Praca Doktorska

Zespoły Klasyfikatorów Oparte na Cechach DTW i Głębokich Reprezentacjach Akcji Ludzi na Mapach Głębi

Jacek Treliński

Promotor rozprawy:

prof. dr hab inż. Bogdan Kwolek

Praca wykonana: Akademia Górniczo-Hutnicza

Wydział Informatyki

Instytut Informatyki

Kraków 2024

Ph.D. Dissertation

ENSEMBLE LEARNING BUILT ON DEEP EMBEDDING AND DTW FEATURES FOR HUMAN ACTIVITY RECOGNITION ON DEPTH MAPS

Jacek Treliński

Supervisor:

prof. dr hab inż. Bogdan Kwolek

Completed in: AGH University of Krakow

Faculty of Computer Science

Institute of Computer Science

Kraków 2024

Pragnę wyrazić serdeczną wdzięczność mojemu promotorowi, profesorowi Bogdanowi Kwolkowi, za wsparcie i pomoc w przeprowadzeniu badań opisanych w niniejszej rozprawie.

CZĘŚĆ PRAC ZREALIZOWANO W RAMACH GRANTU NCN 2017/27/B/ST6/01743

Streszczenie

Rozpoznawanie akcji jest ważnym zagadnieniem w dziedzinie widzenia komputerowego, które polega na identyfikacji i zrozumieniu działań oraz aktywności człowieka na podstawie sekwencji obrazów lub map głębi. Jest to zadanie złożone i wymagające, ze względu na liczne praktyczne trudności, takie jak zakłócenia tła, zmiany skali, przysłonięcia, a także zmiany punktu widzenia, oświetlenia i wyglądu osoby.

Tematyka rozprawy koncentruje się na rozpoznawaniu akcji w przypadku małych zbiorów sekwencji map głębi. W tym kontekście została zaproponowana nowa metoda uczenia zespołowego oparta na sieciach neuronowych, nazwana NECSCF. Przebadano trzy wersje algorytmu NECSCF - jedną dla danych tabelarycznych i dwie dla rozpoznawania akcji. W pracach badawczych przeprowadzonych na danych tabelarycznych pokazano eksperymentalnie, że NECSCF osiąga dobre wyniki na zbiorach o ograniczonej liczbie przykładów. Algorytm NECSCF wymaga opracowania odpowiednich cech wspólnych. W celu zastosowania nowego algorytmu w problemie rozpoznawania akcji, proponuje się nowy zestaw cech opartych na odległości DTW dla szeregów czasowych.

Przebadano eksperymentalnie dwie wersje algorytmu NECSCF do rozpoznawania akcji. Pierwsza wersja wykorzystuje cechy oparte na DTW (obliczone na podstawie cech tworzonych ręcznie) jako cechy wspólne, natomiast cechy specyficzne dla każdej klasy są wyznaczane w dwóch etapach. W pierwszym etapie cechy są wyodrębniane poprzez sieci konwolucyjne na mapach głębi, a następnie w drugim etapie cechy są wyznaczane na szeregach czasowych za pomocą jednej z trzech metod: sieci konwolucyjnej 1D, sieci Siamese 1D lub cech statystycznych.

Druga wersja algorytmu NECSCF została uproszczona i nie korzysta z cech tworzonych ręcznie. Cechy specyficzne dla każdej klasy są wyznaczane przez sieci TD-LSTM trenowane w jednym etapie (end-to-end). W tej wersji wykorzystuje się dwa rodzaje cech wspólnych: oparte na uczeniu nienadzorowanym (konwolucyjny autoenkoder) oraz cechy oparte na DTW. Cechy DTW nie są wyznaczane na cechach stworzonych ręcznie, lecz na cechach wyznaczonych przez sieć Siamese.

Zaproponowana metoda NECSCF osiągnęła konkurencyjne wyniki zarówno na danych tabelarycznych, jak i w rozpoznawaniu akcji. Proponowane algorytmy umożliwiają połączenie zalet różnych architektur uczenia głębokiego oraz pozwala na skuteczne wykorzystanie algorytmu DTW wraz z sieciami neuronowymi.

Abstract

Action recognition is a field within computer vision that involves identifying and understanding human actions and activities based on sequences of images or depth maps. It is a complex and challenging task due to various practical difficulties such as background noise, changes in scale, occlusions, as well as variations in viewpoint, lighting, and appearance of individuals.

This thesis focuses on action recognition with small sets of depth map sequence. In this context, a new ensemble learning method based on neural networks, named NECSCF, has been proposed. Three versions of the NECSCF algorithm were examined - one for tabular data and two for action recognition. In the research conducted on tabular data, it was experimentally demonstrated that NECSCF achieves good results on datasets with a limited number of examples. However, the NECSCF algorithm requires the development of appropriate common features. To apply the new algorithm to the problem of action recognition, a new set of features based on Dynamic Time Warping (DTW) distances for time series is proposed.

Two versions of the NECSCF algorithm were investigated for action recognition. The first version utilizes features based on Dynamic Time Warping (DTW) calculated from handcreated features as common features, while specific features for each class are determined in two stages. In the first stage, features are extracted through convolutional networks on depth maps, and then in the second stage, features are determined on time series using one of three methods: 1D convolutional network, Siamese 1D network, or statistical features.

The second version of the NECSCF algorithm has been simplified and does not utilize handcrafted features. Specific features for each class are determined by TD-LSTM networks trained in a single stage (end-to-end). In this version, two types of common features are utilized: those based on unsupervised learning (convolutional autoencoder) and features based on DTW. DTW features are not calculated on handcreated features but on features determined by the Siamese network.

The proposed NECSCF method has achieved competitive results both in the case of tabular data and action recognition. This family of algorithms enables the combination of advantages from various deep learning architectures and facilitates effective utilization of the DTW algorithm in conjunction with neural networks.

Spis treści

Wykaz skrótów i oznaczeń

C - liczba klas w zbiorze danych

N - liczba próbek w zbiorze treningowym

K - liczba klasyfikatorów w zespole klasyfikatorów

fr - pojedyncza mapa głębi

$dms(t)$ - sekwencja map głębi ($dms(t) = fr_1 \dots fr_{|dms|}$)

$|dms|$ - liczba ramek w sekwencji map głębi

$\vec{v}$ - wektor cech

$|\vec{v}|$ - rozmiar wektora cech

$v_1 \dots v_{|\vec{v}|}$ - współrzędne wektora cech

D - zbiór wszystkich wektorów cech w zbiorze treningowym

t - czas

$ts(t)$ - wielowymiarowy szereg czasowy

$|ts|$, $|l|, |s|$ - długość szeregu czasowego

l_t, s_t - szeregi czasowe, niekoniecznie wielowymiarowe

$l_t[d]$ - w przypadku gdy l jest wielowymiarowym szeregiem czasowym, d-współrzędna wektora t.

y - etykieta próbki

$D_{l,s}(i, j)$ - odległość DTW między podsekwencjami $l_1 \dots l_i$ i $s_1 \dots s_j$

x, y, z - współrzędne punktu na mapie głębi

θ - parametry sieci neuronowej

Skróty

1D-CNN - 1 Dimensional Convolutional Neural Network

CAE - Convolutional Autoencoder

CNN - Convolutional Neural Network

DTW - Dynamic Time Warping

FSL - Few-Shot Learning

LR - Logistic Regression

LSTM - Long Short Term Memory

NECSCF - Neural Ensemble built on Class Specific and Common Features

RF - Random Forest

RFE - Recursive Features Elimination

SVC - Support Vector Classifier

TD-LSTM - Time Distributed Long Short Term Memory

1. Wstęp

1.1. Rozpoznawanie akcji na mapach głębi

Rozpoznawanie akcji, jako istotny element i jedno z najbardziej aktywnych zagadnień badawczych w dziedzinie widzenia komputerowego, jest przedmiotem badań od wielu lat. Rozpoznawanie działań człowieka na sekwencjach obrazów jest złożonym i trudnym zadaniem z powodu trudności praktycznych, takich jak zakłócenia tła, zmiany skali, przysłonięcia oraz zmiany punktu widzenia, oświetlenia i wyglądu osoby.

1.2. Cel pracy

Celem niniejszej pracy jest opracowanie metod rozpoznawania akcji na małych zbiorach sekwencji map głębi, a także pokazanie eksperymentalnie, że uzyskują one porównywalne lub lepsze wyniki niż algorytmy state-of-art na ogólnodostępnych zbiorach danych. Cel osiągnięto poprzez opracowanie i udoskonalenie metod uczenia zespołowego. W szczególności zaproponowano nową metodę uczenia zespołowego opartą na głębokich sieciach neuronowych. Nowa metoda łączy nieliniową metodę dynamicznego marszczenia czasu DTW (ang. *Dynamic Time Wraping*) z sieciami głębokimi. Metody te pozwalają osiągnąć konkurencyjne wyniki w rozpoznawaniu działań człowieka.

Zasadnicza część rozprawy dotyczy wybranych aspektów uczenia zespołowego. Zaproponowano nową metodę wyznaczania reprezentacji akcji ludzi w oparciu o sieci neuronowe i rozpoznawania ich przez zespoły klasyfikatorów. Metody proponowane w pracy pozwalają na łączenie cech opartych na uczeniu głębokim oraz cech tworzonych ręcznie. Przebadano kilka modeli głębokich sieci neuronowych, takie jak sieci neuronowe bliźniacze (ang. Siamese). W pracy przebadano również użyteczność cech wielowymiarowych szeregów czasowych wyznaczanych w oparciu o DTW i algorytm Shapelets w połączeniu z uczeniem głębokim.

1.3. Przegląd zawartości rozprawy

Praca dotyczy rozpoznawania akcji człowieka na mapach głębi. Zasadniczym wkładem niniejszej pracy doktorskiej jest:

– opracowanie modelu zespołu klasyfikatorów do rozpoznawania akcji ludzi, analiza wyników uzyskiwanych na danych z akcjami ludzi przez sieci o różnych architekturach: podstawowe sieci konwolucyjne, sieci bliźniacze Siamese i sieci LSTM

– zaproponowanie metody ekstrakcji cech z wielowymiarowych szeregów czasowych opartej na metryce DTW

– opracowanie nowej metody augmentacji danych dla szeregów czasowych opisujących ruch człowieka

– opracowanie nowej metody ekstrakcji cech z wideo (sekwencji obrazów/map głębi) opartej na sieciach bliźniaczych Siamese i cechach Shapelets

– eksploracja sieci bliźniaczych Siamese w różnych rolach: klasyfikacji szeregów czasowych, ekstrakcji cech na pojedynczych mapach głębi

– ocena użyteczności różnych architektur sieci neuronowych w ramach jednego zespołu sieci neuronowych: sieci Siamese, 1D CNN, autoenkodera, LSTM, sieci konwolucyjnych

Struktura rozprawy jest następująca. W rozdziale 2. przedstawiono bardziej szczegółowo problematykę pracy oraz pokrewne prace. W rozdziale 3. zaproponowano rozwiązanie oparte na cechach szeregów czasowych charakteryzowanych przez DTW. W rozdziale 4. zaprezentowano nowe podejście do uczenia zespołowego. Zaproponowano nowy zespół klasyfikatorów NECSCF (Zespół Neuronowy oparty na Cechach Specyficznych dla Klasy i Cechach Wspólnych). Dokonano także ewaluacji algorytmu na zbiorach UCI oraz zbiorach OpenML. Rozdział 5. łączy NECSCF i cechy szeregów czasowych opartych na dynamicznym marszczeniu czasu DTW. W rozdziale porównano zaproponowane podejście do uczenia zespołowego do bardziej konwencjonalnego algorytmu zespołowego opartego na metodzie baggingu. W rozdziale 6. przedmiotem badań były algorytmy, które nie wykorzystują cech tworzonych ręcznie (ang. *handcrafted*). Zbadano użyteczność sieci bliźniaczych Siamese w algorytmach zespołowych. Wielowymiarowe szeregi czasowe cech map głębi zostały wykorzystywane jako podstawa dla algorytmu DTW i algorytmu opartego na Shapelets. Rozdział 7 stanowi zakończenie rozprawy i przedstawia kierunki dalszych prac.

2. Problematyka pracy i powiązane prace

2.1. Rozpoznawanie akcji człowieka

Rozpoznawanie akcji człowieka (Human Action Recognition, HAR) to zadanie w zakresie widzenia komputerowego, polegające na identyfikacji i rozumieniu działań i aktywności człowieka na podstawie sekwencji wideo lub obrazów. Dane dla HAR są rejestrowane za pomocą sensorów takich jak kamery RGB, kamery głębi, noszone czujniki inercyjne itp. Jest to obszar aktywnych badań w dziedzinie widzenia komputerowego [1, 2, 3, 4].

Z HAR wiąże się wiele zagadnień badawczych o znaczącym potencjale aplikacyjnym, w tym wykrywanie człowieka w sekwencji obrazów/map głębi, szacowanie pozy człowieka, śledzenie człowieka, analiza i rozumienie działań człowieka. Rozpoznawanie działań i akcji ludzi znajduje zastosowanie w systemach wspierających interakcje człowiek-komputer, inteligentnym nadzorze, sporcie [5], monitorowaniu domu [6], przechowywaniu i wyszukiwaniu wideo oraz robotyce [7, 8]. W tym celu stosuje się sekwencje obrazów RGB, sekwencje obrazów głębi lub ich połączenie.

Autorzy pracy [9] definiują akcje jako proste wzorce ruchu wykonywane przez pojedynczą osobę o krótkim czasie trwania. Wang et al. [10] podkreślają skutki akcji (w kontekście interakcji z otoczeniem) jako czynnik definiujący. Na tej podstawie Herath et al. [11] proponują własną definicję: "Akcja to najbardziej elementarna interakcja człowieka z otoczeniem mająca znaczenie".

Rozpoznawanie aktywności człowieka jest złożonym zagadnieniem, ponieważ wiąże się z wieloma problemami, takimi jak:

1. Wariacje wewnątrzklasowe i międzyklasowe - ludzie zachowują się różnie podczas wykonywania takich samych akcji, np. w przypadku akcji "bieganie" osoby mogą biegać szybko lub wolno, a nawet skakać podczas biegu. Jedna klasa akcji może więc zawierać wiele różnych stylów ludzkich ruchów. Ponadto, filmy przedstawiające tę samą akcję mogą być nagrane przez różne osoby, a więc przedstawiać wykonawców akcji z różnych punktów widzenia (z góry lub z boku) [12].

2. Niewystarczające rozmiary danych (zbyt mała liczba obrazów/map głębi). Metody uczenia głębokiego uzyskały obiecujące wyniki, ale wymagają zbiorów danych zawierających duże ilości oznakowanych treningowych sekwencji map głębi [13, 14, 15].

3. Dynamiczne tło i inne problemy związane z interakcją osoby z otoczeniem. Ruch kamery jest kolejnym czynnikiem, który należy uwzględnić w aplikacjach operujących na rzeczywistych danych. Ze względu na znaczący ruch kamery, cechy akcji nie mogą być dokładnie wyodrębniane. Inne problemy związane z otoczeniem, jak warunki oświetleniowe, zmiany perspektywy, dynamiczne tło, również stanowią wyzwania, które utrudniają stosowanie algorytmów rozpoznawania akcji w praktycznych zastosowaniach [16].

4. Nie wszystkie mapy głębi są równie dyskryminujące. Sekwencje można skutecznie reprezentować za pomocą niewielkiego zbioru kluczowych map głębi. Oznacza to, że wiele map głębi zawiera obrazy z małą zawartością informacji, a dyskryminujące mapy głębi mogą występować tylko w niektórych fragmentach sekwencji. Na przykład wiele różnych rodzajów akcji ma podobny początek (np. stanie w miejscu, schylanie się) [12, 17].

W tym kontekście warto zauważyć, że większość zbiorów danych do oceny algorytmów rozpoznawania akcji człowieka, z wyjątkiem niedawno wprowadzonego repozytorium NTU-RGB+D [18], nie jest wystarczająco duża dla większości powszechnie stosowanych podejść do uczenia głębokich sieci neuronowych.

2.1.1. Modalności w repozytoriach danych do rozpoznawania akcji człowieka

Publicznie dostępne zbiory danych zawierają zazwyczaj kilka rodzajów modalności takich jak:

RGB - najprostsza modalność, w której każdy punkt obrazu opisywany jest trzema liczbami reprezentującymi intensywność trzech podstawowych kolorów. Choć jest ona najłatwiejsza do zarejestrowania, w wielu zastosowaniach jest mało użyteczna.

Mapy głębi - istnieją trzy główne typy sensorów głębi: kamery stereo, sensory światła strukturalnego i sensory czasu przelotu (ToF). Kamery stereo imitują wzrok człowieka - obrazy z dwóch nieco przesuniętych punktów widzenia są wykorzystywane do rekonstrukcji obrazu w 3D. Sensory światła strukturalnego używają podczerwonego źródła światła do projekcji skomplikowanego wzoru na scenę, który jest niewidoczny dla ludzkiego oka, ale jest widoczny dla sensora. Takie rozwiązanie było stosowane w Microsoft Kinect [19]. Sensory ToF używają impulsu światła podczerwonego i mierzą czas potrzebny na powrót odbitego światła do sensora. Wspomniana technika została zastosowana w Kinect 2. Mapy głębi mogą dostarczać informacji o strukturze 3D, dzięki czemu informacje o ruchu akcji mogą być bardziej dyskryminacyjne.

Ponadto dane reprezentujące głębię sceny są odporne na zmiany oświetlenia, a zmienność koloru i tekstury spowodowana przez odzież, włosy, skórę i tło może zostać zredukowana. Wewnętrznie różnią się od danych RGB, gdyż piksele na mapie głębi kodują informacje o odległości sceny, a nie miarę intensywności koloru [20, 6].

Sensory inercyjne - sensory inercyjne zawierają akcelerometry i żyroskopy wykonujące pomiary przyspieszenia i prędkości kątowej. Sensory inercyjne zostały wykorzystane do rozpoznawania akcji człowieka np. w pracach [21, 22, 23, 5].

Szkielet człowieka - reprezentowany przez pozycje 2D lub 3D stawów człowieka dla każdej mapy głębi w czasie rzeczywistym. Mapy głębi znacznie ułatwiają ekstrakcję szkieletów. Skuteczną metodę estymacji pozy człowieka zaproponowano w [24]. Autorzy stworzyli duży, realistyczny i zróżnicowany zbiór treningowy obrazów syntetycznych, które zostały wykorzystane do wytrenowania Lasu Drzew Losowych (RF), umożliwiającego efektywne wyznaczenie położeń stawów ciała. Omawiany algorytm jest używany w sensorze Kinect.

2.1.2. Podział akcji i zachowań

W [25, 26] działania człowieka zostały podzielone na cztery typy: gesty, akcje, interakcje (z przedmiotami i innymi osobami) oraz aktywności grupowe. Gesty to elementarne ruchy ciała człowieka, które mogą być opisane na poziomie części ciała. Przykładami gestów są "rozciąganie ramion" i "kopnięcie nogą". Akcje są działaniami jednoosobowymi składającymi się z wielu gestów o ustalonej kolejności czasowej. Charakteryzują się ruchami całego ciała. Przykładami akcji są "machanie", "chodzenie" i "bieganie". Interakcje to działania dwóch lub więcej osób i/lub przedmiotów. Przykładami interakcji mogą być "przesunięcie krzesła" lub "zamiatanie". Aktywności grupowe to natomiast działania wykonywane przez wiele osób i/lub przedmiotów, a ich typowymi przykładami są "grupa osób maszerujących" lub "grupa odbywająca spotkanie".

2.2. Algorytmy rozpoznawania akcji i zachowań ludzkich

Wczesne badania skupiały się na cechach tworzonych ręcznie [27], zwłaszcza tych opartych na stawach szkieletowych. Cechy te były tworzone w sposób dostosowany do konkretnego zbioru danych i mogą nie osiągać dobrych rezultatów na innych zbiorach danych. Wraz z rosnącą popularnością uczenia głębokiego opracowano bardziej skuteczne metody oparte na sieciach neuronowych.

Metodami używanymi do rozpoznawania akcji człowieka są m.in.:

1. Sieci konwolucyjne 2D - stosowanie sieci konwolucyjnej 2D w rozpoznawaniu akcji człowieka wymaga adaptacji wejścia - większość sensorów HAR generuje szeregi czasowe lub sekwencje obrazów. Przed zastosowaniem 2D CNN należy przekształcić sekwencje/szeregi czasowe do postaci pojedynczego obrazu. Przykładem takiego przekształcenia jest przekształcenie oparte o Motion history images (MHI) [28, 29]. W MHI jaśniejsze fragmenty obrazu korespondują do obszarów niedawnego ruchu. Otrzymany obraz może opierać się wyłącznie na sekwencji map głębi/obrazów RGB lub też używać innych modalności. Praca [30] jest przykładem rozwiązania używającego sieci kon-

wolucyjnej 2D oraz dwóch różnych modalności (sekwencji RGB oraz stawów szkieletowych), natomiast praca [31] używa trzech modalności (RGB, mapy głębi oraz stawów szkieletowych).

