

A G H

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

Projekt dyplomowy

Wykrywanie postawy człowieka za pomocą stereowizji

Human posture detection using stereovision

Autor: Jakub Mieszczak
Kierunek studiów: Automatyka i Robotyka
Opiekun pracy: dr inż. Waldemar Bauer

Kraków, 2025

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpościera bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, videogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Niniejszą pracę pragnę zadeklować mojej mame, Lucynie Mieszczak, za wsparcie, zaangażowanie oraz pomoc, które towarzyszyły mi na każdym etapie mojej edukacji i odegrały kluczową rolę w realizacji moich planów, w tym rozpoczęcia oraz ukończenia studiów.

Serdecznie dziękuję mojemu promotorowi, dr inż. Waldemarowi Bauerowi, za inspirację, wyrozumiałość oraz nieocenioną pomoc, które znacząco przyczyniły się do realizacji tej pracy. Pańskie wsparcie i cenne wskazówki były dla mnie niezwykle wartościowe na każdym etapie tworzenia tego projektu.

Spis treści

Wstęp.....	7
Cele i zakres pracy	9
1. Przegląd literatury	11
1.1. Przegląd metod kalibracji	11
1.2. Algorytmy estymacji pozycji	15
1.3. Triangulacja	16
1.4. Rozpoznawanie czynności.....	18
1.4.1. Dwuścieżkowe sieci neuronowe	18
1.4.2. Trójwymiarowe sieci konwolucyjne	19
1.4.3. Modele czasu rzeczywistego.....	20
1.5. Przegląd metod uczenia maszynowego dla danych szeregow czasowych.....	21
1.5.1. Jednokierunkowa sieć neuronowa.....	22
1.5.2. Rekurencyjna sieć neuronowa	22
1.5.3. GRU	23
1.5.4. LSTM.....	23
1.5.5. Przykład zastosowania modeli rekurencyjnych	25
2. Kalibracja i synchronizacja kamer oraz zbieranie danych	27
2.1. Przygotowanie stanowiska do nagrania.....	27
2.2. Kalibracja kamer	27
2.2.1. Parametry wewnętrzne i zewnętrzne.....	27
2.3. Synchronizacja kamer	29
2.3.1. Cele synchronizacji.....	29
2.3.2. Badanie wydajności synchronizacji.....	29
3. Przygotowanie i przetwarzanie danych	31
3.1. Zbieranie danych	31
3.1.1. Przygotowanie algorytmu do zbierania danych 2D za pomocą BlazePose	31

3.1.2. Triangulacja danych 2D na dane 3D.....	32
3.2. Zapoznanie się z danymi i ich struktura	33
3.3. Przygotowanie danych do analizy i modelowania.....	34
4. Uczenie modeli detekcji czynności.....	37
4.1. Implementacja modeli	37
4.2. Dobór długości próbek, liczby epok, parametrów modelu.....	38
4.2.1. Wyniki dla RNN	40
4.2.2. Wyniki dla GRU.....	42
4.2.3. Wyniki dla LSTM	45
4.2.4. Wyniki dla FNN.....	46
4.3. Porównanie czasu uczenia oraz wykonywania.....	48
4.3.1. Wyniki dla RNN	49
4.3.2. Wyniki dla GRU.....	49
4.3.3. Wyniki dla LSTM	50
4.4. Proces finalnego trenowania modelu	51
5. Aplikacja	55
5.1. Schemat aplikacji.....	55
5.2. Działanie aplikacji	55
5.2.1. Instrukcja.....	55
5.2.2. Kalibracja.....	56
5.2.3. Uruchom	58
6. Podsumowanie i wnioski.....	61
6.1. Podsumowanie wyników pracy	61
6.2. Możliwości dalszego rozwoju	62
6.3. Wnioski.....	63

Wstęp

Lokalizowanie obiektów w przestrzeni trójwymiarowej od zawsze było istotne w kwestiach zarządzania, planowania oraz zbierania potrzebnych danych [1][2]. Niestety, obraz z pojedynczej kamery nie może dostarczyć informacji o głębi ani o odległości danego obiektu od urządzenia rejestrującego. Odpowiedzią na tę potrzebę było powstanie systemów stereometrii, które za pomocą obrazów z różnych perspektyw, są w stanie dostarczyć cennych informacji.

Rozwój stereometrii był możliwy poprzez postęp dziedzin takich jak matematyka oraz systemy wizyjne. Dzięki odkryciom w matematyce opracowano zaawansowane metody kalibracyjne oraz sposoby na znalezienie odpowiadających sobie punktów na obrazach. Z kolei systemy wizyjne pozwoliły zamieścić matematyczną teorię na konkretne narzędzia i rozwiązania.

Główną motywacją do stworzenia systemu określającego postawę człowieka była chęć zbadania możliwości opracowania niskobudżetowego środowiska umożliwiającego analizę czynności wykonywanych przez osobę znajdującą się w jego obrębie. Szczególnie interesujące było zrozumienie ograniczeń i trudności, jakie mogą się pojawić zarówno po stronie tworzenia odpowiedniej architektury, jak i zbierania danych i uczenia odpowiednich modeli z rodziny rekurencyjnych sieci neuronowych.

Potencjalne zastosowania takiego środowiska można znaleźć w miejscach, gdzie istotne jest posiadanie ciągłych informacji o czynnościach wykonywanych przez człowieka. Przykładem może być miejsce pracy, w którym system mógłby monitorować poprawność wykonywania zadań oraz wykrywać zagrożenia BHP, takie jak omdlenia czy utrata przytomności. Innym miejscem mogą być obszary, w których szczególnie istotne jest automatyczne mierzenie produktywności pracownika w celu zbierania potrzebnych informacji do planowania, optymalizowania i tworzenia strategii na przyszłość.

Cele i zakres pracy

Celem pracy jest stworzenie środowiska testowego poprzez kalibracje, synchronizacje oraz triangulacje obrazu z kamer, a następnie zebranie za jego pomocą danych oraz trenowanie modeli rozpoznających czynności wykonywane przez użytkownika. Plan pracy obejmuje również porównanie różnych modeli pod względem prędkości, skuteczności oraz czasu potrzebnego na ich uczenie.

Pierwszy rozdział obejmuje wstęp teoretyczny, który pozwala zrozumieć mechanizmy oraz narzędzia wykorzystywane podczas tworzenia rozwiązania. Omówione zostaną zagadnienia związane z kalibracją, modelami estymacji pozycji, triangulacją, metodami rozpoznawania czynności oraz architekturami modeli rekurencyjnych.

W drugim rozdziale opisane są kroki potrzebne do przygotowania środowiska testowego, takie jak ułożenie i kalibracja kamer oraz synchronizacja obrazów z wielu perspektyw. Szczególna uwaga będzie skupiona na otrzymywaniu parametrów kamery oraz badaniu wydajności rozwiązania.

Trzeci rozdział koncentruje się na analizie zgromadzonych danych, w tym na badaniu ich struktury, procesie oczyszczania oraz podziale na zestawy treningowe i testowe. Szczególnie istotne w tym rozdziale jest, aby zrozumieć, co oznaczają dane oraz z czego wynika ich kompozycja.

Rozdział czwarty dotyczy trenowania rekurencyjnych sieci neuronowych oraz porównania ich wyników. Zawiera informacje o architekturze modeli, ich parametrach oraz długości próbek. Zakończony jest analizą wybranego modelu za pomocą popularnych metryk ocenяjących jego jakość.

Kolejny rozdział zajmuje się aplikacją działającą w czasie rzeczywistym, która ma na celu informowanie o czynnościach wykonywanych przez użytkownika. Elementem aplikacji jest sekcja pozwalająca na przygotowanie oraz konfigurację kamer wraz z przestrzenią roboczą.

Ostatni rozdział obejmuje analizę otrzymanych wyników oraz wyciągnięciu z nich wniosków. Można tam znaleźć informacje o ewentualnym rozwoju rozwiązania, jego mocnych stronach oraz najważniejszych ograniczeniach.

1. Przegląd literatury

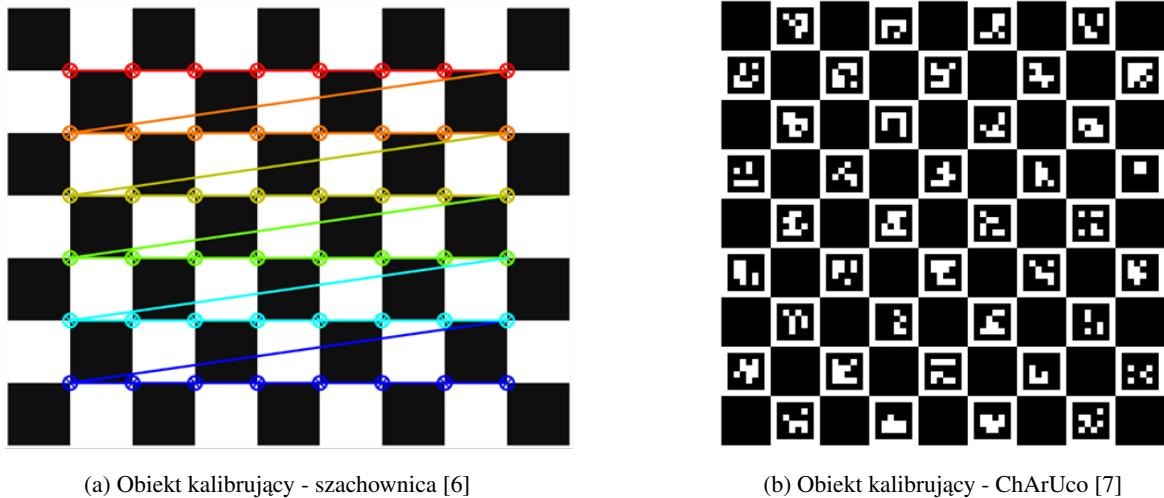
Rozdział ten zawiera przegląd zagadnień i metod rozważanych w trakcie pracy nad opracowywanym rozwiązaniem, w oparciu o dotychczasową wiedzę, literaturę oraz odkrycia z ich zakresu. Wyniki wdrożonych narzędzi są opisane szczegółowo w poświęconych im rozdziałach.

1.1. Przegląd metod kalibracji

Kalibracja kamer jest elementem tworzenia systemów wizyjnych, robotycznych [3] oraz trójwymiarowych rekonstrukcji [4]. Badania nad nimi możemy obserwować już od blisko 60 lat. Jednym z etapów kalibracji jest określenie jej parametrów, takich jak **parametry wewnętrzne i zewnętrzne kamery** [3][4].

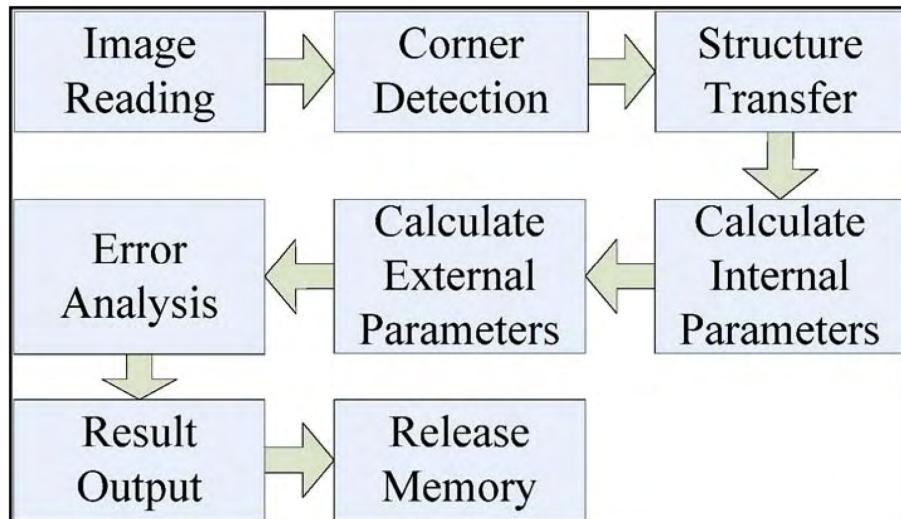
W trójwymiarowej wizji komputerowej kalibracja jest niezbędnym elementem do pozyskania informacji z dwuwymiarowych obrazów [3][4]. Techniki wyznaczania takich parametrów można podzielić na dwa główne rodzaje: **kalibracja fotogrametryczna** oraz **autokalibracja**. Pierwsza metoda opiera się na obserwacji obiektu kalibracyjnego i charakteryzuje się dużą precyzją oraz prostotą. Drugi sposób nie wykorzystuje specjalistycznych obiektów, wystarczy poruszyć kamerą w statycznej scenie, aby uzyskać potrzebne parametry [5].

Powszechnie używana jest kalibracja Zhengyou Zhang'a, będąca kalibracją fotogrametryczną, wyznaczającą parametry wewnętrzne i zewnętrzne kamer. Polega ona na zabraniu co najmniej dwóch obrazów z różnych perspektyw oraz pod różnymi kątami. Zdjęcia muszą zawierać specjalny wzór, na którym będzie można w łatwy sposób znaleźć charakterystyczne punkty. Często wykorzystuje się w tym celu obrazy szachownicy [3] 1.1a lub innych obiektów takich jak ChArUco 1.1b. W porównaniu do innych metod, które wymagają więcej informacji oraz parametrów, ta jest uważana za jedną z prostszych w zastosowaniu. [4]



Rys. 1.1. Porównanie dwóch obiektów kalibratorycznych.

OpenCV to ogólnodostępna biblioteka opracowana przez firmę Intel, która znajduje szerokie zastosowanie w dziedzinie wizji komputerowej, w tym implementuje metodę kalibracji Zhang. Dzięki wykorzystaniu wydajnych funkcji napisanych w języku C oraz nowoczesnych klas w języku C++ pozwala na efektywne wykonywanie operacji macierzowych. OpenCV wspiera popularne systemy operacyjne, takie jak Linux i Windows, co czyni ją wszechstronnym narzędziem dla programistów na różnych platformach. [4]



Rys. 1.2. Proces kalibracji biblioteki OpenCV [4]

Parametry wewnętrzne:

Opisują one wewnętrzne cechy optyczne i geometryczne urządzenia, między innymi: ogniskowa, główny punkt, współczynnik rozciągnięcia oraz parametry zniekształcenia optycznego [4]. Zawierają one informacje o konstrukcji kamery oraz jej matrycy [3]. Można za ich pomocą wyznaczyć macierz kalibracji kamery K . Każdy parametr określa pewną cechę urządzenia:

- ogniskowe (ang. focal Length) - opisuje odległość między optycznym środkiem soczewki, a sensorem kamery (f_x, f_y),
- główny punkt (ang. Principal Point) - punkt na płaszczyźnie obrazu, w którym oś optyczna kamery przecina tę płaszczyznę (c_x, c_y),
- współczynnik rozciągnięcia (ang. skew), odchylenie od kąta prostego pomiędzy osiami układu współrzędnych obrazu, czyli między osią X, a osią Y, zazwyczaj równy 0 (s),
- parametry zniekształcenia optycznego (ang. lens distortion), opisują nieliniowe deformacje obrazu wprowadzone przez soczewki obiektywu (k_1, k_2, p_1, p_2)

Parametry zewnętrzne:

Parametry zewnętrzne kamery mówią o rotacji i położeniu kamery w przestrzeni, są one szczególnie potrzebne w projektach, w których mamy do czynienia z wieloma kamerami, ustawionymi pod różnymi kątami, w niejednakowych miejscach, takich jak rekonstrukcja 3D [3]. Opisywane są też jako parametry określające relacje między kamerami, a sceną [8] lub trójwymiarową pozycją i orientacją układu współrzędnych kamery w porównaniu do układu współrzędnych świata [4]. Można je wymienić jako:

- macierz rotacji (ang. rotation matrix) - opisuje orientację kamery w przestrzeni, czyli jej obrót względem osi układu współrzędnych (R),
- wektor translacji (ang. translation vector) - określa położenie kamery w przestrzeni jako przesunięcie od środka układu współrzędnych (t),

Matematyczna reprezentacja:

Punkty w przestrzeni 2D reprezentujemy jako u oraz v , które odpowiadają odpowiednio współrzędnym x oraz y na obrazie:

$$m = \begin{bmatrix} u \\ v \end{bmatrix} \quad \tilde{m} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Punkty w przestrzeni 3D reprezentujemy jako:

$$M = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \tilde{M} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Macierz parametrów wewnętrznych kamery (ang. intrinsic parameters matrix):

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_x \\ 0 & 0 & 1 \end{bmatrix}$$

- c_x, c_y – współrzędne punktu głównego,
- f_x, f_y – współczynniki skalowania,
- s – parametr opisujący zniekształcenie kątowe dwóch osi obrazu.

Macierz rotacji wokół osi x , reprezentująca obrót o kąt α .

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

Macierz rotacji wokół osi y , reprezentująca obrót o kąt β .

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

Macierz rotacji wokół osi z , reprezentująca obrót o kąt γ .

$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha)$$

Wektor translacji kamery, opisujący jej położenie.

$$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

- t_1 – translacja kamery wzdłuż osi x ,
- t_2 – translacja kamery wzdłuż osi y ,
- t_3 – translacja kamery wzdłuż osi z .

Macierz parametrów zewnętrznych kamery (ang. extrinsic parameters matrix):

$$[R | t] = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{bmatrix}$$

Relacja pomiędzy punktami w przestrzeni 2D oraz 3D:

$$\lambda \tilde{m} = K \begin{bmatrix} R & t \end{bmatrix} M$$

- λ – współczynnik skalowania,
- R, t – parametry zewnętrzne kamery, odpowiadające za położenie i orientację w przestrzeni.

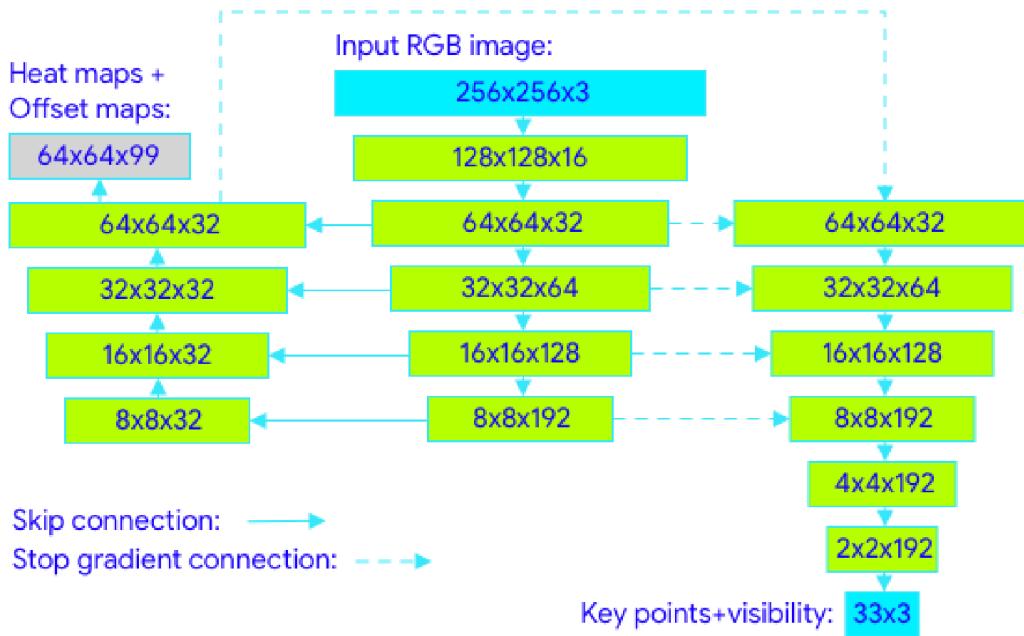
1.2. Algorytmy estymacji pozycji

Rozpoznawanie pozycji ciała człowieka ze zdjęć i nagrań znajduje szerokie zastosowanie w interfejsach użytkownika opartych na ruchu, systemach bezpieczeństwa oraz analizie aktywności fizycznej. Jest to jednak zagadnienie skomplikowane ze względu na mnogość pozycji, wielu stopni swobody, oraz problemów związanych z zasłonięciem częścią ciała [9][10]. Metody estymacji pozycji (ang. pose estimation) w ostatnich latach zawdzięczają swój dynamiczny rozwój odkryciom w dziedzinie sieci neuronowych oraz głębokiego uczenia, takich jak sieci konwolucyjne, które są podstawą dla bardziej zaawansowanych architektur wizyjnych [1][10].

