rozdział 2

Dane eksperymentalne oraz symulacyjne

Dane pochodzą z symulacji zderzeń proton-proton w których został wyprodukowany bozon Higgsa i rozpadł się na parę $\tau\tau$. Dane są wstępnie przetwarzane tak, aby pozostawić tylko najbardziej potrzebne informacje. Ostatecznie, otrzymane dane mają 19 zmiennych objaśniających [4]:

- **byCombinedIsolationDeltaBetaCorrRaw3Hits**
- **chargedIsoPtSum** - suma pędów naładowanych produktów rozpadu wewnątrz stożka sygnałowego.
- **decayDistMag** - odległość między zderzeniem *pp* a rozpadem taonu.
- **decayMode** - kanał rozpadu taonu.
- **dxy** - odległość między zderzeniem *pp* a rozpadem taonu w rzucie na płaszczyznę prostopadłą do osi wiązki.
- **dxy_Sig** - zmienna **dxy** podzielona przez niepeność pomiaru.
- **eRatio** - stosunek energii taonu do energii produktów rozpadu.
- **flightLengthSig** - zmienna **decayDistMag** podzielona przez niepewność pomiaru.
- **gjAngleDiff** - kąt między kierunkiem ruchu taonu oraz kierunkiem ruchu głównego produktu rozpadu.
- **hasSecondaryVertex** - informacja czy rozpad posiada wierzchołek wtórny, którym może być rozpad taonu.
- **ip3d**
- **nPhoton** - liczba protonów przypisana do rozpadu taonu.
- **neutralIsoPtSum** - suma pędów neutralnych produktów rozpadu wewnątrz stożka sygnałowego.
- **photonPtSumOutsideSignalCone** - suma pędów fotonów poza stożkiem sygnałowym.

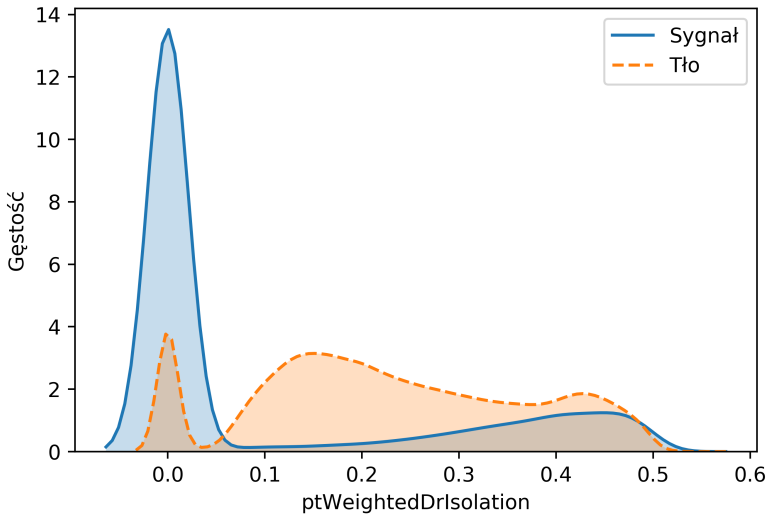
- **ptWeightedDetaStrip**
- **ptWeightedDphiStrip**
- **ptWeightedDrIsolation**
- **ptWeightedDrSignal**
- **puCorrPtSum** - suma pędów cząstek neutralnych znajdujących się wewnątrz stożka sygnałowego, ale pochodząca z innych rozpadów.

Dodatkowo dane zawierają 4 zmienne będące odpowiedziami wcześniej wykorzystywanych klasyfikatorów:

- **DPFTau_2016_v1tauVSall**
- **byIsolationMVArun2v1DBnewDMwLTraw2017v2**
- **deepTau2017v1tauVSall**
- **deepTau2017v1tauVSjet**

Jednakże, dane eksperymentalne nie dostarczają informacji o wystąpieniu taonu. Dlatego do treningu modeli użyte zostały dane symulacyjne wytworzone metodami Monte Carlo. Symulowane są rozpady H $\rightarrow \tau\tau$, a za tło uznane są jety QCD. Dane te zawierają takie same zmienne objaśniające jak dane eksperymentalne, jednak występuje tam również zmienna objaśniana, czyli informacja binarna o wystąpieniu taonu. Przyjmuje on wartość 1 jeśli taon wystąpił (sygnał) oraz 0 jeśli nie wystąpił (tło).