2. Sieci konwolucyjne 3D - najprostszym podejściem do zastosowania uczenia głębokiego w rozpoznawaniu akcji jest użycie sieci z trójwymiarowymi filtrami konwolucyjnymi. Wykorzystywane są dwa wymiary przestrzenne i jeden wymiar czasowy [32]. W przeciwieństwie do metod opartych na sieciach konwolucyjnych 2D operujących na pojedynczych obrazach to podejście nie wymaga stosowania skomplikowanych przekształceń map głębi.

3. Sieci rekurencyjne - rekurencyjne sieci neuronowe (RNN) są powszechnie stosowane w rozpoznawaniu mowy i przetwarzaniu języka naturalnego. Przykład zastosowania RNN do rozpoznawania działań człowieka znajduje się w pracach [33, 34, 35].

4. Metody oparte na cechach tworzonych ręcznie polegają na ekstrakcji cech z danych wideo, takich jak na przykład HOG (ang. *Histogram of Oriented Gradients*) [36], LBP (ang. *Local Binary Patterns*) [37], a następnie na zastosowaniu tradycyjnych algorytmów uczenia maszynowego, takich jak Maszyny Wektorów Wspierających (SVM) [38] lub Lasu Drzew Losowych (ang. *Random Forests*) [39] do klasyfikacji wektorów cech. Metody gradientowe ekstrakcji cech potwierdziły swą użyteczność w wielu metodach [40]. Na przykład LBP wykorzystuje informacje o wartościach gradientów wokół każdego punktu w wektorze binarnym, a następnie oblicza ich histogram. Inne podejście zaprezentowano w pracy [41], w której zaproponowano metodę zwaną jako Histogram of Templates.

5. Sieci dwustrumieniowe - praca [42] zaproponowała metodę klasyfikacji sekwencji obrazów za pomocą zmodyfikowanej dwuwymiarowej sieci konwolucyjnej. Metoda ta została nazwana *Late Fusion*. Polega ona na użyciu dwóch zestawów (strumieni) warstw konwolucyjnych o takich samych wagach. Strumienie przetwarzają obrazy w sekwencji, a następnie są łączone w gęstej warstwie. Wariant tej metody wykorzystano w pracy [43], w której wykorzystano dwa strumienie: przestrzenny i czasowy. Na wejściu strumienia przestrzennego podawany jest pojedynczy obraz RGB. Na wejściu strumienia czasowego przekazywany jest obraz opisujący ruch między obrazami RGB. Aby sklasyfikować sekwencje wybieramy obrazy RGB w równych odstępach czasowych (25 w pracy [43]) i podajemy je na wejście sieci. Ostateczna decyzja jest podejmowana przez uśrednienie wyjścia sieci konwolucyjnych (rozkładzie prawdopodobieństwa przynależności do klasy).

6. Sieci Temporal Segment Networks (TSN) - w celu rozwiązania problemu niemożności modelowania długich struktur czasowych, Wang i inni [32] zaproponowali Temporal Segment Networks (TSN). W omawianym podejściu sekwencja RGB/mapa głębi jest

dzielona na k przedziałów o równej długości, a z każdego przedziału wylosowywana jest jedna mapa głębi/obraz RGB i przekazywana na wejście sieci konwolucyjnej. Ostateczna decyzja jest podejmowana na podstawie wyjścia k sieci konwolucyjnych poprzez funkcję agregującą. Istnieje wiele możliwych funkcji agregujących, poprzez uśrednianie (ang. *averaging*), max pooling etc. W pracy [32] najlepsze rezultaty uzyskano w oparciu o metodę agregacji opartą na uśrednianiu.

7. Transformer z mechanizmem *Attention* - mechanizmy Attention [44] umożliwiają skupienie się na istotnych obszarach lub klatkach w sekwencji wideo, pozwalając modelowi koncentrować się selektywnie na tych częściach danych, które posiadają największą siłę dyskryminacyjną. Pomaga to w uchwyceniu relacji przestrzenno-czasowych przydatnych dla rozpoznawania akcji. Mechanizm *Attention* oryginalnie opracowano dla problemów z dziedziny przetwarzania języka naturalnego, gdzie osiągnięto bezprecedensową poprawę wyników [44]. Metoda ta uzyskuje również lepsze wyniki w zadaniach klasyfikacji sekwencji obrazów [45, 46, 47], dla zbiorów takich jak Kinetics [48], Moments in Time [49], Epic-Kitchens-100 [50], Something-Something V2 [51].

8. Grafowa sieć konwolucyjna (GCN) - grafowa sieć neuronowa to rodzaj sieci neuronowej, która bezpośrednio działa na strukturze grafu. W pracy [52] zaproponowano nową reprezentację sekwencji szkieletowych do rozpoznawania akcji rozszerzając grafowe sieci neuronowe o model grafu przestrzennego-czasowego, zwany Spatial-Temporal Graph Convolutional Networks (STGCN).

2.3. Uczenie zespołowe

Uczenie zespołowe (ang. *Ensemble Learning*) to rodzaj uczenia maszynowego, w którym wiele modeli jest łączonych w celu uzyskania lepszej dokładności niż mogłaby zostać osiągnięta przez pojedynczy model [53]. Jeśli każdy indywidualny klasyfikator popełnia niezależne błędy, to agregacja wyjść klasyfikatorów może zmniejszyć całkowity błąd klasyfikacji [54, 55].

Możliwa jest dekompozycja całkowitego błędu na bias i wariancję. Zilustrowano to na przykładzie problemu regresji [54].

$$Err(x_i) = E[(Y - \hat{f}(x_i))^2 | X = x_i] \tag{2.1}$$

$$Err(x_i) = \sigma_\epsilon^2 + [E\hat{f}(x_i) - f(x_i)]^2 + E[\hat{f}(x_i) - E\hat{f}(x_i)]^2 \tag{2.2}$$

$$Err(x_i) = Nieredukowalny\ blad + Bias^2 + Wariancja \tag{2.3}$$

gdzie:

X - zmienna losowa.

$Y = f(X) + \epsilon$

$f(X)$ - prawdziwy model

ϵ - zmienna losowa o średniej równej zero

$\hat{f}(X)$ - model wyznaczony na podstawie danych treningowych

σ_ϵ^2 - wariancja zmiennej ϵ

Uczenie zespołowe na podstawie zbioru modeli o dużej wariancji tworzy jeden model o mniejszej wariancji. W przypadku algorytmu Lasu Drzew Losowych (RF) wariancja określana jest na podstawie zależności [54]:

$$Var[F_{RF}(x)] = \rho(x)\sigma^2 \tag{2.4}$$

gdzie:

F_{RF} - model wyznaczony przez RF na podstawie danych treningowych

$\rho(x)$ - korelacja między dwoma losowo wybranymi drzewami w zespole

σ^2 - wariancja pojedynczego drzewa

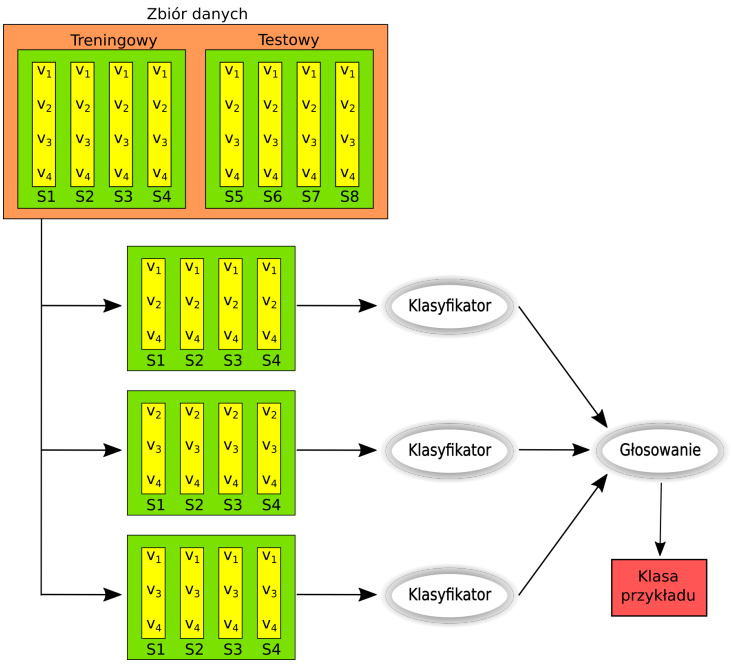
Główną regułą w uczeniu zespołowym jest więc użycie jak najbardziej zróżnicowanych klasyfikatorów pod względem błędnie sklasyfikowanych przykładów. Taki zestaw klasyfikatorów jest uważany za różnorodny (ang. *diverse*), a różnorodność klasyfikatorów można osiągnąć na kilka sposobów [56, 57, 58]:

1. Bagging - poszczególne klasyfikatory są trenowane na różnych podzbiorach pełnego zbioru danych, które są losowo wybierane bez zwracania [59].
2. Używanie różnych hiperparametrów dla różnych klasyfikatorów.
3. Losowe podprzestrzenie - poszczególne klasyfikatory są trenowane na różnych podzbiorach cech (losowo wybieranych bez zwracania) [39]. Przykładem takiego algorytmu jest Las Drzew Losowych (gdzie użyte klasyfikatory to drzewa decyzyjne).
4. Boosting (np. AdaBoost, gradient boosting) - klasyfikatory są trenowane w kilku iteracjach, a w każdej iteracji próbki źle sklasyfikowane przez klasyfikatory z poprzednich iteracji otrzymują większą wagę podczas trenowania nowych klasyfikatorów [60].
5. Wykorzystanie klasyfikatorów trenowanych na różnych modalnościach.

Istnieje wiele różnych miar różnorodności klasyfikatorów. Są to między innymi: statystyka Q [61], współczynnik korelacji, miara podwójnej pomyłki [62], wariancja Kohavi-Wolperta itp. [63].

2.3.1. Random Forest

Kluczową ideą stojącą za algorytmem Lasu Drzew Losowych jest połączenie wielu drzew decyzyjnych [39]. Każde drzewo lasu jest trenowane na innym podzbiorze cech, co zapewnia



Rysunek 2.1: Zespół klasyfikatorów RF. Zestaw treningowy zawiera przykłady $S_1....S_N$. Każdy przykład składa się z równiej liczby cech $v_1....v_k$. Każdy klasyfikator w zespole jest trenowany na tych samych przykładach, ale na różnych podzbiorach cech.

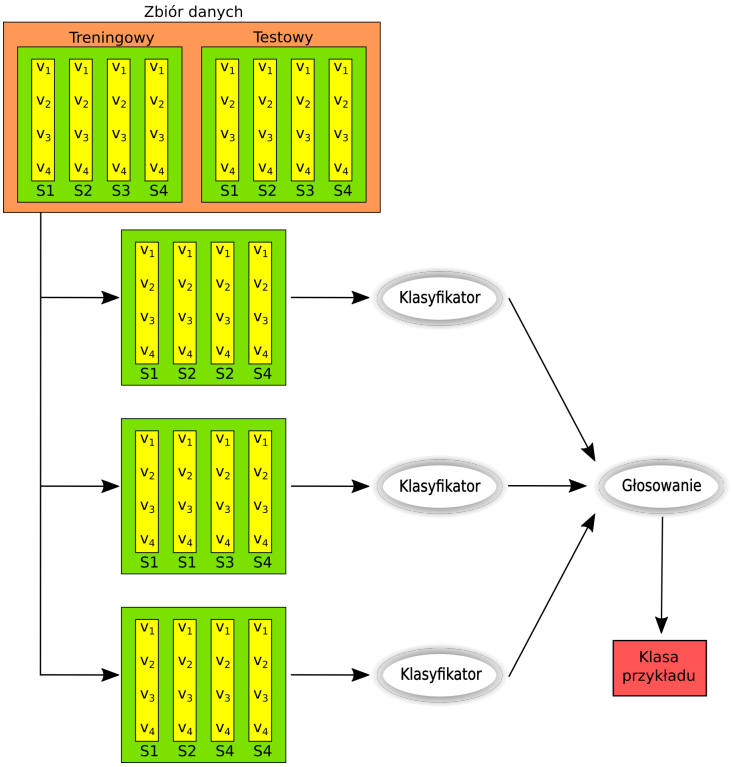
różnorodność klasyfikatorów w procesie uczenia. Ta różnorodność pomaga w redukcji nadmiernego przetrenowania i poprawia ogólną wydajność generalizacji. Algorytm Lasu Drzew Losowych zaprezentowany jest na rysunku 2.1. Aby dokonać predykcji na nowych danych, każde drzewo decyzyjne lasu niezależnie generuje predykcję. W przypadku klasyfikacji jako końcową predykcję wybiera się klasę, która otrzymała większość głosów od wszystkich drzew. W przypadku regresji jako końcową predykcję przyjmuje się średnią predykcji z poszczególnych drzew.

2.3.2. Bagging

Algorytm Bagging (Bootstrap Aggregating) to metoda uczenia zespołowego, która łączy wiele modeli bazowych, trenowanych na różnych podzbiorach danych, w celu dokonania predykcji [59]. Na rysunku 2.2 zilustrowano w sposób schematyczny działanie algorytmu bagging.

1. Przygotowanie danych: na podstawie zbioru treningowego algorytm Bagging tworzy wiele próbek bootstrapowych przez losowe próbkowanie z powtórzeniami z oryginalnego zbioru danych. Każda próbka bootstrapowa ma ten sam rozmiar co oryginalny zbiór danych, ale może zawierać zduplikowane instancje.

2. Trenowanie podstawowych modeli: dla próbki bootstrapowej trenuje się niezależnie model (często nazywany słabym klasyfikatorem) na odpowiadającej próbce. Model pod-



Rysunek 2.2: Zespół klasyfikatorów Bagging. Zestaw treningowy zawiera przykłady $S_1....S_N$. Każdy przykład składa się z równej liczby cech $v_1....v_k$. Każdy klasyfikator w zespole jest trenowany na tych samych cechach, ale na różnych podzbiorach przykładów.

stawowy może być uzyskany za pomocą dowolnego algorytmu uczenia maszynowego, takiego jak drzewo decyzyjne, sieć neuronowa czy maszyna wektorów nośnych.

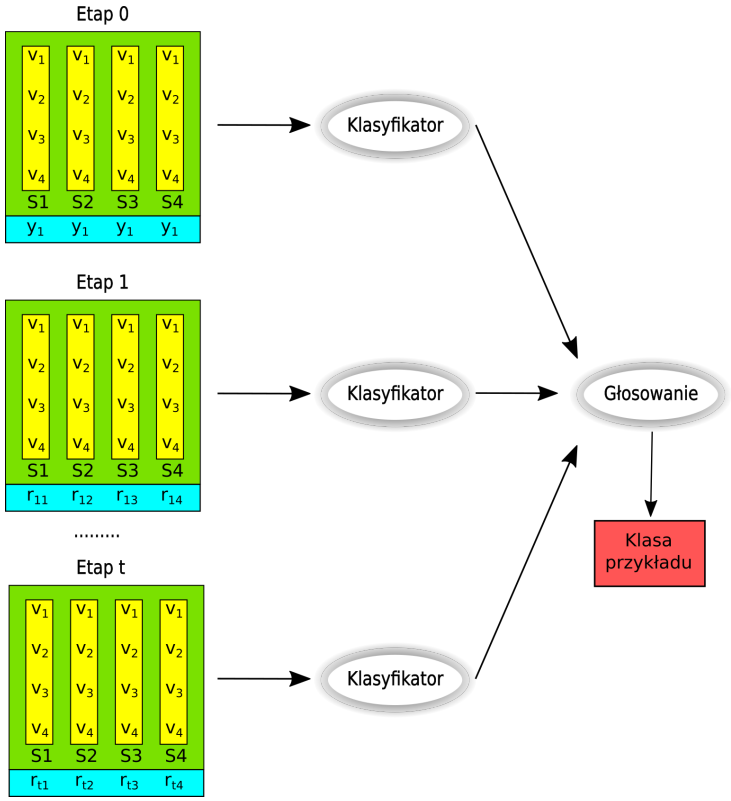
3. Agregacja: po trenowaniu wszystkich modeli bazowych Bagging łączy ich predykcje przez uśrednianie (dla zadań regresji) lub głosowanie (dla zadań klasyfikacji). Dla regresji, jako końcową predykcję przyjmuje się średnią indywidualnych predykcji. Dla klasyfikacji wybierana jest klasa, która otrzymała większość głosów jako końcowa predykcja.

Kluczowa idea stojąca za metodą Bagging polega na trenowaniu wielu modeli na różnych podzbiorach próbek wylosowanych z pełnego zbioru danych i łączeniu ich predykcji, co redukuje wariancję i pomaga zredukować przeuczenie. Otrzymany zespół modeli ma lepsze zdolności uogólniające [64].

2.3.3. Gradient Boosting

Gradient Boosting jest algorytmem uczenia zespołowego, który łączy wiele słabych modeli predykcyjnych (zwykle drzew decyzyjnych) w celu stworzenia silnego modelu predykcyjnego [60]. Główna idea stojąca za Gradient Boosting polega na iteracyjnym budowaniu zbioru słabych modeli, gdzie każdy kolejny model jest trenowany w celu korygowania błędów popełnianych przez poprzednie modele. Końcowy model zespołu jest ważoną kombinacją tych

słabych modeli. Wagi słabych modeli (w głosowaniu ważonym) są dobierane w celu minimalizacji funkcji strat. Najczęściej używaną funkcją strat jest błąd średnio-kwadratowy. Używanie innych funkcji strat jest możliwe, ale muszą one być różniczkowalne [54]. Algorytm Gradient Boosting zaprezentowano na rysunku 2.3. Działanie algorytmu można zilustrować w następujący sposób:



Rysunek 2.3: Zespół klasyfikatorów Gradient Boosting. Zbiór treningowy zawiera próbki $S_1....S_N$. Każda próbka składa się z równej liczby cech $v_1....v_k$. Klasyfikatory są uczone etapami. Klasyfikatory są trenowane na wszystkich przykładach i wszystkich cechach, ale każdy etap ma inną funkcję straty. Funkcja straty na etapie k przykłada większą wagę do próbek źle sklasyfikowanych na etapie $k-1$.

1. Inicjalizacja: na początku tworzony jest prosty model podstawowy (na przykład drzewo decyzyjne).

2. Iteracyjne trenowanie modeli: następnie w każdej iteracji trenowany jest słaby model, który koryguje błędy poprzednich modeli. Błąd jest wyznaczany jako różnica między prawdziwymi etykietami (ang. *ground-truth*), a predykcjami poprzednich modeli. Słaby model jest trenowany tak, aby minimalizować ten błąd.

3. Kombinacja modeli: końcowy model zespołu jest ważoną kombinacją wszystkich słabych modeli. Predykcja końcowa jest dokonywana przez ważone zsumowanie predykcji wszystkich słabych modeli.

2.3.4. Uczenie zespołowe i sieci neuronowe

Uczenie zespołowe może wykorzystać każdy algorytm klasyfikacji jako podstawowy model, w tym sieci neuronowe (zarówno płytkie jak głębokie). W ostatnich latach rośnie zainteresowanie zarówno tradycyjnym uczeniem zespołowym jak i uczeniem zespołowym opartym na modelach głębokich [65]. Modele głębokie mogą zawierać miliardy parametrów, co może utrudniać wykorzystanie uczenia zespołowego. Z drugiej strony uczenie zespołowe może wykorzystać różnorodność architektur głębokich sieci neuronowych. Przykładem jest praca [66], w której w zespole użyto trzech rodzajów architektury sieci (RetinaNet, Deep SVDD, i CNN). Stworzony w ten sposób zespół głębokich sieci neuronowych został zastosowany do diagnozy medycznej. Zespół różnorodnych architektur sieci głębokich znalazł zastosowanie w badania sentymentu tekstu w pracy [67]. Użyto w niej pięciu rodzajów architektury uczenia głębokiego (LSTM, GRU, CNN, RCNN, DNN). Inną właściwością modeli głębokich jest duża liczba hiperparametrów (np. algorytm optymalizacji, współczynnik uczenia, dropout, regularyzacja L1). Można więc stworzyć zespół składający się z modeli głębokich o różnej wartości hiperparametrów.

W zespołach opartych na sieciach neuronowych często zamiast prostego głosowania stosuje się stacking [65]. Oznacza to, że wyjście klasyfikatorów jest przekazywane na wejście następnego klasyfikatora. Przykładem takiego podejścia jest praca [68], w której stacking użyto do rozpoznawania obrazów.

Głębokie sieci neuronowe są często stosowane wraz z metodą boosting. W pracy [69] zespół boosting z siecią konwolucyjną jako podstawowym modelem został użyty do klasyfikacji ubrań. W pracy [70] zespół boosting oparty na sieciach CNN został użyty do zliczania obiektów na obrazie. W niektórych pracach stosuje się metodę bagging, przykładowo w pracy [71], w której bagging wykorzystano do badania sentymentu tekstu. W pracy [72] zastosowano zespół modeli głębokich wykorzystujący boosting i sieć DNN do przewidywania chorób serca.