Determinowanie pozycji przy pomocy uczenia maszynowego można podzielić na dwie główne kategorie. Pierwszą jest detekcja, która wykorzystuje analizę obrazu w celu oszacowania prawdopodobieństwa wystąpienia konkretnych punktów kluczowych (ang. keypoints) w danym obszarze obrazu RGB. Wciąż nie można jednak powiedzieć na tym etapie o konkretnych współrzędnych (x, y), które mogłyby być w łatwy sposób interpretowane. Do ich wyznaczenia konieczne jest zastosowanie podejścia maksymalizacji prawdopodobieństwa a posteriori (MAP) z wykorzystaniem operacji argmax. Drugim sposobem jest podejście regresyjne, pozwala na bezpośrednie określenie współrzędnych (x, y) konkretnego punktu kluczowego, poprzez wykorzystanie funkcji nieliniowych. [1][9][10]

W pracy [9] zauważono, że metody oparte o wyznaczanie prawdopodobieństwa występowania punktów dla każdego piksela (wyznaczanie heatmapy prawdopodobieństwa), pomimo możliwości rozpoznawania wielu osób jednocześnie, są zbyt złożone obliczeniowo, aby można było z nich korzystać w czasie rzeczywistym na urządzeniach mobilnych. Modele regresyjne za to, pomimo mniejszej złożoności i większej stabilności, często nie radzą sobie ze stopniem skomplikowania zadania. Autorzy podkreślają, że w pracy [11], architektura stacked hourglass pozwoliła na znaczne poprawienie jakości nawet przy mniejszej liczbie parametrów. Jednocześnie proponują rozwinięcie tego podejścia o dodatkowy mechanizm ender-decoder network, wykorzystany do predykcji heatmapy wszystkich szukanych punktów. Kolejnym ważnym elementem jest enkoder, który określa bezpośrednio współrzędne (x, y) punktów. Dzięki wprowadzeniu tego typu zmian złożoność modelu jest zredukowana wystarczająco, żeby można było z nich korzystać na urządzeniach mobilnych.

Dane, na których był uczony Blazepose, zawierają 60 tysięcy zdjęć, na których znajduje się jedna lub kilka osób w zwyczajnych, codziennych pozycjach oraz 25 tysięcy zdjęć jednej osoby wykonującej ćwiczenia fitness. Dane były przypisywane manualnie przez ludzi. [9]



Rys. 1.3. Architektura sieci Blazepose [9]

1.3. Triangulacja

Proces odwzorowania współrzędnych trójwymiarowych punktów, na podstawie obrazów z różnym perspektywą, nazywamy triangulacją. Polega on na przecięciu się co najmniej dwóch prostych, w przestrzeni, wyznaczając współrzędne punktu. W idealnych warunkach jest to trywialne zadanie. W two-rzeniu rzeczywistych implementacji tego mechanizmu istnieje potrzeba brania pod uwagę istnienie dodatkowego czynnika, jakim jest szum wynikający z trudności wyznaczenia dokładnych informacji o środowisku, w jakim znajdują się punkty. Z tego powodu problem ten staje się bardziej skomplikowany oraz wymaga zastosowania odpowiednich metod oszacowania lokalizacji wyznaczanego punktu [12]. Wartych wymienienia jest kilka metod opisanych w [12], które są często stosowane do rozwiązywania tej przeszkody:

- **Poly** - najlepsza z wszystkich metod (z wyjątkiem Poly-Abs, która w niektórych sytuacjach może mieć takie same wyniki), jeśli wykorzystywane są dwie perspektywy oraz czas wykonywania nie jest istotny,
- **Poly-Abs** - zachowuje się podobnie do Poly, znajduje globalne minimum sumy wielkości błędu,
- **Mid-point** - metoda, która nie jest polecana. Działa gorzej, nawet dla rekonstrukcji Euclidesowej, od innych metod liniowych,
- **Linear-Eigen, Linear-LS** - charakteryzują się dużą prędkością oraz prostotą. Metody poradzą sobie z małymi przekształceniami, takimi jak przesunięcie, ale mogą mieć problemy z bardziej złożonymi, przykładowo ze zmianą perspektywy,

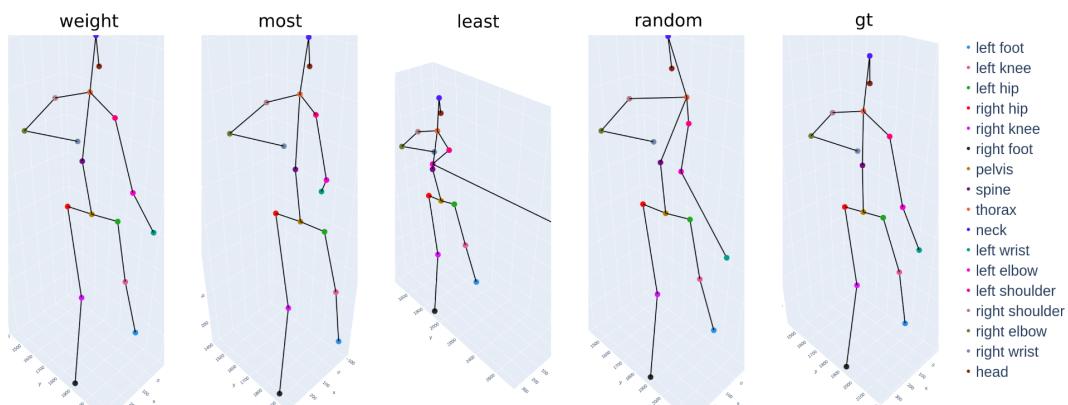
- **Iterative-Eigen, Iterative-LS** - nie mają tak dobrych rezultatów jak Poly, ale osiągają bardzo dobre wyniki i są lepsze od modeli liniowych. Jednym z istotnych problemów tych metod jest ich zawodność – nie zawsze są w stanie znaleźć rozwiązanie, co wymusza konieczność korzystania z metod zapasowych. [12]

Techniki triangulacji często można łączyć z innymi w celu stworzenia bardziej zaawansowanych architektur. Wykorzystanie modeli estymacji pozycji do otrzymania obrazów z dwuwymiarowymi punktami oraz triangulacji, w celu otrzymania ich trójwymiarowych reprezentacji, pozwala na estymację pozycji w przestrzeni 3D. [10]

Korzystanie z modeli estymacji pozycji może powodować pewne ograniczenia. Model może nie być wytrenowany na odpowiedniej ilości danych, aby wykryć każdą pozycję, czego konsekwencją może być utworzenie punktów w nieprawidłowych miejscach. Kolejny problem wynika z możliwości ułożenia pozycji w taki sposób, że niektóre części ciała będą zasłonięte przez inne, w wyniku czego punkty będą niedokładne. Niektóre architektury nie są zoptymalizowane do wydajnego funkcjonowania na urządzeniach mobilnych, co mogłoby być ograniczające podczas tworzenia aplikacji działającej w czasie rzeczywistym. Z powodu tych problemów należy dobrać odpowiedni model, który będzie spełniał oczekiwania, a punkty kluczowe, które zostały niepoprawnie wykryte, zostały pominięte. Jednym ze sposobów pominięcia takich punktów jest zastosowanie RANSAC, który oznacza punkty kluczowe jako odstające/wyjątki, jeśli wartość ich błędu jest większa od pewnego progu. [10]

Plusem wynikającym z modeli estymacji pozycji, jest fakt określania przez nie konkretnych punktów charakterystycznych ciała. W ten sposób podczas triangulacji występuje dodatkowa informacja o odpowiadających sobie punktach, co eliminuje konieczność operowania na chmurze niezidentyfikowanych elementów przestrzeni. [10]

Jak widać na Rysunku 1.4, osiągnięcie poprawnej triangulacji może być skomplikowane. Różnice w samych metodach, dają różne efekty dla tej samej pozy. W aplikacjach mogą występować dodatkowo błędy wynikające z kalibracji kamer, działania modeli estymacji pozycji lub synchronizacji obrazów. Trzeba zdawać sobie z tego sprawę podczas próby tworzenia poprawnie działającego rozwiązania.



Rys. 1.4. Porównanie triangulacji na danych z datasetu Human3.6M [10]

1.4. Rozpoznawanie czynności

Dynamiczny rozwój algorytmów pozwalających rozpoznawać konkretne czynności, można powiązać z rozwojem uczenia maszynowego [1][13]. Dodatkowym aspektem wpływającym na intensywność prowadzonych prac nad tym zagadnieniem, jest duża ilość zastosowań tych rozwiązań, między innymi analiza zachowania, interakcje człowiek-robot czy gry oraz rozrywka [1][14][13]. Pomimo dużej użyteczności, mierzą się one z wieloma utrudnieniami, takimi jak uwzględnianie informacji z długich przebiegów czasowych, złożoności obliczeniowej, trudności w porównywaniu wyników z powodu dużej ilości datasetów oraz sposobów oceniania modeli. Zazwyczaj wraz ze zwiększającą się liczbą danych wysokiej jakości, zwiększa się skuteczność modeli, z tego powodu aby osiągnąć satysfakcyjujące rezultaty konieczne jest przejście przez czasochłonny i często drogi proces zbierania danych, który można podzielić na:

1. definiowanie listy akcji do rozpoznania,
2. zbieranie danych z różnych źródeł,
3. przypisywanie ręcznie danych,
4. czyszczenie danych, usuwanie duplikatów [13]

Rozpoznawanie oraz określanie czynności na wideo jest określane jako rozpoznawanie działań w sekwencjach wideo (ang. video action recognition). Przed 2015 rokiem główną metodą realizującą to zadanie była IDT, przede wszystkim dzięki dużej dokładności. Największą wadą tego rozwiązania była złożoność obliczeniowa, niska skalowalność oraz trudność wdrożenia. Sieci konwolucyjne ze względu na problemy ze zrozumieniem informacji czasowych, nie były często stosowane w tego typu problemach. W pracy [13], gdzie opisano ponad 200 prac naukowych związanych z rozpoznawaniem czynności na podstawie wideo z wykorzystaniem głębokiego uczenia, autorzy wyróżniają trzy trendy rozwoju tych metod:

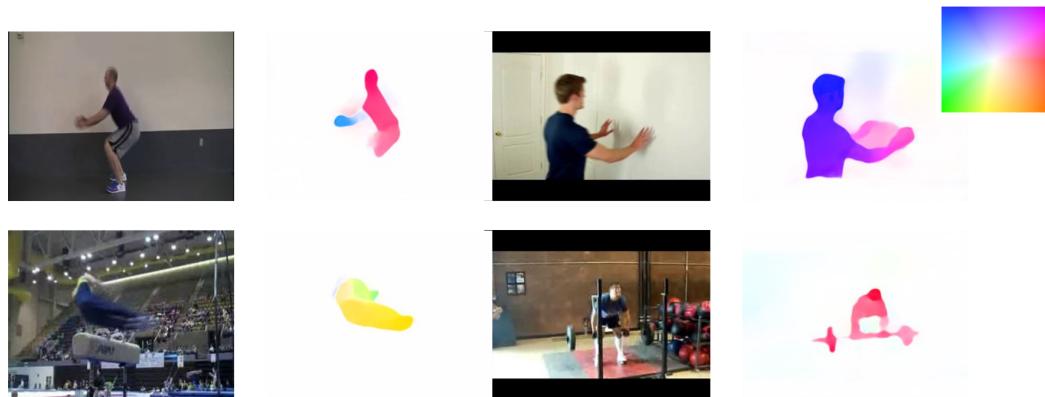
- Dwuścieżkowe sieci neuronowe (ang. Two-Stream Networks),
- Trójwymiarowe sieci konwolucyjne,
- Modele czasu rzeczywistego. [13]

1.4.1. Dwuścieżkowe sieci neuronowe

Pierwszy trend to tworzenie modeli opartych o dwuścieżkowe sieci neuronowe, polega na tworzeniu dwóch ścieżek, obie wyodrębniają informacje za pomocą sieci konwolucyjnych. Pierwsza z nich bazująca na statycznych zdjęciach, wydobywa istotne cechy, zawierają one informacje o lokalizacji oraz charakterystyce obiektu. Druga jest odpowiedzialna za zrozumienie informacji związanych z danymi czasowymi dzięki **przepływowi optycznemu**. Przepływ optyczny (ang. Optical Flow) to efektywny sposób

na przedstawianie ruchu obiektu lub sceny, dzięki uwzględnianiu jedynie elementów poruszających się, nie tylko wyodrębniane są najistotniejsze informacje, ale również ułatwia proces uczenia w porównaniu ze zwykłym obrazem RGB zawierającym często skomplikowane, statyczne tło. Dodatkowo przepływ optyczny jest w stanie przekazywać informacje o kierunku poruszania się danego obiektu, jak pokazuje Rysunek 1.5.

Dwuścieżkowe sieci neuronowe wykorzystywane w analizie wideo operują dwoma strumieniami: pierwszy strumień przetwarza obrazy wejściowe, koncentrując się na ekstrakcji statycznych cech obiektów, natomiast drugi strumień analizuje sekwencje przepływu optycznego, umożliwiając modelowi uchwycenie informacji związanych z dynamiką ruchu pomiędzy kolejnymi klatkami wideo. Dzięki takiej architekturze, modele oparte na CNN, osiągnęły wyniki podobne do IDT, 88% do 87.9% dla datasetu UCF101 oraz 59.4% do 61.1% dla HMDB51. Wielki sukces tego podejścia zainspirował kolejne rozwiązania takie jak: TDD, LRCN, Fusion, TSN. [13]



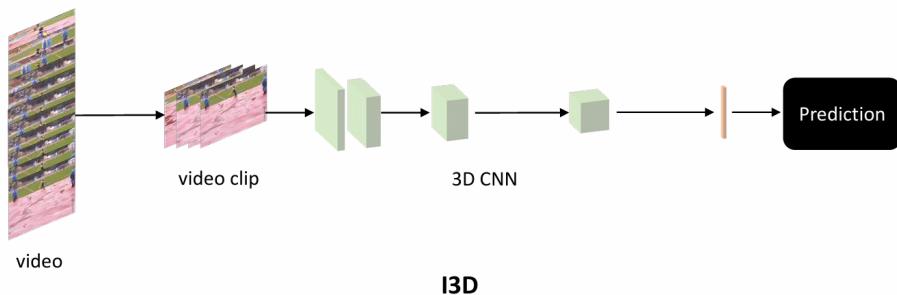
Rys. 1.5. Wizualizacja przepływu optycznego [13]

1.4.2. Trójwymiarowe sieci konwolucyjne

Z powodu dużej złożoności obliczeniowej oraz ilości danych, związanej z wyznaczaniem przepływu optycznego, skalowanie oraz uruchamianie algorytmów w czasie rzeczywistym było bardzo ograniczone. Zaprowadziło to do prac nad nowym typem architektury związanym z reprezentacją wideo jako tensora 3D, o dwóch wymiarach przestrzennych i jednym wymiarze czasowym. Kolejnych trendem był rozwój trójwymiarowych sieci konwolucyjnych. Ich pierwsze wersje, takie jak C3D, które można porównać do zastosowania VGG16 dla danych 3D, nie osiągały satysfakcyjnych wyników, ale prezentowały duże zdolności generalizowania i mogły służyć jako ekstraktor cech dla wideo. Kolejny problem był związany z wielkością datasetów oraz czasu, jaki był potrzebny do trenowania modeli, który często wynosił kilka tygodni. [13]

Przełomem w roku 2017 roku stał się model I3D. Jego twórcy wykorzystali wytrenowane modele konwolucyjne 2D, takie jak ImageNet, do stworzenia ich reprezentacji 3D. Po wstępny trenowaniu na wielkoskalowym datasetie Kinetics400, I3D uzyskał 95.6% skuteczności dla UCF101 oraz 74.8% skuteczności dla HMDB51. I3D stworzyło nowe standardy uczenia modeli do określania czynności,

między innymi: trenowanie przy użyciu datasetów dużych rozmiarów oraz korzystanie z wyuczonych modeli konwolucyjnych dla danych 2D.



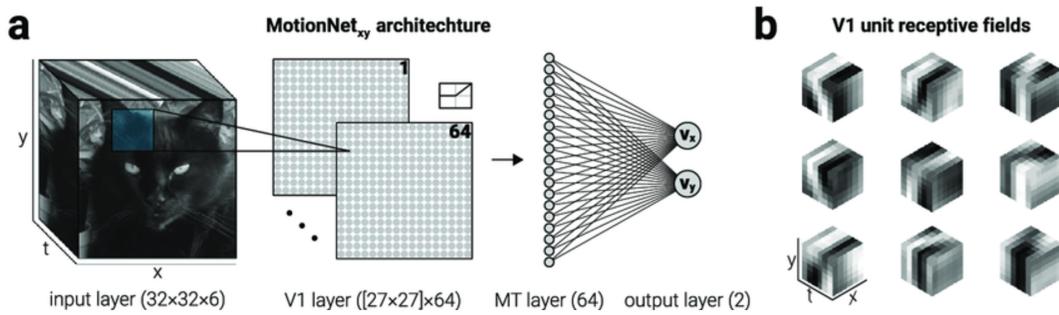
Rys. 1.6. I3D - 3D CNN [13]

1.4.3. Modele czasu rzeczywistego

Kolejny etap rozwoju metod rozpoznawania czynności skupiał się na ich wydajności oraz możliwości wykorzystania na urządzeniach o mniejszych zdolnościach obliczeniowych. Metody oparte o dwuścieżkowe sieci neuronowe wymagają obliczenia przepływu optycznego oraz przechowywania go na dysku. Zbiór danych Kinetics400 posiada zdjęcia, które potrzebują 4.5 TB pamięci do przechowywania. Dodatkowo obliczanie przepływu optycznego dla każdego obrazu utrudnia wykorzystanie tego rozwiązania w czasie rzeczywistym. Metody oparte o trójwymiarowe sieci konwolucyjne są ciężkie w uczeniu oraz późniejszym wykorzystaniu. Czas uczenia SlowFast network na zbiorze danych Kinetics 400 przy użyciu 8 wydajnych GPU, wynosił 10 dni. Tak wysokie koszty w zasobach oraz czasie pozwalały jedynie dużym firmą lub laboratoriom na rozwijanie tych modeli dla swoich potrzeb. [13]

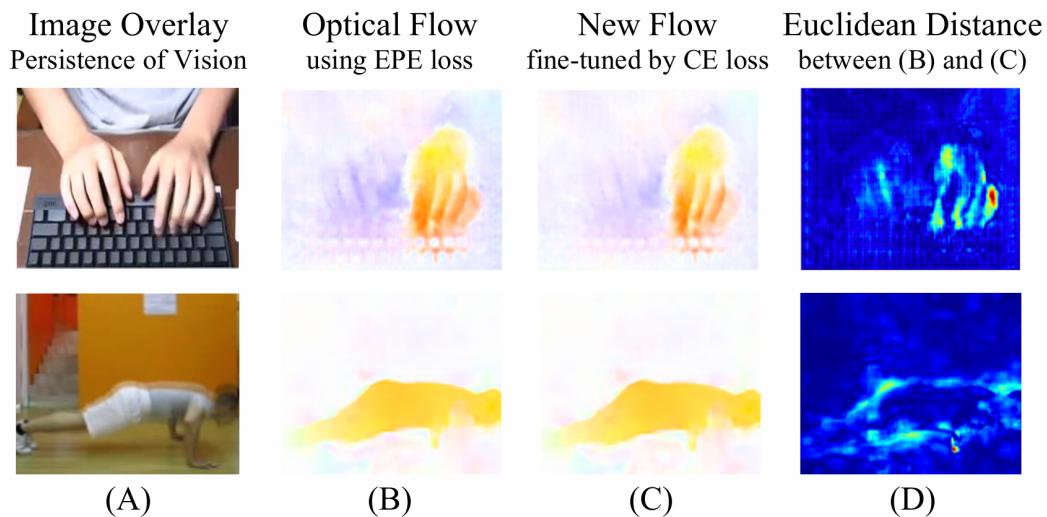
Jednym ze sposobów poradzenia sobie ze złożonością modeli było podejście symulujące przepływ optyczny (ang. Flow-mimic approach), którego główną cechą miało być znalezienie sposobu na otrzymywanie informacji o ruchu bez potrzeby obliczania przepływu optycznego. Niektóre podejścia polegały na próbie zamodelowania przepływu optycznego w sieci neuronowej, dzięki czemu był on potrzebny tylko w procesie uczenia modelu, a nie podczas korzystania z niego.