Na rysunku 2.1 przedstawiono rozkład przykładowej zmiennej dla sygnału oraz tła. Widać zauważalne różnice w rozkładach, także można na podstawie tej zmiennej dokonywać klasyfikacji. Rozkłady częściowo się nakładają, więc klasyfikacja nie byłaby idealna.



Rysunek 2.1: Różnice w rozkładzie przykładowej zmiennej dla sygnału i tła

Rozdział 3

Architektura modeli

Aby dokonywać klasyfikacji sygnału zastosowano różne modele predykcyjne opierające się na metodach opisanych w rozdziale 1.

3.1. Model oparty o same klasyfikatory

Prosty model, składający się z jednego neuronu, który przyjmuje jako zmienne objaśniające odpowiedzi wcześniej używanych klasyfikatorów (patrz rozdział 2). Jest to więc średnia ważona odpowiedzi klasyfikatorów, na którą nałożona jest funkcja sigmoid. Wielkość batcha została ustalona na 128, do optymalizacji został użyty algorytm *Adam* ze stałą uczenia $\gamma = 10^{-3}$. Model uczony był przez cztery epoki. Później używana jest nazwa *ensemble*.

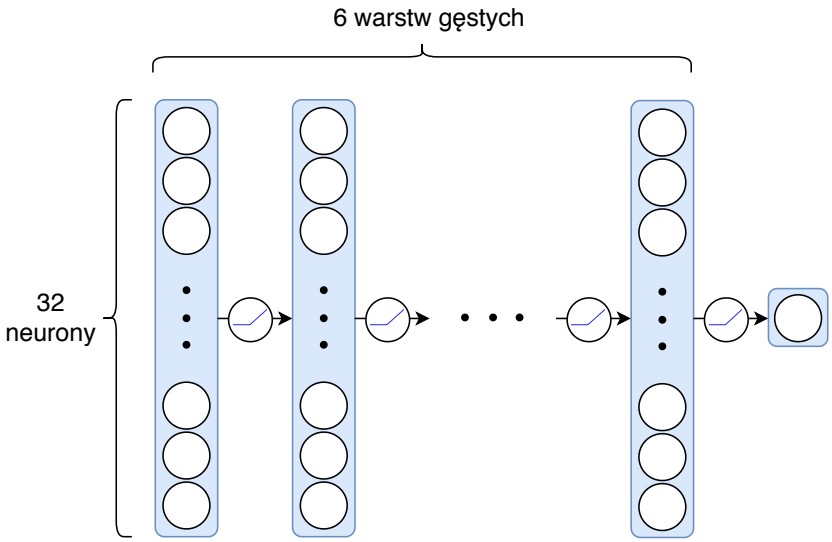
3.2. Sieć neuronowa

Kolejny model był oparty o wszystkie zmienne objaśniające, jego architekturę przedstawiono na rysunku 3.1. Składał się z sześciu warstw gęstych, o wielkości 32 neuronów każda. Jako funkcje aktywacji zastosowano funkcje ReLU. Warstwę wyjściową tworzy jeden neuron z sigmoidem jako funkcją aktywacji. Dzięki temu wyjściem sieci jest pojedyncza wartość z zakresu [0, 1]. Inne parametry zostały ustalone tak samo jak dla modelu w sekcji 3.1. Ten model zwany jest później jako *baseline*.

