DOI: https://doi.org/10.53052/9788367652124.02

Marcin BERNAŚ [1], Vasyl MARTSENYUK[1]

MODEL KLASYFIKACJI AKTYWNOŚCI NA BAZIE CZUJNIKÓW DLA SYSTEMÓW VR

Podsumowanie: Artykuł proponuje rozwiązanie zwiększające imersję w grach poprzez zaproponowany model klasyfikacji aktywności wykonywanych za pomocą kończyn dolnych. Zaprezentowany system jest nisko kosztowy i opiera się na wykorzystaniu czujników z urządzeń komórkowych (smartfonów) oraz głębokiej sieci neuronowej, co pozwala na dokładne rozpoznawanie dziesięciu najczęstszych aktywności użytkownika z dokładnością przekraczającą 99%. Wyniki zostały uzyskane na rzeczywistych danych. Wskazano także sposoby minimalizacji opóźnień rozwiązania nawet do 0.2 sekundy. Artykuł wskazuje także kierunki przyszłych badań i rozwoju tej technologii oraz podkreśla znaczenie tego rodzaju systemów w kontekście popularyzacji technologii interaktywnych gier.

Słowa kluczowe: środowisko wirtualnej rzeczywistości, sensory, monitorowanie ruchu, sieci neuronowe.

SENSOR-BASED ACTIVITY CLASSIFICATION MODEL FOR VR SYSTEMS

Summary: The article proposes a solution that increases immersion in games through a proposed model for classifying activities performed with the lower limbs. The presented system is low-cost and based on the use of sensors from cellular devices (smartphones) and a deep neural network, which allows for accurate recognition of the ten most common user activities with an accuracy exceeding 99%. The results were obtained on real data. Methods of minimizing solution delays of up to 0.2 seconds were also indicated. The article also indicates directions for future research and development of this technology and emphasizes the importance of this type of systems in the context of popularizing interactive game technologies.

Keywords: virtual reality environment, sensory, motion tracking, neural networks

1. Wprowadzenie

Wirtualna rzeczywistość (Virtual Reality) to technologia tworzenia trójwymiarowych środowisk wirtualnych, które użytkownik może eksplorować i wchodzić w interakcję z jego obiektami za pomocą specjalnych urządzeń, takich jak gogle VR oraz manipulatory. Ta zaawansowana technologia znajduje zastosowanie w różnych dziedzinach, takich jak szkolenie pilotów wojskowych i cywilnych [1], treningu

[1] Uniwersytet Bielsko-Bialski, Wydział Budowy Maszyn i Informatyki, mbernas@ubb.edu.pl

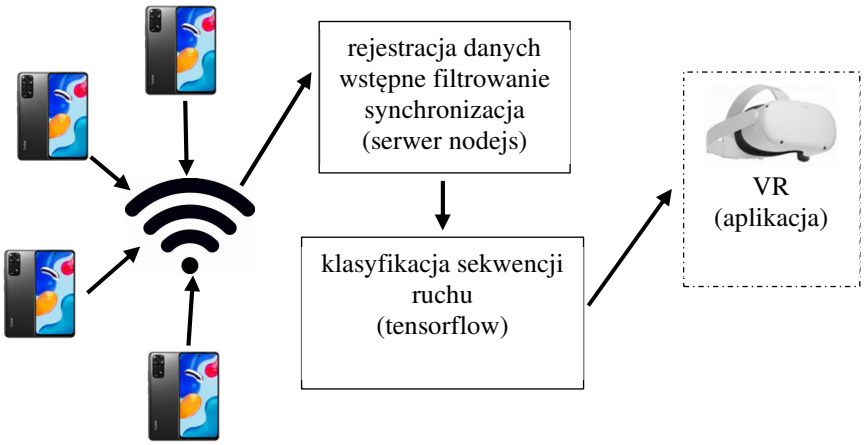
chirurgów [2], a także jako narzędzie edukacyjne i rozrywkowe [3]. Zaletą rozwiązania jest zaangażowanie ruchowe użytkownika, który dzięki temu odczuwa większą imersje z otoczeniem. Ze względu na koszty większość systemów jest ograniczona do śledzenia rąk i głowy w wirtualnej rzeczywistości. Technologia śledzenia rąk i głowy staje się coraz dokładniejsza, przez co zaczyna być odczuwalny brak śledzenia dolnych kończyn. Obecnie dostępne systemy wspierające te funkcje charakteryzują się wysoką ceną. Rysunek 1a przedstawia przykład komercyjnego rozwiązania obejmującego buty oraz specjalistyczną bieżnie.