MotionNet jest siecią, która przy pomocy uczenia nienadzorowanego określa informację o ruchu. Przyjmuje ona klatki z wideo oraz na ich podstawie określa kategorie klasy, bez potrzeby obliczania przepływu optycznego.



Rys. 1.7. MotionNet [13]

PAN naśladuje cechy przepływu optycznego za pomocą różnic pomiędzy kolejnymi mapami cech. Na jego podstawie powstawały kolejne architektury wykorzystujące informacje o ruchu w danych wideo.



Rys. 1.8. PAN [15]

1.5. Przegląd metod uczenia maszynowego dla danych szeregow czasowych

Metody uczenia maszynowego opisane w rozdziale 1.2 koncentrowały się na rozpoznawaniu punktów charakterystycznych na ciele człowieka. W niniejszej pracy zostaną wykorzystane również inne rodzaje modeli, w tym modele rekurencyjne operujące na danych czasowych. Ze względu na specyfikę takich modeli oraz ich zastosowanie, wymagają one szczegółowego omówienia w osobnym, dedykowanym rozdziale.

Dodatkowo, ze względu na trudności w znalezieniu odpowiednich polskich odpowiedników terminów takich jak GRU (Gated Recurrent Unit) oraz LSTM (Long Short-Term Memory), a także biorąc pod uwagę, że tłumaczenie tych nazw w literaturze fachowej jest często uznawane za niezalecane, w niniejszej pracy zdecydowano się na używanie ich oryginalnych, anglojęzycznych nazw. Dzięki temu

zachowana zostanie spójność z terminologią powszechnie stosowaną w literaturze naukowej i technicznej.

1.5.1. Jednokierunkowa sieć neuronowa

W 1943 roku McCulloch oraz Pitts zaproponowali pierwszy matematyczny model inspirowany budową ludzkiego mózgu, który opierał się na abstrakcji działania neuronów biologicznych. Był to moment, w którym rozpoczęto badania nad sztucznymi sieciami neuronowymi (ang. artificial neural network). Są one zdolne rozwiązywać skomplikowane zadania, rozpoznając wzorce i podobieństwo w danych. Jednokierunkowa sieć neuronowa (ang. Feedforward Neural network) jest szczególnym rodzajem sztucznych sieci neuronowych, którą można przedstawić w postaci strukturalnej. Jej postać umożliwia określenia aproksymacji dowolnej funkcji ciągłej, co pozwala na modelowanie, analizowanie oraz rozwiązywanie wielu problemów z dziedzin rozpoznawania schematów, klasyfikacji, sterowania, bioinżynierii, teorii sygnałów, rozpoznawania mowy i tym podobne. [16]

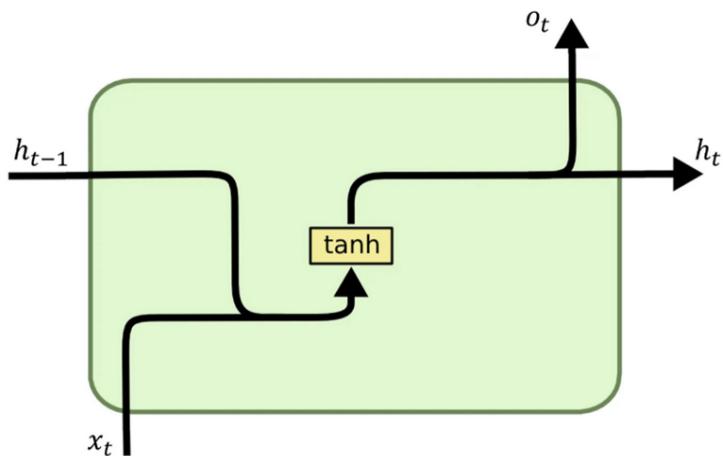
Jednokierunkowa sieć neuronowa jest podstawową siecią neuronową, na podstawie której są budowane bardziej złożone architektury. Jak wskazuje nazwa, dane wejściowe przechodzą przez poszczególne warstwy zbudowane z neuronów, które za pomocą wag, określają wynik modelu. Trenowanie takiego modelu sprowadza się do dostosowania odpowiednich wartości wag w neuronach.

1.5.2. Rekurencyjna sieć neuronowa

Rekurencyjna sieć neuronowa (ang. Recurrent neural network) – sieć neuronowa zmodyfikowana o mechanizm zapamiętywania informacji z poprzednich próbek czasu. Dzięki temu możliwe jest przewidywanie wyników na podstawie więcej niż jednego elementu ciągu. Głównym ograniczeniem sieci jest problem z przekazaniem informacji poprzez gradient w architekturach posiadających wiele warstw ukrytych:

Problem zanikającego gradientu (ang. Vanishing gradient problem) – zanikanie gradientu podczas propagacji wstecznej (ang. backpropagation), ze względu na zbyt dużą liczbę warstw ukrytych, w wyniku czego występuje problem z optymalizacją funkcji celu. [17]

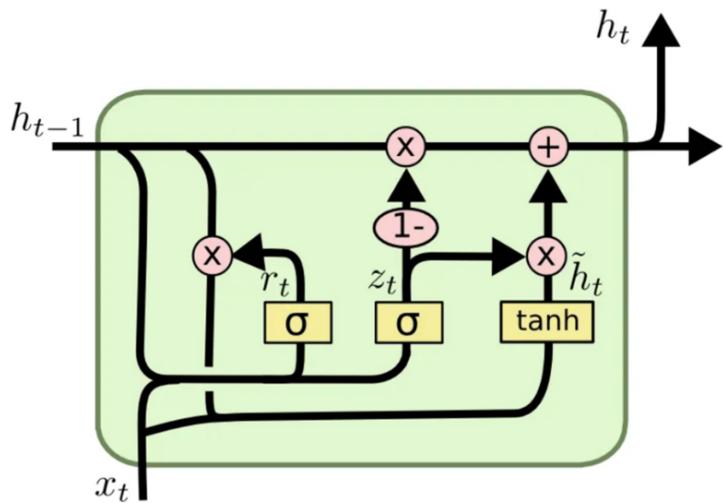
Problem eksplodującego gradientu (ang. Exploding gradient problem) – osiąganie przez gradient na tyle wysokich wartości, że zmiany w parametrach modelu są zbyt duże, żeby znaleźć minimum funkcji celu. [17]



Rys. 1.9. Schemat RNN [18]

1.5.3. GRU

Działanie tego mechanizmu potwierdza praca [17], w której udowodniono, że komórki bramkowe, pomagają w ograniczeniu problemów z zanikającym/eksploducującym gradientem. Bramki te umożliwiają modelowi skuteczne uczenie się zależności w danych sekwencyjnych poprzez dynamiczne kontrolowanie jakie informacje są zachowywane, a jakie ignorowane w każdym kroku czasowym.



Rys. 1.10. Schemat GRU [18]

1.5.4. LSTM

LSTM – Long-short term memory – jest architekturą opartą o komórki bramkowe (ang. gated units), powstały w odpowiedzi na problemy w uczeniu modeli opartych o długie próbki czasowe [17]. Tak jak wspomniano w [19], tradycyjne sieci rekurencyjne (RNN), nie dają sobie rady w operowaniu na długich

szeregach czasowych. W celu rozwiązania tego problemu stworzono LSTM, którego zaletą jest mechanizm pamięci długotrwałej, dzięki któremu modele są w stanie zapamiętywać kluczowe informacje dla dłuższych sekwencji czasowych. Warto zauważyć, że efekty zanikającego oraz eksplodującego gradientu są ograniczone, ale nie rozwiązane. LSTM będzie mógł się uczyć na dłuższych szeregach czasowych niż tradycyjne sieci rekurencyjne, nie znaczy to jednak, że dla dowolnie długiego ciągu LSTM będzie prawidłowo działał [17].

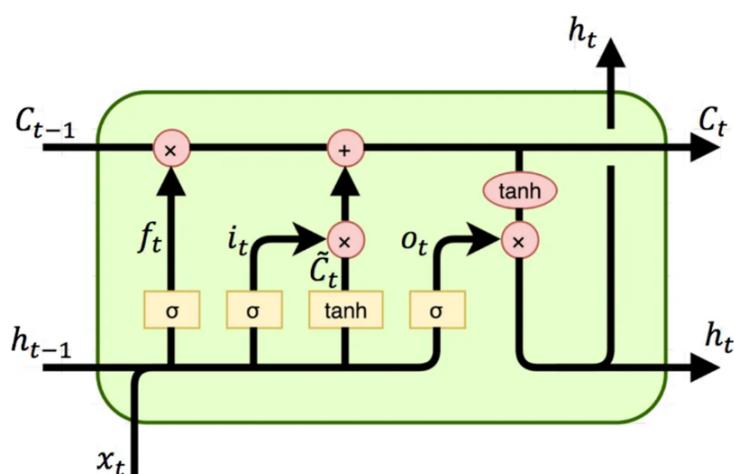
Schemat LSTM:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ u_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(W \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} \right)$$

$$c_t = i_t \cdot u_t + f_t \cdot c_{t-1}, \quad h_t = o_t \cdot \tanh(c_t)$$

gdzie:

- i_t – input gate,
- f_t – forget gate,
- o_t – output gate,
- u_t – input modulation gate,
- h_t – output state,
- c_t – internal memory,
- x_t – dane wejściowe od sieci w czasie t .



Rys. 1.11. Schemat LSTM [18]

1.5.5. Przykład zastosowania modeli rekurencyjnych

Rozpoznawanie czynności oparte o modele głębokiego uczenia głównie skupiają się na trzech rodzajach danych wejściowych: obrazy RGB, dane odległościowe oraz współrzędne szkieletu człowieka. W przeszłości przede wszystkim obrazy RGB były głównym elementem badań tego typu algorytmów. Z czasem, okazały się one mniej odporne na zmieniające się środowisko testowe (zmiana oświetlenia, różnica kolorów), niż dane oparte o sensory odległości, których dodatkową zaletą było dostarczanie dodatkowych informacji o trójwymiarowości przestrzeni. Wraz z rozwojem tych systemów oraz przystosowanym odpowiednich algorytmów, zaczęto korzystać z danych opierających się budowę ciała. Obecnie, z pomocą uczenia maszynowego w celu określenia oraz rozpoznania wykonywanej czynności, można skorzystać sieci konwolucyjnych (CNN) i rekurencyjnych (RNN). [19]

Metody oparte na sieciach konwolucyjnych są ograniczone ze względu na operowanie na obrazach 2D, tracąc w ten sposób część informacji [19]. Kolejnym ograniczeniem tego rozwiązania jest brak informacji czasowych, które mogłyby pomóc w określeniu czynności, które są niemożliwe do określenia tylko z jednego obrazu.

Wykorzystanie sekwencji czasowej jest możliwe przy wykorzystaniu modeli rekurencyjnych. Dają one możliwość do analizowania, nie tylko obecnej próbki, ale też poprzednich w celu podniesienia skuteczności działania algorytmu. Tego typu podejście zastosowano w pracy [19], gdzie wyodrębniając konkretne części ciała wytrenowano model LSTM w celu rozpoznania czynności wykonywanej przez użytkownika.

2. Kalibracja i synchronizacja kamer oraz zbieranie danych

2.1. Przygotowanie stanowiska do nagrań

Przygotowanie stanowiska jest ważnym procesem, który może mieć duży wpływ na późniejsze działanie algorytmu. Nieprawidłowe rozplanowanie ułożenia kamer może skutkować problemami z triangulacją oraz zbieraniem danych. Stanowisko powinno zapewniać użytkownikowi bezproblemowe wykonywanie ruchów rejestrowanych, jednocześnie przez wszystkie kamery. Urządzenia powinny być zamontowane stabilnie, tak aby nie poruszały się w trakcie korzystania ze stanowiska, najlepiej do tego wykorzystać statwy.

Stanowisko wykorzystane do zbierania danych składało się z dwóch urządzeń nagrywających, zlokalizowanych w dobrze oświetlonym pomieszczeniu przed osobą nagrywaną. Rejestratorami obrazu używanymi w badaniu były: smartfon iPhone oraz tablet iPad. Kąt między pierwszą kamerą, osobą rejestrowaną, a drugą kamerą wynosił w okolicach 90 stopni.

Każda z kamer była ustawiana w taki sposób, aby w centralnej części nagrania znajdowała się nagrywana osoba. Pomógł w tym stały podgląd obrazu. Poprawność ustawienia środowiska została zweryfikowana za pomocą sprawdzenia możliwości kalibracji kamer oraz triangulacji. Po otrzymaniu popranej rekonstrukcji szkieletu człowieka można było uznać przygotowanie stanowiska za zakończone.

2.2. Kalibracja kamer

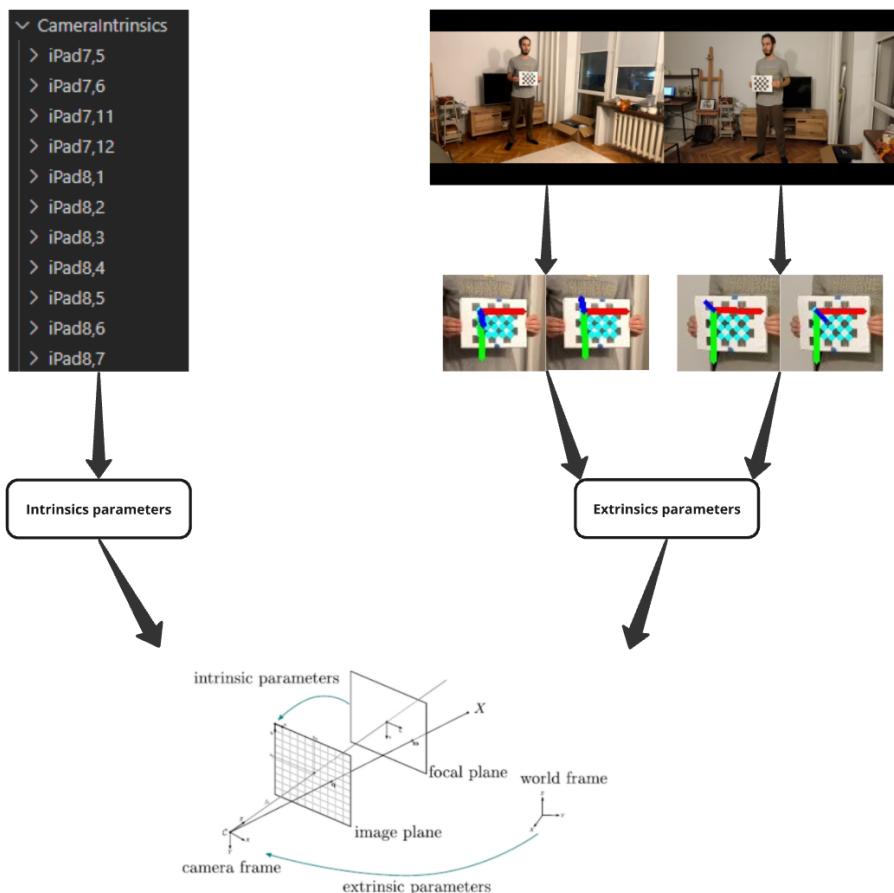
W celu przeprowadzenia popranej kalibracji kamer potrzebna jest znajomość jej charakterystyki. Każda kamera znajdująca się na stanowisku, posiada swoje specyfikacje techniczne, a informacje na ich temat znajdują się w parametrach wewnętrznych kamery. Dodatkowo są one zlokalizowane w przestrzeni trójwymiarowej, mają konkretne położenie, są często skierowane w różnych kierunkach, za te informacje odpowiadają parametry zewnętrzne. W procesie kalibracji potrzebne jest ustalenie obu rodzajów parametrów dla każdej kamery.

2.2.1. Parametry wewnętrzne i zewnętrzne

Parametry wewnętrzne: W celu ich ustalenia potrzebna jest znajomość wykorzystywanego urządzenia. Jeśli znamy konfigurację techniczną danej kamery, jesteśmy w stanie ustalić parametry wewnętrzne.

Rozwiązań korzysta z zebranych przez [20] parametrów dla dużej liczby telefonów i tabletów firmy Apple.

Parametry zewnętrzne: Do określenia parametrów zewnętrznych zachodzi potrzeba skorzystania z obiektu, który będzie dobrze widoczny dla wszystkich kamer. Zastosowano w tym celu obraz szachownicy. Istotne jest, aby podczas kalibrowania, szachownica znajdowała się w tym samym miejscu w przestrzeni dla wszystkich obrazów wykorzystywanych do kalibracji kamer. Następnie odpowiednie algorytmy wyodrębniają istotne cechy, które są zapisywane jako parametry zewnętrzne.



Rys. 2.1. Proces kalibracji kamer [21]

Po ukończeniu kalibracji na podstawie [20], tworzony jest plik `sessionMetadata.yaml`, który przechowuje informacje na temat użytkownika, urządzeń wykorzystanych do nagrywania i szachownicy, między innymi rozmiar oraz ilość kwadratów. Będzie on wykorzystywany do uruchomienia programu bez potrzeby ponownej kalibracji. Pomimo tego, że część parametrów jest niewykorzystywana, zostały one zachowane z myślą o rozwoju rozwiązania w przyszłości.

```
mediapipe > init > ! sessionMetadata.yaml
1  augmentermodel: v0.3
2  calibrationSettings:
3    | overwriteDeployedIntrinsics: false
4    | saveSessionIntrinsics: false
5  checkerBoard:
6    black2BlackCornersHeight_n: 4
7    black2BlackCornersWidth_n: 5
8    placement: backWall
9    squareSideLength_mm: 35.0
10   filterfrequency: default
11   gender_mf: m
12   height_m: 1.85
13   iphoneModel:
14     Cam0: iPhone8,1
15     Cam1: iPhone14,7
16   markerAugmentationSettings:
17     markerAugmenterModel: LSTM
18   mass_kg: 76.0
19   openSimModel: LaiUhlrich2022
20   posemodel: hrnet
21   subjectID: Jakub
```

Rys. 2.2. Zawartość pliku sessionMetadata

2.3. Synchronizacja kamer

Algorytm wykorzystuje kamery z telefonów, z tego powodu wystąpiła potrzeba przekazania synchronizowanego obrazu do komputera. W tym celu wykorzystano programu Iriun Webcam, który pozwala na udostępnianie widoku z kamer w urządzeniach podłączonych do tej samej sieci Wi-Fi. Można za jego pomocą wykorzystać do 4 kamer. Zwiększenie tej liczby może wymagać przygotowana dedykowanego oprogramowania.