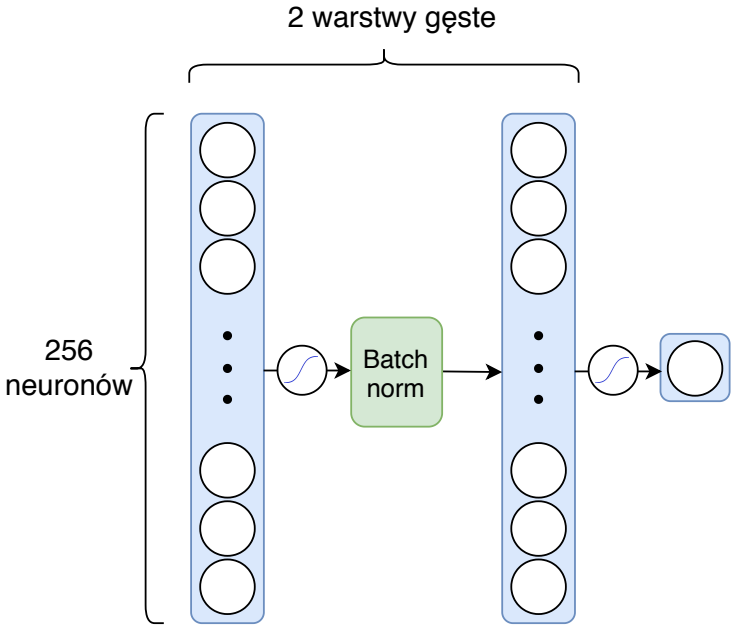
3.3. Poprawiona sieć neuronowa

Na podstawie optymalizacji hiperparametrów przy pomocy losowego przeszukiwania został wybrany najlepszy model oparty o gęste sieci neuronowe. Optymalizowano parametry takie jak: wielkość batcha, stała uczenia, wielkość i liczba warstw ukrytych, funkcja aktywacji, liczba epok, występowanie warstwy *batch normalization* i występowanie harmonogramu stałej uczenia. Otrzymana architektura znajduje się na rysunku 3.2.

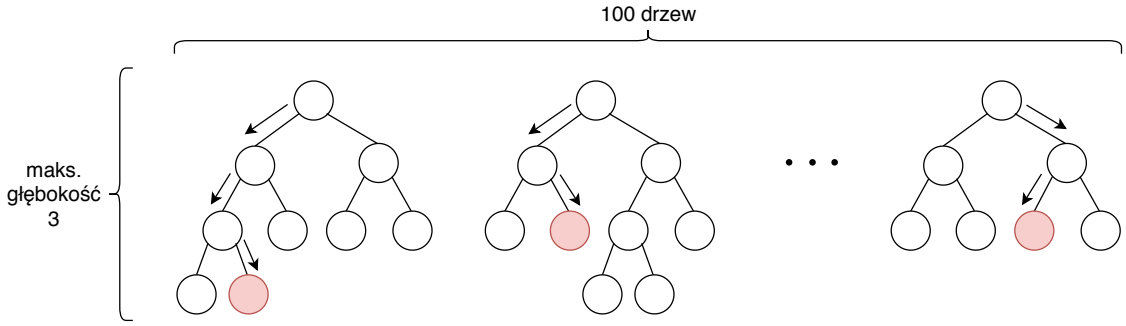
Wybrany model ma dwie warstwy gęste po 256 neuronów każda, zatem widać że model jest płytszy i szerszy niż opisany w sekcji 3.2. Jako funkcja aktywacji jest tutaj użyty sigmoid, a dodatkowo zastosowana jest warstwa *batch normalization*. Użyta wielkość batcha to 256, do optymalizacji zastosowano algorytm *Adam* ze stałą uczenia $\gamma = 5 \cdot 10^{-4}$ oraz harmonogram zmniejszający stałą uczenia γ na wypłaszczeniu funkcji straty przez czynnik 0.2 do minimalnej wartości $\gamma = 10^{-5}$. Model był uczony przez sześć epok. Ten model zwany jest później jako *best-nn*.



Rysunek 3.1: Bazowa sieć neuronowa (*baseline*). Składa się z 6 ukrytych warstw gęstych po 32 neurony. Jako funkcji aktywacji użyto ReLU.



Rysunek 3.2: Zoptymalizowana sieć neuronowa (*best-nn*). Składa się z dwóch ukrytych warstw gęstych oraz warstwy *batch normalization*. Jako funkcji aktywacji użyto sigmoida.



Rysunek 3.3: Architektura modelu typu XGBoost opartego na drzewach decyzyjnych. Tutaj użyto modelu zawierającego 100 drzew o maksymalnej głębokości 3 (nie licząc korzenia).

3.4. XGBoost

Model oparty na XGBoost składał się ze 100 drzew decyzyjnych o maksymalnej głębokości 3. Wykorzystano stałą uczenia $\gamma = 10^{-1}$, funkcją kosztu była binarna entropia krzyżowa (tak samo jak w przypadku sieci neuronowych). Schematyczny rysunek architektury znajduje się na rysunku 3.3.

Rozdział 4

Wyniki

Każdy model trenowany był w dwóch trybach. Pierwszy z nich korzystał ze wszystkich zmiennych objaśniających, drugi zaś pomijał odpowiedzi klasyfikatorów. Dzięki temu można porównać wyniki z wcześniej używanymi modelami. Po dokonaniu analizy istotności zmiennych zauważono, że zmienna byCombinedIsolationDeltaBetaCorrRaw3Hits (patrz rozdział 2) jest najbardziej istotna dla wszystkich modeli. Jako, że rozkład tej zmiennej różni się na danych symulacyjnych oraz eksperymentalnych, postanowiono wytrenować również modele bez tej zmiennej.

