Načrtovanje podatkovnega cevovoda za strojno učenje na podatkih senzorjev

Val Vec[1], Anton Umek[1], Sašo Tomažič[1], Andrej Kos[1], Urban Sedlar[1]

[1]Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška cesta 25, 1000 Ljubljana, Slovenija
E-pošta: val.vec@fe.uni-lj.si

Designing a data pipeline for machine learning on sensor data

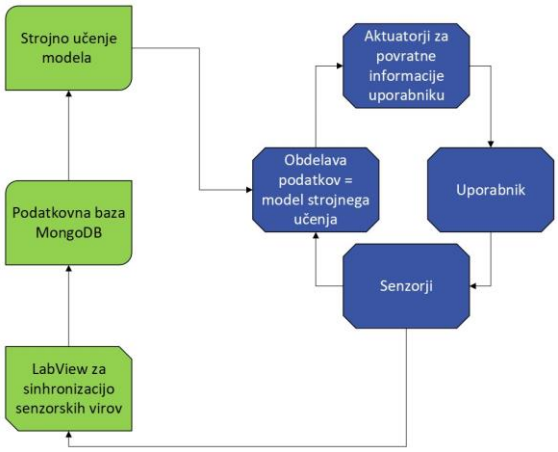
Abstract. *Biomechanical feedback loop in sports is a concept where athletes train with the help of technology. Sensors are used to capture their movements, data from sensors is then processed and feedback that can be useful for faster learning is provided back to the athlete. As the processing pard of the bio-feedback loop is being replaced with machine learning algorithms, data solutions for capturing movement data needed to train ML models is needed. This paper describes a data pipeline for collecting signals from sensors, storing them in a database and then training machine learning models in a way, where training can be done on a regular PC regardless of the size of collected dataset. We found, that when collecting data from different sensors synchronized using LabView, writing directly into a database is too slow; therefore, a more optimized data format - TDMS was selected due to high-speed streaming support and small footprint. Furthermore, LabView program was designed to run in a regular while loop with a timer as it proved to be faster than a timed loop. For the choice of database, MongoDB fits the structure of measurement data the best. We have written Python scripts for both transferring data from local measurements in TDMS format to the database and for machine learning with fetching only current batch of data from the database.*

1 Uvod

Biološka povratna zanka v športu je koncept sestavljen iz 4 delov: subjekta (uporabnika oziroma športnika), senzorjev, ki med športom merijo njegovo gibanje, procesne enote, ki senzorske signale obdela, in aktuatorjev, ki v različnih modalnostih podajo povratno informacijo športniku. Tako izdelan sistem omogoča učinkovitejše učenje športniku [1]. Trenutno večina raziskovalcev za procesiranje uporablja algoritme, ki ne vključujejo strojnega učenja [2]. V okviru širše raziskave, katere del je ta prispevek, želimo raziskati možnosti uporabe strojnega učenja za podajanje povratne informacije športniku. V pregledu področja z naslovom »Trends in Real-time Artificial Intelligence Applications In Sports: A Systematic Review«, ki trenutno čaka na recenzijo, smo ugotovili, da se področje uporabe strojnega učenja v športu hitro razvija – čedalje več člankov ugotavlja, da izstopajo športi s preprostimi gibi ter da večina člankov še ne implementira povratne informacije športniku in tako ne zaključi biološke povratne zanke, tako da je na tem področju še veliko neraziskanega.

Raziskovali bomo, kaj lahko s strojnim učenjem izluščimo iz podatkov senzorjev in kako to informacijo podati športniku, da mu bo koristna pri učenju. Narediti torej želimo celotno biološko povratno zanko, pri čemer bo procesiranje potekalo na osnovi strojnega učenja. Ta članek opisuje enega izmed začetnih korakov in sicer razvoj sistema, ki nam bo omogočal zbiranje velikih količin podatkov senzorjev in učenje modelov strojnega učenja na teh podatkih. Ker predpostavljamo, da bo za najboljše modele strojnega učenja potrebna velika količina podatkov, želimo razviti skalabilen cevovod od zbiranja podatkov, do učenja modela. V ta namen bomo razvili pol-avtomatiziran cevovod z vsemi koraki od zbiranja podatkov, do njihove pred-obdelave in končno, treniranja modela. Prednosti cevovoda, ki nam največ pomenijo so modularnost in skalabilnost [3], saj želimo veliko različnih metod preizkusiti na veliki količini podatkov. Za rešitev podatkovnega cevovoda zahtevamo, da zmore shranjevati senzorske podatke s vzorčno frekvenco najmanj 200Hz, da omogoča shranjevanje različno dolgih gibov z čim manj redundance in da omogoča pridobivanje relevantnih senzorskih kanalov in metapodatkov iz baze z čim manjšim overheadom.