2.3.1. Cele synchronizacji

W celu działania programu w czasie rzeczywistym konieczne jest wysyłanie obrazu z wielu kamer, aby zastosować triangulacje. Z tego powodu program w danym momencie przechwytuje obraz z wszystkich kamer, przełączając się pomiędzy nimi, a zbiór obrazów przesyłany jest do dalszej części kodu.

2.3.2. Badanie wydajności synchronizacji

Z powodu wielu operacji występujących podczas wykonywania programu, takich jak triangulacja, estymacja pozycji dla każdej kamery oraz wyświetlenia wyników, synchronizacja mimo poprawnego działania nie ma dużej wydajności. Program osiąga w okolicach 10 FPS przy bardziej zoptymalizowanych modelach oraz 5 przy dokładniejszych. Jest ona wystarczająca do odróżnienia między sobą konkretnych czynności, pojawia się tutaj jednak potencjał do poprawienia tych wyników.

3. Przygotowanie i przetwarzanie danych

3.1. Zbieranie danych

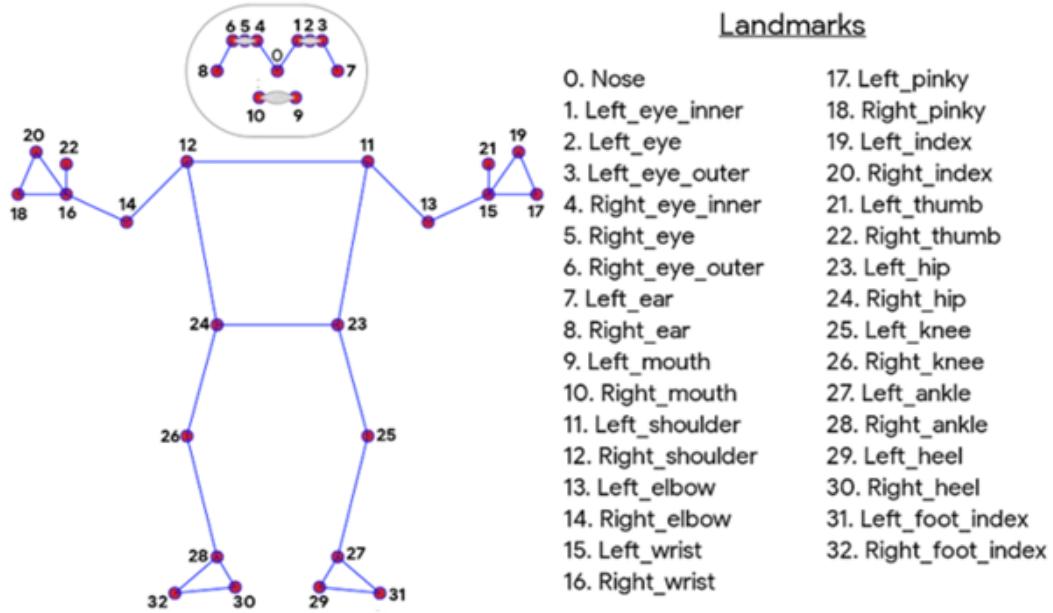
Do nauczenia modeli rekurencyjnych potrzebne są duże zbiory danych. W tym przypadku to dane zawierające przebiegi czasowe wykonywanych czynności, reprezentowane jako przebiegi współrzędnych 2D punktów kluczowych – zebranych za pomocą modelu estymacji pozycji oraz 3D – zebranych za pomocą triangulacji punktów 2D na 3D. Podczas zbierania danych wykonywane były różne czynności, które następnie były przypisywane do odpowiednich kategorii. W ten sposób udało się zebrać 2 zestawy danych:

- mniejszy (5000 klatek treningowych, 5000 klatek walidacyjnych) - wykorzystany do eksploracyjnego uczenia na podzbiorze danych w celu określenia odpowiednich parametrów modelu rekurencyjnego, jednocześnie oszczędzając czas wymagany na uczeniu modelu przy użyciu większej liczby danych,
- większy (20709 klatek treningowych, 5000 klatek walidacyjnych) - w celu trenowania finalnego modelu na podstawie parametrów dobranych za pomocą mniejszego zbioru danych

3.1.1. Przygotowanie algorytmu do zbierania danych 2D za pomocą BlazePose

W celu określenia lokalizacji konkretnych punktów na ciele człowieka wykorzystano model estymacji pozycji. Dla każdego obrazu dostarczonego przez program, uruchomiony został model BlazePose, który charakteryzuje się lokalizowaniem 33 punktów kluczowych. Model występuje w 3 wersjach Lite, Full i Heavy, które różnią się prędkością oraz dokładnością detekcji.

W tym przypadku zdecydowano się na wersje Heavy, ze względu na największą dokładność działania, która musi być zapewniona, aby zebrane dane były wysokiej jakości. Model Lite, mimo największej wydajności, często niedokładnie określał charakterystyczne punkty na ciele, co prowadziło do błędnych wyników rekonstrukcji trójwymiarowej.



Rys. 3.1. Punkty kluczowe modelu BlazePose [9]

3.1.2. Triangulacja danych 2D na dane 3D

Do triangulacji wykorzystano parametry uzyskane podczas kalibracji kamer oraz wyniki działania modelu estymacji pozycji. Kod do triangulacji jest oparty o [22], który jest częścią projektu OpenCap. Po triangulacji uzyskane punkty, które w zależności od lokalizacji oraz parametrów kamer mogą osiągać różne zakresy. O ile nie jest to problemem w przypadku wizualizacji punktów, może być to duże utrudnienie w procesie uczenia modeli, które dla identycznych czynności będą otrzymywać zupełnie inne współrzędne. W tym celu powstała potrzeba normalizacji danych do pewnego, stałego zakresu. Aby pozbyć się tego problemu, zanim zaczęto zbierać dane, wykonano dodatkową kalibrację polegającą na przyjęciu i zarejestrowaniu ‘neutralnej pozy’. Dzięki takiej operacji możliwe będzie określenie maksymalnych przedziałów liczbowych dla wyświetlanych punktów, a następnie podzielenie odpowiadających ich współrzędnych w celu normalizacji do przedziałów od 0 do 1. Dane nie tylko stają się bardziej przejrzyste, ale również zrozumiałe dla modeli rekurencyjnych. Kalibracja ma miejsce na początku zbierania danych, dla jednej pozycji, następnie wartości mogą przekraczać zakres 0 - 1, ponieważ osoba nagrywana będzie się przemieszczać po pomieszczeniu.

Poza problemem związanym z zakresami wartości punktów, wystąpił również problem z ich orientacją w przestrzeni. Zjawisko polegało na obróceniu szkieletu względem osi x, y lub z. W ten sposób osoba, która w rzeczywistości stała pionowo, mogła wyglądać na rekonstrukcję jakby leżała płasko na jakiejś powierzchni. W rozwiązaniu tego problemu również skorzystano z ‘neutralnej pozy’. W zależności od lokalizacji punktów kluczowych względem siebie zostały określone odpowiednie operacje modyfikujące punkty. Przykładowo, jeśli punkt kluczowy określający lokalizację nosa, będzie miał mniejszą wartość współrzędnej "z" od wartości współrzędnej "z" punktu kluczowego reprezentującego lokalizację

pięty, oznacza to, że szkielet jest odbiciem lustrzanym względem płaszczyzny x-y lub jest obrócony o 180 stopni względem osi "z".

Po wprowadzeniu tych zmian triangulacja działała poprawnie, o ile osoba znajdowała się w obszarze nagrywania, przynajmniej dwóch kamer. W innym przypadku rekonstrukcja jest niemożliwa

3.2. Zapoznanie się z danymi i ich struktura

Dane zostały zebrane dla przykładowej sekwencji czynności. W tym przypadku zasymulowany był proces pakowania paczek przez pracownika magazynu. W tym celu osoba nagrywana powtarzała proces składający się z pięciu czynności:

- sięganie po karton
- rozpakowywanie kartonu
- pakowanie kartonu
- odkładanie kartonu
- inne - żadna z powyższych

Warto zauważyć, że brak kontekstu czasowego uniemożliwia określenie niektórych czynności. Pojedyncza klatka, w której osoba trzyma karton, może być zarówno sięganiem po niego jak i jego odkładaniem, podobnie z pakowaniem i rozpakowywaniem. Z tego właśnie powodu konieczne będzie skorzystanie z modeli rekurencyjnych, które pozwolą wyciągnąć potrzebne informacji z kolejnych klatek filmu.

Dane 2D zawierają informacje o współrzędnych x i y każdego z 33 punktów kluczowych, ich wielkość zależy od liczby kamer, ponieważ zbierane są punkty dla każdej perspektywy. Model estymacji pozycji - BlazePose, wykorzystany do lokalizacji punktów na ciele człowieka, zwraca je jako procent szerokości obrazu (osi x) lub wysokości obrazu (osi y). Wartości te są następnie dostosowywane do rozdzielczości kamery, dzięki czemu otrzymywane są konkretne piksele, w których znajduje się środek zlokalizowanego elementu.

X_0_cam0	Y_0_cam0	X_0_cam1	Y_0_cam1	X_1_cam0	Y_1_cam0
60.121188	124.111080	198.868580	97.322121	59.632320	112.731609
109.100714	130.206661	200.692673	104.140835	107.172003	118.396416
152.693806	134.187279	225.636883	126.796350	149.645824	121.043682
209.673557	143.754139	259.847298	134.082184	207.140770	132.342072
265.107365	142.201195	281.061954	141.297474	263.207760	131.291924

Rys. 3.2. Dane 2D wykorzystane do triangulacji

Dane 3D zawierają informacje o współrzędnych x, y i z każdego z 33 punktów kluczowych, ich wielkość jest stała, ponieważ niezależnie od ilości kamer, wykorzystywane jest odwzorowanie za pomocą triangulacji. Wartości po przeskalowaniu, niezależnie od kalibracji, powinny być:

- < 0 - jeśli punkty znajdują się niżej od minimalnej wartości punktów kluczowych w 'neutralnej pozie'
- $(0 - 1)$ - jeśli wartość punktów zawiera się w przedziale między maksymalną, a minimalną wartością punktów kluczowych w 'neutralnej pozie'
- > 1 - jeśli punkty znajdują się powyżej maksymalnej wartości punktów kluczowych w 'neutralnej pozie'

X_0	Y_0	Z_0	X_1	Y_1	Z_1	X_2	Y_2	Z_2	X_3	...
1.365182	0.192908	0.818807	1.377669	0.195188	0.820432	1.371873	0.196587	0.822049	1.368904	...
1.389916	0.191036	0.813465	1.405227	0.192856	0.814365	1.401851	0.193930	0.815385	1.399036	...
1.423944	0.188148	0.797858	1.440570	0.190160	0.798602	1.438390	0.191275	0.799438	1.435256	...
1.410974	0.186051	0.788210	1.418296	0.187452	0.790803	1.413732	0.188313	0.792098	1.409785	...
1.403729	0.186978	0.788912	1.410188	0.188412	0.792112	1.406336	0.189178	0.793506	1.403209	...

Rys. 3.3. Dane 3D wykorzystane do uczenia modeli

3.3. Przygotowanie danych do analizy i modelowania

Podczas tworzenia modelu rekurencyjnego będzie potrzebny zbiór treningowy, na którym zostanie nauczony model do rozpoznawania czynności. Zbiór walidacyjny będzie służył jako wyznacznik tego, jak model działa na danych, których nigdy nie widział, co dostarczy istotnych informacji o jego skuteczności w docelowym zastosowaniu.

Zebrane zostały dwa zestawy danych w formie szeregów czasowych:

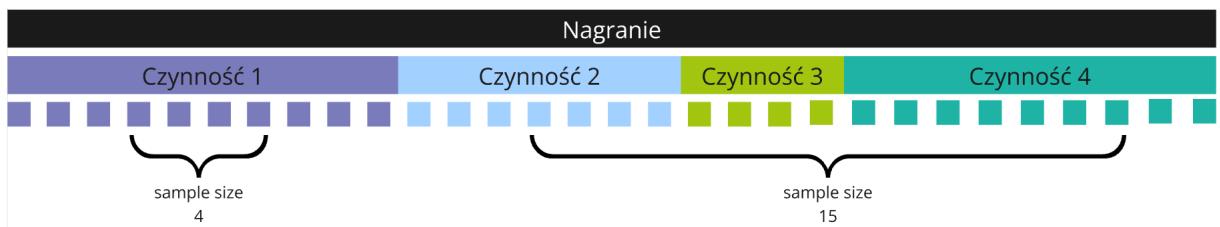
- 5000 klatek treningowych, 5000 klatek walidacyjnych
- 20709 klatek treningowych, 5000 klatek walidacyjnych

Dane muszą posiadać pewną strukturę, modele rekurencyjne przyjmują informacje z określonego przedziału czasowego. W tym przypadku będzie to jednoznaczne z przekazaniem przebiegów punktów kluczowych dla kolejnych klatek nagrania. Długość tego przedziału jest jednym z hiperparametrów, który będzie określony podczas budowy modelu i nazywany jako sample size. Jak widać na schemacie 3.4, w zależności od doboru tego hiperparametru można spodziewać się różnych podziałów danych. Ustawienie sample size, na zbyt niską wartość może powodować dostarczeniem zbyt małej ilości informacji czasowych, aby model mógł poprawnie określić wykonywaną czynność. Zbyt duże wartości mogą za to niepotrzebnie przekazywać informacje, które już nie mają znaczenia, a jednocześnie znacząco przedłużają proces uczenia modeli.

Wielkość zbiorów danych będzie się między sobą różnić w zależności od sample size, według wzoru:

$$d = n - s + 1$$

- d – wielkość zbioru danych do uczenia modelu,
- n – całkowita liczba zdjęć w zbiorze,
- s – wielkość próbki czasowej sample size



Rys. 3.4. Schemat danych

4. Uczenie modeli detekcji czynności

Ten rozdział skupia się na procesie uczenia modeli oraz porównywaniu ich wyników w zależności od dobranych parametrów, analizując, jak zmienne te wpływają na efektywność i dokładność działania. Dodatkowo analizowana będzie prędkość uczenia modeli oraz generowania odpowiedzi, aby ocenić ich wydajność w różnych konfiguracjach.

4.1. Implementacja modeli

Podczas uczenia modeli rekurencyjnych dobierane są parametry, które będą definiowały ich strukturę oraz determinowały ostateczne wyniki. W celu automatyzacji niektórych procesów przygotowany został kod uczący jednocześnie wiele modeli.

```
train_model(X, Y, X_test, Y_test, model_type, epochs, sample_size,  
           hidden_units, seed)
```

- X, Y - dane treningowe, na podstawie których model będzie uczyony
- X_test, Y_test - dane walidacyjne, wykorzystywane do określenia poprawności działania modelu
- model type – typ modelu rekurencyjnego [Simple RNN, GRU, LSTM]
- epochs – liczba epok decydująca o długości uczenia modelu
- sample size – długość próbki czasowej, odpowiada liczbie klatek, którą będzie brana pod uwagę, podczas decyzji o klasyfikacji czynności
- hidden units – parametr bloku rekurencyjnego, określający możliwość sieci do przechowywania informacji na temat konkretnego kroku czasu
- seed - pozwala na otrzymywanie powtarzalnych wyników poprzez ustalenie stałego sposobu losowania początkowych parametrów sieci oraz rozkładu danych

Dzięki standaryzacji danych model może łatwiej optymalizować gradient, co ułatwia znalezienie lepszego dopasowania do danych. Parametry standaryzacji dla **StandardScaler** są określone na zbiorze treningowym, natomiast standaryzacja jest przeprowadzona zarówno dla danych treningowych jak i testowych.

```
scaler = StandardScaler()
X = scaler.fit_transform(X)
X_test = scaler.transform(X_test)
```

Każdy model posiada funkcję celu **sparse_categorical_crossentropy**, która jest używana w problemach klasyfikacji wieloklasowej, model będzie się starał ją optymalizować podczas uczenia. Wykorzystywaną metryką jest **accuracy**, dostarczy ona informacji o dokładności modelu. Funkcją aktywacji jest **tanh** ze względu na jej zdolność do mapowania wartości wejściowych na symetryczny zakres od -1 do 1.

```
model = Sequential([LSTM(hidden_units, input_shape=(sample_size, 99),
activation='tanh'), Dense(5, activation='softmax')])
model.compile(loss="sparse_categorical_crossentropy",
optimizer='adam', metrics=["accuracy"])
```

Po przemieszaniu danych w celu zapewnienia losowego rozkładu za pomocą parametru **shuffle**, można przejść do uczenia modeli.

```
model.fit(X, Y, validation_data=(X_test, Y_test), epochs=epochs,
shuffle=True)
```

4.2. Dobór długości próbek, liczby epok, parametrów modelu

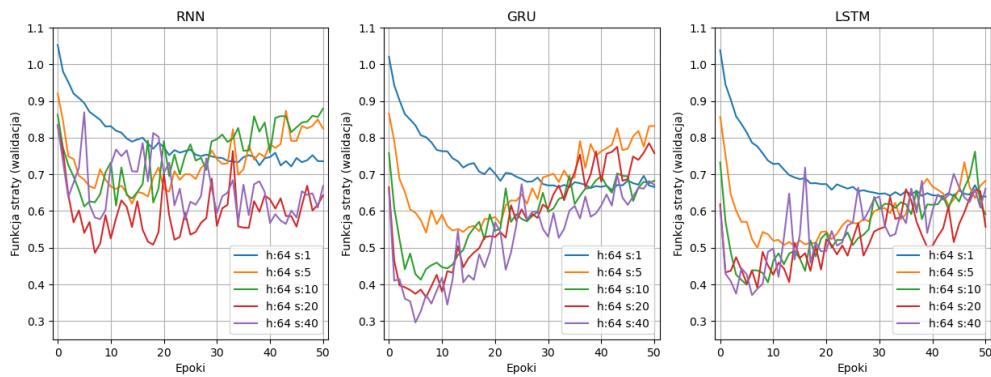
Ten rozdział skupia się na określeniu w jaki sposób trenować model, aby minimalizować jego funkcję celu na danych testowych. Dla każdego modelu sprawdzona została kombinacja różnych wartości parametrów, których wyniki można zobaczyć na wykresach oraz w tabelach. Rysunki 4.1 oraz 4.3 przedstawiają wykresy funkcji celu, a 4.2 oraz 4.4 dokładność modeli RNN, GRU oraz LSTM, na danych walidacyjnych dla różnych kombinacji parametrów. Wszystkie modele były uczone na takiej samej liczbie epok, równej 100. Wartość ta nie jest optymalna do trenowania docelowego modelu, ponieważ może prowadzić do przeuczenia (ang. overfitting), pozwala ona za to zaobserwować proces uczenia oraz znalezienia odpowiedniej liczby epok, a także w przejrzysty sposób porównać wiele modeli jednocześnie.

W większości przypadków dla tych samych wartości parametrów najlepiej z rozpoznawaniem konkretnych czynności poradziły sobie modele GRU, osiągając najlepsze wyniki, wyprzedzając między innymi modele LSTM. Można też zauważyc chaotyczne przebiegi uczenia dla podstawowych modeli RNN, które mają największe problemy z optymalizowaniem funkcji celu. Może to wynikać ze zbyt prostej architektury do zaproponowanego problemu lub problemów z gradientem.