Rysunek 1. Śledzenie ruchów nóg na stanowisku VR:
a) rozwiązanie komercyjne, b) propozycja rozwiązania

W celu zwiększenia imersji gry możliwe jest użycie dodatkowych czujników i sensorów. Zakup ich wiąże się jednak z dodatkowymi kosztami. W tym artykule zaproponowano zastosowanie w tym celu budżetowych telefonów, które na czas rozgrywki mogą stać się czujnikiem (rys. 1b). Takie urządzenie jest w posiadaniu większości użytkowników. Dodatkowo budżetowy smartfon posiada wystarczającą ilość pamięci, moc obliczeniową i możliwości transmisji, aby stać się nie tylko sensorem, ale także narzędziem do klasyfikacji aktywności człowieka [4]. Czujniki w telefonie zbierają, przetwarzają i przesyłają dodatkowe dane za pomocą łączności bezprzewodowej do serwera [5]. Transmisja z serwerem może być realizowana za pomocą komunikacji bezprzewodowej. Głównym założeniem pracy jest rejestracja a następnie detekcja akcji wykonywanych przez użytkowników środowiska VR, stąd też wyróżniono 10 charakterystycznych aktywności gracza. Typ aktywności rozpoznawany jest za pomocą dedykowanej sieci neuronowej. W kolejnym rozdziale opisane zostanie zaproponowany model. Proces trenowania sieci oraz wyniki zostaną opisane w rozdziale 3. Rozdział 4 zawiera podsumowanie oraz opis dalszych kierunków badań.

2. Model aplikacji do śledzenia i wykrywania ruchu

Model obejmujący detekcję ruchu kończyn dolnych przy użyciu smartfonów jako czujników do monitorowania ruchu kończyn został zaprezentowany na rysunku 2.



Rysunek 2. Śledzenie ruchów nóg na stanowisku VR

Model obejmuje trzy etapy. W pierwszym dane pobierane są z urządzeń umieszczonych za pomocą opasek do ciała użytkownika. Preferowane miejsca obejmują okolice kostki oraz uda. Ze względu na wielkość urządzenia opaski nie powinny blokować stawów. Dane po uruchomieniu aplikacji są zbierane w paczkach, a następnie przesyłane do miejsca docelowego, gdzie są synchronizowane. Po synchronizacji przeprowadzana jest detekcja typu ruchu. Dane dotyczące wykrytej akcji mogą być przekazywane do aplikacji VR w celu zwiększenia poziomu immersji – wpływania na przebieg rozgrywki.

2.1. Pozyskiwanie danych z sensorów

Pozyskiwanie danych z sensorów jest realizowane poprzez dedykowaną aplikację, która zbiera dane z ustalonym poziomem próbkowania, a następnie przesyła je do serwera w standardzie JSON. Komunikacja odbywa się za pomocą protokołu HTTP. Każdy z czujników ma zdefiniowany identyfikator, który jednocześnie jest elementem adresu URL. W przypadku tej implementacji łańcuch URL ma postać:

```
http://[ip_serwera]/data_[nr_czujnika]
```

Format przesyłanych danych w JSON jest ujednolicony i zawiera znacznik czasu oraz dane z czujnika w postaci:

```
{"name":"t_czujnika","values":{"z":0,"y":0,"x":0,
"time":16…}
```

W celu rejestracji danych serwer nasłuchuje żądań na wybranym porcie. W celu ograniczeniu narzutu i minimalizacji czasu przetwarzania pakietu wykorzystano

środowisko *nodejs*. Dzięki temu sam czas odpowiedzi na ramkę wynosi poniżej 0.5 ms. Dane po odebraniu z żądania są przetwarzane do postaci tablicowej. W celu połączenia danych w jedną sekwencję porównywane są wartości *time* z ramek. Algorytm przetwarzania danych przedstawiono w następujących krokach dla serwera:

```
W reakcji na żądanie http:
1. zidentyfikuj sensor x po nagłówku żądania (mapa: data-x),
2. skonwertuj dane z JSON na obiekt,
3. dla każdego pomiaru przekształć dane na mapę postaci
   data-x[time]->[wartości], gdzie x to numer czujnika.
```