Na sliki 1 prikazujemo planirano arhitekturo modela biološke povratne zanke z modro barvo. Vsebuje prej opisane elemente, kjer je procesiranje zamenjano z uporabo modela strojnega učenja. Tridelni sistem, opisan v tem članku, je predstavljen z zeleno barvo in služi

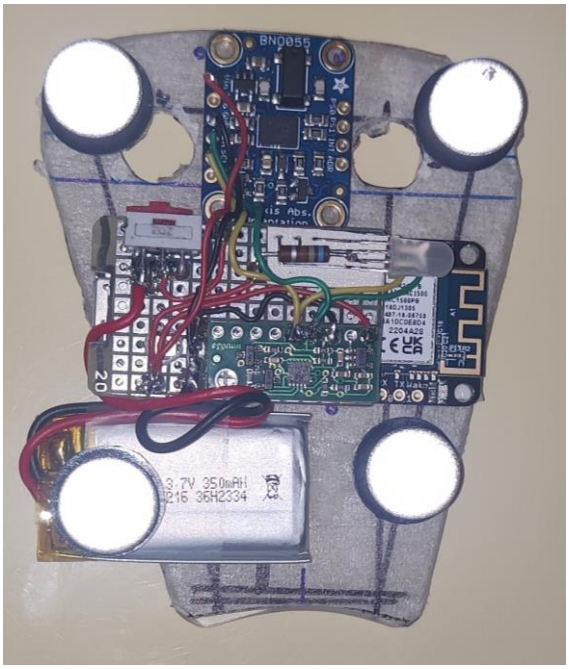


Slika 1.Diagram biološke povratne vezave z modro barvo in elementov našega sistema, ki nam omogočajo učenje modela znotraj zanke biološke povratne vezave

učenju modela znotraj biološke povratne vezave. Vsak izmed treh elementov je podrobno opisan v posameznem poglavju tega članka. Končni izdelek širše raziskave pa bo ta zanka s povratno vezavo. Trenutni sistem služi kot orodje, s katerim bomo pridobili modele strojnega učenja za uporabo v zanki.

2 Meritve

Pri merjenju gibov hkrati uporabljamo več senzorjev. V poskusu uporabljenem za predstavitev sistema uporabljamo nosljivo napravo z žiroskopom in pospeškometrom iz senzorja BNO085 povezanega na mikrokrmilnik Adafruit Feather M0 ter meritve



Slika 2. Senzorska naprava je sestavljena iz arduina, senzorjev ter markerjev, ki jih prepozna Qualysis

pridobljene iz sistema za snemanje gibanja (angl. motion capture) Qualisys. Sestavljena naprava je na sliki 2 in poleg omenjenega senzorja vsebuje še 6DOF senzor (senzor pospeška in žiroskop, ki podaja neobdelane podatke - ne preračunava orientacije in ne odšteva gravitacije od izmerjenega pospeška) za kasnejšo uporabo, ter 4 pasivne markerje – refleksne kroglice, na podlagi katerih Qualisys zazna pozicijo objekta v prostoru. Merilna naprava bo bolj podrobno opisana v prihodnjem članku in je bila razvita v okviru te raziskave.