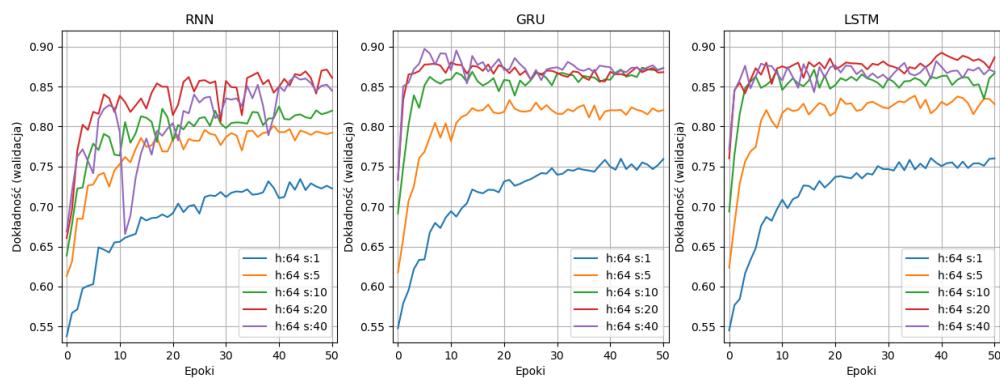
Wykresy 4.1, 4.2 skupiają się na wpływie długości danych czasowych na wyniki modeli. Najgorzej radzące sobie modele to podstawowe RNN, osiągają one największe wartości funkcji celu oraz najniższe wartości skuteczności dla danych walidacyjnych.

Znacząca część modeli ma tendencje do osiągania najniższych wartości funkcji celu na początku procesu uczenia. Następnie wartości zaczynają znacznie wzrastać, najprawdopodobniej wynika to z występowania nadmiernego dopasowania się modeli do danych treningowych, co skutkuje brakiem zrozumienia ogólnych wzorców oraz możliwości wykonywania prawidłowych predykcji na wcześniej niewidzianych danych.

Niskie wartości parametru **sample state** powodują wolniejszy spadek funkcji celu w pierwszych epokach uczenia, co może wskazywać na powiązanie między wzrostem tego parametru, a przyspieszeniem przeuczenia. Podobnie w przypadku dokładności najszybciej osiąga swoje maksymalne wartości wraz ze wzrostem tego parametru.

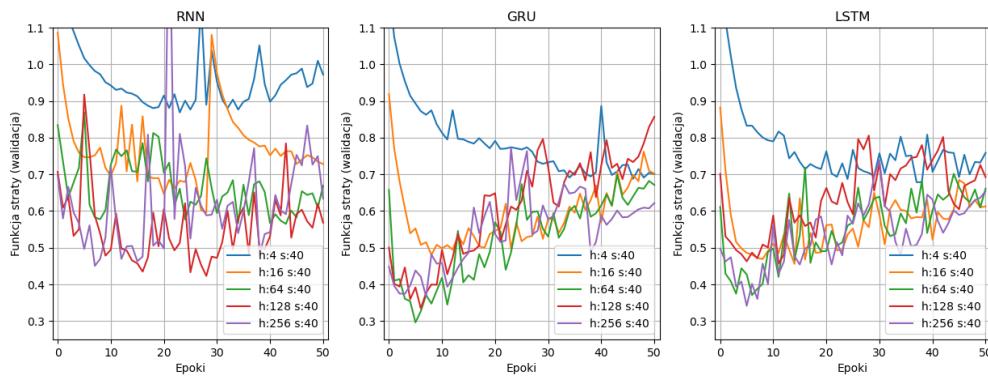


Rys. 4.1. Przebieg funkcji celu - sample size

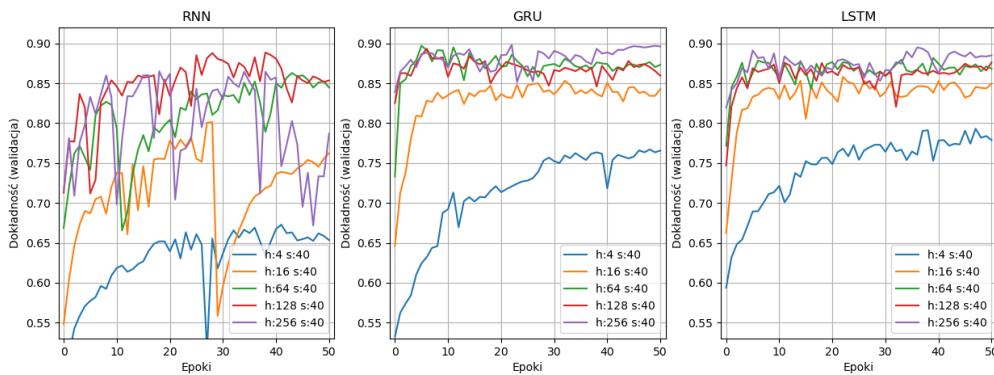


Rys. 4.2. Przebieg dokładności - sample size

Dla porównania wykresy 4.3, 4.4, większą wagę przykładają do analizy parametru **hidden units**, który określa wewnętrzną złożoność sieci oraz umożliwia uczenie jej bardziej skomplikowanych zagadnień. Rosnąca wartość parametru prowadzi do szybszego osiągania minimalnych wartości funkcji celu w początkowej fazie uczenia. Jednocześnie niskie wartości tworzą sieci o zbyt małej złożoności, żeby wyciągnąć wszystkie istotne informacje ze zbiorów uczących.



Rys. 4.3. Przebieg funkcji celu - hidden units



Rys. 4.4. Przebieg dokładności - hidden units

Ilość zebranych danych jest zbyt duża, aby można było je przedstawić na pojedynczym wykresie. Każdy model posiada własny podrozdział, w którym będzie można zapoznać się z dokładniejszą analizą wyników.

4.2.1. Wyniki dla RNN

Model RNN jest najprostszym z użytych modeli rekurencyjnych, można spodziewać się po nim najgorszych wyników dla długich szeregów czasowych. Dane zebrane w procesie uczenia znajdują się w tabelach 4.1, 4.2, 4.3, 4.4.

Jak można zauważyć w tabeli 4.1, dla zbyt niskich wartości hidden units, model RNN ma trudności optymalizować funkcje celu. Może to wynikać ze zbyt małej złożoności modeli dla problemu, który ma zostać rozwiązany. Dane treningowe najlepiej są optymalizowane dla najwyższej wartości hidden units oraz długości próbki z przedziału 5-15, gdzie modele zredukowały funkcje celu niemal do zera. Fakt, że dla dłuższych próbek model nie był w stanie tego osiągnąć, może wynikać z problemu zanikania gradientów. Są to wyniki dla danych treningowych, więc szukanie optymalnych modeli tylko na tych

danych jest niewystarczające, daje to jednak informacje dla jakich parametrów model jest w stanie w rozsądnym czasie nauczyć się danych wzorców, a jakie parametry uniemożliwią uczenie modelu.

Tabela 4.2 zawiera informacje o wynikach dla danych walidacyjnych. Wiele obserwacji wynikających z modeli uczonych na danych treningowych pokrywa się. Zwiększenie wartości hidden units poprawia skuteczność modeli do pewnego momentu, po którym korzyści stają się marginalne lub żadne. W tym przypadku można zauważać, że dla wysokiej liczby jednostek ukrytych modele o najdłuższych próbkach osiągały najwyższe wyniki.

Tabela 4.1. RNN - wartość funkcji celu dla danych treningowych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.782	0.649	0.530	0.419	0.331	0.287	0.261
3	0.713	0.518	0.326	0.151	0.037	0.007	0.005
5	0.663	0.458	0.248	0.083	0.013	0.001	0.000
10	0.646	0.443	0.148	0.033	0.004	0.000	0.000
15	0.617	0.398	0.132	0.043	0.014	0.001	0.000
20	0.592	0.351	0.134	0.058	0.013	0.001	0.004
40	0.546	0.337	0.206	0.072	0.027	0.020	0.044

Tabela 4.2. RNN - wartość funkcji celu dla danych walidacyjnych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.871	0.810	0.753	0.725	0.718	0.717	0.730
3	0.838	0.771	0.716	0.633	0.619	0.597	0.603
5	0.806	0.738	0.697	0.617	0.620	0.560	0.563
10	0.817	0.813	0.657	0.550	0.612	0.515	0.513
15	0.850	0.882	0.600	0.498	0.574	0.469	0.463
20	0.864	0.858	0.559	0.531	0.486	0.476	0.451
40	0.869	0.895	0.596	0.520	0.565	0.423	0.451

W przypadku dokładności modeli dla danych treningowych przy małej liczbie jednostek ukrytych wraz ze wzrostem długości próbki zwiększała się skuteczność modeli. Efekt ten przestał być widoczny dla wyższych wartości hidden units, gdzie długość próbek z zakresu 3 - 20 jest w stanie niemal za każdym razem prawidłowo kategoryzować przykłady.

Wyniki dla danych testowych pokazują, że każdorazowe zwiększenie wartości sample size od 1 do 20, w modelach z liczbą ukrytych jednostek większą lub równą 16, poprawia dokładność modelu. Metryka, która została użyta potwierdza przesłanki wskazujące na to, że modele RNN najlepiej radzą sobie przy długich sekwencjach czasowych oraz średnich lub dużych wartościach hidden units.

Tabela 4.3. RNN - dokładność dla danych treningowych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.670	0.735	0.792	0.840	0.879	0.895	0.904
3	0.707	0.797	0.880	0.947	0.993	1.000	0.999
5	0.739	0.826	0.916	0.974	0.999	1.000	1.000
10	0.755	0.840	0.953	0.991	1.000	1.000	1.000
15	0.752	0.864	0.958	0.987	0.998	1.000	1.000
20	0.759	0.879	0.957	0.982	0.998	1.000	1.000
40	0.808	0.883	0.931	0.978	0.993	0.994	0.986

Tabela 4.4. RNN - dokładność dla danych walidacyjnych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.646	0.684	0.714	0.740	0.755	0.762	0.760
3	0.666	0.705	0.753	0.790	0.790	0.801	0.802
5	0.691	0.734	0.775	0.795	0.801	0.822	0.830
10	0.692	0.715	0.791	0.835	0.836	0.858	0.865
15	0.686	0.715	0.818	0.846	0.859	0.871	0.875
20	0.672	0.726	0.838	0.852	0.877	0.886	0.887
40	0.681	0.730	0.801	0.867	0.864	0.889	0.866

Na podstawie zebranych danych, najlepszym modelem RNN okazał się ten o parametrach: sample size = 40, hidden units = 128, osiągając nie tylko najniższą wartość funkcji celu równą 0.423, ale również najwyższą wartość dokładności na poziomie 88.9%, na danych walidacyjnych. Jest on kandydatem do bycia modelem wykorzystanym w aplikacji.

4.2.2. Wyniki dla GRU

Dla modeli GRU funkcja celu osiąga wartość niemal równą zero dla długości próbek dłuższych od 3 oraz liczbie jednostek ukrytych większej niż 128. Podobnie jak dla RNN, wynika to z przeuczenia modeli dla dużej liczby epok, tutaj jednak można zauważać o wiele większy wpływ zwiększania sample size na poprawę funkcji celu. Wynika to prawdopodobnie z umiejętności lepszego radzenia sobie z

dłuższymi szeregami czasowymi, dzięki ograniczenia problemu zanikającego/eksplodującego gradientu przez architekturę GRU, wpływa to jednoznacznie na poprawę wyciągania cennych informacji dla dłuższych szeregów czasowych. Warto też zauważyć, że sample size ma mniejszy wpływ na sieci o większej złożoności.

Dla danych walidacyjnych, modele o największych wartościach sample size nie osiągają zawsze najlepszych wyników, może to wynikać z paru powodów. Modele mogły otrzymać zbyt mało danych, aby dostosować wszystkie parametry bardziej złożonych modeli oraz nauczyć się relacji wynikających z dłuższych szeregów czasowych, takich jak przy sample size = 40, za to dla wartości 1-5 zawierały zbyt mało informacji potrzebnych do odróżnienia podobnych czynności. Z kolei modele o zbyt małych wartościach hidden state mogą nie być w stanie zrozumieć tak długich ciągów czasowych i ich wyniki pozostawałyby niskie niezależnie od zwiększającej się liczby danych.

Tabela 4.5. GRU - wartość funkcji celu dla danych treningowych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.619	0.487	0.380	0.311	0.252	0.218	0.197
3	0.489	0.304	0.138	0.051	0.014	0.007	0.001
5	0.368	0.202	0.065	0.011	0.002	0.000	0.000
10	0.272	0.095	0.018	0.003	0.000	0.000	0.000
15	0.282	0.085	0.014	0.001	0.000	0.000	0.000
20	0.278	0.064	0.016	0.002	0.000	0.000	0.000
40	0.263	0.068	0.016	0.002	0.001	0.000	0.000

Tabela 4.6. GRU - wartość funkcji celu dla danych walidacyjnych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.784	0.740	0.662	0.665	0.657	0.678	0.686
3	0.771	0.687	0.588	0.576	0.568	0.584	0.579
5	0.724	0.640	0.545	0.535	0.541	0.551	0.516
10	0.631	0.569	0.444	0.429	0.413	0.449	0.412
15	0.673	0.514	0.430	0.381	0.384	0.391	0.394
20	0.671	0.427	0.430	0.362	0.365	0.387	0.362
40	0.688	0.470	0.480	0.304	0.296	0.335	0.367

Dokładność modeli GRU dla danych treningowych, podobnie jak dla funkcji celu, sygnalizuje lepsze wyniki dla wyższych wartości sample size. Ponownie, wyniki mogą wynikać z przeuczenia. Dla

porównania modele RNN często cechowała niższą skuteczność dla tej samej liczby epok. Warto zwrócić uwagę na lokalizacje najwyższych wartości dokładności modelu oraz w jaką stronę jest skierowany ten wzrost. Na podstawie tej tabeli można uznać, że zarówno parametr simple size oraz hidden units są wprost proporcjonalne do poprawy wyników modeli GRU.

W modelu GRU wyniki dokładności dla zbioru testowego osiągają wartości przekraczające 90%. Wskazuje to na bardzo wysoką skuteczność, o ile dane walidacyjne dobrze reprezentują rozwiązywany problem oraz różnią się od danych treningowych.

Tabela 4.7. GRU - dokładność dla danych treningowych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.750	0.813	0.857	0.886	0.908	0.920	0.929
3	0.817	0.892	0.956	0.986	0.997	0.999	1.000
5	0.876	0.933	0.981	0.999	1.000	1.000	1.000
10	0.912	0.968	0.998	1.000	1.000	1.000	1.000
15	0.909	0.973	0.999	1.000	1.000	1.000	1.000
20	0.901	0.979	0.997	1.000	1.000	1.000	1.000
40	0.904	0.978	0.998	1.000	1.000	1.000	1.000

Tabela 4.8. GRU - dokładność dla danych walidacyjnych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.689	0.723	0.760	0.769	0.776	0.776	0.778
3	0.733	0.761	0.804	0.810	0.818	0.821	0.825
5	0.758	0.790	0.816	0.822	0.835	0.841	0.844
10	0.800	0.810	0.852	0.867	0.882	0.879	0.886
15	0.783	0.838	0.856	0.881	0.887	0.886	0.900
20	0.787	0.863	0.858	0.881	0.880	0.901	0.899
40	0.770	0.849	0.853	0.900	0.897	0.893	0.907

Model GRU o parametrach sample size = 40, hidden units = 64 osiągnął najniższą wartość funkcji celu dla danych walidacyjnych równą 0.296. Pomimo gorszych osiągnięć dla metryki zostaje on wybrany ze względu na dwa główne powody. Pierwszy powód to fakt, że model osiąga dokładność na poziomie 89.7%, czyli tylko 0.1 punktu procentowego mniej od modelu o najwyższej wartości. Po drugie, istotniejsze w tym problemie jest wybieranie modeli o niższych wartościach funkcji celu, ponieważ nie mówi ona tylko o poprawności klasyfikacji, ale też o pewności predykcji. Z tych powodów będzie on brany pod uwagę podczas wyboru modelu wykorzystywanego w aplikacji.

4.2.3. Wyniki dla LSTM

Wyniki dla modeli LSTM są podobne do wyników modeli GRU, występuje jednak parę różnic. Podobnie jak modele GRU, LSTM optymalizują funkcje celu dla danych treningowych lepiej przy dużych wartościach sample size. Podobnie jak dla GRU, wartości sample size oraz hidden units są niemal wprost proporcjonalne do poprawy wyników modeli.

Modele LSTM osiągają zbliżone wartości funkcji celu do GRU. Jeśli mowa o wartościach minimalnych, modele GRU zdołały lepiej ją zminimalizować w przypadku danych walidacyjnych.

Tabela 4.9. LSTM - wartość funkcji celu dla danych treningowych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.679	0.509	0.381	0.297	0.231	0.192	0.167
3	0.473	0.283	0.141	0.053	0.019	0.014	0.003
5	0.355	0.161	0.065	0.017	0.006	0.003	0.000
10	0.260	0.066	0.018	0.003	0.001	0.000	0.000
15	0.231	0.052	0.018	0.002	0.000	0.000	0.000
20	0.178	0.040	0.014	0.002	0.001	0.000	0.000
40	0.140	0.055	0.012	0.003	0.001	0.000	0.000

Tabela 4.10. LSTM - wartość funkcji celu dla danych walidacyjnych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.843	0.719	0.652	0.646	0.635	0.639	0.654
3	0.750	0.681	0.616	0.579	0.552	0.571	0.583
5	0.701	0.601	0.539	0.531	0.500	0.495	0.517
10	0.631	0.487	0.463	0.442	0.399	0.415	0.429
15	0.591	0.507	0.407	0.435	0.385	0.380	0.387
20	0.554	0.430	0.371	0.450	0.391	0.446	0.407
40	0.679	0.547	0.456	0.445	0.371	0.457	0.342

Podobnie jak w przypadku modeli GRU, LSTM osiąga w okolicach 100% skuteczności na danych treningowych dla wysokich wartości parametrów.

Dane walidacyjne wskazują, że model LSTM niemal za każdym razem osiąga najlepsze wyniki dla sample size = 20 w porównaniu do innych kombinacji parametrów oraz osiąga gorsze wyniki dla sample size = 40 od modeli GRU.