Dane z czujników służą do nauki sieci oraz do klasyfikacji. W pierwszym przypadku dane są grupowane w jedną tablicę dla każdej z analizowanych aktywności. W drugim przypadku dane mogą być przesyłane jako paczki o długości zgodnej z założoną długością okna równą *w,* co minimalizuje opóźnienie. Opis wielkości okna opisano w rozdziale 2.2.
Synchronizacja danych z czujników przebiega dla wyznaczonego przedziału czasu. W tej implementacji brane pod uwagę są tylko sekwencje danych, gdzie dane dla wszystkich czujników zostały pobrane. Algorytm łączenia danych w jedną tablicę zdefiniowano następująco:

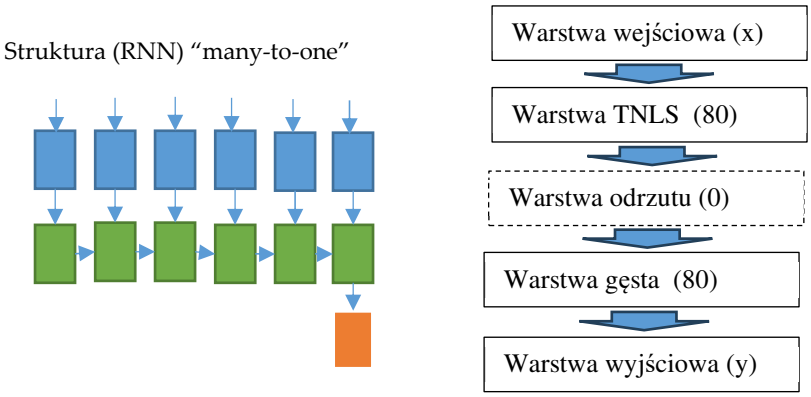
```
1. Wyznacz min=min(time) oraz max=max(time) dla pierwszego
   sensora.
2. Podziel przedział [min, max] na podprzedziały o wielkości
   równej czasie próbkowania czujników.
3. Dla każdego przedziału zmapuj wartość klucza time mapy data-
   x na wartości sensorów.
4. Zapisz w tablicy jeżeli wektor danych jest kompletny – dane
   dla wszystkich czujników.
```

Dane uzyskane z czujników smartfona są próbkowane z określoną częstotliwością. Badania pokazują [6], że częstotliwość próbkowania wynosząca 10 Hz jest wystarczająca do rozróżnienia różnych rodzajów prostych aktywności jak marsz czy bieg. Ta kwestia ma szczególne znaczenie, ponieważ zmniejszenie częstotliwości próbkowania o osiem razy może zwiększyć żywotność urządzenia trzykrotnie [7]. Jednak autorzy w pracy [8] wykazali, że próbkowanie równe 20 Hz umożliwia rozpoznawanie bardziej skomplikowanych ruchów. Dlatego też, ta wartość została przyjęta jako bazowa do dalszych badań. Tablica będąca reprezentacją szeregu czasowego dla badanych czujników jest następnie przekazywana do modułu wnioskowania.

2.2. Wnioskowanie z zastosowaniem rekurencyjnej sieci neuronowej

Dane przed nauką modelu należy dostosować do jego wymagań. Przekształcenia danych mogą mieć charakter przekształceń przestrzennych lub czasowych. W pracy [9] została zaproponowana analiza widmowa, która może przynieść dobre rezultaty podczas analizy aktywności ruchowej. Także zamiana szeregu na cechy związane z danymi statystycznymi (średnia lokalna, wariancja, odchylenie bezwzględne, korelacja lub gradienty) dla okna czasowego były analizowane przez naukowców. W niektórych badaniach zaproponowano również [10] wykrywanie powtarzających

się wzorców podczas aktywności takich jak chodzenie czy bieganie. Po analizie zdecydowano się na użycie głębokich sieci neuronowych, gdzie dane są traktowane jako surowe szeregi czasowe [11] lub przetworzone wektory cech w dziedzinie czasu lub częstotliwości [12]. Ostatecznie wybrano naukę na nieprzetworzonych danych aby maksymalnie uprościć proces i zmniejszyć opóźnienie. Struktura sieci dla tego zadania została przedstawiona na rys. 3.