V nadaljnjih raziskavah želimo število senzorjev še povečati. Vsi ti senzorji pa morajo biti sinhronizirani med seboj. Za sinhronizacijo uporabimo programsko okolje LabView (verzija 24.1.1f1), v katerem smo razvili aplikacijo za meritve. Aplikacija zbira podatke iz senzorjev, pri čemer so podatki iz nosljivega senzorja poslani preko protokola UDP, za pridobitev podatkov iz Qualisysa pa uporabljamo njihov modul za povezavo z LabViewom, ki deluje preko protokola TCP. Vsi podatki se prenašajo zgolj preko lokalnega omrežja. V okolju LabView podatke sinhroniziramo in jih obdelamo, tako da upoštevamo odstopanje (angl. bias) senzorjev. Nato moramo vsako meritev posebej shraniti. Na koncu želimo imeti vse meritve v eni sami podatkovni bazi. Trenutno je vzorčna frekvenca senzorjev zadosti nizka, da shranjevanje deluje v vsakem primeru. Želimo pa narediti sistem, ki ga bo mogoče v prihodnje dobro skalirati. Zato smo preizkusili več oblik shranjevanja podatkov.

Meritve sinhroniziramo z časovno zanko [4] v LabView, ki v enakomernih korakih zapiše pridobljene podatke. Na operacijskem sistemu Windows je največja frekvenca izvajanja časovne zanke 1kHz. Za več bi potrebovali namensko napravo z operacijskim sistemom v realnem času. Zato se z našo rešitvijo shranjevanja želimo približati tej vzorčni frekvenci. Idealno tudi želimo, da dolžina posnetka ni omejena z spominom v računalniku, zato preizkusimo direkten vpis v bazo in direktno pisanje na disk v primerjavi z začasnim shranjevanjem v delovni pomnilnik in pisanjem po končani meritvi. V dokumentaciji za LabView [5] trdijo, da je najhitrejši format za zapis meritev TDMS. Ta format omogoča tudi, da določimo, kdaj podatke iz pomnilnika zapišemo na disk. Za testiranje shranjevanja smo ustvarili časovno zanko z frekvenco 1 kHz, kjer shranjujemo zgolj trenutni časovni žig in eno število. Poleg časovne zanke smo preizkusili še delovanje navadne *while* zanke z časovnikom, ki počaka do naslednje iteracije. Test smo izvedli na računalniku s procesorjem Intel i7-1355U in 16 GB delovnega pomnilnika. Primerjali smo naslednje 3 možnosti:

1. Konstantno vpisovanje v SQL bazo
2. Zapis v TDMS format, kjer vse podatke nemudoma pišemo na disk
3. Zapis v TDMS format, kjer podatke vpisujemo v delovni pomnilnik in shranimo na disk šele ob koncu meritve.

Baza SQL je izbrana za ta test, saj za razliko od MongoDB (in večine ostalih baz), LabView podpira vnose direktno v SQL bazo preko uradne knjižnice.

Časov za shranjevanje v podatkovno bazo MongoDB tako kot je opisano v naši rešitvi ni v tabeli, ker se shranjevanje izvede v 2 korakih; prvi korak v realnem času shranjuje podatke v pomnilnik, drugi korak pa podatke v paketu shrani v podatkovno bazo MongoDB. Shranjevanje v MongoDB je mnogo počasnejše od shranjevanja v pomnilnik, vendar pa ne blokira samega procesa meritve in zato v tabeli ni posebej izpostavljeno.

Po opravljenih meritvah izračunamo razlike med shranjenimi časovnimi žigi in dobimo povprečne čase med 2 shranjenima vzorcema.

Tabela 1. Primerjava časov med 2 zapisoma podatkov senzorskih signalov zajetih v okolju LabView v podatkovno shrambo glede na način zapisovanja

Način shranjevanja	Povprečni čas med vzorci [ms]
SQL baza	95.9
TDMS v spomin (časovna zanka)	19.17
TDMS na disk (časovna zanka)	21.7
TDMS v spomin (while zanka)	1.5
TDMS na disk (while zanka)	33.4

Kot vidimo, način, kjer bi meritve ob vsakem vzorcu shranjevali v bazo, ne pride v poštev. Edini način, ki nas pri shranjevanju ne ovira je uporaba LabViewega formata TDMS, kjer vsako meritev najprej shranimo v delovni pomnilnik. Če bi bile posamezne meritve predolge, pa je druga možnost vpis na disk.