Tabela 4.11. LSTM - dokładność dla danych treningowych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.697	0.805	0.856	0.893	0.921	0.933	0.941
3	0.820	0.897	0.951	0.985	0.996	0.997	0.999
5	0.870	0.943	0.981	0.996	0.999	1.000	1.000
10	0.916	0.981	0.997	1.000	1.000	1.000	1.000
15	0.929	0.984	0.997	1.000	1.000	1.000	1.000
20	0.945	0.988	0.998	1.000	1.000	1.000	1.000
40	0.958	0.983	0.998	1.000	1.000	1.000	1.000

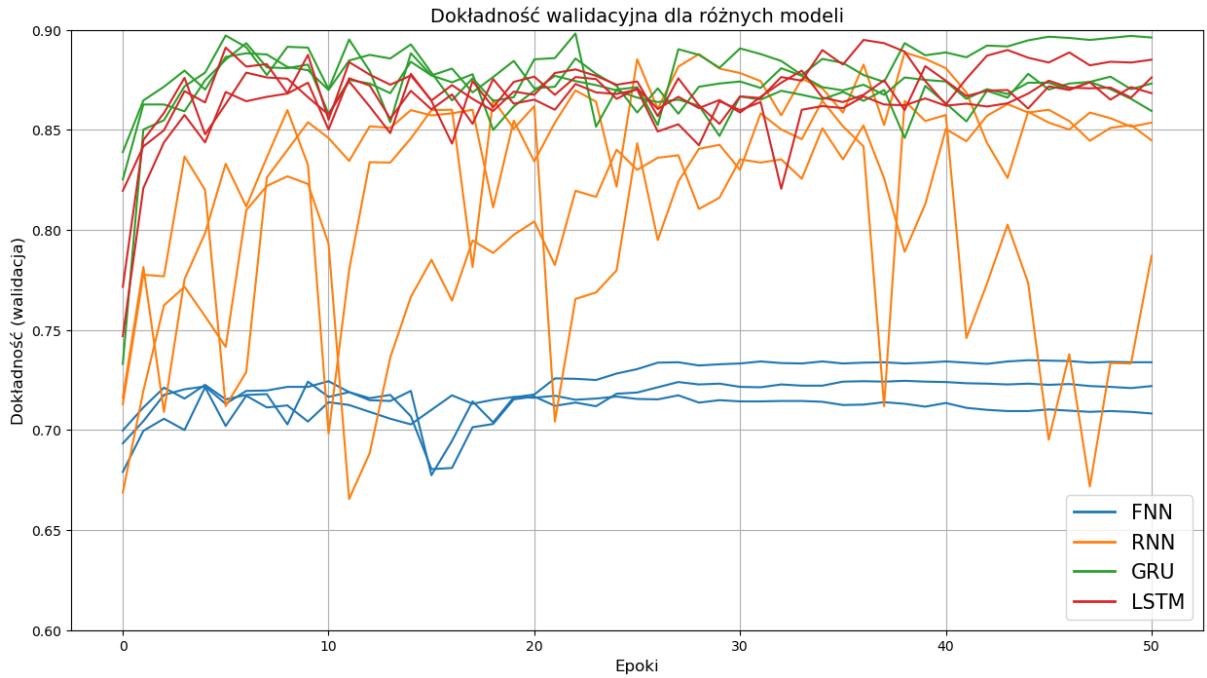
Tabela 4.12. LSTM - dokładność dla danych walidacyjnych

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.642	0.719	0.751	0.766	0.776	0.783	0.784
3	0.717	0.765	0.800	0.806	0.817	0.821	0.824
5	0.757	0.807	0.822	0.820	0.845	0.843	0.846
10	0.795	0.837	0.861	0.860	0.877	0.872	0.881
15	0.821	0.852	0.876	0.870	0.878	0.882	0.897
20	0.835	0.866	0.887	0.879	0.892	0.891	0.891
40	0.803	0.862	0.858	0.869	0.883	0.886	0.895

Najniższą wartość funkcji celu równą 0.342 osiąga model o parametrach: sample size = 40, hidden units = 256. Ma on również dokładność na poziomie 89.5%. Jest on ostatnim modelem branym pod uwagę do wykorzystania w aplikacji.

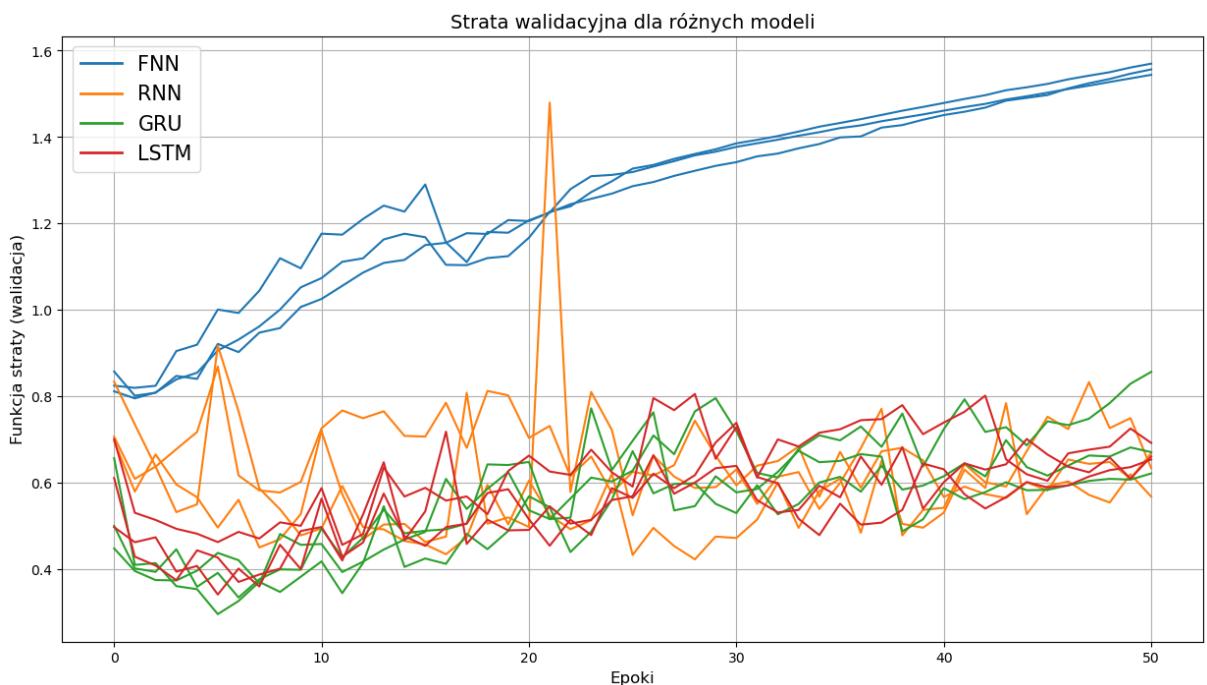
4.2.4. Wyniki dla FNN

W tym fragmencie zostanie wyjaśnione, dlaczego model FNN nie nadaje się do pracy z długimi szeregami czasowymi. Brak mechanizmów, takich jak połączenia zwrotne w przypadku RNN lub bramki dla GRU oraz LSTM ogranicza skuteczne zapamiętywanie informacji i wyciąganie z nich wniosków. W tym celu zostały wytrenowane modele dla sample size równego 40 oraz różnych wartości parametru hidden units . Jak widać na Rysunku 4.5, dokładność walidacyjna modeli FNN wzrasta nieznacznie przez pierwsze kilka epok uczenia, następnie się stabilizuje na poziomie 70-75% dokładności. W porównaniu do innych modeli, FNN radzi sobie najgorzej, niezależnie od parametrów.



Rys. 4.5. Dokładność walidacyjna dla różnych modeli

W przypadku wyników funkcji celu dla danych walidacyjnych model od samego początku nie jest w stanie jej poprawnie optymalizować, zwiększać jej wartość przez niemal cały proces uczenia. Z tych powodów modele FNN nie były brane pod uwagę podczas doboru modeli do aplikacji.

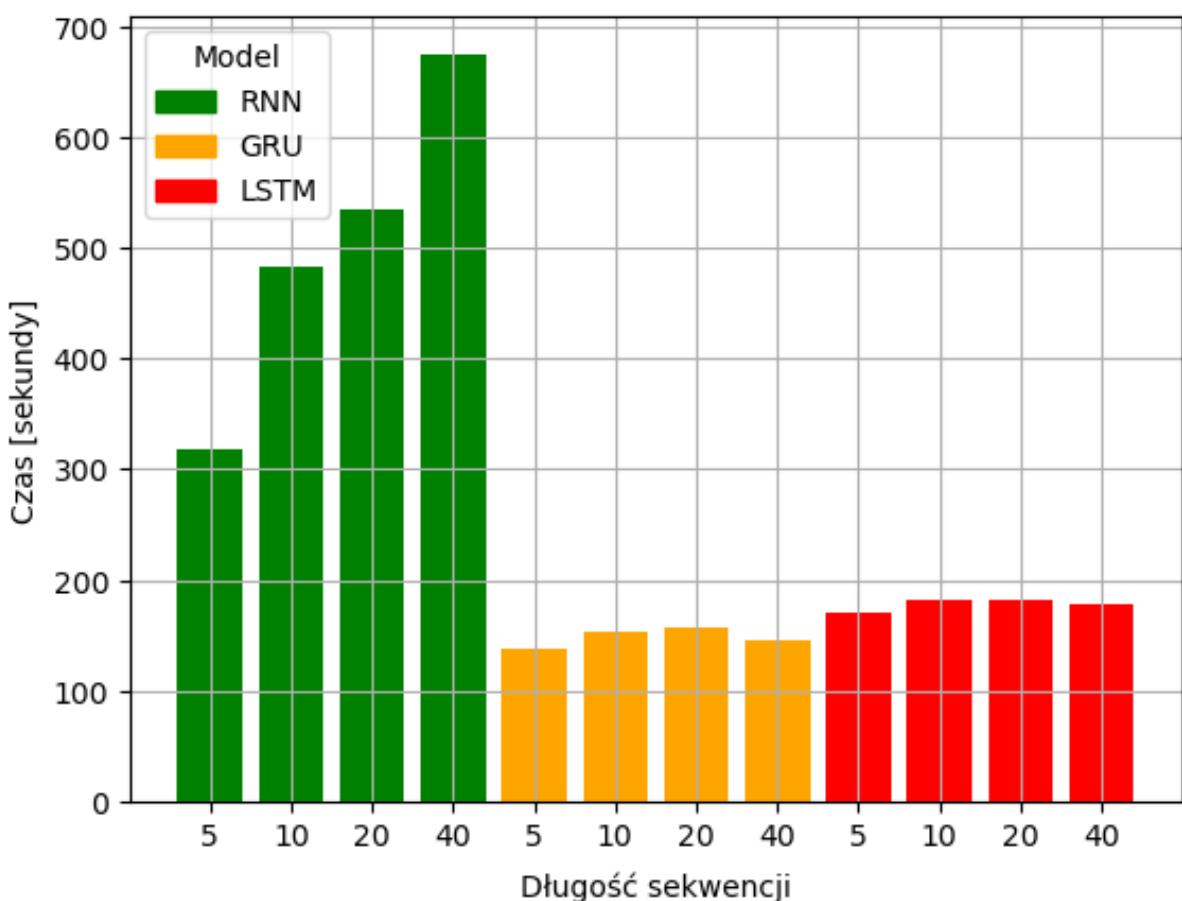


Rys. 4.6. Czas uczenia modeli ze względu na długość próbki

4.3. Porównanie czasu uczenia oraz wykonywania

Modele, z których korzystano mają różne architektury. RNN ze względu na swoją prostą budowę ma gorsze wyniki od bardziej złożonych modeli, takich jak GRU czy LSTM. Z drugiej strony, prostsze modele zazwyczaj są w stanie być uczone oraz wykonywać predykcje szybciej dzięki mniejszej liczbie potrzebnych do wykonania operacji. Wbrew oczekiwaniom, modele Simple RNN uczyły się często kilkukrotnie dłużej dla tych samych parametrów. Może to wynikać z problemów modelu w optymalizacji funkcji celu, którego wynikiem mogło być przedłużenie procesu uczenia. Czas uczenia był mierzony od inicjalizacji modelu, do końca procesu uczenia. Czas odpowiedzi modelu był brany jako średnia czasów odpowiedzi dla 100 przykładowych próbek danych. Parametry sprzętowe urządzenia wykorzystanego do zebrania danych były następujące:

- NVIDIA GeForce GTX 1660 TI - 6GB,
- Intel(R) Core(TM) i5-9300H CPU @ 2,40GHZ,
- 8 GB RAM



Rys. 4.7. Czas uczenia modeli ze względu na długość próbki

4.3.1. Wyniki dla RNN

Modele RNN mają najdłuższy czas uczenia z wszystkich modeli dla odpowiadających sobie parametrów. Można zauważyć, że w zależności od ich kombinacji, modele mogą wydłużyć proces uczenia nawet kilkukrotnie. Zazwyczaj zmiana parametru sample size z 1 na 40 wydłuża czas trenowania 3-5 razy w porównaniu do hidden units, którego modyfikacja nie zawsze wpływa na zwiększenie czasu uczenia.

Średni czas predykcji dla przykładu treningowego jest bardzo niski i utrzymuje się pomiędzy 0.037, a 0.048 sekundy. Jest to wynik zbliżony do innych modeli, przy czym maksymalny czas odpowiedzi jest w tym przypadku najdłuższy. Niska skuteczność, najdłuższy średni czas uczenia oraz porównywalne prędkości generowania odpowiedzi przez modele sprawiają, że zwykłe modele RNN nie są najlepszym wyborem dla tego problemu.

Tabela 4.13. RNN - Czas uczenia

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	125.9	148.5	165.0	169.2	149.8	175.7	131.9
3	183.2	165.8	237.6	205.9	212.9	183.0	197.9
5	234.0	211.8	282.2	240.0	317.5	230.8	247.0
10	445.9	335.6	384.7	384.7	482.5	328.9	357.9
15	476.0	427.8	444.1	506.4	532.7	456.4	454.3
20	528.5	678.7	518.8	627.7	535.8	559.4	566.5
40	610.0	699.4	605.2	631.6	675.4	593.5	722.9

Tabela 4.14. RNN - Średni czas generowania odpowiedzi

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.038393	0.037686	0.038202	0.038675	0.038146	0.038228	0.041898
3	0.041221	0.038092	0.040721	0.037902	0.040970	0.038467	0.042684
5	0.037860	0.038203	0.038479	0.038344	0.038575	0.038631	0.040397
10	0.039426	0.039149	0.039422	0.039131	0.039162	0.039796	0.040737
15	0.040313	0.039877	0.042326	0.040623	0.040192	0.040506	0.041011
20	0.040712	0.042697	0.043720	0.041954	0.041654	0.041609	0.042476
40	0.044401	0.045636	0.045135	0.045455	0.045227	0.045677	0.047850

4.3.2. Wyniki dla GRU

Modele GRU przy niskich wartościach parametru sample size charakteryzują się porównywalnym czasem uczenia do modeli RNN. Dopiero po zwiększeniu tej wartości, różnice zaczynają stawać się

znaczące, sięgając nawet do poziomu 4-5 krotnie krótszych czasów uczenia. Warto zauważyc, że w tym przypadku maksymalny czas uczenia wynosi 3.8 minut, co jest wartością o wiele niższą niż 12.05 minut w przypadku modeli RNN przy relatywnie małym zbiorze danych. Trzeba brać to pod uwagę podczas rozważania korzystania z tej architektury.

Czas predykcji jest porównywalny do czasów osiąganych przez zwykłe modele RNN. Można zaobserwować niższy wpływ parametru sample size na zwiększenie się czasu uczenia modeli, w porównaniu do poprzedniej klasy modeli.

Tabela 4.15. GRU - Czas uczenia

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	149.0	143.0	128.1	120.2	128.0	143.3	139.1
3	154.9	155.1	128.0	123.7	135.0	143.8	148.8
5	174.1	170.3	129.7	129.2	137.1	143.9	146.3
10	190.0	186.7	143.4	143.4	153.4	153.0	152.2
15	207.2	177.5	149.3	142.1	152.9	154.5	162.4
20	212.1	152.6	153.2	144.2	156.4	163.7	172.8
40	227.5	143.0	139.8	138.9	145.4	150.5	188.8

Tabela 4.16. GRU - Średni czas generowania odpowiedzi

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.039244	0.039023	0.039451	0.039596	0.039228	0.039927	0.039820
3	0.038129	0.038307	0.041325	0.038527	0.038556	0.041878	0.039199
5	0.038182	0.038820	0.038234	0.038239	0.038327	0.038575	0.039209
10	0.038349	0.038708	0.038561	0.038643	0.038397	0.039040	0.039603
15	0.038434	0.038567	0.038770	0.038641	0.038816	0.039292	0.042901
20	0.041915	0.038957	0.038696	0.041968	0.038949	0.039244	0.040341
40	0.039678	0.039798	0.039673	0.039924	0.039658	0.040193	0.041302

4.3.3. Wyniki dla LSTM

Czas uczenia oraz predykcji modeli LSTM jest zbliżony do czasu osiąganego przez modele GRU. Częściej niż w przypadku GRU, zwiększenie parametru hidden units wpływało na przedłużenie procesu uczenia. Maksymalna wartość czasu odpowiedzi jest wyższa dla modeli LSTM niż GRU i wynosi ona 0.044309 sekundy.

Tabela 4.17. LSTM - Czas uczenia

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	142.6	139.5	143.2	151.0	159.3	168.8	173.0
3	145.2	144.5	152.8	156.8	170.1	174.3	179.2
5	151.6	151.8	156.2	166.9	170.4	176.7	186.0
10	161.0	155.9	166.3	179.0	181.1	193.1	190.0
15	168.9	169.4	165.1	171.5	180.6	206.2	204.9
20	167.3	178.5	179.5	179.5	182.0	203.8	213.8
40	166.5	164.4	164.1	165.3	177.6	227.1	225.5

Tabela 4.18. LSTM - Średni czas generowania odpowiedzi

Sample Size	Hidden Units						
	4	8	16	32	64	128	256
1	0.039417	0.042624	0.038894	0.039311	0.039632	0.039563	0.040099
3	0.038602	0.038743	0.038646	0.038755	0.039260	0.039436	0.040144
5	0.038939	0.038931	0.043474	0.038878	0.039486	0.039653	0.040379
10	0.038706	0.038926	0.041714	0.039982	0.039786	0.040050	0.043858
15	0.038860	0.040257	0.039720	0.039883	0.039908	0.040193	0.041423
20	0.039450	0.039448	0.039501	0.043039	0.040064	0.040435	0.041883
40	0.040242	0.040540	0.040079	0.040427	0.041149	0.041527	0.044309

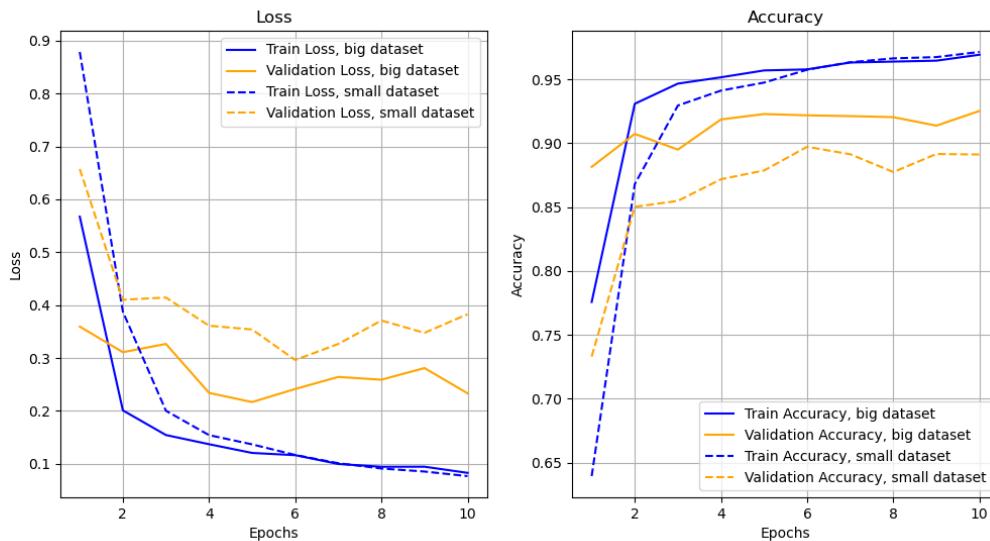
4.4. Proces finalnego trenowania modelu

Na podstawie analizy wyników modeli dla różnych parametrów został wyłoniony model o najniższej wartości funkcji celu. Jest to model GRU, który uzyskał wynik 0.296 dla danych walidacyjnych. Model osiągnął ten efekt dla podanych parametrów:

- sample size = 40,
- hidden units = 64,
- epochs = 6.