Losowe podprzestrzenie są podejściem rzadko stosowanym razem z sieciami głębokimi. Istnieją jednak wyjątki jak na przykład praca [73]. Również popularna metoda regularyzacji sieci neuronowych dropout może być uznawana jako forma uczenia zespołowego [74]. Dropout polega na losowym wyłączeniu (z prawdopodobieństwem p) połączeń między neuronami w czasie trenowania sieci neuronowych.

3. Cechy DTW do rozpoznawania akcji ludzi

DTW jest jednym z najpopularniejszych algorytmów do klasyfikacji szeregów czasowych. Jak pokazuje praca [75] trudno jest stworzyć algorytm, który osiąga lepsze wyniki niż DTW w klasyfikacji szeregów czasowych. Pomimo tego algorytm DTW jest stosunkowo rzadko używany do rozpoznawania akcji ludzkich. Wynika to z faktu, że w typowym podejściu rozpoznawanie akcji ludzi realizowane jest na sekwencjach obrazów RGB lub map głębi. W oparciu o wspomniany algorytm, zaproponowano nowy zestaw cech dla szeregów czasowych (cechy DTW) i wykorzystano je do rozpoznawania akcji człowieka. Opracowano również zestaw cech tworzonych ręcznie (wyznaczanych na mapach głębi) na potrzeby rozpoznawania akcji człowieka.

3.1. Algorytm DTW

Dynamiczne marszczenie czasu (DTW, ang. *Dynamic Time Wraping*) jest efektywnym algorytmem do mierzenia podobieństwa między dwoma szeregami czasowymi, które mogą różnić się długością. DTW wyznacza optymalne dopasowanie między dwiema sekwencjami danych [76]. Jednym z najskuteczniejszych algorytmów do klasyfikacji szeregów czasowych jest 1-NN-DTW, który jest specjalnym przypadkiem klasyfikatora k-najbliższych sąsiadów z k=1, oraz DTW do pomiaru odległości [77] między szeregami czasowymi. W DTW sekwencje są nieliniowo deformowane w wymiarze czasowym, aby uzyskać najlepsze dopasowanie między dwoma szeregami czasowymi.

Jako dane wejściowe do DTW mamy dwa szeregi czasowe $l = l_1 \dots l_{|l|}$ i $s = s_1 \dots s_{|s|}$, gdzie $|l|$ i $|s|$ to odpowiednio długości sekwencji l i s (niekoniecznie równe). W przypadku wielowymiarowego szeregu czasowego l oznaczamy wymiar k jako $l[k]$, zaś wartość wymiaru k w wektorze i jako $l_i[k]$.

Oznaczmy $D_{l,s}(i, j)$ jako odległość DTW między podsekwencjami $l_1 \dots l_i$ i $s_1 \dots s_j$. Odległość DTW między l i s można obliczyć za pomocą algorytmu programowania dynamicznego [78], zgodnie z następującym równaniem iteracyjnym:

$$D_{l,s}(i,j) = \begin{cases} D_{l,s}(0,0) = 0 & \\ D_{l,s}(i,0) = \infty & i{>}0 \\ D_{l,s}(0,j) = \infty & j{>}0 \\ min\{D_{l,s}(i-1,j-1), D_{l,s}(i-1,j), D_{l,s}(i,j-1)\} + \|l_i, s_j\| & i{>}0\ j{>}0 \end{cases} \tag{3.1}$$

Złożoność obliczeniowa wyznaczania odległości DTW wynosi $O(|l| \times |s|)$, gdzie $|l|$ i $|s|$ to odpowiednio długości sekwencji l i s. Porównanie algorytmów aproksymujących odległość DTW znajduje się w pracy [79].

Do zalet DTW należy:

1. Możliwość stosowania do szeregów czasowych o różnej długości.
2. Odporność na szum. Dwa szeregi mogą być podobne nawet jeśli istnieją znacznie różniące się subszeregi.
3. Użyteczność w klasyfikacji akcji wykonywanych z różną prędkością.

Zazwyczaj DTW jest obliczany dla szeregów jednowymiarowych. Dla szeregów wielowymiarowych DTW można uogólnić na dwa sposoby (opisane w pracach [80, 81]). W metodzie wektorowej (zilustrowanej na rysunku 3.1b) DTW obliczamy dla wszystkich wymiarów razem. Odległość między poszczególnymi wektorami szeregu czasowego jest obliczana za pomocą metryki Euklidesowej.

$$\|l_i, s_j\| = \sqrt{\sum_{k=1}^{|v|} (l_i[k] - s_j[k])^2} \tag{3.2}$$

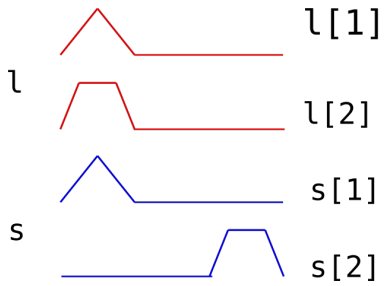
W metodzie skalarnej (zilustrowanej na rysunku 3.1c) obliczamy DTW dla każdego wymiaru osobno, a następnie sumujemy obliczone odległości:

$$DTW_{l,s}(i,j) = \sum_{k=1}^{|v|} (D_{l[k],s[k]}(i,j)) \tag{3.3}$$

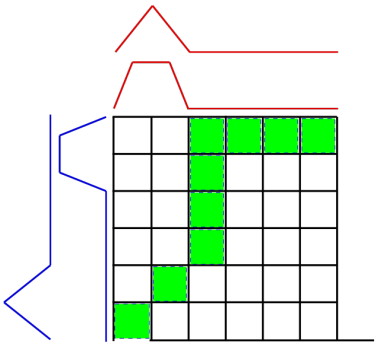
W niniejszej pracy zawsze używana jest metoda wektorowa.

3.2. Cechy map głębi tworzone ręcznie

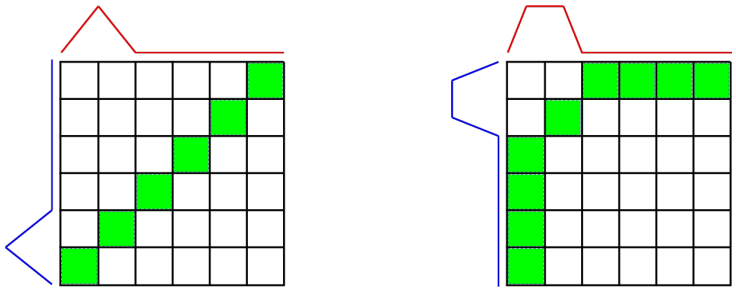
Z uwagi na to, że zasadnicza część niniejszej pracy dotyczy rozpoznawania akcji ludzi na mapach głębi nie możemy zastosować algorytmu DTW bezpośrednio. Przed użyciem DTW konieczne jest wyznaczenie cech z pojedynczych map głębi. Poniżej zaproponowano zestaw cech tworzonych ręcznie [82].



(a) Dane wejściowe



(b) Metoda wektorowa



(c) Metoda skalarna

Rysunek 3.1: Dwie metody uogólnienia DTW na przypadek wielowymiarowy. Pokazano dwa dwuwymiarowe szeregi czasowe l i s. Rysunek b ilustruje DTW obliczone za pomocą metryki euklidesowej dla dwuwymiarowego szeregu czasowego. Rysunek c ilustruje DTW wyznaczone dla poszczególnych wymiarów dwuwymiarowego szeregu czasowego. Jak można zaobserwować omawiane metody mogą dawać różne wyniki.

Dla każdej mapy głębi obliczamy cechy opisujące kształt osoby. Przed rozpoczęciem obliczania cech na mapach głębi wymagane jest wydzielenie osoby na mapie głębi. Rzutujemy otrzymane mapy głębi na trzy ortogonalne płaszczyzny kartezjańskie, aby modelować trójwymiarowy kształt i informacje o ruchu czynności ludzkich. Oznacza to, że wyznaczamy rzutowania z boku, z przodu i z góry map głębi. Mapa głębi otrzymana przez sensory głębi jest rzutowana na trzy dwuwymiarowe płaszczyzny kartezjańskie, gdzie płaszczyzna xy przedstawia widok frontalny, płaszczyzna yz ilustruje widok z boku, a płaszczyzna xz widok z góry.

Do obliczania cech wykorzystywane są tylko piksele reprezentujące wyodrębnioną osobę na mapach głębi. W zbiorach danych badanych w niniejszym rozdziale tło usunięto zawczasu. Jedynym przetwarzaniem wstępnym było wyodrębnienie prostokąta ograniczającego dla pikseli niezerowych (prostokąt ma takie same rozmiary dla wszystkich map głębi w sekwencji). Wybrano układ współrzędnych, w którym środek masy (obliczony dla punktów dla wszystkich map głębi w sekwencji) jest równy zeru. Współrzędne punktów są przeskalowane do zakresu [-1, 1], a większe wartości z reprezentują piksele bliżej kamery (obserwatora).

Na takich mapach głębi zostały wyznaczone następujące cechy map głębi:

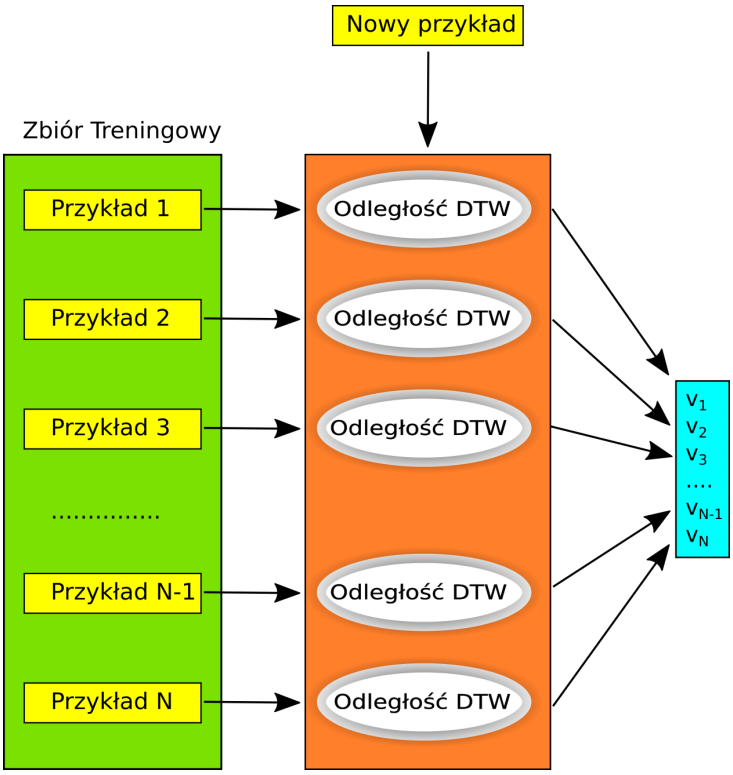
1. Proporcje obszaru (obliczany tylko dla frontalnej projekcji mapy głębi na osiach x, y) zajmowanego przez osobę w stosunku do całkowitej liczby pikseli na mapie głębi.
2. Odchylenie standardowe (osie x, y, z).
3. Skośność (osie x, y, z).
4. Korelacja między współrzędnymi pikseli (osie xy, xz i zy).
5. Współrzędne (x i y), dla których odpowiadająca wartość głębi z reprezentuje najbliższy piksel do kamery.

Oznacza to, że kształt osoby na każdej mapie głębi jest opisany dwunastoma cechami. Akcja człowieka reprezentowana przez sekwencje map głębi jest opisana jako wielowymiarowy (liczba wymiarów to 12) szereg czasowy o długości równej liczbie map głębi w danej sekwencji.

3.3. Cechy DTW

Metoda wyznaczania cech DTW jest zilustrowana na rysunku 3.2. Obliczono odległości DTW między wszystkimi możliwymi parami sekwencji map głębi w zbiorze treningowym. Dla każdej sekwencji map głębi obliczono odległości DTW dla zestawów cech 1, 2, 3 i 4, które przedstawiono w tabeli 3.1. Następnie odległości DTW zostały użyte jako cechy. Oznacza to, że końcowy wektor cech ma rozmiar $N \times 4$, gdzie N oznacza liczbę sekwencji map głębi w zbiorze treningowym. Do obliczania cech DTW użyto implementacji algorytmów DTW z biblioteki DTADistance [83].

W pseudokodzie 1 został zaprezentowany algorytm ekstrakcji cech DTW dla zagadnienia rozpoznawania akcji. Wejściem algorytmu jest lista wszystkich szeregów czasowych (all_series) oraz lista treningowych szeregów czasowych (train_series). Obliczamy odległość DTW między danym szeregiem czasowym (funkcja oznaczona jako DTW) i każdym szeregiem czasowym w zbiorze treningowym. W oparciu o odległości DTW wyznaczone tym sposobem tworzony jest wektor cech, który opisuje dany szereg czasowy (linie 3-4). Na wyjściu zwracane są wektory cech DTW.



Rysunek 3.2: Cechy DTW obliczone na zbiorze treningowym składającym się z N przykładów. Każdy przykład to wielowymiarowy szereg czasowy reprezentujący akcję człowieka. Liczba cech (rozmiar wektora cech DTW) jest równa liczbie przykładów w zbiorze treningowym (N). Każda cecha to odległość DTW nowego przykładu od jednego z przykładów w zbiorze treningowym.

Podsumowując, cechy mają postać:

$$dtw\_feats(s) = [DTW(s, l_1), DTW(s, l_2), \ldots, DTW(s, l_{|v|})] \tag{3.4}$$

gdzie:

s - nowy przykład

$l_1, l_2, \ldots, l_{|v|}$ - przykłady ze zbioru treningowego.

Po wyznaczeniu cech DTW na całym zbiorze treningowym otrzymujemy tablicę o wymiarach $|all\_samples| \times |train\_samples|$, gdzie $|all\_samples|$ oznacza liczbę wszystkich przykładów, zaś $|train\_samples|$ oznacza liczbę przykładów w zbiorze treningowym. Oznacza to, że problem klasyfikacji akcji człowieka reprezentowanych przez wielowymiarowe szeregi czasowe został sprowadzony do wyznaczenia dwuwymiarowej tablicy. Postać tablicy (w wierszach kolejne przykłady, zaś w kolumnach cechy) jest podobna do postaci, jakie są zwyczajowo wykorzystywane w zadaniach klasyfikacji danych tabelarycznych (ang. *tabular data*).

W pseudokodzie 2 zaprezentowano proponowany algorytm klasyfikacji na podstawie cech DTW oznaczony jako DTW_CLF. Wejściem algorytmu jest lista sekwencji map głębi (all_actions), lista etykiet (all_labels) oraz listy indeksów zbioru treningowego (train_indexes) i testowego (test_indexes). Przekazane jako argumenty listy all_actions i all_labels zawierają

Algorytm 1 Pseudokod dla algorytmu ekstrakcji cech DTW.

```
1: function DTW_FEATURES(all_series,train_series)
2:     for series_s ∈ all_series do
3:         for series_l ∈ train_series do
4:             all_dtw_feats[s][l] ← DTW(series_s, series_l)
5:         end for
6:     end for
7:     return all_dtw_feats
8: end function
```

wszystkie przykłady zarówno ze zbioru treningowego jak i testowego. Dodatkowo przekazujemy indeksy zbioru treningowego oraz zbioru testowego, które są używane do wydzielenia przykładów składających się na zbiór treningowy i testowy. Najpierw na podstawie akcji (sekwencji map głębi) obliczamy cechy - dla każdej indywidualnej mapy głębi wyznaczany jest osobny wektor cech o stałej długości. Wektory są wyznaczane przez funkcję COMPUTE_HC_FEATURES. W rezultacie otrzymujemy wielowymiarowy szereg czasowy (linie 2-6). Wektory cech wyznaczone za pomocą funkcji DTW_FEATURES są dzielone na zbiór testowy i treningowy. Dokonujemy selekcji cech (linie 11-13) za pomocą algorytmu RFE (ang. *Recursive Feature Elimination* [84]). Funkcja RFE.FIT wyszukuje optymalny podzbiór cech, natomiast RFE.TRANSFORM przekształca wektory cech. Zbiór treningowy jest używany do wytrenowania klasyfikatora (linia 15). Klasyfikator jest następnie używany do predykcji etykiet wektorów ze zbioru testowego (linia 16).

Algorytm k-NN-DTW operuje bezpośrednio na szeregach czasowych (szeregi czasowe mogą być różnej długości). Algorytm wyznaczania cech DTW przyporządkowuje szeregom czasowym wektory o stałej długości. W przypadku cech DTW możliwe jest więc zastosowanie standardowych algorytmów selekcji cech, co jest niemożliwe w k-NN-DTW. Cechy DTW umożliwiają połączenie zalet DTW i selekcji cech. W szczególności, selekcja cech umożliwia zmniejszenie globalnie dużego kosztu obliczeniowego cech DTW. Nie zmniejsza złożoności obliczeniowej wyliczenia pojedynczej odległości DTW. Redukuje jedynie liczbę wyliczanych odległości.

Jak już wspomniano, do selekcji cech użyto algorytmu RFE [84]. RFE poszukuje najbardziej dyskryminującego podzbioru cech. Dzieje się to przez dopasowanie klasyfikatora (w naszym przypadku SVM (Support Vector Machine) [85]) do początkowego zbioru cech, ocenę znaczenia cech, odrzucenie najmniej istotnych cech i ponowne dopasowanie klasyfikatora. Procedura ta jest powtarzana rekurencyjnie, aż do osiągnięcia określonej liczby cech. W niniejszym rozdziale selekcje cech realizowano na tablicy o rozmiarze $number\_samples \times number\_of\_training\_samples$ (tzn. tablicy wyznaczonej przez Algorytm 1.). Wynikiem końcowym jest tablica o rozmiarze $number\_samples \times 400$, tzn. wybrano 400 cech o największej sile dyskryminacyjnej.

Algorytm 2 Pseudokod dla klasyfikacji akcji na podstawie cech DTW.

function DTW_CLF(all_actions,all_labels,train_indexes,test_indexes)
  for $action\_i \in all\_actions$ **do**
    for $frame\_j \in action\_i$ **do**
      $all\_series[i][j] \leftarrow COMPUTE\_HC\_FEATURES(frame\_j)$
    end for
  end for
  $train\_series \leftarrow all\_series[train\_indexes]$
  $all\_dtw\_feats \leftarrow DTW\_FEATURES(all\_series, train\_series)$
  $train\_dtw\_feats \leftarrow all\_dtw\_feats[train\_indexes]$
  $test\_dtw\_feats \leftarrow all\_dtw\_feats[test\_indexes]$
  $RFE.FIT(train\_dtw\_feats, train\_labels)$
  $train\_dtw\_feats \leftarrow RFE.TRANSFORM(train\_dtw\_feats)$
  $test\_dtw\_feats \leftarrow RFE.TRANSFORM(test\_dtw\_feats)$
  $train\_labels \leftarrow all\_labels[train\_indexes]$
  $CLF\_ALG.FIT(train\_dtw\_feats, train\_labels)$
  $predicted\_labels \leftarrow CLF\_ALG.PREDICT(test\_dtw\_feats)$
  return $predicted\_labels$
end function

W niniejszym rozdziale do klasyfikacji akcji reprezentowanych przez proponowane cechy DTW użyto klasyfikatora SVM. Wykorzystano implementację ze scikit-learn [86] i użyto domyślnej wartości parametrów. Używamy RBF (ang. *radial basis function*) jako kernel. Współczynnik regularyzacji jest równy 1 (im mniejsza wartość tym silniejsza regularyzacja) - problem nie wymaga silnej regularyzacji. Współczynnik gamma ma wartość 'scale' - oznacza to, że wartość gamma jest wyznaczana ze wzoru:

$$gamma = \frac{1}{number\_of\_features \times variance} \tag{3.5}$$

gdzie:

$number\_of\_features$ - liczb cech w zbiorze treningowym

$variance$ - wariancja wszystkich wartości cech w zbiorze danych (z każdego wektora)

Złożoność obliczania cech DTW to w najlepszym przypadku $O(N \times TS_{min})$, a w najgorszym przypadku $O(N \times TS_{max})$

gdzie:

N - liczba wszystkich szeregów czasowych w zbiorze treningowym.

TS_{min} - długość najkrótszego szeregu w zbiorze.

TS_{max} - długość najdłuższego szeregu w zbiorze.

Podobna metoda do opisanej została opublikowana mniej więcej w tym samym czasie w pracy [87]. Konkurencyjny algorytm operuje na danych szkieletowych, zaś liczba cech

jest równa liczbie klas (reprezentant klasy wybierany jest losowo). W niniejszej pracy używamy wyłącznie map głębi i badamy możliwości jakie daje łączenie cech DTW opartych na różnych zestawach cech tworzonych ręcznie (wyznaczanych na podstawie pojedynczej mapy głębi). Dodatkowo w naszym algorytmie obliczamy odległość dla każdego przykładu w zbiorze treningowym, a następnie dokonujemy selekcji cech o największej sile dyskryminacyjnej za pomocą algorytmu RFE.