Pri tem moramo še paziti, da »timed loop« zanka, ki je namenjena sinhronizaciji, deluje počasneje, kot navadna *while* zanka, pri kateri počakamo na naslednjo milisekundo iz sistemske ure.

Na podlagi teh meritev se odločimo, da bomo posamezne meritve shranjevali v formatu TDMS, iz katerega bomo po meritvi podatke prenesli v izbrano podatkovno bazo.

3 Podatkovna baza

Naši podatki so oblike, ki ni najbolj primerna za klasično relacijsko bazo, čeprav jo v prvotni verziji sicer uporabljamo. Vsaka meritev ima nekaj podatkov, ki nam opišejo parametre meritve. To so na primer oznaka razreda podatka - oznaka do kakšne napake je v gibu prišlo v kolikor je prišlo do napake, katera oseba je izvajala gib, čas merjenja in še več drugih, katere želimo imeti možnost kasneje tudi dopolniti. Poleg tega pa ima vsaka meritev še nedoločeno število senzorskih podatkov. Število je namreč odvisno od časa trajanja giba, ki pa ni vedno enako. V relacijski bazi smo to rešili tako, da smo v eno tabelo zapisovali parametre meritev, v drugo tabelo pa senzorske meritve. Senzorska meritev predstavlja eno točko v času, v podatkovni bazi pa je z tujim ključem povezana z tabelo, kjer so zapisani parametri meritve. Težava tega pristopa je, da moramo ob vsaki meritvi v tabelo senzorskih meritev dodati nekaj tisoč zapisov. Z izbiro drugačne baze želimo to nekoliko optimizirati.

Rešitev, ki je najbolj prilagojena obliki naših meritev je nerelacijska baza MongoDB [6]. Ta vrsta podatkovne baze nam omogoča, da eno meritev vstavimo kot objekt in temu objektu podamo vse parametre naše meritve. Časovne vrste senzorjev podamo kot pod-objekte naši meritvi. Ta način nam omogoča enostavne poizvedbe, kjer iz baze pogledamo zgolj časovne vrste, ki nas pri učenju posameznega modela zanimajo. Arhitektura posameznega objekta meritve je naslednja:

- ID meritve
- Parametri meritve (več polj v objektu). Parametri meritve nam služijo pri obdelavi podatkov, ter kot oznake pri strojnem učenju
- Časovne vrste:
  - Senzor pospeška
    - Meritev ob posameznem času
  - Senzor orientacije
    - Meritev ob posameznem času

Pri postavljanju baze smo naleteli tudi na »Time Series Collections« modul baze MongoDB [7], ki je

Slika 3: Oblika dokumenta v bazi MongoDB je povsem

prilagojen za delo s časovnimi vrstami. Ta modul predvideva, da posamezno meritev vstavljamo takoj ko je izmerjena in optimizira delo strežnika, če vstavljamo podatke na tak način. To kot smo v prvem delu ugotovili ni primerno za našo vzorčno hitrost. Poleg tega je natančnost merjenja časa tam prenizka. Še vedno bi lahko svoj časovni žig vstavljali kot dodaten podatek, vendar prednosti uporabe tega modula izginejo zaradi omenjenih 2 omejitev.

Ker smo meritve vse shranili v TDMS datotekah, smo napisali Python skripto, ki po odpravljenih meritvah vse meritve shranjene v mapi, kamor naša LabView aplikacija shranjuje meritve naloži v bazo. Deluje tako, da jo kopiramo v mapo s shranjenimi meritvami in poženemo. Skripta najprej poišče vse datoteke s končnico ».tdms«, nato podatke preoblikuje v željeno obliko, kakšno smo opisali zgoraj. Podatkov dodamo še zaporedno številko meritve, tako da najprej pogledamo količino meritev v bazi, in nato ročno izračunamo številko naslednje meritve. To nam bo omogočalo lažje poizvedbe med strojnim učenjem, saj pri deljenju podatkovnega seta pridobimo indekse podatkov, ki padejo v učno in testno množico, pri čemer pa nameravamo za vsak batch posebej poizvedovati po bazi, saj planiramo za primer, da bo podatkov preveč, da bi lahko vse shranili lokalno med strojnim učenjem. Podatkovna baza MongoDB pa nima vgrajene funkcije samodejnega dodeljevanja zaporednih indeksov (angl.

autoincrement) oziroma kaj podobnega, kot smo navajeni iz baz SQL.