Znając parametry modelu uczonego na podzbiorze danych, możemy zakładać, że będą one odpowiednie dla całego zbioru. Dzięki temu zaoszczędzono wiele czasu, który musiałby być poświęcony do uczenia wszystkich modeli na pełnym zbiorze. Model, który był trenowany i finalnie wykorzystany w aplikacji będzie miał takie same parametry. Na poniższym wykresie można zauważać porównanie wyników modeli dla różnych wielkości zbiorów uczących. Jak widać większa liczba danych pozwoliła na

osiągnięcie lepszych wyników zarówno funkcji celu jak i dokładności. Wartość funkcji celu oraz dokładność poprawiły się odpowiednio z 0.296 oraz 89.7% przy 6 epokach na 0.2113 i 92.58% przy 5 epokach.



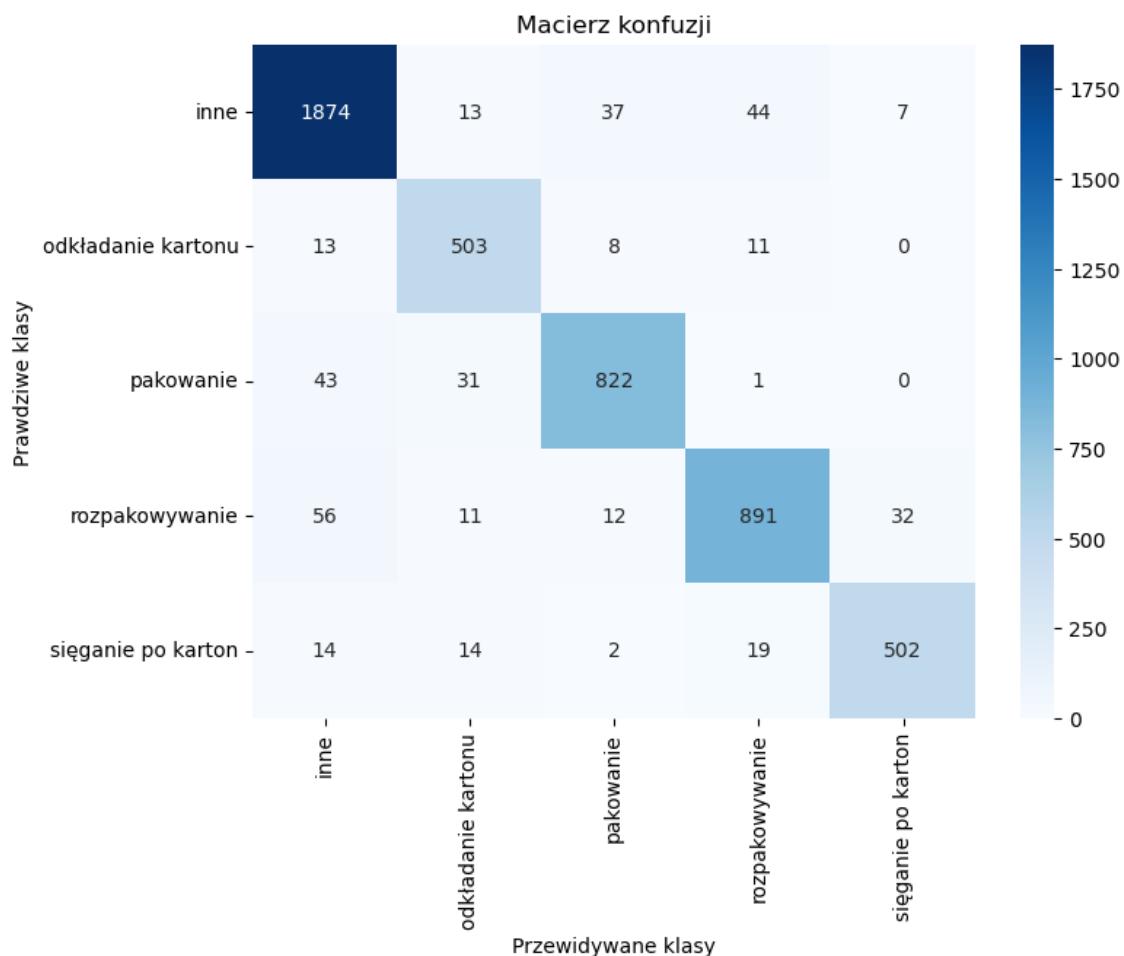
Rys. 4.8. Wykres funkci丢失 (Loss) i dokładności (Accuracy) dla modelu GRU

W tabeli 4.19 zawarto różne wskaźniki jakości modelu. Precyzaja określa, jaki odsetek przewidzianych pozytywnych przykładów jest faktycznie poprawny. Czułość mierzy, jaki odsetek rzeczywistych pozytywnych przykładów został poprawnie przewidziany. F1-score bierze pod uwagę zarówno precyzaję jak i czułość, obliczając ich średnią harmoniczną. Liczebność określa liczbę próbek w każdej kategorii. Największa liczebność występuje w kategorii "inne" osiągając wartość 1975.

Klasa	Precyzaja	Czułość	F1-score	Liczebność
inne	0.9370	0.9489	0.9429	1975
odkładanie kartonu	0.8794	0.9402	0.9088	535
pakowanie	0.9330	0.9164	0.9246	897
rozpakowywanie	0.9224	0.8892	0.9055	1002
sięganie po karton	0.9279	0.9111	0.9194	551
Średnia makro	0.9199	0.9211	0.9202	4960
Średnia ważona	0.9261	0.9258	0.9257	4960

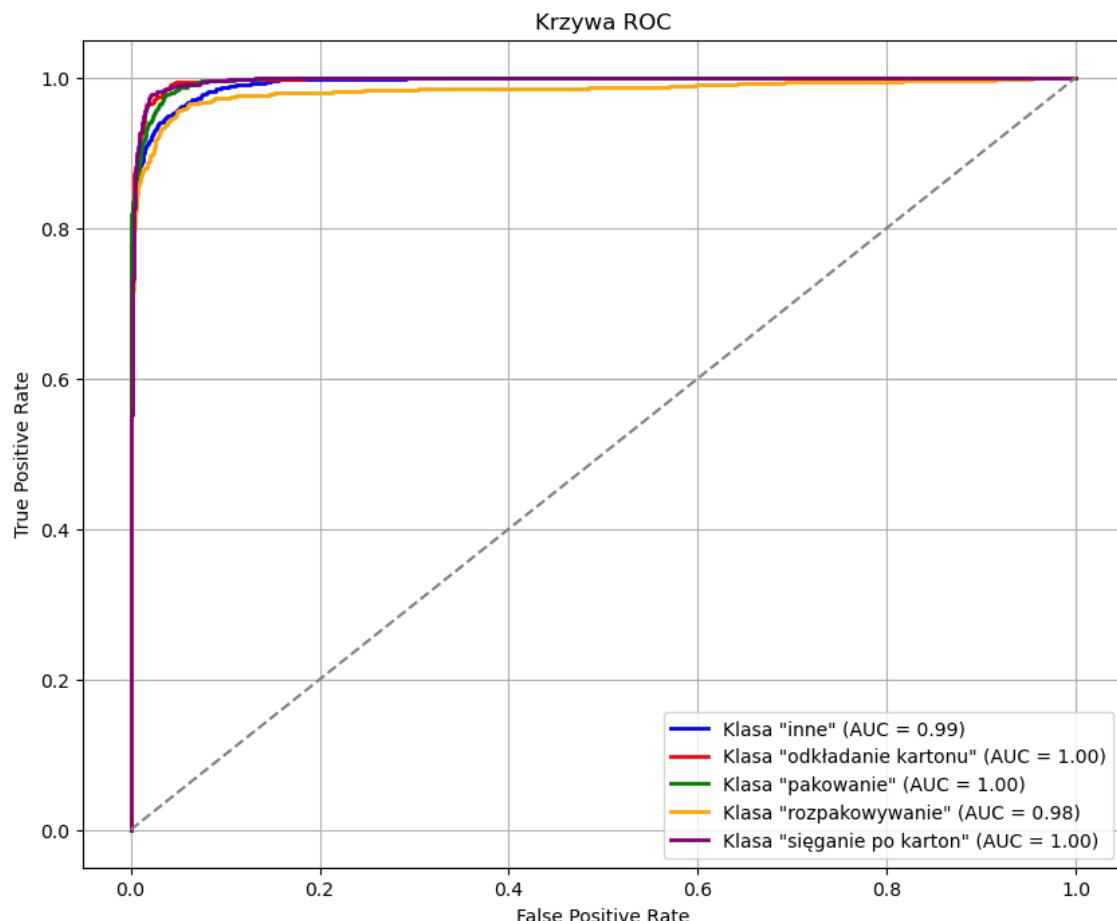
Tabela 4.19. Precyzaja, czułość i F1-score dla każdej klasy.

Macierz konfuzji pozwala określić, które klasy były błędnie przydzielane do innych klas, co może oznaczać, że model ma problem z ich rozróżnianiem. W tym przypadku "rozpakowywanie" oraz "inne" było najczęściej mylone niezależnie od tego, która z kategorii była przewidywana. Wiele pomyłek wynikało prawdopodobnie z trudności w precyzyjnym rozdzieleniu momentu, w którym jeden ruch się kończył, a drugi zaczynał. Inne mogą być rezultatem podobieństwa niektórych czynności, a jeszcze inne – zwykłej zawodności modelu wynikającej z niewystarczającej liczby danych.



Rys. 4.9. Macierz konfuzji modelu GRU

Wykres ROC analizuje różne warianty macierzy konfuzji w zależności od progów decyzyjnych, które definiują do której klasy przypisywany jest przykład. Wynika z niego, że model ma największe trudności z dopasowaniem klas w przypadku "inne" oraz "rozpakowywanie", ponieważ ich wartość AUC (pola pod wykresem), jest najmniejsza.



Rys. 4.10. Wykres krzywej ROC dla modelu GRU

5. Aplikacja

5.1. Schemat aplikacji

Aplikacja została stworzona w języku Python przy pomocy biblioteki Gradio, pozwalającą na proste tworzenie interfejsów użytkownika do testowania modeli uczenia maszynowego. Aplikacja umożliwia w prosty sposób kalibrować kamery oraz otrzymać wyniki klasyfikacji dla osób nieposiadających wiedzy z dziedziny systemów wizyjnych oraz machine learningu. Wspiera ona do czterech urządzeń nagrywających.

Głównym celem aplikacji była demonstracja detekcji czynności. Jest ona minimalistyczna oraz przejrzysta. Działa w czasie rzeczywistym oraz na bieżąco zwraca informacje dotyczące wykonywanych aktywności. Złożona jest z ciemnego menu głównego, w którym znajdują się kontrastujące, białe napisy. Występują też trzy zakładki:

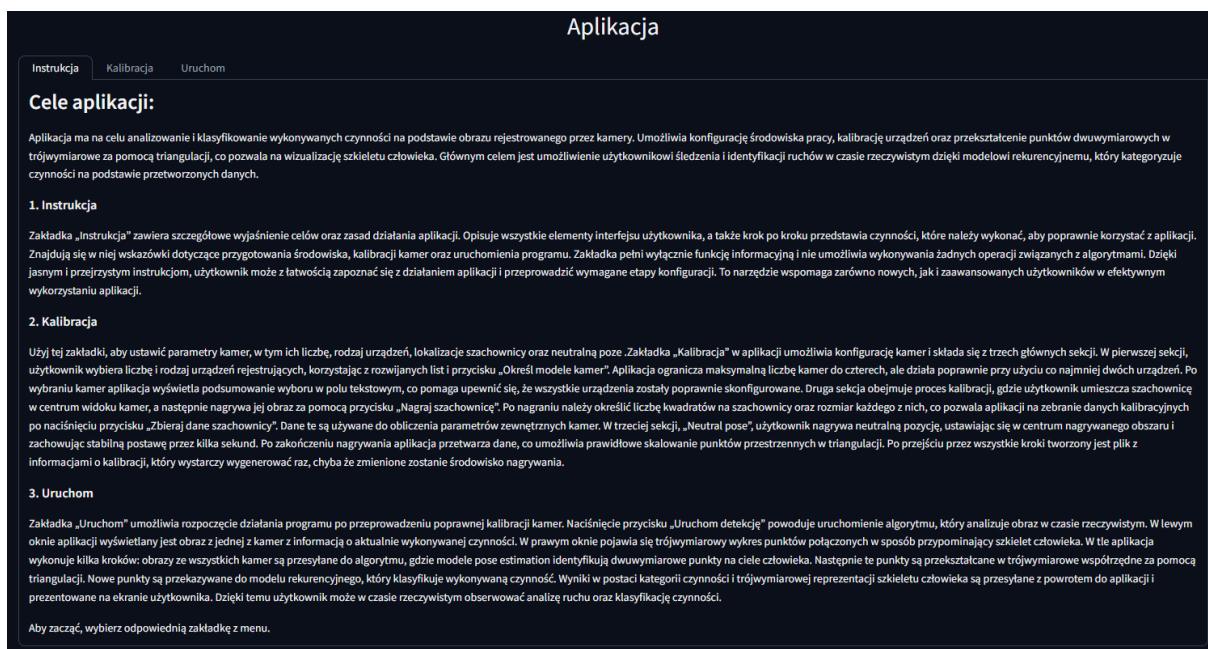
1. Instrukcja - wyjaśnia działanie i cele aplikacji,
2. Kalibracja - pozwala skalibrować parametry wewnętrzne i zewnętrzne kamery, jak i ustalić liczbę kamer, rodzaje urządzeń, charakterystykę szachownicy oraz parametry pozy neutralnej,
3. Uruchom - jeśli kalibracja została skończona, można uruchomić program, który ma na celu rekonstrukcję obrazu z kamer, a następnie określenie wykonywanej czynności, zaprezentować trójwymiarowy szkielet osoby ćwiczącej.

5.2. Działanie aplikacji

Ten rozdział jest przeznaczony na szczegółowe opisanie każdego z głównych okien aplikacji. Zostaną zaprezentowane i wyjaśnione komponenty interfejsu graficznego, jak i wykonywane w tle procesy. Wraz z opisem znajdują się zrzuty ekranu prezentujące wyniki prac.

5.2.1. Instrukcja

Instrukcja zawiera szczegółowe wyjaśnienie celów jak i działania aplikacji. Przedstawia kolejne jej elementy wraz z czynnościami, które powinno się wykonać, aby prawidłowo z niej korzystać. To okno ma charakter informacyjny i nie wykonuje żadnych zadań związanych z uruchomieniem algorytmów.

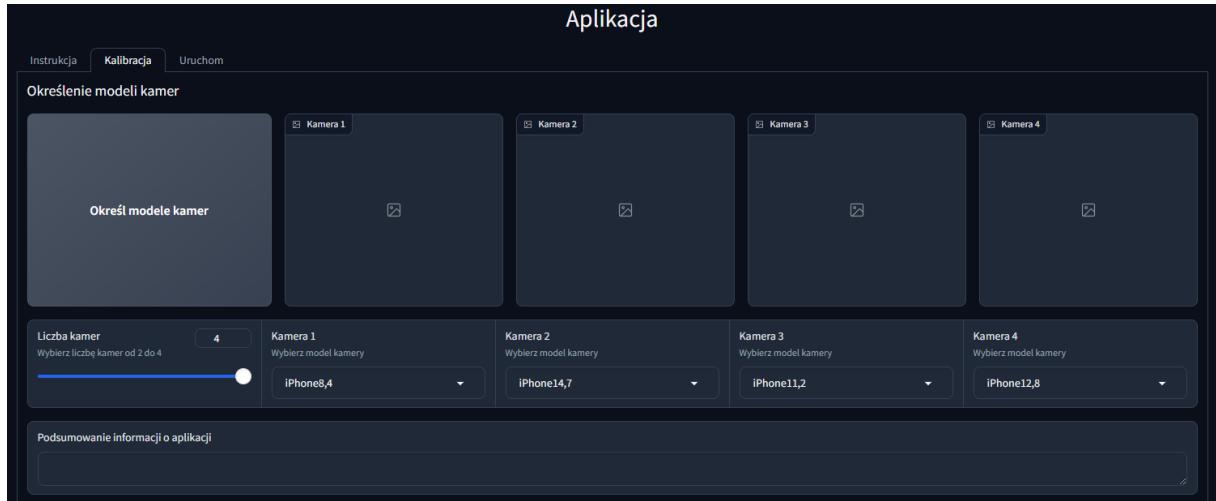


Rys. 5.1. Aplikacja - Instrukcja

5.2.2. Kalibracja

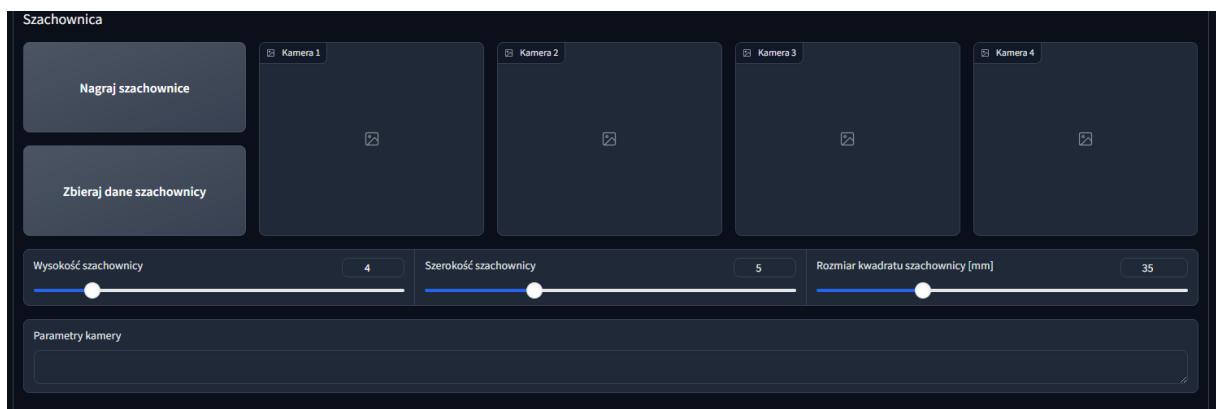
Zakładka "Kalibracja" jest rozbudowana, posiada wiele przycisków, suwaków oraz rozwijanych list. Wszystkie te elementy są podpisane i wyjaśnione w instrukcji. Można ją podzielić na trzy główne sekcje, w których znajdują się okna przedstawiające wyniki wykonanych etapów, co pozwala na łatwiejsze przeprowadzenie procesu kalibracji.

Pierwsza sekcja "Określenie modeli kamer" jest przeznaczona do wybrania, z jakiej ilości oraz rodzaju urządzenia będziemy korzystać. Z powodu prędkości oraz przejrzystości aplikacji zdecydowano się na ograniczenie maksymalnej liczby urządzeń do czterech. Po przyciśnięciu przycisku "Określ modele kamer", aplikacja wyświetla wszystkie dostępne kamery podłączone do urządzenia, na którym się znajduje. W ten sposób łatwo można określić, które urządzenie rejestrujące odpowiada któremu indeksowi kamery, a następnie wybrać go z rozwijanej listy. W rozwijanych listach znajdują się modele telefonów firmy Apple. Nie jest wymagane, aby użytkownik korzystał z maksymalnej liczby urządzeń, program powinien działać poprawnie dla przynajmniej dwóch kamer. Podsumowanie przydzielonych kamer zostanie wyświetcone w polu tekstowym po ponownym naciśnięciu przycisku "Określ modele kamer". Dane zebrane w tej części będą wykorzystane do określenia parametrów wewnętrznych kamer.