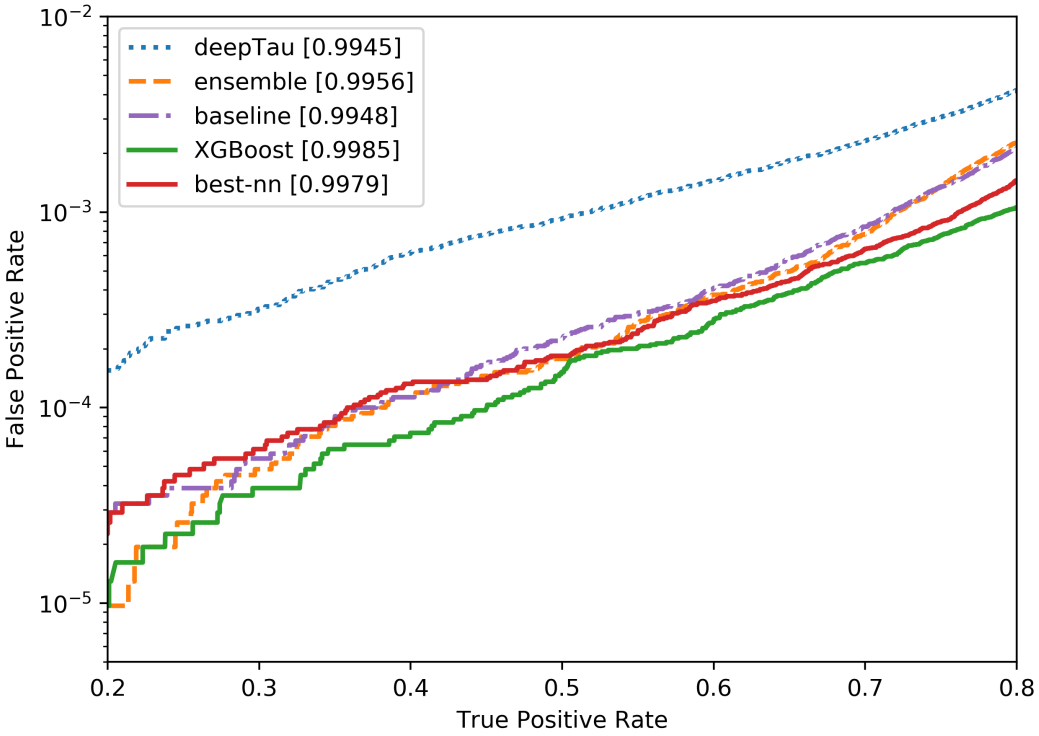
W tabeli 4.1 znajdują się wyniki ROC AUC dla wszystkich modeli. Dla porównania zamieszczono wynik najlepszego z wcześniejszych klasyfikatorów, którym okazał się deepTau2017v1tauVSjet (później nazywany *deepTau*).

Można zauważyć, że model oparty o XGBoost osiąga najwyższą skuteczność spośród zaproponowanych oraz wcześniej używanych modeli. W szczególności przy treningu na danych z wyłączeniem klasyfikatorów zarówno *best-nn* jak i *XGBoost* osiągnęły lepsze wyniki niż *deepTau*. Widać również, że wyrzucenie głównej zmiennej nie pogarsza znacząco skuteczności modeli, a może skutkować lepszymi wynikami na danych eksperymentalnych. Ponadto, model oparty o same klasyfikatory osiąga lepszy wynik niż każdy klasyfikator osobno.

Tabela 4.1: Wartość ROC AUC dla wszystkich modeli na zbiorze testowym, pogrubioną czcionką zaznaczono najlepszy wynik.