Ena meritev enega giba tako postane en MongoDB dokument. Pri tem se zavedamo, da so dolžine naših gibov v športu dovolj kratke, da omejitve baze za velikost dokumenta (16 MB) ne bomo v praksi nikoli dosegli. V kolikor bi to postal problem, je rešitev drugačna oblika dokumenta. Če bi ena točka v času predstavljala en dokument, bi dosegli, da dolžina meritve ni omejena. Vendar pa to pomeni, da potrebujemo pri strojnem učenju prevzeti veliko količino dokumentov v vsakem koraku. Časovno vrsto posamezne meritve bomo namreč vedno potrebovali naenkrat.

Tak način shranjevanja nam omogoča, da pri merjenju nismo omejeni s počasnostjo sprotnega vpisovanja v bazo, hkrati pa je baza skalabilna, omogoča zbiranje podatkov velikega števila merjenj in primerne oblike za nadaljnje strojno učenje na podatkih. Oblika vsakega dokumenta v bazi – kot je na sliki 3, se ujema z prej definirano obliko posamezne meritve.

4 Strojno učenje na podatkih iz baze

Želimo napisati skripto, ki nam omogoča, da naučimo model strojnega učenja tudi, kot bomo imeli zares veliko količino podatkov. Takrat bomo omejeni z količino delovnega spomina na delovni postaji, kjer učimo model. Šolski primer kjer uvozimo podatkovni set v Python, nato pred-procesiramo celotni podatkovni set in nato učimo model na celotnem podatkovnem setu, ne deluje, če imamo podatkov več, kot jih lahko shranimo v spomin. Tako našo rešitev ob strojni opremi, ki jo imamo (PC z 16GB rama), nujno potrebujemo za bazo, če zberemo nekaj manj kot 16GB podatkov (del spomina je potreben še za sistem, program in model).

Za sistem želimo da deluje na običajni delovni postaji in da ne zahteva celotnega podatkovnega centra za učenje modela, ker so naša sredstva bolj omejena, kot pri npr. sodobnih generativnih modelih.

Rešitev vidimo v tem, da imamo podatke meritev ves čas shranjene izključno v bazi, ko pa učimo model iz baze prenesemo izključno trenutni paket podatkov. Napisali in preizkusili smo skripto v programskem jeziku Python, ki nam omogoča strojno učenje na neomejeni količini senzorskih podatkov. Skripta je narejena modularno, kar je pomembno za nadaljnje delo, da bomo z njo lahko preizkušali različne metode strojnega učenja in pred-procesiranja podatkov.

Skripta deluje na naslednji način: najprej indekse podatkov razdelimo na učno in testno množico, nato pa v zanki izvajamo učenje. Glavna zanka izvaja strojno učenje po epohah in paketih podatkov. To naredi tako, da v zaporedju pokliče 3 funkcije.

Prva funkcija je napisana tako, da prevzame podatke z zahtevanimi indeksi iz MongoDB podatkovne baze in jih zapiše v obliko za nadaljnjo obdelavo. Ta funkcija je vezana na bazo (ki je definirana z uporabljenimi senzorji) in ni mišljeno, da jo karkoli spreminjamo tekom poskusa.

Druga funkcija izvede pred-procesiranje podatkov. Za naše delo, bomo preizkušali veliko različnih pred-procesiranj, zato želimo imeti shranjene neobdelane podatke in sproti obdelovati samo tiste, ki jih potrebujemo. Napisali smo funkcijo, ki sprejme neobdelane podatke in vrne značilke in oznake za strojno učenje. Ta funkcija je ločena od ostalih, kar nam omogoča lahko menjavo različnih pred-procesiranj.

Model strojnega učenja je prav tako definiran posebej. Za testiranje je uporabljena preprosta nevronska mreža, vendar je mogoče model enostavno zamenjati brez spremembe arhitekture aplikacije.. V kasnejšem delu bomo preizkušali veliko različnih modelov, uspešnost posameznega modela pa ni del tega poročila.