Aby użyć metryki DTW do klasyfikacji szeregów czasowych zazwyczaj używa się ich jako odległości w algorytmie k-NN-DTW [75]. Oryginalność proponowanego rozwiązania polega na użyciu odległości do reprezentacji szeregu czasowego jako wektora cech. Umożliwia to zastosowanie go razem z dowolnym algorytmem klasyfikacji.

Przykład zastosowania DTW do rozpoznawania akcji znajduje się w pracy [88]. Dokonywana jest tam klasteryzacja szeregów czasowych, a następnie konstruowany jest szereg czasowy reprezentujący każdy klaster. Jako cechy używane są odległości Subsequence DTW (wariant odległości DTW zaproponowany w pracy [88]) od szeregów czasowych reprezentujących daną klasę. Inny przykład znajduje się w pracy [89]. Do klasyfikacji akcji zastosowano tam k-NN-DTW z pięcioma wybranymi szeregami czasowymi z każdej klasy.

We wspomnianych wyżej pracach algorytm DTW klasyfikuje szeregi czasowe uzyskane z akcelerometru - zaproponowany algorytm operuje na cechach wyznaczanych na mapach głębi. W pracy [75] wybiera się losowo szeregi czasowe (z każdej klasy). Natomiast w pracy [88] konstruuje się mniejszy zbiór reprezentujący zbiór treningowy poprzez klasteryzację szeregów czasowych, a następnie konstruuje się reprezentanta każdego klastra. W zaproponowanym algorytmie dokonujemy selekcji szeregów umożliwiających najlepszą dyskryminację klas. W [75] używa się k-NN-DTW, zaś w [88] jako cech używa się odległości od szeregów czasowych reprezentujących klastry. W proponowanym algorytmie używa się odległości od szeregów czasowych w zbiorze treningowym.

Wyniki badań uzyskane w niniejszej pracy pokazują, że podzielenie wielowymiarowego szeregu czasowego na wiele mniejszych (o mniejszej liczbie wymiarów) szeregów czasowych, a następnie wyznaczenie cech DTW na podstawie podzbiorów cech daje lepsze rezultaty niż na pierwotnym zestawie cech. Podzbiory cech tworzonych ręcznie są przedstawione w tabeli 3.1.

W pracy [90] pokazano jak użyć DTW do łączenia różnych modalności danych. Szczegółowy opis wykorzystania k-NN-DTW na podstawie danych szkieletowych znajduje się w pracy [91].

Tabela 3.1: Podzbiór cech tworzonych ręcznie do opisu kształtu osoby na pojedynczej mapie głębi.

ID	Nazwa	Liczba cech	Opis	Osie
I	corl	3	Korelacja	zy,xz,zy
II	max z	2	Współrzędne największej wartości piksela na mapie głębi	x,y
III	std	3	Odchylenie standardowe	x,y,z
IV	skew	3	Skośność	x,y,z

3.4. Wyniki prac eksperymentalnych

Ewaluację algorytmu ekstrakcji cech DTW zrealizowano na trzech ogólnie dostępnych zbiorach danych: MSR-Action3D [92], UTD-MHAD [93] oraz 3DHOI [94].

3.4.1. MSR-Action3D

MSR-Action3D [92] jest jednym z najczęściej wykorzystywanych zbiorów danych w badaniach oraz do oceny algorytmów klasyfikacji sekwencji map głębi. Zbiór danych zawiera 567 sekwencji głębi i 20 akcji. Sekwencje map głębi zostały zarejestrowane przez sensor Kinect. Akcje zostały wykonane przez 10 osób. Mapy głębi pobrano z częstotliwością 15 klatek na sekundę. W sumie w zbiorze danych (we wszystkich sekwencjach) znajduje się 23797 map głębi. Rozmiar mapy głębi to 640×480 pikesli. Przykładowe mapy głębi ze zbioru MSR są zaprezentowano na rys. 3.3.

Akcje znajdujące się w zbiorze MSR to: uniesienie ręki, wymach ręką, uderzenie młotkiem, łapanie ręką, cios w przód, rzut w górę, rysowanie litery "x", rysowanie kreski, rysowanie okręgu, klaskanie w dłonie, wymach dwoma rękami, boksowanie na boki, pochylenie się, kopnięcie w przód, kopnięcie na bok, bieganie, zamach tenisowy, serwis tenisowy, zamach golfowy, podniesienie i rzut.

Ewaluacja algorytmów na zbiorze danych MSR jest realizowana na dwóch protokołach. Pierwszy protokół ewaluacji (oznaczony jako split I) wykorzystywano w pracach [95, 96]. W omawianym protokole ewaluacji międzyosobowej w pojedynczym eksperymencie użyto wszystkie 20 klas. Połowa osób oznaczonych liczbami nieparzystymi (1, 3, 5, 7 i 9) jest wykorzystywana do trenowania modeli, a połowa osób oznaczonych liczbami parzystymi (2, 4, 6, 8 i 10) do ewaluacji.

Drugi protokół (oznaczony jako split II) wykorzystano w pracy [92]. W omawianym protokole 20 akcji podzielono na trzy podzbiory, z których każdy zawiera 8 klas. Tabela 3.2 przedstawia trzy podzbiory akcji. W omawianym protokole dokładność klasyfikacji jest obliczana dla każdego podziału danych, a następnie uśredniana. Podobnie jak w poprzednim protokole, połowa osób jest używana do trenowania modeli, a reszta do ewaluacji. Jak pokazano w pracy [97] osiągnięcie dobrych wyników jest znacznie trudniejsze na split I niż na split II - większość przebadanych algorytmów osiąga lepsze wyniki na split II.

3.4.2. UTD-MHAD

Zbiór danych UTD-MHAD [93], zawiera cztery rodzaje modalności danych: RGB, mapy głębi, pozycje stawów szkieletowych i sygnały z sensorów inercyjnych. Sekwencje danych zostały zarejestrowane przez sensor Kinect oraz noszony sensor inercyjny. W skład zbioru danych wchodzi 27 różnych akcji wykonanych przez osiem osób (cztery kobiety i czterech mężczyzn). Każda osoba powtórzyła każdą akcję cztery razy. W sumie zbiór danych zawiera

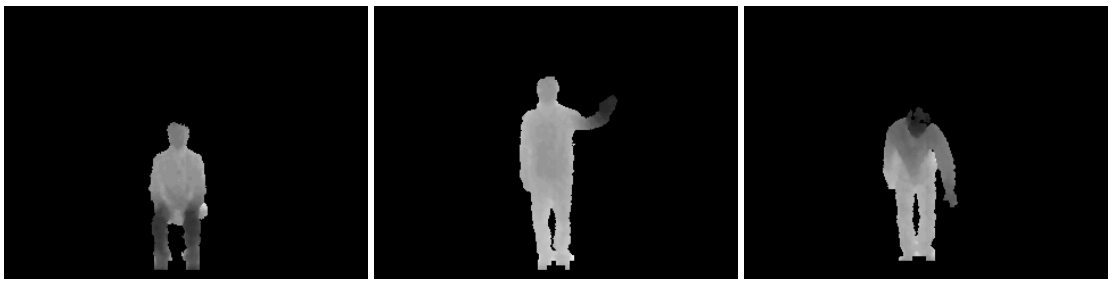
Rysunek 3.3: Przykładowe mapy głębi ze zbioru danych MSR-Action3D.

Tabela 3.2: Trzy podzbiory klas używane w protokole split II.

A1	A2	A3
Wymach ręką	Uniesienie ręki	Rzut w górę
Uderzenie młotkiem	Łapanie ręką	Kopnięcie w przód
Cios w przód	Rysowanie litery "X"	Kopnięcie na bok
Rzut w górę	Rysowanie kreski	Bieganie
Klaskanie w dłonie	Rysowanie okręgu	Zamach tenisowy
Pochylenie się	Wymach dwoma rękami	Serwis tenisowy
Serwis tenisowy	Kopnięcie w przód	Zamach golfowy
Podniesienie i rzut	Boksowanie na boki	Podniesienie i rzut

861 sekwencji map głębi oraz 59160 indywidualnych map głębi. Mapy głębi pobrano z częstotliwością 30 klatek na sekundę. Rozmiar mapy głębi to 320×240 pikseli. Przykładowe mapy głębi ze zbioru UTD-MHAD są zaprezentowane na rys. 3.4.

Akcje znajdujące się w zbiorze UTD-MHAD to: ruchy ręki (jednej lub obu) lub nogi: przesunięcie w lewo, przesunięcie w prawo, machanie, klaskanie, rzut, skrzyżowanie ramion, rzut koszykarski, rysowanie litery "x", rysowanie okręgu zgodnie z ruchem wskazówek zegara, rysowanie okręgu przeciwnie do ruchu wskazówek zegara, rysowanie trójkąta, kręgle, boksowanie, zamach baseballowy, zamach tenisowy, skręcanie ramienia, serwis tenisowy, pchnięcie, pukanie, łapanie, podniesienie i rzut, jogging, chodzenie, wstawanie, siadanie, wykrok, przysiad. Akcje są wykonywane przez ośmiu uczestników z czterema powtórzeniami. Protokół ewaluacji używa osób 1, 3, 5, 7 do trenowania modeli i uczestników 2, 4, 6, 8 do testowania.



Rysunek 3.4: Przykładowe mapy głębi ze zbioru danych UTD-MHAD.

3.4.3. 3DHOI

Zestaw danych 3DHOI [94], zawiera trzy rodzaje modalności danych: RGB, mapy głębi i pozycje stawów szkieletowych. Sekwencje danych zostały zarejestrowane przez sensor Kinect. W skład zbioru danych wchodzi 12 różnych akcji wykonanych przez 40 osób. W sumie zbiór danych zawiera 480 sekwencji map głębi i 107697 indywidualnych map głębi. Mapy głębi pobrano z częstotliwością 30 klatek na sekundę. Rozmiar mapy głębi to 640×480 pikseli. Przykładowe mapy głębi ze zbioru 3DHOI są zaprezentowane na rys. 3.5.

Akcje znajdujące się w zbiorze 3DHOI to: dzwonienie przez telefon komórkowy, zabawa telefonem komórkowym, picie, nalewanie, przesuwanie krzesła, siadanie na krześle, pakowanie plecaka, noszenie plecaka, zamiatanie, mopowanie, wyjmowanie czegoś z portfela i wyjmowanie portfela. Każda aktywność stanowi rodzaj interakcji człowiek-obiekt. Akcje były wykonywane przez 40 osób. Omawiany zbiór danych jest wyzwaniem dla rozpoznawania ludzkich akcji, ponieważ wiele akcji ma podobne ruchy lub takie same działające na obiekt w początkowych mapach sekwencji.

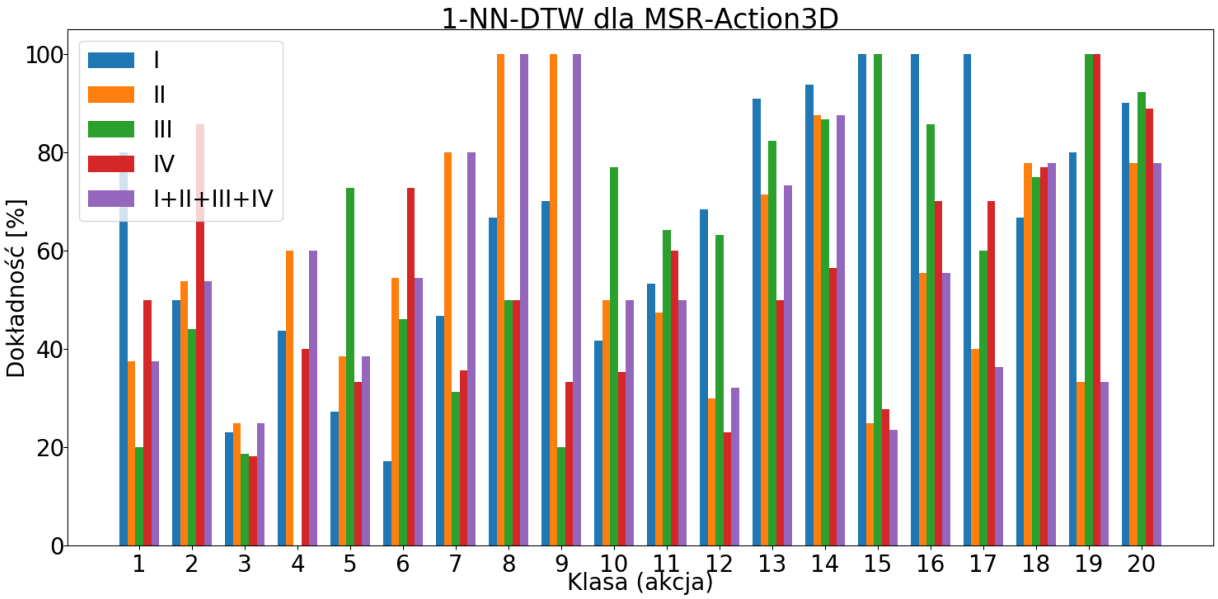
Ewaluacja algorytmów na zbiorze 3DHOI realizowana jest na dwóch protokołach:

Rysunek 3.5: Przykładowe mapy głębi ze zbioru danych 3DHOI. Na zbiorach wykorzystywanych w badaniach eksperymentalnych, osoby zostały wydzielone ręcznie. Z każdej sekwencji wyodrębniono prostokąt zawierajacy wycinek mapy głębi, który następnie przeskalowano do rozmiaru wejścia sieci.

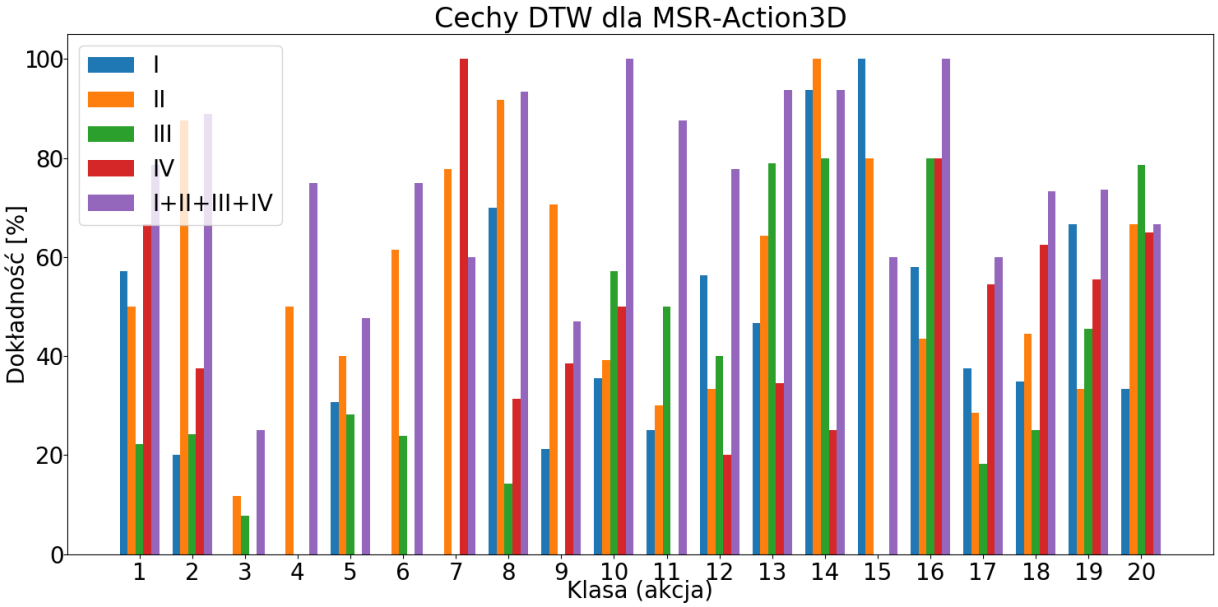
1. Setting 1 - połowa przykładów jest wybrana do trenowania modeli, a reszta do testowania. Liczba przykładów z każdej klasy w zbiorze testowym i treningowym nie musi być równa.
2. Setting 2 - połowa przykładów z każdej klasy jest wybrana do trenowania modeli, a reszta do testowania. Liczba przykładów z każdej klasy w zbiorze testowym i treningowym jest równa.

3.4.4. Omówienie wyników uzyskanych w pracach eksperymentalnych

W niniejszym rozdziale porównano cztery różne algorytmy: 1-NN-DTW oraz klasyfikatory wytrenowane na trzech różnych typach cech. Przebadano cechy oparte na DTW, cechy oparte na DTW z selekcją cech RFE oraz cechy statystyczne. Jako klasyfikator używany jest SVM. Algorytmy przebadano na trzech zbiorach danych (MSR-Action3D, UTD-MHAD, 3DHOI). W



Rysunek 3.6: Dokładności klasyfikacji na zbiorze MSR-Action3D uzyskiwane przez algorytm 1-NN-DTW na zestawach cech: I, II, III, IV oraz na połączonych cechach (I+II+III+IV).



Rysunek 3.7: Dokładności klasyfikacji na zbiorze MSR-Action3D uzyskiwane przez algorytm SVM wykorzystujący cechy oparte na DTW na zestawach cech: I, II, III, IV oraz na połączonych cechach (I+II+III+IV).

tabelach 3.3, 3.4, 3.5, porównano wyniki uzyskane przez cztery algorytmy. Poniżej opisano oznaczania użyte w tabelach 3.3, 3.4 oraz 3.5:

1. stat_feats - trzy proste cechy obliczone na podstawie szeregów czasowych - średnia, odchylenie standardowe i skrzywienie.
2. 1-NN-DTW - algorytm k-NN z miarą DTW.

3. dtw_feats - cechy oparte na DTW.

4. dtw_feats (RFE) - cechy oparte na DTW po selekcji algorytmem RFE.

Z czterech przebadanych algorytmów, algorytm wykorzystujący cechy oparte na DTW po selekcji RFE osiąga dobre wyniki na zbiorach MSR-Action3D i UTD-MHAD, natomiast dużo gorsze dokładności klasyfikacji uzyskuje na 3DHOI. Należy przypuszczać, że powodem mniejszej dokładności klasyfikacji jest to, że zbiór 3DHOI zawiera akcje polegające na interakcji z obiektami.

Tabela 3.3: Wyniki uzyskane na zbiorze MSR przez algorytmy opierające się na odległości DTW.

Zbiór Danych	Alg.	Zestaw cech	Liczba cech	Protokół	Accuracy	Precision	Recall	F1-score
MSR	stat_feats	I	9	Split 1	0,2982	0,2834	0,2891	0,2698
MSR	stat_feats	I+II+ III+IV	33	Split 1	0,3382	0,2682	0,3332	0,2798
MSR	1-NN-DTW	I	-	Split 1	0,5818	0,5798	0,6547	0,5842
MSR	dtw_feats	I	292	Split 1	0,4364	0,4323	0,4339	0,3864
MSR	dtw_feats	I+II+ III+IV	1168	Split 1	0,7273	0,7156	0,7385	0,7086
MSR	dtw_feats (RFE)	I+II+ III+IV	400	Split 1	**0,7382**	0,7262	0,7399	0,7168

Tabela 3.4: Wyniki uzyskane na zbiorze UTD-MHAD przez algorytmy opierające się na odległości DTW.

Zbiór danych	Alg.	Zestaw cech	Liczba cech	Protokół	Accuracy	Precision	Recall	F1-score
UTD MHAD	stat_feats	I	9	-	0,2395	0,2283	0,2394	0,2210
UTD MHAD	stat_feats	I+II III+IV	33	-	0,5535	0,5808	0,5539	0,5440
UTD MHAD	1-NN-DTW	I	-	-	0,6233	0,6235	0,6765	0,6074
UTD MHAD	dtw_feats	I	431	-	0,4535	0,4539	0,5034	0,4463
UTD MHAD	dtw_feats	I+II+ III+IV	1724	-	0,6837	0,6850	0,6827	0,6745
UTD MHAD	dtw_feats (RFE)	I+II+ III+IV	400	-	**0,6907**	0,6920	0,7062	0,6832

Dodatkowo porównujemy różne podzbiory cech tworzonych ręcznie. Zestawienie podzbiorów cech znajduje się w tabeli 3.1. Na rysunkach 3.6 i 3.7 znajduje się porównanie dokładności klasyfikacji uzyskiwanych przez algorytm 1-NN-DTW oraz przez cechy DTW dla poszczególnych klas ze zbioru danych MSR-Action3D. Na rysunku 3.6 znajdują się wyniki dla algorytmu 1-NN-DTW, zaś na rysunku 3.7 dla algorytmu SVM wykorzystującego cechy oparte na DTW. Nie zamieszczono wyników uzyskanych na zbiorze UTD-MHAD z uwagi na to, że nie wnoszą one nic nowego. Porównujemy cztery podzbiory cech (zob. tabela 3.1) oraz połączenie wszystkich cech. Jak można zaobserwować różne podzbiory uzyskają różne dokładności klasyfikacji dla rożnych klas (algorytm operujący na cechach zestawu II uzyskuje najlepsze dokładności dla klas 7 i 8, zaś algorytm operujący na cechach z zestawu I uzyskuje najlepsze wyniki dla klas 10 i 11). Jak można zaobserwować proste połączenie zestawów cech przed obliczaniem DTW nie powoduje poprawy wyników. Jak można zauważyć w tabelach 3.3, 3.4, 3.5 algorytm dtw_feats (w przeciwieństwie do 1-NN-DTW) uzyskuje lepsze wyniki jeśli łączone są zestawy cech. Połączone zestawy cech dają najlepsze wyniki dla 15 klas. W niektórych przypadkach (np. klasy 10, 11 oraz 12) połączone zestawy dają znacznie lepsze wyniki niż każdy pojedynczy zestaw.