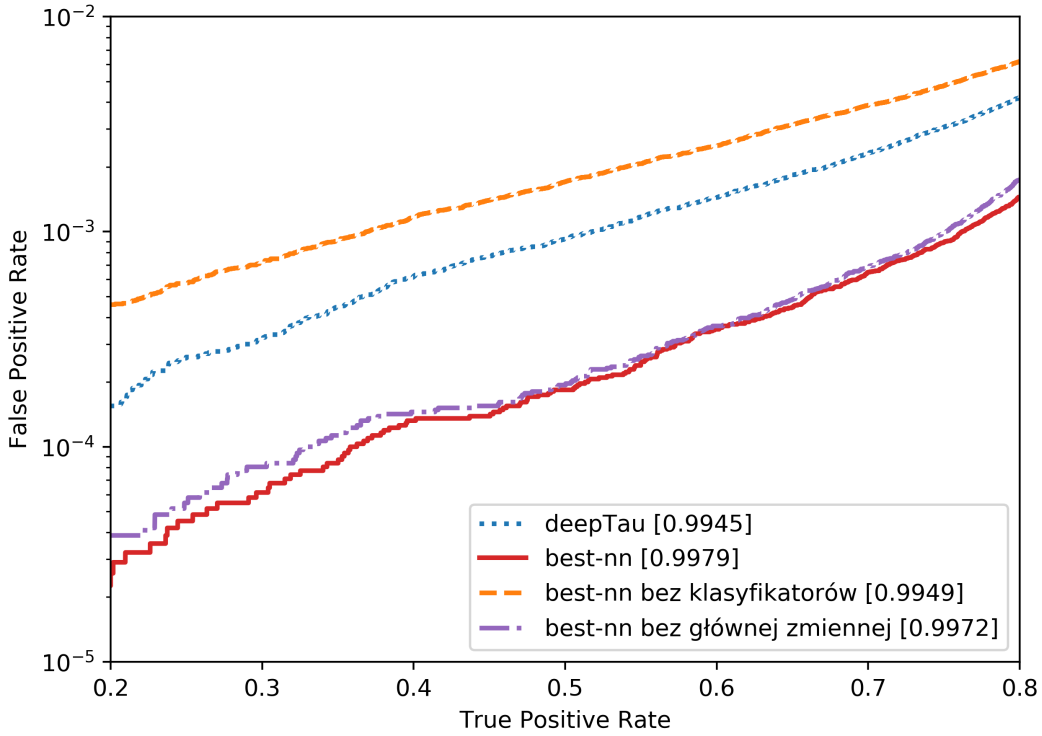
Tryb	*baseline*	*best-nn*	*XGBoost*	*deepTau*	*ensemble*
Pełne dane	.9948	.9979	**.9985**	–	–
Bez klas.	.9940	.9949	**.9956**	.9945	–
Bez najlepszej	.9977	.9972	**.9985**	–	–
Same klas.	–	–	–	–	.9956

Na rysunkach 4.1, 4.2 i 4.3 przedstawiono fragmenty krzywych ROC dla wybranych modeli. Na wykresach osie są zamienione, więc lepszy model ma mniejsze pole pod krzywą. Zakres TPR na wykresach został zmniejszony, aby lepiej uwidocznić różnice między modelami.



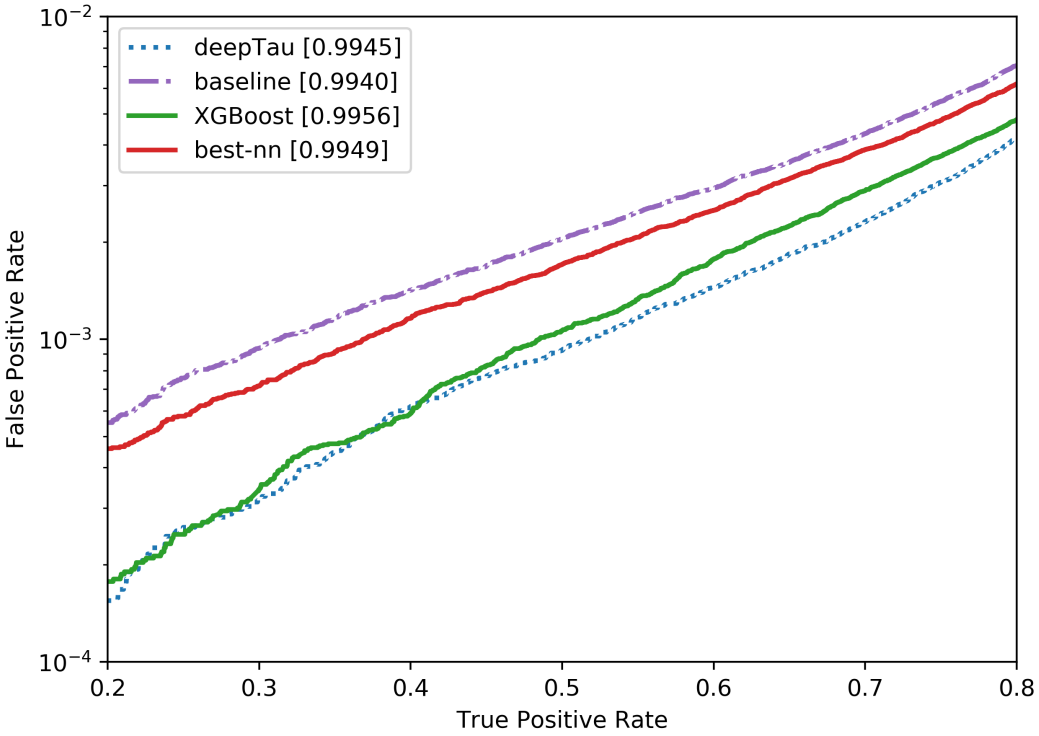
Rysunek 4.1: Porównanie najlepszych modeli na podstawie krzywej ROC. *baseline*, *best-nn* oraz *XGBoost* były uczone na pełnych danych, a *deepTau* na danych bez klasyfikatorów. W nawiasach kwadratowych podano wartość ROC AUC dla danego modelu.