Skripto smo preizkusili in ugotovili, da se model uspešno uči, pri čemer smo določili, da se baza razdeli na 5 delov in v vsakem paketu treniramo na petini zbranih podatkov. Na koncu skripte smo model ovrednotili na vseh testnih podatkih naenkrat, kar ni sporno, glede na to, da je testna množica manjša in lahko celotno naenkrat pridobimo iz baze.

5 Zaključek

Uspešno smo sestavili tridelni sistem, ki nam omogoča shranjevanje in upravljanje z veliko količino meritev, kakršne izvajamo v okviru doktorskega dela. Začeli smo s testiranjem načinov meritev, in ugotovili, da je uporaba LabViewovega formata TDMS za lokalno shranjevanje edina dovolj hitra možnost, da lahko zapisujemo v realnem času. Nato smo izbrali podatkovno bazo, ki ustreza našim zahtevam, glavno vlogo pri izbiri pa je igrala struktura naših podatkov. Nato smo pripravili še skripti za shranjevanje podatkov iz TDMS datotek v podatkovno bazo in za strojno učenje na podatkih iz baze, ki deluje ne glede na količino podatkov v bazi.

Nadaljnje delo zajema najprej sistematično zbiranje podatkov. Načrtovani sistem bomo lahko uporabili pri več različnih športih, prvi poskus pa smo si zamislili na pikadu. Senzor prikazan na sliki 2 bo nalepljen na roko športnika, ki meče pikado. S senzorsko napravo bomo izmerili pospeške v vseh treh dimenzijah, ter orientacijo roke. Hkrati bomo pozicijo in orientacijo roke izmerili tudi z sistemom Qualisys. Za vsakega športnika bomo zbrali časovne vrste vsaj 50 metov, pri čemer bomo za vsaj nekaj športnikov zbrali preko 200 metov. Pri vsakem metu bomo poleg časovnih vrst iz senzorjev zbirali tudi podatek o rezultatih metov, ki nam bodo pri strojnem učenju služili kot oznake razreda.

Tako zbrani podatki, nam bomo omogočali primerjave metod strojnega učenja za klasifikacijo časovnih vrst senzorskih signalov v športu. Poleg samih metod strojnega učenja, bomo pozornost posvetili tudi primerjavi kako kvalitetni morajo biti podatki za uspešno klasifikacijo ter kakšne so razlike med uporabo modelov treniranih na celotni populaciji in posameznemu športniku prilagojenih modelov.

Zahvala

Raziskovalni program ICT4QoL - Information and Communications Technologies for Quality of Life št. programa P2-0246 je sofinancirala Javna agencija za raziskovalno dejavnost Republike Slovenije iz državnega proračuna.

Literatura

[1] A. Kos and A. Umek, *Biomechanical Biofeedback Systems and Applications*. in Human–Computer Interaction Series. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-91349-0.

[2] M. Hribernik, A. Umek, S. Tomazic, and A. Kos, "Review of Real-Time Biomechanical Feedback Systems in Sport and Rehabilitation," *Sensors*, vol. 22, p. 3006, Apr. 2022, doi: 10.3390/s22083006.

[3] "What Is a Machine Learning Pipeline? | IBM." Accessed: Jul. 12, 2024. [Online]. Available: https://www.ibm.com/topics/machine-learning-pipeline

[4] "Timed Loop - NI." Accessed: Jul. 12, 2024. [Online]. Available: https://www.ni.com/docs/en-US/bundle/labview-api-ref/page/structures/timed-loop.html

[5] "NI TDMS File Format - What is a TDMS File?" Accessed: Jul. 12, 2024. [Online]. Available: https://www.ni.com/en/support/documentation/supplemental/06/the-ni-tdms-file-format.html

[6] "MongoDB: The Developer Data Platform," MongoDB. Accessed: Jul. 12, 2024. [Online]. Available: https://www.mongodb.com

[7] "Time Series - MongoDB Manual v7.0." Accessed: Jul. 18, 2024. [Online]. Available: https://www.mongodb.com/docs/manual/core/timeseries-collections/