Rysunek 3. Model rekurencyjnej sieci neuronowej

Model jest implementacją rekurencyjnej sieci neuronowej (rysunek 3) gdzie podstawowa struktura została dobrana na podstawie publikacji [8, 13]. Struktura ta została dostosowana do ciągu wejściowego oraz liczby klas. Dodatkowo zredukowano w niej liczbę neuronów aby przyspieszyć proces wnioskowania. Liczbę epok dostosowano na podstawie analizowanej krzywej uczenia, a warstwa odrzutu (ang. Dropout) została wykorzystana w celu uniknięcia przetrenowania modelu. Wagi na warstwie wyjściowej są traktowane jako przynależność do danej klasy. W analizowanej implementacji klasa o najwyższej wadze jest przyjmowana jako odpowiadająca danej sekwencji wejściowej. Zmodyfikowane parametry wykorzystane w badaniach zostały opisane w kodzie źródłowym biblioteki Tensor Flow Keras [14]:

```
def ocen_model(DT, KT, DTe, KTe):
    v = 0
    e = 80
    nt = DT.shape[1]
    nf = DT.shape[2]
    no = KT.shape[1]
    m = Sequential()
    m.add(LSTM(100, input_shape=(nt, nf)))
    m.add(Dropout(0.5))
    m.add(Dense(100, activation='relu'))
    m.add(Dense(no, activation='softmax'))
    m.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])
    m.fit(DT, KT, epochs=e, verbose=v)
    a = m.evaluate(DTe, KTe, verbose=0)
```

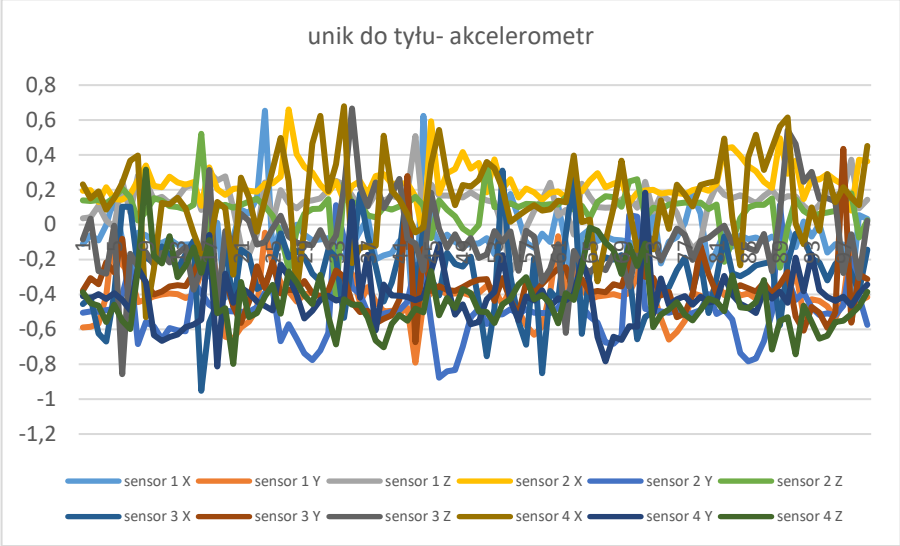
Wytrenowana sieć określa typ aktywności. Wartość ta może być następnie przesłana do aplikacji aby wpłynąć na rozgrywkę.

3. Wyniki eksperymentów

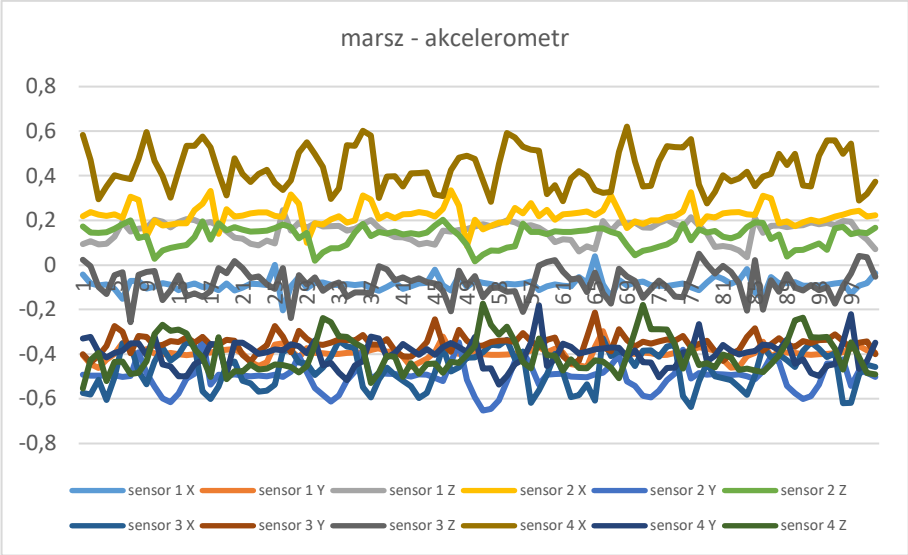
W celu weryfikacji założeń wykorzystano cztery urządzenia mobilne, które umieszczono na nogach użytkownika zgodnie z rysunkiem 4.