Na rysunku 4.1 przedstawiono najlepsze modele każdego typu. Można zauważyć, że największa różnica jest między *deepTau* a innymi modelami, natomiast najskuteczniejszy jest *XGBoost*. Na tym fragmencie krzywej różnice między wytrenowanymi modelami są dosyć niewielkie.



Rysunek 4.2: Porównanie wszystkich trybów uczenia nowej sieci z najlepszym klasyfikatorem na podstawie krzywej ROC. W nawiasach kwadratowych podano wartość ROC AUC dla danego modelu.

Na rysunku 4.2 znajdują się wszystkie tryby treningu nowej sieci w porównaniu z *deepTau*. Wyniki dla treningu na pełnych danych i z wyłączeniem głównej zmiennej nie różnią się znacząco, natomiast trening bez klasyfikatorów widocznie pogarsza wygląd krzywej ROC. Na tym fragmencie osiąga gorsze wyniki niż *deepTau*. Jednak patrząc na wartość ROC AUC można podejrzewać, że osiąga lepsze rezultaty na skrajnych fragmentach.



Rysunek 4.3: Porównanie modeli uczonych na danych bez klasyfikatorów na podstawie krzywej ROC. W nawiasach kwadratowych podano wartość ROC AUC dla danego modelu.

Na rysunku 4.3 widnieje porównanie każdego typu modelu uczonego na danych bez klasyfikatorów. Można zauważyć, że na wyróżnionym fragmencie najlepsze wyniki osiąga *deepTau*, jednak są one zbliżone do wyników *XGBoosta*. Patrząc na wartości ROC AUC możemy przypuszczać, że zarówno *XGBoost* jak i *best-nn* uzyskują lepsze rezultaty niż *deepTau* na skrajnych zakresach TPR.

Rozdział 5

Dyskusja

Porównując dane symulacyjne i eksperymentalne można zauważyć również różnice w rozkładach zmiennych objaśniających, także nie ma pewności, że model dający dobre wyniki na danych symulacyjnych będzie dawał równie dobre wyniki na danych eksperymentalnych. Jest to jednak jedyne dostępne źródło danych.

Rozdział 6

Podsumowanie

Bibliography

[1] GL Bayatian et al. *CMS Physics: Technical Design Report Volume 1: Detector Performance and Software*. Tech. rep. CMS-TDR-008-1, 2006.

[2] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM. 2016, pp. 785–794.

[3] CMS Collaboration. "Performance of reconstruction and identification of τ leptons decaying to hadrons and ν_τ in pp collisions at $\sqrt{s} = 13$ TeV". In: (2018). arXiv: 1809.02816 [hep-ex].

[4] CMS collaboration et al. "Reconstruction and identification of τ lepton decays to hadrons and $\nu\tau$ at CMS". In: *Journal of Instrumentation* 11.01 (2016), P01019.

[5] Particle Data Group Collaboration. "Review of particle physics". In: (2010). DOI: 10.1088/0954-3899/37/7A/075021.

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

[7] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: (2015). arXiv: 1502.03167.

[8] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980.

[9] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: In *Proceedings of ICML'10*. 2010, pp. 807–814.