Tabela 3.5: Wyniki uzyskane na zbiorze 3DHOI przez algorytmy opierające się na odległości DTW.

Zbiór danych	Alg.	Zestaw cech	Liczba cech	Protokół	Accuracy	Precision	Recall	F1-score
3DHOI	stat_feats	I	9	Setting 1	0,2691	0,2761	0,2688	0,2623
3DHOI	stat_feats	I+II+III+IV	33	Setting 1	0,3049	0,2612	0,3019	0,2549
3DHOI	1-NN-DTW	I	-	Setting 1	0,2675	0,2675	0,2815	0,2565
3DHOI	dtw_feats	I	252	Setting 1	0,2544	0,2544	0,2586	0,2423
3DHOI	dtw_feats	I+II+III+IV	1008	Setting 1	0,3465	0,3465	0,3527	0,3277
3DHOI	dtw_feats (RFE)	I+II+III+IV	400	Setting 1	**0,3816**	0,3816	0,4223	0,3756

3.5. Podsumowanie.

Zaproponowano cechy DTW, a następnie porównano wyniki uzyskane przez różne klasyfikatory oparte na odległości DTW. Wykazano eksperymentalnie, że algorytm oparty na cechach DTW uzyskuje lepsze wyniki niż k-NN-DTW. Pokazano również, że cechy DTW obliczone na różnych podzbiorach cech tworzonych ręcznie dają znacznie różniące się wyniki. Cechy DTW umożliwiają poprawę wyników poprzez połączenie cech DTW obliczonych na różnych zestawach cech. Algorytmy oparte na DTW nie osiągają jednak dobrych wyników na każdym z

przebadanych zbiorów danych (na przykład na 3DHOI). Mając na uwadze powyższe, w rozdziałach 5 i 6 zaproponowano algorytmy łączące DTW i sieci neuronowe.

4. Zespół Neuronowy oparty na cechach specyficznych dla klas i cechach wspólnych

W niniejszym rozdziale omówiono autorski algorytm uczenia zespołowego oparty o sieci neuronowe - Neural Ensemble built on Class Specific and Common Features (NECSCF). Jest to nowe podejście do uczenia zespołowego możliwe do zaimplementowania w wielu wariantach dostosowanych do poszczególnych zagadnień - różnych typów danych (sekwencje map głębi, dane tabelaryczne) oraz różnych problemów (rozpoznawanie akcji ludzkich, rozpoznawanie obiektów). W niniejszym rozdziale zaproponowano wariant algorytmu NECSCF dla danych tabelarycznych, a w kolejnym rozdziale do rozpoznawania akcji człowieka.

Sieci neuronowe trenowane w oparciu o metody uczenia głębokiego umożliwiają uzyskanie bezprecedensowych wyników w dziedzinie rozpoznawania obrazów [98] oraz przetwarzania języka naturalnego [44], jednak na danych tabelarycznych nadal najlepsze wyniki są uzyskiwane przez modele oparte na drzewach decyzyjnych [99].

W pracy [99] zaprezentowano wyniki badań, w których porównano dokładności klasyfikacji uzyskane przez sieci neuronowe i algorytmy oparte o drzewa decyzyjne. Dane tabelaryczne mają postać wektora cech o stałej długości między którymi nie ma relacji przestrzenno czasowych (nie są obrazami ani szeregami czasowymi). Ewaluacje algorytmów przeprowadzono na 45 zbiorach danych. Zbiory zawierają od 3000 do 10000 przykładów i mniej niż 500 cech. Najlepsze wyniki uzyskały algorytmy uczenia zespołowego (RF, boosting) oparte o drzewa decyzyjne. Rezultaty uzyskane przez algorytmy uczenia zespołowego (RF [86], XGBoost [100], GradientBoosting [86]) są lepsze niż rezultaty trzech algorytmów opartych o sieci neuronowe przebadane w pracy [99], MLP [101], ResNet [102] i prosty transformer [44].

Przewagę modeli opartych na drzewach decyzyjnych nad sieciami neuronowymi wykazano również w pracy [103]. Ewaluacje przeprowadzono na 11 zbiorach danych. Przebadane zbiory zawierają większą liczbę przykładów niż zbiory przebadane w pracy [99] (od 7 tysięcy przykładów do miliona przykładów). Zbiory zawierają od 2 do 7 klas oraz od 10 do 2000 cech. Pomimo przebadania kilku wariantów sieci neuronowych (w tym wariantów zaprojektowanych dla danych tabelarycznych) nie uzyskano wyników wskazujących na przewagę nad modelami opartymi na drzewach decyzyjnych.

Zaproponowany w niniejszej pracy algorytm NECSCF łączy zalety sieci neuronowych, uczenia zespołowego oraz drzew decyzyjnych. W niniejszym rozdziale zaprezentowano wari-

ant algorytmu dla danych tabelarycznych, który przebadano eksperymentalnie na zbiorach UCI [104] (opisanych w podrozdziale 4.3.1) oraz zbiorach OpenML [105] (opisanych w podrozdziale 4.3.2). Proponowany algorytm uzyskuje dobre wyniki w porównaniu do wyników uzyskiwanych w pracach [106] i [105]. Zaproponowany algorytm uzyskał na wielu zbiorach danych lepsze wyniki, a w kilku przypadkach poprawa była statystycznie istotna.

Algorytm 3 Pseudokod dla algorytmu zespół neuronowy oparty na cechach specyficznych dla klasy i cechach wspólnych (NECSCF).

1: **function** NECSCF(train_samples,train_labels,test_samples)
2: $number\_of\_classes \leftarrow COUNT\_CLASSES(train\_labels)$
3: **for** $i \leftarrow 1 \ldots number\_of\_classes$ **do**
4: **for** $label\_j \in \_labels$ **do**
5: $binary\_labels\_i[j] \leftarrow INT(label\_j == i)$
6: **end for**
7: $class\_specific\_models[i] \leftarrow TRAIN\_MODEL(train\_samples, binary\_labels\_i)$
8: **end for**
9: **for** $sample\_j \in train\_samples$ **do**
10: $common\_feats\_j \leftarrow GEN\_COMMON\_FEATS(sample\_j)$
11: **for** $model\_i \in class\_specific\_models$ **do**
12: $cs\_feats\_ij \leftarrow GEN\_CLASS\_SPECIFIC\_FEATS(model\_i, sample\_j)$
13: $concat\_feats[i][j] \leftarrow CONCATENATE(common\_feats\_j, cs\_feats\_ij)$
14: **end for**
15: **end for**
16: **for** $i \leftarrow 1 \ldots nunber\_of\_classes$ **do**
17: $multiclass\_clf[i] \leftarrow TRAIN\_CLF(concat\_feats[i], train\_labels)$
18: **end for**
19: **for** $sample\_j \in test\_samples$ **do**
20: $predicted\_labels[j] \leftarrow CLASSIFIER\_AGGREGATION(sample\_j, multiclass\_clf)$
21: **end for**
22: **return** $predicted\_labels$
23: **end function**

4.1. NECSCF - ogólny algorytm

Zespół neuronowy oparty na cechach specyficznych dla klasy i cechach wspólnych (NECSCF) to zaproponowany algorytm uczenia zespołowego, opisany w pseudokodzie 3 oraz zilustrowany na rys. 4.1. Na wejściu znajdują się dane (przykłady) treningowe (train_samples), etykiety przykładów treningowych (train_labels) oraz przykłady testowe (test_labels). W pierwszej fazie algorytmu (opisanej w liniach 2-6) wyznaczamy cechy specyficzne dla klasy (ang.

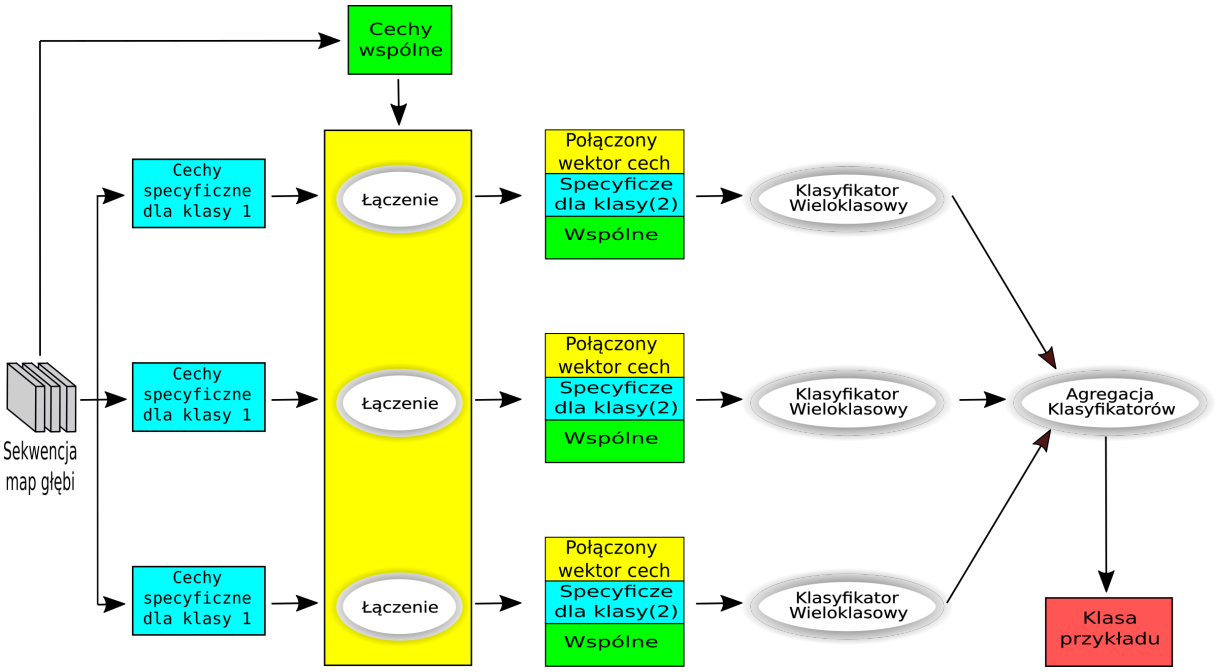
one-vs-all). W tym celu dla każdej klasy trenowana jest sieć neuronowa typu jeden-przeciw-wszystkim. Liczba sieci jest równa liczbie klas w zbiorze danych (wyznaczana przez funkcję COUNT_CLASSES). Cechy specyficzne dla klasy są wyznaczane przez przedostatnią warstwę sieci neuronowej wytrenowanej zawczasu na zbiorze treningowym. Sieci są trenowane przez funkcję TRAIN_MODEL. Sieci neuronowe o dowolnej architekturze mogą zostać użyte do wyznaczania cech specyficznych dla klasy. W następnym podrozdziale zaprezentowano wersje algorytmu NECSCF dla danych tabelarycznych, gdzie użyto prostej sieci trójwarstwowej.

W drugim etapie (linia 10) wyznaczamy cechy wspólne (ang. *common*) - za pomocą funkcji GET_COMMON_FEATS. Cechy wspólne mogą być wyznaczone przez sieć neuronową lub mogą być tworzone ręcznie (ang. *handcrafted*). Następnie (linia 12) dla każdej sieci neuronowej typu jeden-przeciw-wszystkim wyznaczamy cechy specyficzne dla klasy (za pomocą funkcji GEN_CLASS_SPECIFIC_FEATS).

Cechy wspólne są łączone (ang. *concatenated*) z cechami specyficznymi dla klas (linia 13) za pomocą funkcji CONCATENATE. W trzeciej fazie (linijki 16-18) na tak powstałym pełnym zestawie cech trenowane są wieloklasowe klasyfikatory (funkcja TRAIN_CLF). W niniejszej rozprawie używany jest algorytm RF, lecz można zastosować dowolny algorytm klasyfikacji, np. regresję logistyczną lub MLP. Liczba wieloklasowych klasyfikatorów jest równa liczbie klas. Ostateczna decyzja o przynależności nowej próbki do klasy jest rozstrzygana przez (linie 19-21) funkcję CLASSIFIER_AGGREGATION, która agreguje predykcje pojedynczych klasyfikatorów. Przykładem takiej funkcji może być twarde (ang. *hard voting*) lub miękkie głosowanie (ang. *soft voting*).

Warianty algorytmu NECSCF różnią się czterema właściwościami:

1. Cechy specyficzne dla klasy - zawsze wyznaczane są za pomocą dwuklasowej sieci neuronowej. Dowolna architektura sieci może zostać użyta do wyznaczania cech. W niniejszym rozdziale używamy prostej sieci MLP.
2. Cechy wspólne - cechy wspólne dla wszystkich klasyfikatorów - mogą być tworzone ręcznie lub wyznaczane przez wieloklasową sieć neuronową.
3. Klasyfikator wieloklasowy - algorytm, który jest trenowany na połączonych (ang. *concatenated*) cechach. Przykładem algorytmów klasyfikacji, które mogą zostać użyte to między innymi regresja logistyczna (regresja logistyczna zwraca wektor prawdopodobieństwa przynależności próbki do klasy, może zatem być użyta do klasyfikacji) [107]) oraz RF.
4. Agregacja wyników klasyfikatorów - funkcja, która na podstawie decyzji poszczególnych klasyfikatorów wyznacza wynik klasyfikacji. Najprostsza metoda kombinacji to *hard voting*, w której wybiera się klasę na podstawie większości głosów. W metodzie *soft voting* [108] sumuje się rozkłady prawdopodobieństw wyznaczone przez klasyfikatory i wybiera się klasy o największym prawdopodobieństwie. Bardziej złożoną metodą jest ważone głosowanie [109, 110], gdzie w głosowaniu różnym klasyfikatorom przypisuje się

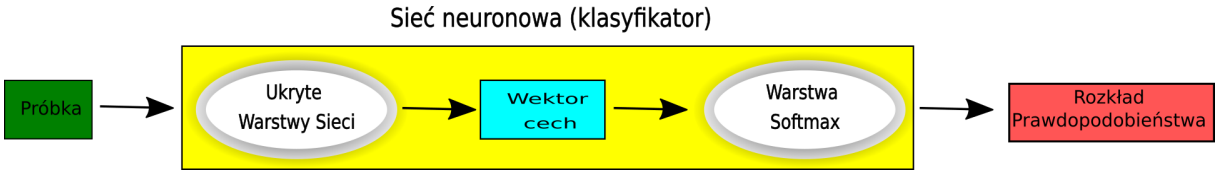


Rysunek 4.1: Ogólna forma autorskiego algorytmu NECSCF (Neural Ensemble built on Class Specific and Common Features). Powyższy rysunek ilustruje ogólny schemat algorytmu NECSCF. W tym i w dwóch następnych rozdziałach przedstawiono konkretne warianty tego algorytmu.

różne wagi. Inną rzadko stosowaną metodą w uczeniu zespołowym, są systemy głosowania wywodzące się z teorii wyboru społecznego [111, 112]. Wspomniane metody były przedmiotem badań w pracy [113]. Warto wspomnieć, że we wspomnianej pracy metody wywodzące się z teorii głosowania społecznego zostały wykorzystane do poprawy algorytmów uczenia maszynowego, w przeciwieństwie np. do pracy [114], w której metody ucznia maszynowego zostały użyte do poprawienia reprezentatywności metody głosowania.

4.1.1. Cechy wyznaczane przez przedostatnią warstwę sieci neuronowej.

W niniejszym podrozdziale omówiono właściwości cech wyznaczanych przez przedostatnią warstwę sieci neuronowej oraz wyjaśniono dlaczego użyto ich w algorytmie NECSCF. Uproszczona architektura klasyfikatora opartego na sieci neuronowej jest zilustrowana na rysunku 4.2. Można wyodrębnić dwie części składowe klasyfikatora opartego na sieci neuronowej: ukryte warstwy sieci oraz warstwę softmax [85].



Rysunek 4.2: Uproszczona budowa klasyfikatora opartego na sieci neuronowej.

Ukryte warstwy sieci wyznaczają wektor cech. Wartości neuronów w ostatniej ukrytej warstwie sieci są przekazywane do warstwy Softmax. Trójwarstwowa sieć neuronowa (z nieliniową funkcją aktywacyjną) jest uniwersalnym aproksymatorem [115]. Niech k i t będą liczbami naturalnymi, funkcja $g : \mathbb{R}^k \to \mathbb{R}^t$ będzie dowolną funkcją gładką, a $\Omega \in \mathbb{R}^p$ dowolną ograniczoną dziedziną. Dla każdego ϵ istnieje m takie, że dla trójwarstwowej sieci ϕ o m neuronach w warstwie ukrytej i wagach β prawdziwe jest równanie 4.1.

$$max_{x\in\Omega} = ||g(x) - \phi(x, \beta)|| < \epsilon \tag{4.1}$$

Warstwa Softmax - w tym modelu prawdopodobieństwo przynależności wektora do klasy c zależy od odległości od separującej hiperpłaszczyzny. Warstwa Softmax jest opisana przez równanie 4.2.

$$p(y = c|x) = \frac{e^{\beta_0^{(c)} + \beta^{(c)} \cdot x}}{\sum_{i=1}^{C} e^{\beta_0^{(c)} + \beta^{(c)} \cdot x}} \tag{4.2}$$

gdzie:

- $p(y = c|x)$ - prawdopodobieństwo przynależności próbki x do klasy x_c
- c - klasa próbki
- x - wektor cech
- $\beta_0^{(i)}$ - bias dla klasy i
- $\beta^{(i)}$ - parametry hiperpłaszczyzny dla klasy i
- C - liczba klas

W przypadku dwóch klas:

$$p(y = c|x) = \frac{e^{\beta_0^{(c)} + \beta^{(c)} \cdot x}}{e^{\beta_0^0 + \beta^0 \cdot x} + e^{\beta_0^1 + \beta^1 \cdot x}} = \frac{e^{\beta_0^{(c)} + \beta^{(c)} \cdot x - \beta_0^0 - \beta^0 \cdot x}}{1 + e^{(\beta_0^1 + \beta^1 \cdot x - \beta_0^0 - \beta^0 \cdot x)}} \tag{4.3}$$

Po podstawieniu $\beta_0 = \beta_0^0 - \beta_0^1$ i $\beta = \beta^0 - \beta^1$ uzyskujemy:

$$p(y = c|x) = \frac{e^{\beta_0 + \beta \cdot x}}{1 + e^{\beta_0 + \beta \cdot x}} = \frac{1}{1 + e^{-(\beta_0 + \beta \cdot x)}} \tag{4.4}$$

Wzór ten jest równoważny zależności:

$$log(\frac{p(y = c|x)}{1 - p(y = c|x)}) = \beta_0 + \beta \cdot x \tag{4.5}$$

Zazwyczaj model opisany w równaniach 4.4 i 4.5 nazywa się regresją logistyczną [116, 107]. Nie jest to jednak algorytm regresji ponieważ nie zwraca zmiennej rzeczywistej lecz wektor prawdopodobieństwa. Regresja logistyczna jest przykładem algorytmu parametrycznego, sieć neuronowa jest przykładem algorytmu nieparametrycznego. Algorytm nieparametryczny

potrafi przybliżać dowolną funkcję [115], ale zwykle wymaga większej liczby przykładów do wytrenowania modelu [117].

Jako funkcje strat użyto kategorycznej entropii krzyżowej [118, 119]:

$$L(p,q) = -\sum_{i=1}^{C} q(y_i) log(p(y_i)) \tag{4.6}$$

gdzie:

– L - funkcja strat

– p - prawdopodobieństwo wyznaczone przez sieć neuronową

– q - prawdziwy rozkład prawdopodobieństwa

– C - liczba klas

Jak można zaobserwować po zastosowaniu reprezentacji wyznaczonej przez przedostatnią warstwę sieci neuronowej próbki z różnych klas są separowalne liniowo (jeśli funkcja strat osiąga małą wartość). Oznacza to, że wektory cech przyporządkowane przez sieć neuronową do przykładów z różnych klas mogą być rozdzielone hiperpłaszczyzną. Wartość $|b + \beta \cdot x|$ jest proporcjonalna do odległości od hiperpłaszczyzny. Gdy odległość od hiperpłaszczyzny dąży do nieskończoności, prawdopodobieństwo $p(y_c|x)$ dąży do 0 lub 1 (w zależności od znaku wartości $b + \beta \cdot x$).