Rysunek 4. Układ oraz rozmieszczenie sensorów

Zastosowany telefon Xiaomi 11 umożliwia wykorzystanie czujników takich jak akcelerometr, magnetometr, mikrofon czy kamerę. W celu minimalizacji opóźnień wykorzystano tylko dane z akcelerometrów. Przyjmując trzy osie x, y oraz z dla każdego z czterech urządzeń otrzymujemy 12 monitorowanych parametrów. Dane z sensorów pobierane były 20 razy na sekundę (częstotliwość próbkowania 20 Hz). W celu nauki zarejestrowano dziesięć rodzajów aktywności obejmujące kopnięcia w przód i w bok, wychylenie do przodu, do tyłu i na boki, stanie, marsz oraz bieg. Klasy uzupełnia przysiad. Aktywności zostały dobrane na bazie obserwacji graczy w trakcie sesji w VR. W badaniu wykorzystano szereg czasowy o długości 14000 obejmujących 12 parametrów. Szereg czasowy został podzielony w stosunku 70% do 30%, gdzie pierwsza wartość określa zbiór uczący a druga testowy. Każdy ze zbiorów został znormalizowany oraz przetworzony za pomocą filtru przyrostowego aby uniknąć wpływu charakterystyki miejsca badania na wyniki. Były to jedyne operacje przetwarzania danych. Następnie ciąg za pomocą okna został podzielony na sekwencje o długości okna w. W sumie zdefiniowano y=10 analizowanych klas aktywności użytkownika. Przykładowe wykresy dla danych reprezentowanych w danych klasach przedstawiono na rys. 5 oraz 6 reprezentujących odpowiednio klasy unik do tyłu (krok w tył) oraz marsz (dane przed procesem filtrowania).



Rysunek 5. Odczyty akcelerometrów przy uniku w tył

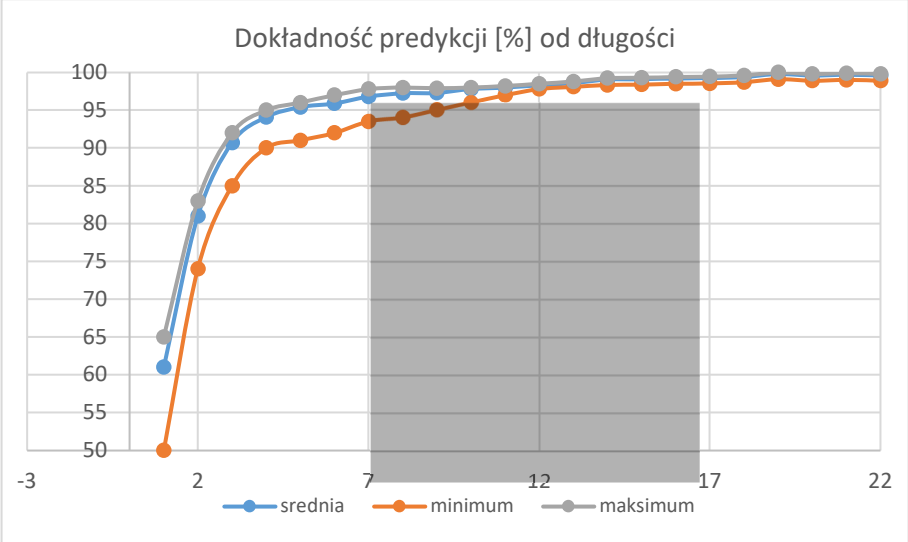


Rysunek 6. Odczyty akcelerometrów podczas marszu

3.1. Weryfikacja modelu dla czterech klas

Badania przeprowadzono dla różnych wielkości okien a tym samym różnej długości sekwencji wejściowej dla sieci. Przeanalizowano okna o długości w = [1, 25] odpowiadającej długości od 0,05 sekundy do 1,25 sekundy. Wielkość graniczna została wybrana jako podwójna wartość okresu badanych czynności