Rys. 5.2. Aplikacja - Kalibracja, parametry wewnętrzne

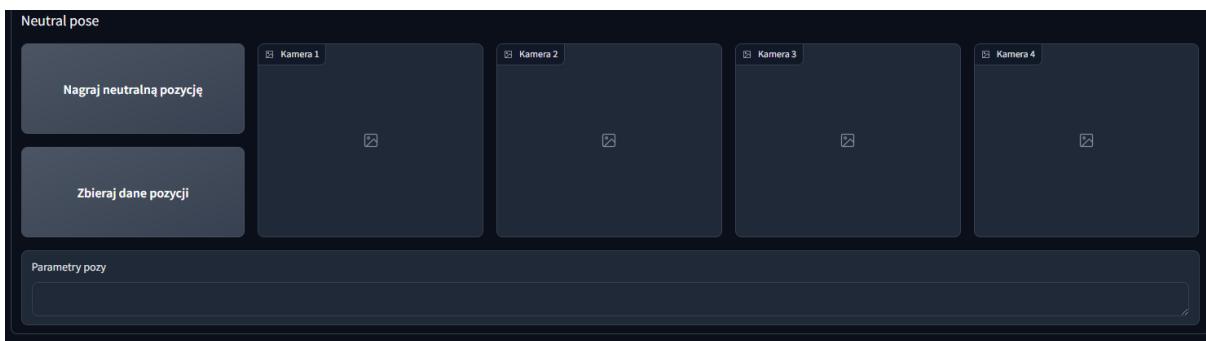
Kolejna sekcja dotyczy kalibracji. Użytkownik aplikacji powinien ustawić obiekt wykorzystywany do kalibracji w takim miejscu, aby był on dobrze widoczny przez wszystkie kamery, najlepiej w środku obszaru testowego. W tym przypadku takim obiektem jest obraz szachownicy. Następnie po naciśnięciu przycisku "Nagraj szachownice", algorytm nagra kilka sekund filmu, dla każdej perspektywy, jednocześnie wyświetlając je w aplikacji. W dalszej kolejności należy wybrać liczbę kwadratów w podstawie i wysokość szachownicy oraz rozmiar samego kwadratu w milimetrach. Jeśli wymienione czynności zostały wykonane prawidłowo, naciśnięcie przycisku "Zbieraj dane szachownicy" spowoduje wykrycie obiektu kalibrującego na nagraniu, a następnie obliczenie parametrów zewnętrznych kamer.



Rys. 5.3. Aplikacja - Kalibracja, parametry zewnętrzne

Ostatnia sekcja to "Neutral pose". Odpowiada ona za określenie parametrów potrzebnych do prawidłowego zmodyfikowania punktów otrzymanych w procesie triangulacji, między innymi maksymalne oraz minimalne wartości punktów w przestrzeni dla danych osi. Dzięki temu punkty będą mogły być zvisualizowane oraz przekazane do modelu rekurencyjnego. Użytkownik po naciśnięciu przycisku "Nagraj neutralną pozycję" musi ustawić się na środku nagrywanego obszaru, w pozycji neutralnej oraz zacząć

kilka sekund aż nagrywanie się zatrzyma. Następnie może wrócić do ekranu aplikacji i nacisnąć przycisk "Zbieraj dane pozycji", co spowoduje wyświetlenie się odpowiedniego komunikatu.



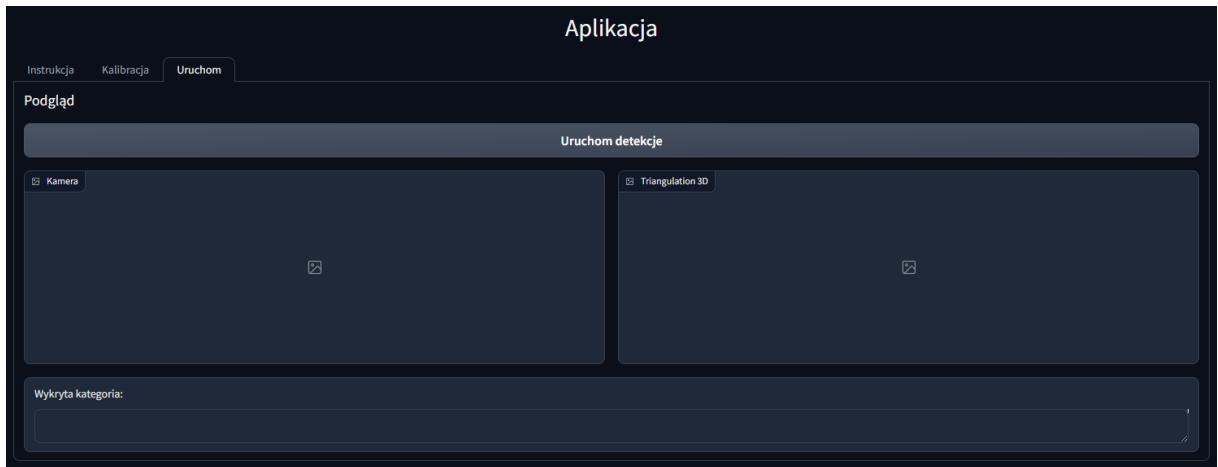
Rys. 5.4. Aplikacja - Kalibracja, poza neutralna

Po przejściu przez wszystkie sekcje zostanie utworzony odpowiedni plik z informacjami o kalibracji. Dzięki temu wystarczy przeprowadzić tylko raz kalibrację dla danego ułożenia kamer. W przypadku zmiany środowiska nagrywania konieczne jest powtórzenie wszystkich czynności.

5.2.3. Uruchom

Zakładka "Uruchom" pozwala na uruchomienie programu po wcześniejszym przeprowadzeniu po prawnej kalibracji. Naciśnięcie przycisku "Uruchom detekcję" spowoduje uruchomienie algorytmu, który rozpoczęcie wyświetlenie obrazu z jednej z kamer w lewym oknie. W prawym oknie będzie wyświetlał się wykres połączonych ze sobą punktów przypominających szkielet człowieka. W oknie "Wykryta kategoria" pojawi się przypis informującym o tym, jaka czynność jest wykonywana w danym momencie. W tle będą się wykonywały następujące operacje:

1. wszystkie działające kamery wysyłają obraz do algorytmu,
2. model estymacji pozycji rozpoznaje dwuwymiarowe punkty na ciele człowieka,
3. otrzymane punkty zostaną przekształcone w punkty trójwymiarowe za pomocą triangulacji,
4. nowe punkty zostaną dodane do danych przesyłanych do modelu rekurencyjnego,
5. model rekurencyjny zwróci kategorie czynności, jaka jest wykonywana w danym momencie,
6. trójwymiarowa reprezentacja szkieletu człowieka oraz informacja o kategorii wykonywanej czynności zostaje wysłana z powrotem do aplikacji.



Rys. 5.5. Aplikacja - Uruchamianie algorytmu

6. Podsumowanie i wnioski

6.1. Podsumowanie wyników pracy

Celem niniejszej pracy inżynierskiej było zaprojektowanie, implementacja oraz demonstracja możliwości określania czynności za pomocą technologii stereowizji. W pierwszym etapie przeprowadzono dogłębną analizę dostępnych metod, opierając się na przeglądzie literatury naukowej, która dostarczyła cennych informacji i posłużyła jako solidna podstawa teoretyczna dla dalszych działań.

Kolejnym etapem było stworzenie środowiska testowego poprzez zapewnienie odpowiednio dużego obszaru roboczego, ustawienia kamer pod odpowiednimi kątami oraz upewnienie się, że nagrywana osoba jest widoczna przez każdą z nich. Rozmieszczone kamery wymagały kalibracji, polegającej na wyznaczeniu ich parametrów wewnętrznych oraz zewnętrznych, co pozwoliło na określenie ich struktury oraz orientacji w przestrzeni. Proces ten przeprowadzono przy użyciu plików zawierających parametry sprzętowe urządzeń oraz obiektu kalibracyjnego, którym w tym przypadku był wzorzec szachownicy. Dodatkowo, obrazy z kamer zostały zsynchronizowane za pomocą zewnętrznej aplikacji, zapewniając spójność danych pomiędzy urządzeniami.

Obrazy z kamer były przetwarzane za pomocą modeli estymacji pozycji w celu precyzyjnego określenia lokalizacji wybranych punktów na ciele człowieka. Do tego zadania wykorzystano model BlaZePose, który umożliwia identyfikację 33 punktów anatomicznych. Model ten dostępny jest w trzech wariantach – Lite, Full oraz Heavy – dostosowanych do różnych wymagań wydajnościowych. Działa on poprawnie, pod warunkiem, że punkty są widoczne na obrazie i nie są zasłonięte przez obiekty lub inne części ciała.

Istotnym elementem pracy było przygotowanie algorytmu triangulacji, który umożliwiałby określenie trójwymiarowych współrzędnych punktów przy pomocy ich dwuwymiarowych odpowiedników. Aby punkty niezależnie od ustawień kamery, znajdująły się w podobnych przedziałach liczbowych, przygotowano dodatkowy etap kalibracji, który pozwalał normalizować wartości punktów.

Następnie zostały zebrane dane, zawierające trajektorie trójwymiarowych punktów dla wykonywanych przykładowych czynności, w tym przypadku była to symulacja pakowania paczek w magazynie. Czynność była nagrywana oraz powtarzana przez cały proces zbierania danych. Można było ją podzielić na pięć kategorii: "inne", "odkładanie kartonu", "pakowanie", "rozpakowywanie" oraz "sięganie po karton". Wiele z tych czynności jest nieroóżnialna bez kontekstu czasowego.

Na podstawie zebranych danych uczono modele rekurencyjne, których zadaniem było rozpoznawanie kategorii czynności, która była wykonywana. Najpierw na podzbiorze danych przeuczono modele rekurencyjne dla różnych kombinacji parametrów sample size oraz hidden units, aby określić jakie powinny być użyte w finalnym procesie uczenia. Głównym kryterium wyboru była wartość funkcji celu dla danych walidacyjnych. Modele, które były brane pod uwagę to Simple RNN, GRU oraz LSTM. Wyłoniono najlepszego kandydata oraz wykorzystano cały zbiór danych w celu osiągnięcia lepszych wyników. Model został zbadany za pomocą różnych wskaźników takich jak: precyza, czujność, F1-score, macierz konfuzji oraz wykresy ROC.

Na końcu, przy użyciu biblioteki Gradio w Pythonie, stworzono aplikację implementującą wypisane algorytmy. Umożliwia ona użytkownikowi w prosty sposób przygotowanie środowiska, kalibrację kamer oraz określenie wykonywanej czynności. Aplikacja cechuje się intuicyjnością i minimalizmem.

6.2. Możliwości dalszego rozwoju

Kalibracja:

Aktualne rozwiązanie kalibruje kamery przy pomocy jednego obrazu dla każdego urządzenia. W przyszłości możliwe jest wykonanie paru zdjęć oraz wyciągnięcie wartości średnich parametrów, aby kalibracja była bardziej odporna na anomalię. Kolejnym problemem możliwym do rozwiązania jest wielkość szachownicy wykorzystywanej do kalibracji. Występują sytuacje, w których obraz kalibratorowy jest niewielki i słabo widoczny, co znacząco utrudnia dokładne wykrycie punktów referencyjnych, prowadząc do ich potencjalnych przesunięć względem oczekiwanych pozycji. Potencjalnym rozwiązaniem tego problemu jest przygotowanie większych obiektów kalibratorowych lub wyciągnięcia części parametrów kamery na obrazach o większej rozdzielcości niż te wykorzystywane podczas triangulacji.

Triangulacja:

Istnieją przypadki, w których dane punkty na ciele są niewidoczne lub są źle lokalizowane przez modele estymacji pozycji, w takich sytuacjach reprezentacja trójwymiarowa punktu bardzo odbiega od faktycznej lokalizacji. W przyszłych rozwiązaniach możliwa jest implementacja mechanizmu wykrywanie oraz ignorowanie klatek, w których występują takie punkty.

Modele estymacji pozycji:

Największymi ograniczeniami modeli estymacji pozycji jest ich dokładność i szybkość. Części ciała nagrywanej osoby, które nie są dobrze widoczne na zdjęciu często nie są wykrywane przez model. Prowadzi to do niepoprawnej rekonstrukcji szkieletu, co może skutkować gorszymi wynikami modelu rozpoznającego czynności. Jednym z rozwiązań mogłoby być doszkolenie istniejącego lub nauczenie nowego modelu do rozpoznania szkieletu człowieka. Większą szybkość modelu można osiągnąć poprzez zmienienie jego wersji z Heavy na Full lub Lite, trzeba jednak zdawać sobie sprawę z wynikającego z tego spadku skuteczności. Potencjalny rozwój koncepcji mógłby też zakładać stworzenie własnego

modelu rozpoznającego punkty na obrazach dla dowolnego obiektu w celu określania wykonywanych przez niego czynności. Zastosowanie takiego rozwiązania mogłoby pomagać w monitorowaniu robotów, pojazdów czy zwierząt.

Modele rozpoznawania czynności:

Możliwe jest poprawienie skuteczności modeli rozpoznających czynności poprzez dostarczenie większej ilości zróżnicowanych danych dobrej jakości. Dane powinny zawierać przebiegi czasowe punktów 3D otrzymane za pomocą triangulacji wykonanej przez różną liczbę kamer, ustawioną pod odmiennymi kątami. Jeśli jakąś czynność może zostać wykonana na parę sposobów też powinno być to uwzględnione.

Aplikacji:

Stworzenie aplikacji miało na celu zawarcie wszystkich potrzebnych algorytmów w jednym miejscu oraz przystosowanie ich do użytkowników w taki sposób, aby mogły z niej korzystać bez posiadania żadnej dodatkowej wiedzy. W przyszłości może być ona rozbudowana o dodatkowe mechanizmy takie jak modyfikowanie parametrów modeli, większą liczbę wykorzystywanych urządzeń lub tworzenie wykresów obrazujących istotne informacje z sesji.

6.3. Wnioski

Przebieg i efekty pracy nad rozpoznawaniem czynności przy pomocy stereowizji prowadzą do wniosków, że jest możliwe stworzenie architektury, która będzie na to pozwalać. Wyniki uczenia modeli rekurencyjnych wskazują na ich wysoką skuteczność w tego typu zadaniach, osiągając dokładność na poziomie powyżej 90% na danych walidacyjnych. Dzięki niewielkim wymaganiom sprzętowym mogą one być wykorzystywane w wielu miejscach po odpowiednim skonfigurowaniu środowiska. Dzięki wykorzystaniu zalet wynikających z systemów wielokamerowych możliwe jest operowanie na danych zawierających informacje o lokalizacji punktów w przestrzeni trójwymiarowej, pozwalającej na lepsze zobrazowanie problemu.

Z drugiej strony, w komercyjnym zastosowaniu możliwa byłaby konieczność zebrania większej ilości danych, ponieważ niektóre czynności mogą być wykonywane w różny sposób, a przebieg prac może nie być tak powtarzalny jak w zaproponowanym rozwiązaniu. System też nie będzie działał w miejscowościach, które nie będą zapewniały warunków do optymalnego zamontowania kamer, przykładowo w zbyt tłocznych budynkach lub takich o małym metrażu.

Podkreślając wielowymiarowy potencjał tej technologii, kluczowe będzie dalsze eksplorowanie jej możliwości poprzez zwiększenie uniwersalności modelu, poprawę skalowalności oraz testowanie w rzeczywistych warunkach, co przyczyni się do zwiększenia jej wartości użytkowej i komercyjnej.

Bibliografia

- [1] Diogo C Luvizon, David Picard i Hedi Tabia. „Multi-task deep learning for real-time 3D human pose estimation and action recognition”. W: *IEEE transactions on pattern analysis and machine intelligence* 43.8 (2020), s. 2752–2764.
- [2] Jun Liu i in. „Spatio-temporal lstm with trust gates for 3d human action recognition”. W: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part III* 14. Springer. 2016, s. 816–833.
- [3] Kang Liao i in. „Deep learning for camera calibration and beyond: A survey”. W: *arXiv preprint arXiv:2303.10559* (2023).
- [4] YM Wang, Y Li i JB Zheng. „A camera calibration technique based on OpenCV”. W: *The 3rd International Conference on Information Sciences and Interaction Sciences*. IEEE. 2010, s. 403–406.
- [5] Zhengyou Zhang. „A flexible new technique for camera calibration”. W: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2000), s. 1330–1334.
- [6] Mark Hedley Jones. *Calibration Checkerboard Collection — markhedleyjones.com*. <https://markhedleyjones.com/projects/calibration-checkerboard-collection>. [Accessed 03-01-2025].
- [7] <https://araintelligence.com/blogs/computer-vision/geometric-vision/cameracalibration>. [Accessed 03-01-2025].
- [8] H Xiangjian i in. „Estimation of internal and external parameters for camera calibration using 1D pattern”. W: *Proceedings-IEEE International Conference on Video and Signal Based Surveillance 2006, AVSS 2006*. 2006.
- [9] V Bazarevsky. „BlazePose: On-device Real-time Body Pose tracking”. W: *arXiv preprint arXiv:2006.10204* (2020).
- [10] Kristijan Bartol i in. „Generalizable human pose triangulation”. W: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, s. 11028–11037.
- [11] Alejandro Newell, Kaiyu Yang i Jia Deng. „Stacked Hourglass Networks for Human Pose Estimation”. W: *CoRR* abs/1603.06937 (2016). arXiv: 1603.06937.
- [12] RICHARD I HARTLEY. „Richard I. Hartley and Peter Sturm y”. W: () .

- [13] Yi Zhu i in. „A comprehensive study of deep video action recognition”. W: *arXiv preprint arXiv:2012.06567* (2020).
- [14] Srijan Das i in. „Deep-temporal lstm for daily living action recognition”. W: *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2018, s. 1–6.
- [15] Can Zhang i in. „Pan: Towards fast action recognition via learning persistence of appearance”. W: *arXiv preprint arXiv:2008.03462* (2020).
- [16] Varun Kumar Ojha, Ajith Abraham i Václav Snášel. „Metaheuristic design of feedforward neural networks: A review of two decades of research”. W: *Engineering Applications of Artificial Intelligence* 60 (2017), s. 97–116.
- [17] Alexander Rehmer i Andreas Kroll. „On the vanishing and exploding gradient problem in Gated Recurrent Units”. W: *IFAC-PapersOnLine* 53.2 (2020), s. 1243–1248.
- [18] Rice Yang. *A Comparison between RNN, LSTM, and GRU* — u9534056.medium.com. <https://u9534056.medium.com/comparison-of-rnn-lstm-and-gru-6ebfd8fee988>. [Accessed 03-01-2025].
- [19] Chuankun Li i in. „Skeleton-based action recognition using LSTM and CNN”. W: *2017 IEEE International conference on multimedia & expo workshops (ICMEW)*. IEEE. 2017, s. 585–590.
- [20] *opencap-core/CameraIntrinsics at main · stanfordnml/opencap-core* — github.com. <https://github.com/stanfordnml/opencap-core/tree/main/CameraIntrinsics>. [Accessed 03-01-2025].
- [21] *cameras &x2014; openMVG library* — openmvg.readthedocs.io. <https://openmvg.readthedocs.io/en/latest/openMVG/cameras/cameras/>. [Accessed 03-01-2025].
- [22] *GitHub - stanfordnml/opencap-core: Main OpenCap processing pipeline* — github.com. <https://github.com/stanfordnml/opencap-core>. [Accessed 03-01-2025].