Oczywiście, możemy zastosować liniowy model do każdych danych, jednak nie każde zbiory wektorów cech można rozdzielić hiperpłaszczyzną. Za pomocą nieliniowej transformacji, można zastosować metody liniowe do nieseparowalnych liniowo danych ("kernel trick" opiera się na podobnej zasadzie [120]).

4.1.2. Porównanie NECSCF z innymi algorytmami uczenia zespołowego.

Tabela 4.1 przedstawia różnice między NECSCF a innymi algorytmami uczenia zespołowego. Bardziej szczegółowy opis algorytmów uczenia zespołowego znajduje się w podrozdziale 2.3. Zarówno w metodzie losowego wyboru podprzestrzeni (której szczególnym przypadkiem jest RF [39]) jak i uczenia zespołowego bagging, trenowany jest zbiór słabych klasyfikatorów [85], a ostateczna decyzja jest podejmowana przez głosowanie. W RF klasyfikatory są trenowane na różnych podzbiorach cech, zaś w algorytmie bagging klasyfikatory są trenowane na różnych podzbiorach przykładów treningowych. W metodzie znanej pod nazwą gradient boosting [60] klasyfikatory są trenowane z różnymi wagami przykładów w funkcji celu. W proponowanym algorytmie klasyfikatory operują na cechach wyznaczonych przez sieci neuronowe typu jeden-przeciw-wszystkim.

Podsumowując, zalety NECSCF są następujące:

1. Łączenie zalet metod opartych o sieci neuronowe i drzewa decyzyjne.

Tabela 4.1: Różnice między różnymi algorytmami uczenia zespołowego. Kolumna "Algorytm" zawiera nazwę algorytmu uczenia zespołowego. Kolumna "Różnice między modelami" pokazuje czym różnią się modele wchodzące w skład zespołu klasyfikatorów.

Algorytm	Różnice między modelami
Gradient boosting [60]	Wagi przykładów (w funkcji straty)
RF [39]	Zbiór cech na którym modele są trenowane
Bagging [59]	Zbiór przykładów treningowych
Ensemble NECSCF (proponowany)	Cechy określone przez sieć jeden-przeciw-wszystkim (jedna dla każdej klasy)

2. Możliwość łączenia cech wyznaczonych w oparciu o DTW i sieci neuronowe.
3. Wykorzystanie cech wyznaczanych przez przedostatnią warstwę sieci neuronowych. Zaletą takich cech jest liniowa separowalność klas [116]. Liniowa separowalność umożliwia zastosowanie modeli parametrycznych (takich jak regresja logistyczna). Modele parametryczne umożliwiają uzyskanie dobrych wyników na zbiorach treningowych o mniejszym rozmiarze [117].

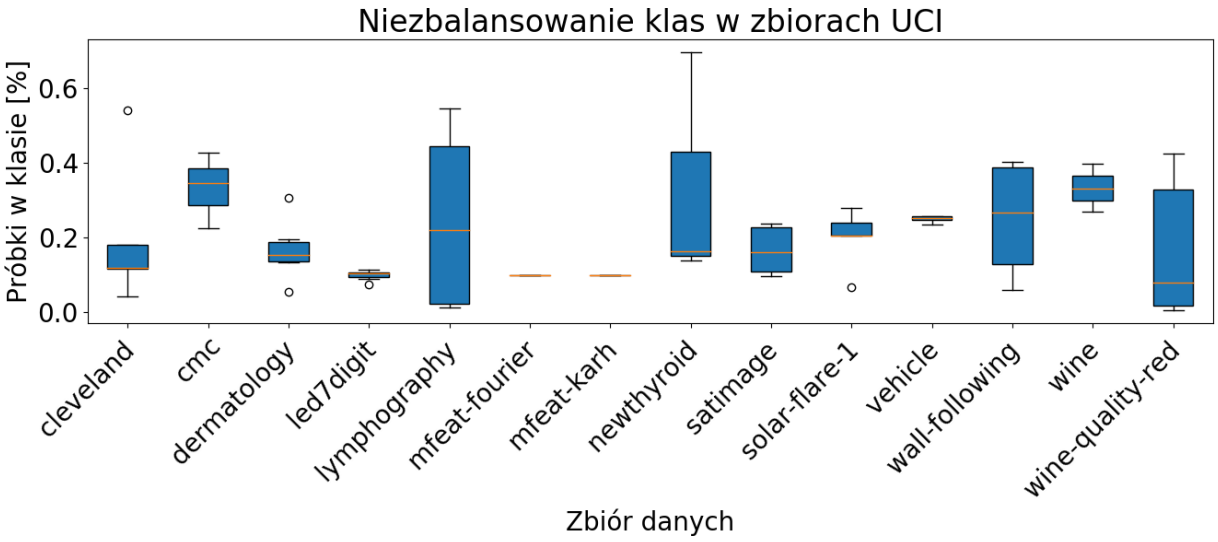
4.2. NECSCF dla danych tabelarycznych

W niniejszej części pracy przedstawiono wersję algorytmu NECSCF dostosowanego do danych tabelarycznych. Cechy specyficzne dla klasy są wyznaczone przez prostą sieć neuronową typu MLP z co najwyżej dwiema warstwami ukrytymi. Szczegóły użytej architektury sieci neuronowej zaprezentowano w tabeli 4.2. Hiperparametry algorytmu zostały zoptymalizowane za pomocą bayesowskiej walidacji krzyżowej [121].

Tabela 4.2: Szczegóły architektury sieci neuronowej.

Parametr	Wartość
Liczba neuronów w pierwszej warstwie ukrytej	Hiperparametr
Liczba neuronów w drugiej warstwie ukrytej	Hiperparametr
Batch normalization	Hiperparametr
Funkcja straty	Entropia krzyżowa binarna
Algorytm optymalizacji	Adam (adaptive moment estimation) [122]
Współczynnik uczenia	0,001

W niniejszym rozdziale prace eksperymentalne realizowano między innymi na niezbalansowanych zbiorach danych (zob. rysunki 4.3 i 4.4). Z tego powodu należy przypisać odpowiednie wagi próbkom z wybranej klasy aby także mniej liczne klasy miały wpływ na trenowany klasyfikator. Hiperparametr α określa wagę wybranej klasy. Wagi przykładów wyznaczane są w oparciu o równanie 4.7.



Rysunek 4.3: Rozkład liczby przykładów w poszczególnych klasach dla zbiorów UCI.

Wagi przykładów w klasyfikatorze wyznaczającym cechy specyficzne dla klasy c:

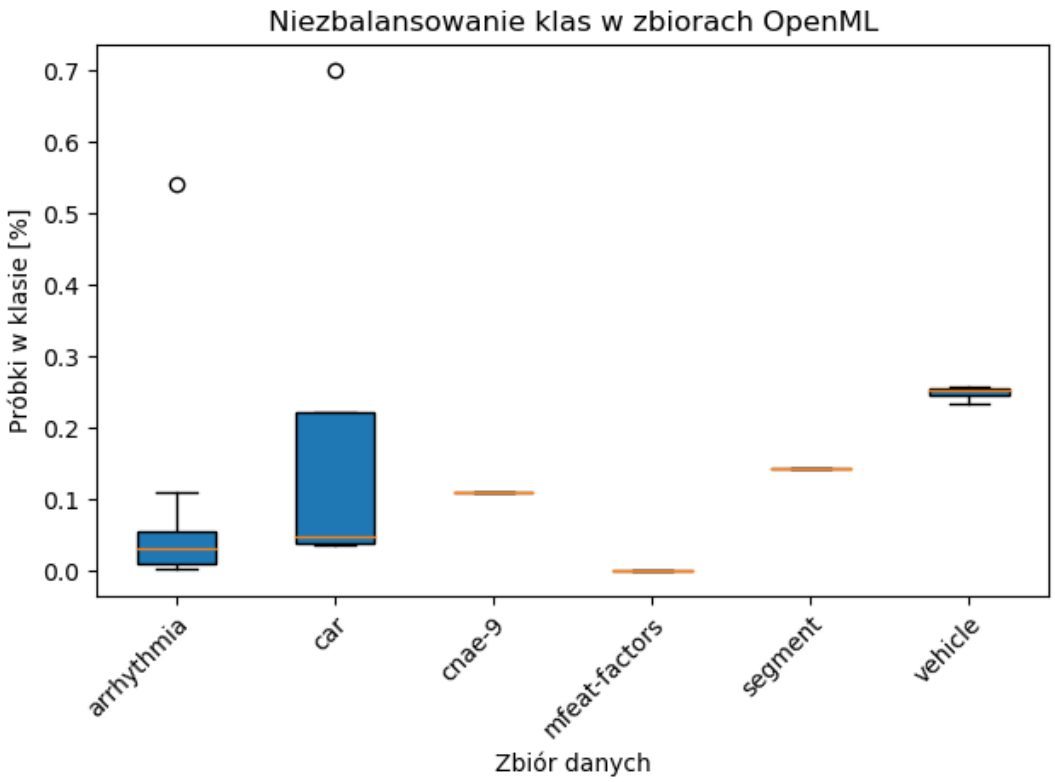
$$w_{c,i} = \begin{cases} w_{c,i} = \alpha \dfrac{1}{|A_c|} & y_i = c. \\ w_{c,i} = (1-\alpha)\dfrac{1}{|D \setminus A_c|} & y_i! = c \end{cases} \tag{4.7}$$

gdzie:

- α - stała liczba rzeczywista. Określa wagę przykładów wybranej klasy w funkcji straty sieci neuronowej.
- $w_{c,i}$ - waga przykładu i użyta w trenowaniu klasyfikatora wyznaczającego cechy specyficzne dla klasy c.
- c - klasa danego klasyfikatora wyznaczającego cechy specyficzne dla klasy.
- y_i - klasa przykładu i.
- A_c - zbiór przykładów dla klasy c.
- D - zbiór wszystkich przykładów.
- C - liczba klas w zbiorze danych.

Na rysunku 4.5 zilustrowano zespół klasyfikatorów dla danych tabelarycznych. Podsumowanie właściwości zespołu NECSCF dla danych tabelarycznych znajduje się w tabeli 4.3. Jako cechy wspólne użyto znormalizowanych oryginalnych cech. Cechy specyficzne są wyznaczane przez prostą sieć neuronową z dwiema warstwami ukrytymi. Hiperparametry i inne szczegóły użytej architektury sieci neuronowej zaprezentowano w tabeli 4.2. Zestawienie hiperparametrów zawarto w tabeli 4.4.

Sieci neuronowe trenowano za pomocą algorytmu Adam. Algorytm Adam to jedna z najpopularniejszych metod optymalizacji używana do trenowania sieci neuronowych. Wspomni-



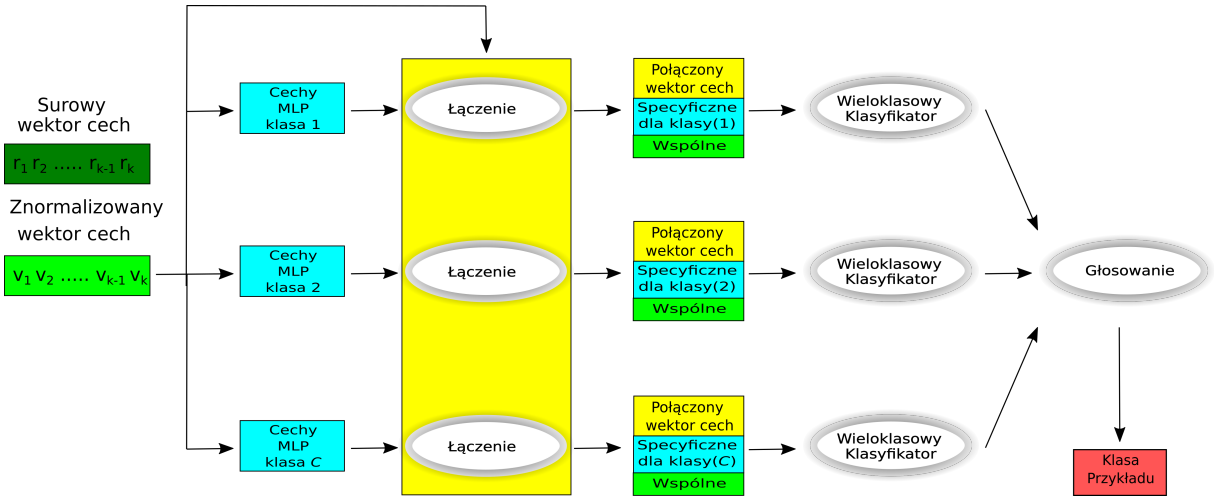
Rysunek 4.4: Rozkład liczby przykładów w poszczególnych klasach dla zbiorów OpenML.

Tabela 4.3: Wersja algorytmu NECSCF dla danych tabelarycznych.

Właściwości NECSCF	Wersja alg. dla danych tabelarycznych
Cechy wspólne	Znormalizowane, oryginalne cechy
Cechy specyficzne dla klasy	Sieć MLP
Klasyfikator wieloklasowy	RF
Agregacja wyników klasyfikatorów	Głosowanie miękkie

anego algorytmu użyto ze względu na wysoką efektywność obliczeniową oraz skuteczne działanie w przypadku rzadkiego gradientu [123]. Rzadki gradient może wynikać z sytuacji, w której niektóre cechy nie zawierają interesujących informacji i przypisuje się im zerowe wagi. Wyniki nie powinny zależeć od wybranego algorytmu optymalizującego zwłaszcza biorąc pod uwagę fakt, że trenujemy stosunkowo proste sieci neuronowe na małej liczbie przykładów.

Jako klasyfikatora wieloklasowego użyto RF, gdyż jak pokazano w pracy [99] jest to algorytm, który osiąga najlepsze wyniki na danych tabelarycznych. Ostateczna decyzja o przynależności do klasy podejmowana jest przez miękkie głosowanie klasyfikatorów.



Rysunek 4.5: Prosty wariant NECSCF dla danych tabelarycznych. Do wyznaczania cech specyficznych dla klasy używamy prostych sieci MLP, zaś jako cechy wspólne używane są znormalizowane (średnia każdej cechy jest równa 0, a odchylenie standardowe 1) oryginalne cechy. Sieci MLP są używane do wyznaczania cech, dowolny algorytm klasyfikacji (np. RF) może zostać użyty jako wieloklasowy klasyfikator.

Tabela 4.4: Zestawienie hiperparametrów. Kolumna 'Parametr' zawiera nazwę hiperparametru. Kolumna 'Typ' ilustruje jakiego typu zmienną jest dany hiperparametr. Kolumna 'Zakres' podaje w jakim przedziale algorytm optymalizacji poszukuje wartości hiperparametru. Kolumna 'A priori' ilustruje jaka dystrybucja a priori została użyta w bayesowskiej optymalizacji hiperparametru.

Parametr	Typ	Zakres	A priori
layer_1	Rzeczywisty	[1,10]	Jednostajne
layer_1	Rzeczywisty	[1,10]	Jednostajne
BN	Binarny	{True,False }	Jednostajne
α	Rzeczywisty	(0,1)	Jednostajne

4.3. Wyniki prac eksperymentalnych

Aby zbadać użyteczność proponowanych metod, przeprowadzono ewaluacje zaproponowanego algorytmu na danych UCI (UCI Machine Learning Repository) [104] oraz na OpenML [105]. Chociaż wybrane zbiory danych nie są związane z rozpoznawaniem akcji, umożliwiają eksplorację algorytmów zespołowych na różnorakich danych tabelarycznych.

Na zbiorach danych UCI i na zbiorach danych OpenML analizowano wyniki uzyskane w rożnych protokołach ewaluacyjnych. W przypadku zbiorów UCI dla każdego zbioru danych przeprowadzono 10 powtórzeń procedury walidacji krzyżowej 10-krotnej. Każdy zbiór jest dzielony na 10 części. W kolejnych iteracjach algorytmu jedna część używana jest jako zbiór testowy, zaś pozostałe 9 częsci jako zbiór treningowy. Taki sam protokół zastosowano w pracy [39]. W przypadku zbiorów OpenML każdy zbiór został losowo podzielony na trzy podzbiory

Tabela 4.5: Charakterystyka zbiorów danych z repozytorium UCI. Kolumna 'Zbiór danych' zawiera nazwę zbioru danych UCI. Kolumna 'Liczba klas' oznacza liczbę klas w zbiorze danych. Kolumna 'Liczba przykładów' zawiera rozmiar zbioru danych. Kolumna 'Liczba cech' oznacza wymiarowość przykładów ze zbioru danych. Kolumna 'Współczynnik Giniego' podaje stopień niezbalansowania liczby przykładów w każdej klasie w danym zbiorze danych.

Zbiór danych	Liczba klas	Liczba przykładów	Liczba cech	Współczynnik Giniego (dla klas)
cleveland	5	303	13	0,43
cmc	3	1473	9	0,13
dermatology	6	366	34	0,24
led7digit	10	500	7	0,06
lymphography	4	148	18	0,5
mfeat-fourier	10	2000	76	0,0
mfeat-karh	10	2000	64	0,0
newthyroid	3	215	5	0,37
satimage	6	6430	36	0,2
solar-flare-1	5	315	12	0,18
vehicle	4	846	18	0,02
wall-following	4	5456	24	0,32
wine	3	178	13	0,09
wine-quality-red	6	1599	11	0,56

zbiory: treningowy (60%), testowy (20%), walidacyjny (20%). Taki sam protokół zastosowano w pracy [124].

W podrozdziałach 4.3.1 i 4.3.2 opisano wybrane zbiory danych na których przeprowadzono eksperymenty. W podrozdziale 4.3.3 szczegółowo omówiono optymalizacje hiperparametrów. Wyniki prac eksperymentalnych zamieszczono w podrozdziałach 4.3.4 i 4.3.5.

4.3.1. Zbiory danych UCI

Wybrano zbiory danych przebadane w pracy [39], które zwierają więcej niż 2 klasy. Właściwości zbiorów danych są opisane w tabeli 4.5. Pokazano cztery właściwości zbiorów: liczba klas, liczba przykładów, liczba cech oraz współczynnik Giniego dla klas. Współczynnik Giniego to miara niezbalansowania rozkładu - wartość 0,0 oznacza jednostajny rozkład, zaś wartość 1,0 maksymalne niezbalansowanie. Przebadane zbiory zawierają od 3 do 10 klas, od 148 do 6430 przykładów i od 5 do 76 cech. Przebadano zarówno zbalansowane jak i niezbalansowane zbiory (współczynnik Giniego od 0,0 do 0,56).

Aby lepiej przedstawić niezbalansowanie klas na rysunku 4.3 pokazano rozmiary poszczególnych klas jako odsetek całości zbioru. Jak można zaobserwować zbiory danych

są zróżnicowane, a w niektórych zbiorach wszystkie klasy mają takie same rozmiary (mfeat-fourier, mfeat-karh), zaś innych zbiorach danych (jak np. clevaland, wine-quality-red) niektóre klasy są reprezentowane przez większą liczbę przykładów niż pozostałe klasy.

Tabela 4.6: Charakterystyka zbiorów danych z repozytorium OpenML. Kolumna 'Zbiór danych' zawiera nazwę zbioru danych OpenML. Kolumna 'Liczba klas' podaje liczbę klas w zbiorze danych. Kolumna 'Liczba przykładów' opisuje rozmiar zbioru danych. Kolumna 'Liczba cech' podaje wymiarowość przykładów ze zbioru danych. Kolumna 'Współczynnik Giniego' podaje stopień niezbalansowania liczby przykładów w każdej klasie w danym zbiorze danych.

Zbiór danych	Liczba klas	Liczba przykładów	Liczba cech	Współczynnik Giniego (dla klas)
arrhythmia	13	452	279	0,66
car	4	1728	6	0,543
cnae-9	9	1080	856	0,00
mfeat-factors	10	2000	216	0,00
segment	7	2310	19	0,00
vehicle	4	846	18	0,018

4.3.2. Zbiory danych OpenML

Aby zbadać użyteczność proponowanych metod, ewaluacje zaproponowanego algorytmu przeprowadzono także na zbiorach OpenML [105]. Wybrano zbiory danych przebadane w pracy [124], które zwierają więcej niż 2 klasy i mniej niż 3000 przykładów. Charakterystykę zbiorów danych zamieszczono w tabeli 4.6. Pokazano cztery właściwości zbiorów: liczba klas, liczba przykładów, liczba cech oraz współczynnik Giniego dla klas. W przeciwieństwie do zbiorów UCI niektóre zbiory OpenML mają wyższą wymiarowość (kilkaset cech). Wybrano zarówno zbiory zbalansowane (współczynniki Giniego bliski 0) jak i niezbalansowane (współczynnik Giniego większy niż 0,5).