(umożliwiających wykrycie cykliczności). Wyniki uzyskanej dokładności (mierzonej miarą liczby poprawnie sklasyfikowanych przypadków do łącznej ich liczby) w zależności od długości okna przedstawiono na rys. 7.



Rysunek 7. Wynik dokładności klasyfikacji w zależności od wielkości okna

Wyniki wskazują, że najlepsze wyniki i najlepiej dopasowany model otrzymujemy dla długości szeregów czasowych w przedziale powyżej 11. Dla wartości w=7 można już zauważyć, że detekcja aktywności jest stabilna i wynosi 97,8%. Wartości dla okna w=16 umożliwia najdokładniejsze odwzorowanie dynamiki ruchu oraz jej cykliczności (dokładność równa 99%). Niemniej należy zwrócić uwagę, że czas wymagany do zebrania 7 próbek wynosi 0,35s a 16 już 0,8s co może wpłynąć na opóźnienia w grze. Przeanalizowano także błędy klasyfikacji w poszczególnych klasach. W przypadku podziału na poszczególne klasy (rys. 8 i 9) można zauważyć pewne zależności dla błędów klasyfikacji.

	unik_tył	piłka_kop	unik_p	niski_kop	bieg	kop_bok	krok_bok	przysiad	stanie	marsz
unik_tył	400	1	1	0	0	3	6	0	0	1
piłka_kop	11	380	9	2	0	7	0	1	0	2
unik_p	23	0	372	10	0	2	3	1	1	0
niski_kop	0	1	0	404	2	5	0	0	0	0
bieg	0	0	0	0	412	0	0	0	0	0
kop_bok	0	0	1	1	1	401	8	0	0	0
krok_bok	5	0	0	2	2	0	403	0	0	0
przysiad	1	6	4	0	0	2	1	382	1	12
stanie	0	0	0	0	0	0	0	0	412	0
marsz	3	0	0	0	0	0	0	0	0	409

Rysunek 8. Macierz błędów dla 10 klas aktywności i okna w=7

	unik_tył	piłka_kop	unik_p	niski_kop	bieg	kop_bok	krok_bok	przysiad	stanie	marsz
unik_tył	406	0	3	0	0	0	1	0	0	0
piłka_kop	2	396	2	4	0	6	0	0	0	0
unik_p	2	0	405	0	0	0	3	0	0	0
niski_kop	0	0	0	399	0	11	0	0	0	0
bieg	0	0	3	1	406	0	0	0	0	0
kop_bok	0	0	0	0	0	410	0	0	0	0
krok_bok	0	0	0	0	0	0	410	0	0	0
przysiad	0	0	0	0	0	0	0	406	2	2
stanie	0	0	3	0	0	0	0	0	407	0
marsz	0	0	0	0	0	1	0	0	0	409

Rysunek 9. Macierz błędów dla 10 klas aktywności i okna w=16

W przypadku wielkości okna *w*=7 (rys. 8) błędy klasyfikacji pojawiają się nielicznie prawie dla każdej klasy. Najczęstsze błędy można zauważyć, gdy część ruchu jest wspólna jak kopnięcie piłki i krok do tyłu, czy kopnięcie w przód lub w bok. W trakcie testów czujniki były przymocowane paskami co nie gwarantowało ich całkowitej stabilności (jak w przypadku kostiumu) co mogło nieznacznie wpłynąć na wyniki. W przypadku okna *w*=16 (rys. 9) jedyne błędy jakie występują są związane z klasyfikacją typu kopnięcia lub uniku. Warto zauważyć, że nawet dla okna o długości *w*=5 dokładność klasyfikacji wynosi powyżej 95%. Taka długość okna umożliwia zmniejszenie opóźnienia detekcji do 25ms co po uwzględnieniu komunikacji oraz modułu wnioskowania umożliwia detekcję aktywności poniżej 0.2s.