Aby lepiej zilustrować niezbalansowanie klas na rysunku 4.4 pokazano rozmiary poszczególnych klas jako odsetek całości zbioru. Jak można zaobserwować zbiory danych są zróżnicowane, w niektórych zbiorach wszystkie klasy zawierają równą liczbę przykładów (cnae-9, mfeat-factors), zaś w innych zbiorach danych (jak np. arrhythmia, car) niektóre klasy są reprezentowane przez większą liczbę przykładów niż pozostałe klasy.

4.3.3. Optymalizacja hiperparametrów.

Hiperparametry algorytmu zostały zoptymalizowane za pomocą bayesowskiej walidacji krzyżowej [121]. W omawianej metodzie modelujemy rozkład hiperparametrów za pomocą metod statystyki bayesowskiej. Rozpoczynamy z początkowym rozkładem (określanym jako rozkład a priori) i aktualizujemy go po otrzymaniu nowych danych zgodnie z twierdzeniem

Tabela 4.7: Wyniki optymalizacji hiperparametrów na poszczególnych zbiorach danych z UCI. Kolumny 'units_1' i 'units_2' ilustrują liczbę neuronów w odpowiednio w pierwszej i drugiej warstwie sieci neuronowej. Kolumny 'layers_1' i 'layers_2' podają liczbe neuronów jako wielokrotność rozmiarów wektora wejściowego.

Zbiór danych	units_1	layer_1	units_2	layer_2	batch	alpha
cleveland	113	8,69	63	4,85	1	0,1
cmc	9	1,00	9	1,00	0	0,8
dermatology	124	3,65	284	8,35	0	0,3
led7digit	57	8,14	17	2,43	0	0,8
lymphography	78	4,33	128	7,11	1	0,6
mfeat-fourier	556	7,32	356	4,68	0	0,3
mfeat-karh	574	8,97	604	9,44	0	0,4
newthyroid	15	3,00	5	1,00	0	0,9
satimage	76	2,11	316	8,78	0	0,9
solar-flare	62	5,17	22	1,83	0	0,5
vehicle	178	9,89	-	-	0	0,8
wall-following	24	1,00	204	8,50	0	0,9
wine	113	8,69	53	4,08	0	0,1
wine-quality-red	61	5,55	-	-	1	0,8

Bayesa. Znaczenie parametrów wyjaśniono w tabeli 4.4, zaś wartości hiperparametrów podano w tabeli 4.7.

Optymalizowane hiperparametry to rozmiary ukrytych warstw sieci neuronowej oraz użycie batch normalization (BN). Parametr BN określa czy sieć neuronowa używa metody regularyzacji znanej jako batch normalization [125]. W batch normalization wszystkie cechy wyznaczone przez daną warstwę mają znormalizowaną średnią i wariancję [126]. Dodatkowym hiperparametrem jest α - określa on wagi przyporządkowane próbkom w funkcji straty, zob. równanie 4.7. Wszystkie parametry poza parametrem BN są zmiennymi rzeczywistymi. Z uwagi na to że używamy optymalizacji bayesowskiej musimy podać początkowy rozkład wartości parametru (określany jako a priori). We wszystkich przypadkach używamy rozkładu jednostajnego.

W tabeli 4.7 pokazano bezwzględną liczbę neuronów w warstwie (units_0 i units_1) oraz rozmiar warstwy jako wielokrotność wymiarowości danych wejściowych (layers_0 i layers_1). Rozmiar danych wejściowych (liczba cech) zestawiono w tabelach 4.5 i 4.6. Równanie 4.8 przedstawia zależność między hiperparametrem $layers\_i$ a liczbą neuronów w warstwie i.

$$units\_i = number\_of\_features \times layers\_i \tag{4.8}$$

Wyniki optymalizacji hiperparametrów znajdują się w tabeli 4.7 (dla zbiorów UCI) i 4.8

Tabela 4.8: Wyniki optymalizacji hiperparametrów na poszczególnych zbiorach danych z OpenML. Kolumny 'units_1' i 'units_2' ilustrują liczbę neuronów w odpowiednio w pierwszej i drugiej warstwie sieci neuronowej. Kolumny 'layers_1' i 'layers_2' podają liczbę neuronów jako wielokrotność rozmiarów wektora wejściowego.

Zbiór danych	units_0	layer_0	units_1	layer_1	batch	alpha
arrhythmia	320	1,14	219	0,785	0	0,75
car	26	4,33	32	5,33	0	0,25
cnae-9	680	0,79	167	0,1951	0	0,50
mfeat-factors	680	3,14	180	0,833	1	0,25
segment	129	6,79	111	5,84	1	0,25
vehicle	108	6	42	2,33	0	0,25

Tabela 4.9: Kolumna 'Dataset' podaje zbiór danych na którym przeprowadzono ewaluacje. Kolumna 'RF' przedstawia wyniki uzyskane przez algorytm RF. Kolumna 'NECSCF' przedstawia wyniki uzyskane przez algorytm NECSCF. Kolumna 'Pvalue' pokazuje p-wartość różnicy między wynikami klasyfikatorów RF i NECSCF. Kolumna 'Sig' pokazuje czy ($\rho < 0,05$). Pogrubiono zbiory danych dla których algorytm NECSCF osiąga statystycznie istotną poprawę w stosunku do algorytmu RF. Podkreślono zbiory danych dla których NECSCF osiąga lepsze wyniki niż RF, lecz nie są one statystycznie istotne.

Zbiór danych	RF	NECSCF	Pvalue	Sig
cleveland	0,5698±0,0520	0,5771±0,0607	0,3646	False
cmc	**0,5204±0,0363**	**0,5311±0,0377**	**0,0427**	**True**
dermatology	0,9736±0,0274	0,9689±0,0276	0,2334	False
led7digit	0,7088±0,0606	0,7134±0,0597	0,5911	False
lymphography	0,8510±0,1055	0,8717±0,0912	0,1411	False
mfeat-fourier	0,8310±0,0204	0,8299±0,0224	0,7061	False
mfeat-karh	**0,9611±0,0137**	**0,9696±0,0104**	**0,0**	**True**
newthyroid	0,9549±0,0421	0,9637±0,0397	0,1313	False
satimage	0,9165±0,0096	0,9165±0,0104	0,9825	False
solar-flare	0,7259±0,0610	0,7171±0,0620	0,3181	False
vehicle	**0,7539±0,0355**	**0,7834±0,0381**	**0,0**	**True**
wine	0,9798±0,0323	0,9788±0,0323	0,8257	False
wine-quality-red	0,7104±0,0306	0,7129±0,0312	0,5803	False

(dla zbiorów OpenML). Jak można zaobserwować optymalna liczba neuronów w warstwie ukrytej jest wielokrotnie większa niż wymiarowość danych wejściowych. Mediana wartości layer_1 równa jest 5,3561, mediana wartości layer_2 równa jest 4,7652. Po optymalizacji hiperparametrów dla większości zbiorów danych BN nie jest używana.

4.3.4. Wyniki eksperymentów. Zbiory danych UCI

Dla każdego zbioru danych przeprowadzono 10 powtórzeń procedury walidacji krzyżowej 10-krotnej, co daje łącznie 100 eksperymentów dla każdego zbioru danych. Dla każdego zbioru wyznaczono średnią i odchylenie standardowe dokładności klasyfikacji z przeprowadzonych powtórzeń eksperymentu.

W tabeli 4.9 zaprezentowano wyniki uzyskane przez algorytm NECSCF oparty o klasyfikator RF i porównano je z wynikami uzyskiwanymi przez algorytm RF. Pokazano również statystyczną istotność wyników. Porównujemy, czy różnica między NECSCF i RF jest statystycznie istotna. Istotność statystyczna jest obliczana za pomocą testu t-student [127]. Użyto implementacji algorytmu z biblioteki SciPy [128].

Uzyskano statystycznie istotną poprawę w stosunku do algorytmu RF w trzech przypadkach. W ośmiu otrzymano statystycznie nieistotną poprawę. Statystycznie istotne pogorszenie następuje w jednym przypadku, zaś statystycznie nieistotne w dwóch przypadkach. Wyniki statystycznie lepsze pogrubiono, zaś wyniki statystycznie nieistotne podkreślono.

Tabela 4.10: Kolumna 'Dataset' podaje zbiór danych na którym przeprowadzono ewaluacje. Kolumna 'Alg' podaje algorytm który osiągnął najlepsze wyniki na zbiorze danych w pracy [106]. Wyniki uzyskane przez ten algorytm podane są w kolumnie 'Alg_Acc'. Kolumna 'NECSCF_Acc' przedstawia wyniki dla algorytmu NECSCF.

Zbiór danych	Alg.	Alg_Acc	NECSCF_Acc
cleveland	RF [129]	57,72±5,29	57,71±6,07
cmc	AdaB [130]	56,3±3,44	53,11±3,77
dermatology	EPIC [131]	97,99±2,14	96,89±2,76
led7digit	MDSQ [132]	73,17±5,77	71,34±5,97
lymphography	**MRMR [133]**	**86,57±9,17**	**87,17±9,12**
mfeat-fourier	MRMR [133]	83,58±2,27	82,99±2,24
mfeat-karh	DISC [133]	97,13±1,09	96,96±1,04
newthyroid	RE [134]	96,88±3,51	96,37±3,97
satimage	RF [129]	91,82±0,95	91,65±1,04
solar-flare	EPIC [131]	75,65±2,82	71,71±6,2
vehicle	**AdaB [130]**	**78,29±4,1**	**78,34±3,81**
wall-following	AdaB [130]	99,88±0,14	99,0±0,41
wine	RF [129]	98,22±3.24	97,88±3,23
wine-quality-red	**RF [129]**	**70,53±3.44**	**71,29±3,12**

Ponadto porównano uzyskane wyniki z wynikami algorytmów zaprezentowanymi w [57] w tabeli 4.10. W omawianej tabeli zarówno dla RF jak i dla proponowanego algorytmu podajemy średnią wartość dokładności klasyfikacji ze 100 eksperymentów dla każdego zbioru danych, jak również odchylenie standardowe uzyskanych dokładności. Zapisujemy wyniki w formie *śred-*

nia ± *odchylenie_standardowe*. Zbiory danych dla których algorytm NECSCF osiąga najlepsze wyniki pogrubiono. Zaproponowana metoda w trzech przypadkach uzyskała najlepsze wyniki.

Dodatkowo zbadano wpływ właściwości zbiorów danych (tabela 4.5) na wyniki. W tym celu przedstawiamy współczynniki dwuklasowego liniowego klasyfikatora (model logistyczny). Jako cech używamy znormalizowanych właściwości zbiorów danych z tabeli 4.5. Liniowy klasyfikator którego współczynniki podano w tabeli 4.11 przewiduje na jakich zbiorach danych algorytm NECSCF osiągnąłby statystycznie istotną poprawę w stosunku do klasyfikatora RF. Liniowy klasyfikator którego współczynniki podano w tabeli 4.12 przewiduje na jakich zbiorach danych algorytm NECSCF osiągnąłby lepsze wyniki od algorytmy z pracy [106].

Tabela 4.11: Wartości współczynników liniowego klasyfikatora, który na podstawie parametrów zbiorów danych (przedstawionych w tabeli 4.5) przewiduje, kiedy algorytm NECSCF uzyskuje lepsze wyniki niż klasyfikator RF.

Liczba klas	Liczba przykładów	Liczba cech	Współczynnik Giniego
-0,0335	-0,2756	0,3140	-0,3769

Tabela 4.12: Wartości współczynników liniowego klasyfikatora, który na podstawie parametrów zbiorów danych (przedstawionych w tabeli 4.5) przewiduje, kiedy algorytm NECSCF uzyskuje lepsze wyniki niż klasyfikatory z pracy [106].

Liczba klas	Liczba przykładów	Liczba cech	Współczynnik Giniego
-0,1575	-0,2360	-0,1146	0,4919

W obu tabelach widać, że algorytm NECSCF osiąga lepsze wyniki na zbiorach danych o mniejszej liczbie przykładów. W przypadku osiągnięcia statystycznie istotnej przewagi nad algorytmem RF, lepsze wyniki są osiągane na zbiorach o małej wartości współczynnika Giniego oraz o dużej liczbie cech. Natomiast wyniki lepsze niż algorytmy z pracy [106] są uzyskiwane na zbiorach o dużej wartości współczynika Giniego oraz o małej liczbie cech. Nie są to sprzeczne rezultaty. Algorytm NECSCF często osiąga wyniki lepsze od wyników uzyskiwanych w pracy [106], które nie są jednak istotne statystycznie.

4.3.5. Wyniki eksperymentów. Zbiory OpenML

W niniejszym podrozdziale, w ramach ewaluacji proponowanego algorytmu przeprowadzano porównanie z wynikami z pracy [124] na zbiorach OpenML. W tym celu wybieramy zbiory danych w których znajduje się więcej niż dwie klasy i mniej niż 5000 przykładów. W protokole ewaluacji użytym w pracy [124] losowo podzielono każdy zbiór na zbiór treningowy (60%), testowy (20%) oraz walidacyjny (20%). W niniejszej pracy dzielimy zbiór losowo 10 razy na część treningową testową i walidacyjną, następnie podajemy średnią z otrzymanych wyników.

Tabela 4.13: Wyniki dla zbiorów OpenML

Zbiór danych	MLP+D	XGB	ASK-G	TabN	Node	AutoGL
arrhythmia	38,704	48,779	46,850	43,562	N/A	48,934
car	99,690	92,376	**100,000**	98,701	46,119	99,675
cnae	90,741	94,907	93,519	89,352	**96,759**	92,593
mfeat	**98,000**	**98,000**	97,500	97,250	97,250	**98,000**
segment	94,589	93,723	93,074	91,775	90,043	91,991
vehicle	82,603	74,973	80,165	79,654	75,541	**83,793**

W tabelach 4.13 i 4.14 znajduje się porównanie wyników uzyskanych przez NECSCF z wynikami z [124] na zbiorach OpenML (z uwagi na dużą liczbę porównywanych algorytmów jedną tabele rozbito na dwie). W użytym protokole ewaluacji algorytmu (z pracy [124]) wydzielono zbiór walidacyjny. Dlatego przebadano eksperymentalnie dwa warianty NECSCF - podstawowy i z optymalizacją hiperparametrów algorytmu RF (optymalizacje przeprowadzono na zbiorze walidacyjnym). Algorytm, który osiąga najlepsze wynik pogrubiono, a algorytm który osiąga drugi najlepszy wynik podkreślono. Dla dwóch zbiorów danych (arrhythmia i segment) NECSCF osiąga najlepsze wyniki.

Tabela 4.14: Wyniki dla zbiorów OpenML otrzymane przez algorytm NECSCF. W algorytmie oznaczonym jako NECSCF (optim) dokonujemy optymalizacji parametru klasyfikatora Lasu Drzew Losowych na zbiorze walidacyjnym.

Zbiór danych	NECSCF	**NECSCF (optim)**	MLP	S MLP+C
arrhythmia	61,29	**62,37**	37,991	61,461
car	96,53	96,82	97,442	99,587
cnae	91,20	89,35	87,500	95,833
mfeat	97,00	97,00	97,750	**98,000**
segment	**97,62**	**97,62**	94,805	93,723
vehicle	82,46	83,63	83,766	82,576

Liniowy klasyfikator, którego współczynniki podano w tabeli 4.15 przewiduje na jakich zbiorach danych algorytm NECSCF osiągnąłby lepsze wyniki niż algorytmy z pracy [124]. Wyniki lepsze od wyników uzyskiwanych przez algorytmy z pracy [124] są uzyskiwane na zbiorach o dużej wartości współczynnika Giniego, dużej liczbie klas oraz o małej liczbie cech. Liczba przykładów nie ma wpływu na wynik.

4.3.6. Dyskusja uzyskanych wyników

W pracy [106] porównano 12 algorytmów na zbiorach UCI. Algorytmy MDSQ, MRMR, DISC uzyskują najlepsze wyniki na jednym ze zbiorów danych. Algorytmy AdaB, EPIC

Tabela 4.15: Wartości współczynników liniowego klasyfikatora, który na podstawie parametrów zbiorów danych (przedstawionych w tabeli 4.6) przewiduje kiedy algorytm NECSCF uzyskuje lepsze wyniki niż klasyfikatory z pracy [124].

Liczba klas	Liczba przykładów	Liczba cech	Współczynnik Giniego
0,4561	0,0745	-0,2734	0,1961

uzyskują najlepsze wyniki na 2 zbiorach danych. Algorytm RF uzyskuje najlepszy wynik na 3 zbiorach. Proponowany algorytm uzyskuje najlepszy wyniki na 3 zbiorach. W pracy [124] porównano 8 algorytmów na zbiorach OpenML. Algorytmy MLP+D, XGB, ASK-G, TabN, Node uzyskują najlepszy wyniki na 1 zbiorze danych. Algorytm AutoGL uzyskuje najlepsze wyniki na 2 zbiorach danych. Proponowany algorytm uzyskuje najlepsze wyniki na 3 zbiorach.

Algorytm przebadano na różnorodnym zestawie danych i na dwóch różnych protokołach ewaluacji algorytmów. Porównano go zarówno z klasycznymi algorytmami uczenia zespołowego ([106]) jak i z metodami opartymi na sieciach neuronowych ([124]). Proponowana metoda osiągnęła jeden z najlepszych wyników pomimo użycia prostej architektury sieci oraz braku wyszukanych (dobieranych) cech wspólnych. Dodatkowo w niektórych przypadkach uzyskano statystycznie istotną poprawę w porównaniu z jednym z najlepszych algorytmów dla danych tabelarycznych (RF).

4.4. Podsumowanie.

W niniejszym rozdziale zaproponowano zespół klasyfikatorów NECSCF łączący algorytmy oparte na sieciach neuronowych oraz na drzewach decyzyjnych. Przebadano eksperymentalnie proponowany algorytm w sumie na 20 (14+6) zbiorach danych i osiągnięto na nich obiecujące wyniki. Algorytm ewaluowano według dwóch protokołów, a ponadto przeprowadzono porównanie z algorytmem RF. Pokazano również statystyczną istotność wyników (na 14 zbiorach z UCI). Uzyskano statystycznie istotną poprawę w stosunku do algorytmu RF w trzech przypadkach. W ośmiu otrzymano statystycznie nieistotną poprawę. Statystycznie istotne pogorszenie następuje w jednym przypadku, zaś statystycznie nieistotne w dwóch przypadkach. Pokazano eksperymentalnie, że zaproponowany algorytm uzyskuje lepsze wyniki niż modele oparte tylko na drzewach decyzyjnych.

Przebadano również wpływ właściwości zbiorów danych (tabela 4.5) na wyniki algorytmu NECSCF. Należy być ostrożnym z wyciąganiem wniosków z przeprowadzonej analizy, gdyż oparta jest ona na stosunkowo małej liczbie zestawów danych. Przeprowadzona analiza pokazuje, że algorytm NECSCF uzyskuje lepsze wyniki w stosunku do innych algorytmów na zbiorach danych o mniejszej liczbie przykładów.

5. Cechy wyznaczane przez CNN i DTW na potrzeby rozpoznawania akcji ludzi

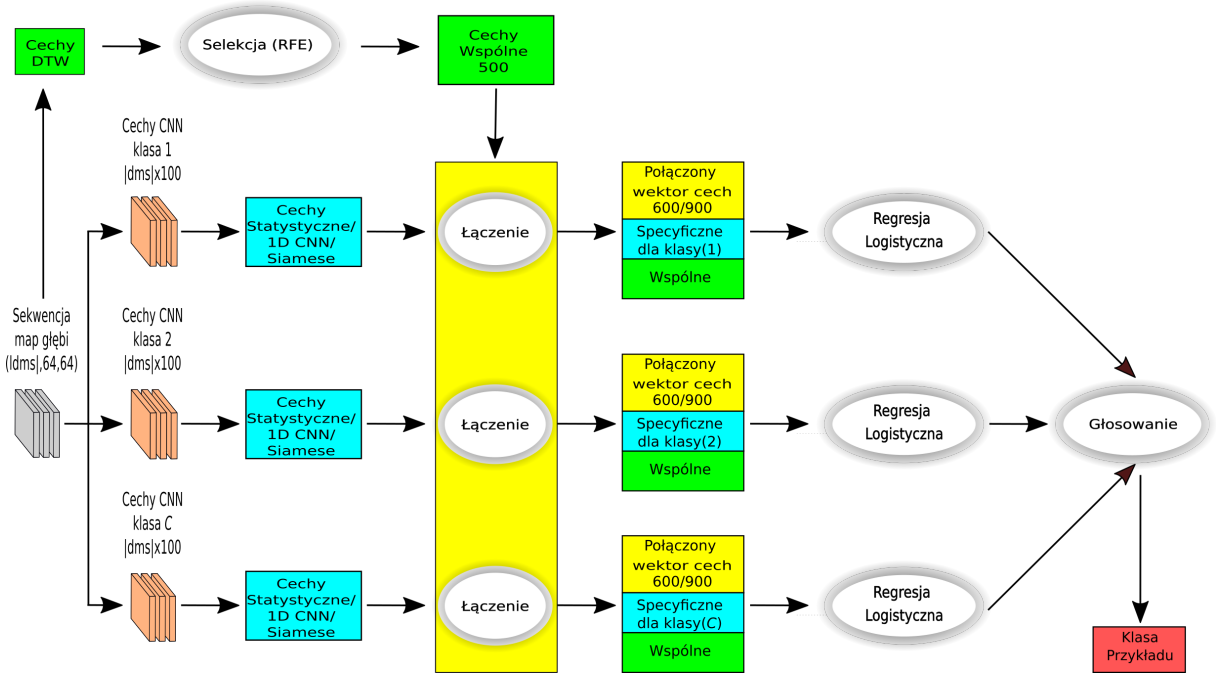
W rozdziale 4 zaproponowano nową metodę uczenia zespołowego opartą na głębokich sieciach neuronowych oraz cechy szeregów czasowych oparte na DTW. W niniejszym rozdziale proponuje się połączenie obu podejść i ich zastosowanie do problemu rozpoznawania akcji człowieka. Cechy oparte na DTW są szczególnie użyteczne w klasyfikacji ruchu, ponieważ są odporne na zmianę prędkości wykonywanych akcji.