4. Podsumowanie

Zaproponowane rozwiązanie w sposób niskobudżetowy umożliwia śledzenie aktywności z dokładnością 99% oraz umożliwia detekcję aktywności gracza dla dziesięciu klas aktywności. Wykorzystując to rozwiązanie możliwe jest wykrycie czy gracz stoi, idzie czy biegnie. Dodatkowo, możliwe jest wykonywanie uniku oraz kopnięć, które mogą stać się elementem rozgrywki. Zaletą rozwiązania jest możliwość zastosowania smartfonu (posiadanego przez większość użytkowników) jako elementu śledzącego gracza, a finalnie sama gra może pełnić role serwera i klasyfikatora. Przy ograniczeniu dokładności detekcji do 95% poprzez zmniejszenie okna do 5 możliwe jest zmniejszenie czasu potrzebnego na detekcję do 0.2 sekundy. Warto podkreślić, że nie wszystkie klasy ruchu w ramach aplikacji muszą być wykorzystywane. Kolejnym krokiem jest integracja klasyfikatora dla wybranej gry oraz przetestowanie rozwiązania na większej grupie użytkowników. Dodatkowo możliwym kierunkiem jest wykorzystanie dodatkowych sensorów komórki w celu jeszcze dokładniejszej detekcji położenia kończyn i śledzenia ich w grze w czasie rzeczywistym. Ostatecznie możliwe jest także strumieniowe przesyłanie lokalizacji kończyn oraz dodanie ich do modelu rozgrywki.

LITERATURA

1. DYMORA P. *et al* Deep recurrent neural networks for human activity recognition. 2021 *IOP Conf. Ser.: Mater. Sci. Eng.* 1024 012099
2. MÜLLER W., BOCKHOLT U., LAHMER A. et al. VRATS – Virtual-Reality-Arthroskopie-Trainingssimulator. Radiologe 40, 290–294 (2000).
3. KAVANAGH, S., LUXTON-REILLY, A., WUENSCHE, B., PLIMMER, B. (2017). A systematic review of Virtual Reality in education. Themes in Science and Technology Education, 10(2), 85-119.
4. STRACZKIEWICZ M., JAMES P. ONNELA JP. A systematic review of smartphone-based human activity recognition methods for health research. npj Digit. Med. 4, 148 (2021).
5. GIANNINI P.; BASSANI G.; AVIZZANO, C.A. Filippeschi, A. Wearable Sensor Network for Biomechanical Overload Assessment in Manual Material Handling. Sensors 2020, 20, 3877.
6. WANNENBURG J., MALEKIAN R. Physical activity recognition from smartphone accelerometer data for user context awareness sensing. IEEE Trans. Syst. Man, Cybern. Syst. 47, 3143–3149 (2017).
7. YURUR O., LABRADOR M., MORENO W. Adaptive and energy efficient context representation framework in mobile sensing. IEEE Trans. Mob. Comput. 13, 1681–1693 (2014).
8. BERNAŚ M., PŁACZEK, B., LEWANDOWSKI M. Ensemble of RNN Classifiers for Activity Detection Using a Smartphone and Supporting Nodes. Sensors 2022, 22, 9451. https://doi.org/10.3390/s22239451
9. LI P., WANG Y., TIAN Y., ZHOU T.-S., LI, J.-S. An automatic user-adapted physical activity classification method using smartphones. IEEE Trans. Biomed. Eng. 64, 706–714 (2017)
10. DERAWI M., BOURS P.: Gait and activity recognition using commercial phones. Comput. Secur. 39, 137–144 (2013).
11. AVILÉS-CRUZ C., FERREYRA-RAMÍREZ A., ZÚÑIGA-LÓPEZ A., VILLEGAS-CORTÉZ J. Coarse-fine convolutional deep-learning strategy for human activity recognition. Sensors 19, 1556 (2019).
12. ZHAO B., LI S., GAO Y., LI C., LI W. A framework of combining short-term spatial/frequency feature extraction and long-term IndRNN for activity recognition. Sensors 20, 6984 (2020).
13. GUILLAUME C., LSTMs for Human Activity Recognition, 2016, https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition
14. ABADI M., AGARWAL A., BARHAM P., ET. AL. TensorFlow: Large-scale machine learning on heterogeneous systems 2015. Software available from *tensorflow.org*.