Aby pokazać zalety algorytmu NECSCF, porównano go z bardziej konwencjonalnym algorytmem uczenia zespołowego (bagging) przy użyciu tych samych cech tworzonych ręcznie. Jako podstawowe klasyfikatory zespołu bagging użyto konwolucyjnych sieci neuronowych, które zostały wytrenowane na szeregach czasowych cech tworzonych ręcznie. Zaproponowano również nowy algorytm augmentacji danych dla szeregów czasowych - augmentację danych poprzez deformację w czasie (ang. *warp*), która posiada część zalet DTW (poprawa klasyfikacji akcji wykonywanych z różną prędkością, przy znacznie mniejszym koszcie obliczeniowym).

5.1. Zespół klasyfikatorów NECSCF oparty na DTW

W niniejszej części pracy opisano wersję algorytmu NECSCF opartą na DTW. Algorytm NECSCF pozwala na połączenie cech wyznaczonych przez sieci neuronowe z cechami wyznaczonymi przez inne metody. W zespole NECSCF opartym na DTW uczenie cech specyficznych dla klasy odbywa się dwuetapowo, co zaprezentowano na rys. 5.1. W pierwszym etapie trenowano oddzielne konwolucyjne sieci neuronowe (CNN) typu jeden-przeciw-wszystkim na pojedynczych mapach głębi w celu wyznaczania cech specyficznych dla klas, reprezentujących kształt osoby. Dla każdej akcji cechy wyznaczone tym sposobem tworzą wielowymiarowe szeregi czasowe.

Liczba map głębi nawet w małych zbiorach danych HAR jest wystarczająca do trenowania konwolucyjnych sieci neuronowych, ale liczba sekwencji map głębi/wielowymiarowych szeregów czasowych jest znacznie mniejsza. Z tego powodu w drugim etapie, oprócz wielokanałowej konwolucyjnej sieci neuronowej 1D [135, 136], przebadano eksperymentalnie dwa podejścia do wyznaczania cech dla wielowymiarowych szeregów czasowych, odpowiednich dla małych zbiorów danych: cechy statystyczne i cechy wyznaczane przez bliźniacze sieci neuronowe



Rysunek 5.1: Zespół NECSCF oparty na cechach DTW i na cechach wyznaczonych przez wielokanałowe sieci konwolucyjne 1D.

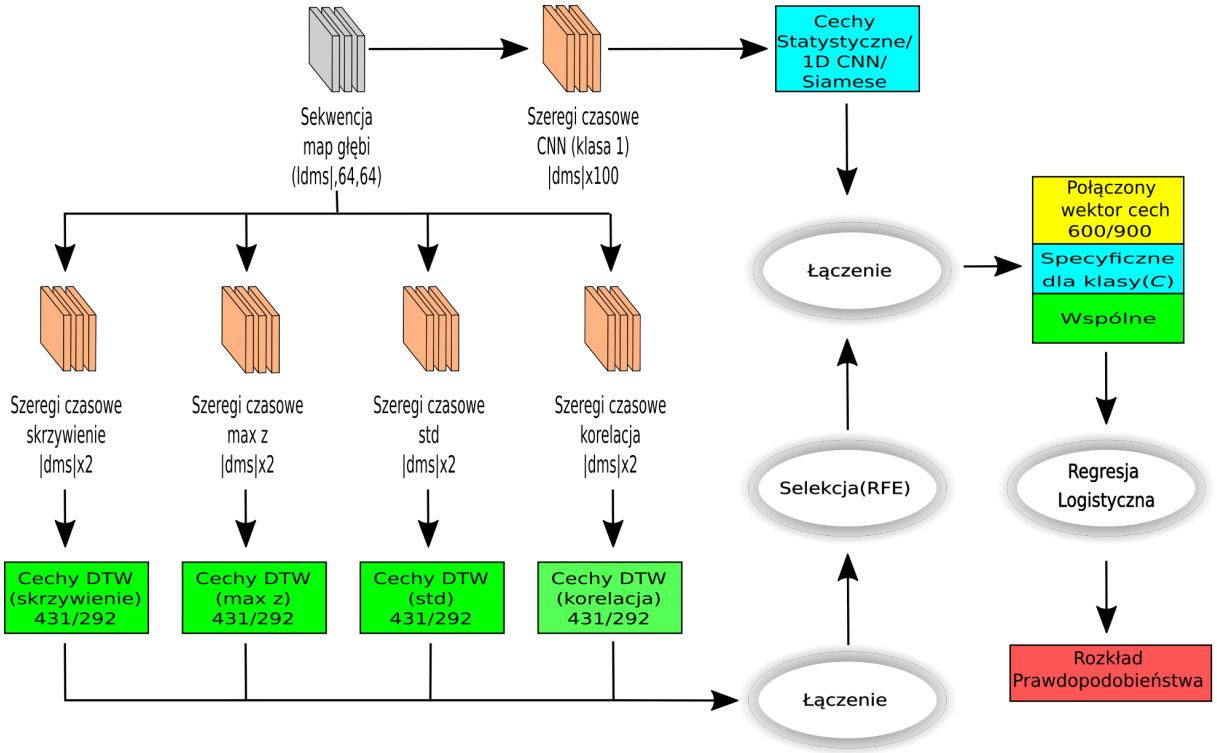
Siamese 1D [97, 35]. Wykorzystanie prostych cech statystycznych zapobiega nadmiernemu przeuczeniu (ang. *overfitting*). Sieci bliźniacze Siamese to rodzaj głębokich sieci neuronowych, które mogą być trenowane na zbiorach o małych rozmiarach. Zbiory na których są trenowane składają się z par przykładów, a nie pojedynczych przykładów.

Tabela 5.1: Cechy używane w zespole opartym na DTW. Kolumna „Nazwa" podaje nazwy cech. Kolumna „Opis" podaje podrozdział, w którym omówione są cechy. Kolumna „Typ danych" opisuje typ danych, na podstawie którego wyznaczane są cechy. Kolumna „Trenowane" podaje, czy cechy zostały wyznaczone przez sieci neuronowe. Kolumna „Typ cech" podaje, czy cechy są używane w zespole NECSCF jako cechy wspólne, czy też jako cechy specyficzne dla klasy.

Nazwa	Opis	Typ danych	Trenowane	Typ cech
Cechy DTW	3.3	Szereg czasowy	Nie	Wspólne
Cechy tworzone ręcznie	3.2	Mapa głębi	Nie	Wspólne
Cechy mapy głębi oparte na CNN	5.1.1	Mapa głębi	Tak	Specyficzne dla klasy
Cechy statystyczne	5.1.2	Szereg czasowy	Nie	Specyficzne dla klasy
Cechy 1D CNN	5.1.3	Szereg czasowy	Tak	Specyficzne dla klasy
Cechy 1D Siamese	5.1.4	Szereg czasowy	Tak	Specyficzne dla klasy

Jako wspólne cechy najpierw wyznaczono cechy tworzone ręcznie dla każdej mapy głębi. Cechy te są zaprojektowane do problemu rozpoznawania akcji człowieka. Jednak nie są one zaprojektowane dla konkretnego zbioru danych - umożliwiają uzyskanie dobrych dokładności klasyfikacji na różnorodnych zbiorach danych. Dla każdej pojedynczej mapy głębi wyznaczane

są zestawy cech tworzonych ręcznie (opisane w podrozdziale. 3.2). Następnie, na podstawie wielowymiarowych szeregów czasowych cech tworzonych ręcznie, wyznaczono cechy DTW. Jak już wspomniano, są one wyznaczane na podstawie odległości DTW między wszystkimi szeregami czasowymi treningowymi (zob. podrozdział 3.3 i rys. 3.2). Wszystkie cechy używane w zespole NECSCF opartym na DTW zebrano w tabeli 5.1.



Rysunek 5.2: Pojedynczy klasyfikator wieloklasowy wchodzący w skład zespołu NECSCF.

Bardziej szczegółowy obraz ilustrujący architekturę pojedynczego klasyfikatora w zespole NECSCF opartym na DTW zaprezentowano na rys. 5.2. Cechy tworzone ręcznie dla pojedynczych map głębi są podzielone na cztery oddzielne podzbiory, co zaprezentowano w tabeli 5.2. Dla każdego podzbioru tworzony jest osobny wielowymiarowy szereg czasowy. Na podstawie utworzonych tym sposobem wielowymiarowych szeregów czasowych wyznaczane są cechy DTW. Dokładna liczba cech DTW (dla każdego podzbioru) zależy od rozmiaru zbioru treningowego (431 w przypadku zbioru danych MHAD i 292 w przypadku zbioru danych MSR). Aby zredukować liczbę cech DTW, wykonano selekcję RFE. Spośród cech DTW obliczonych na podstawie wszystkich czterech podzbiorów (1724 cechy w przypadku zbioru danych MHAD i 1168 cech w przypadku zbioru danych MSR) wybrano 500 cech. Liczba cech specyficznych dla klasy zależy tylko od metody wyznaczania cech (400 dla cech statystycznych, 100 w przeciwnym przypadku).

Wektor cech akcji jest złożeniem cech wspólnych i cech specyficznych dla klasy, co pokazano na rys. 5.2. Klasyfikator wieloklasowy określa rozkład prawdopodobieństwa przynależności do klasy na podstawie wektora cech akcji. Ostateczna decyzja jest podejmowana

za pomocą miękkiego głosowania - rozkłady prawdopodobieństwa wyznaczane przez klasyfikatory są sumowane, a następnie wybierana jest klasa o największym prawdopodobieństwie [108].

Tabela 5.2: Podzbiór cech tworzonych ręcznie do opisu kształtu osoby na pojedynczej mapie głębi.

ID	Nazwa	Liczba cech	Opis	Osie
I	corl	3	Korelacja	zy,xz,zy
II	max z	2	Współrzędne najbliższego punktu (w stosunku do obserwatora) na mapie głębi	x,y
III	std	3	Odchylenie standardowe	x,y,z
IV	skew	3	Skośność	x,y,z

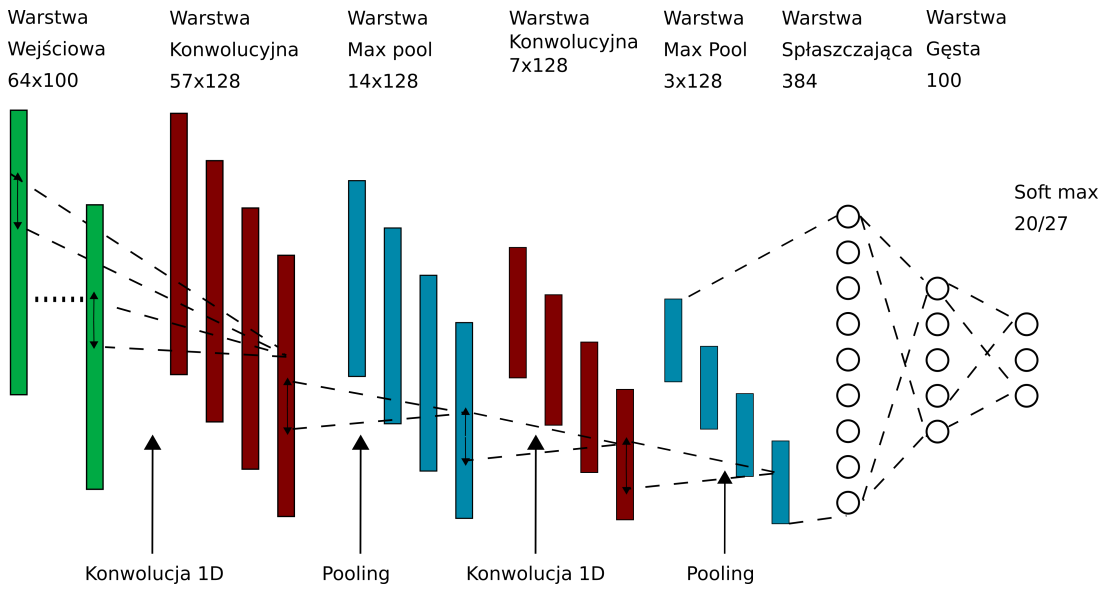
5.1.1. Cechy na mapach głębi wyznaczone przez sieci CNN

Jak już wspomniano w podrozdziale 1.2, w niniejszej rozprawie doktorskiej zajmujemy się rozpoznawaniem akcji na małych zbiorach danych. Zbiory danych do rozpoznawania akcji opartej na mapach głębi nie posiadają wystarczającej liczby przykładów (sekwencji map głębi), aby wytrenować konwencjonalne modele głębokie o odpowiednich zdolnościach generalizacji. Dlatego wykorzystano sieci CNN działające na pojedynczych mapach głębi do wyznaczania cech reprezentujących kształt osoby [137]. W omawianym podejściu, dla każdej klasy akcji trenujemy oddzielną sieć CNN, która ma za zadanie przewidzieć, czy rozważana mapa głębi należy do klasy, dla której sieć została wytrenowana, czy do jednej z pozostałych klas. Jest to wariant klasyfikatora typu jeden-przeciw-wszystkim [138]. Każda sieć jest trenowana na pojedynczych mapach głębi należących do rozważanej klasy oraz mapach głębi pobranych ze zbioru obrazów należących do pozostałych klas. Każda sieć neuronowa jest trenowana w 200 epokach, przy rozmiarze batch równym 32, wykorzystując momentum Nesterova z wartością współczynnika uczenia równą 0,001 i wartością momentum równą 0,9. Jako funkcję straty stosujemy entropię krzyżową kategoryczną (ang. *categorical crossentropy*).

Po wytrenowaniu sieci CNN wykorzystano je do wyznaczania cech reprezentujących kształt osoby na podstawie nieprzetworzonych map głębi. Wyznaczono 100 cech na podstawie przedostatniej warstwy sieci CNN. Architektura sieci neuronowej została przedstawiona na rys. 5.3. Ponieważ liczba klatek w sekwencjach map głębi nie jest identyczna, długości wielowymiarowych szeregów czasowych reprezentujących akcje mogą się różnić.

5.1.2. Statystyczne cechy szeregów czasowych

Dla każdego wielowymiarowego szeregu czasowego opartego na cechach mapy głębi wyznaczonych przez sieć CNN (szeregi czasowe mogą różnić się długością) obliczono cechy statystyczne [139] (STATS). Dla każdej cechy szeregu czasowego obliczono cztery wartości: średnią, odchylenie standardowe, skośność (ang. *skewness*) oraz korelację sygnału ze zmienną



Rysunek 5.3: Wielokanałowa sieć 1D CNN do wyznaczania cech specyficznych dla poszczególnych klas akcji.

czasu (dla szeregu czasowego s obliczono współczynnik korelacji Pearsona pomiędzy czasem $1 \ldots |s|$ a wartościami szeregu czasowego $s_1 \ldots s_{|s|}$).

Cechy wyznaczone tym sposobem nazywane są statystycznymi cechami czasowymi (STATS). Motywacją do uwzględnienia skośności było uwzględnienie informacji o asymetrii w rozkładzie prawdopodobieństwa zmiennej losowej [137]. Liczba cech STATS reprezentujących szeregi czasowe oparte na cechach reprezentujących osoby na mapach głębi z sieci CNN wynosi 400 (4×100).

Choć statystyczne cechy w zasadzie są tworzonymi ręcznie cechami, nie są one projektowane dla konkretnego zbioru danych ani nawet dla konkretnego rodzaju problemu, takiego jak zadanie rozpoznawania czynności człowieka. Ponadto, są one wyznaczane na podstawie cech wyznaczonych przez głęboką sieć neuronową, a nie bezpośrednio na podstawie danych. Są to proste statystyki opisowe, które mogą zostać wyznaczane dla dowolnego szeregu czasowego.

5.1.3. Cechy szeregów czasowych wydzielone przez 1D CNN

W wielokanałowych, czasowych sieciach konwolucyjnych typu 1D CNN, konwolucje 1D są realizowane w wymiarze czasowym [140]. Wielokanałowe 1D CNN są trenowane na szeregach czasowych cech reprezentujących mapy głębi, które zostały wyznaczone za pomocą dwuwymiarowej konwolucyjnej sieci neuronowej, opisanej w podrozdziale 5.1.1. Następnie sieć 1D CNN była używana do wyznaczania reprezentacji sekwencji map głębi. Jak wspomniano w [97], przedostatnia warstwa 1D CNN jest używana do wyznaczania wektora cech opisujących akcję. Liczba cech wynosi 100. Wielowymiarowe szeregi czasowe zostały interpolowane do długości równej 64.

Każda 1D CNN jest trenowana do klasyfikacji wszystkich akcji w rozważanym zbiorze

danych. Sieci operują na cechach specyficznych dla poszczególnych klas, które zostały omówione w podrozdziale 5.1.1. Szeregi czasowe zostały interpolowane do równej długości za pomocą interpolacji funkcją sklejaną (ang. *cubic-spline*). Ze względu na małą ilość danych treningowych w zbiorach danych wykorzystywanych w niniejszej pracy do rozpoznawania akcji na podstawie map głębi, wykorzystano płytką sieć neuronową składającą się dwóch warstw konwolucyjnych. Szczegółowa architektura sieci 1D CNN jest przedstawiona na rys. 5.3.

Pierwsza warstwa 1D CNN to filtr, który działa w wymiarze czasowym. Dla każdego wielowymiarowego szeregu czasowego opartego na cechach map głębi wyznaczonej za pomocą sieci CNN trenowano oddzielną wielokanałową 1D CNN. Pierwsza warstwa jest następnie poprzedzona dwiema warstwami konwolucyjnymi, z filtrem 8×1, max pooling 4×1 i 2×1, a następnie warstwą spłaszczającą (ang. *flattening*) i gęstą warstwą składającą się z 100 neuronów. Liczba neuronów wyjściowych jest równa liczbie klas.

Do uczenia sieci użyto optymalizatora Nesterov Accelerated Gradient (Nesterov momentum) przez 1000 iteracji, z wartością momentum równą 0,9, wartością dropout równą 0,5, współczynnikiem uczenia równym 0,001 i parametrem L1 równym 0,001. Jako funkcję straty użyto kategorycznej entropii krzyżowej. Po wytrenowaniu 1D CNN, wyjścia warstw gęstych zostały wykorzystane do wyznaczenia cech.

5.1.4. Cechy szeregów czasowych wyznaczane przez sieci bliźniacze 1D Siamese

Sieć neuronowa bliźniacza Siamese jest specjalnym rodzajem sieci, który współdzieli wagi podczas jednoczesnej pracy na dwóch różnych wektorach wejściowych w celu obliczenia podobieństwa tych wejść przez wyznaczenie i porównanie wektorów cech. Sieć neuronowa bliźniacza Siamese (SNN), została zaproponowana w pracy [58]. SNN zostały zaproponowane do redukcji wymiarowości [141] oraz do klasyfikacji *one-shot* obrazów [142]. Dla zbioru danych o rozmiarze N liczba par wynosi $\frac{N(N-1)}{2}$. Dlatego omawiana architektura może dawać dobre rezultaty przy stosunkowo małej ilości danych treningowych. SNN osiąga to przez redukcję wymiarowości danych i zastosowanie funkcji straty opartej na odległości, celem znalezienia wielowymiarowej reprezentacji, w której przykłady z tej samej klasy znajdują się blisko siebie, a z różnych klas są oddalone od siebie.

Podstawową ideą leżącą u podstaw sieci SNN jest wyznaczanie małowymiarowej reprezentacji danych, w której podobne pary danych są blisko siebie, a pary z przykładami z różnych klas są oddzielone od siebie, zaś odległość między nimi parametryzowana jest przez zmienną zwaną marginesem. Sieć neuronowa bliźniacza Siamese redukuje wymiarowość przez mapowanie przykładów wejściowych na wektory w przestrzeni o mniejszej wymiarowości w ten sposób, że należące do tej samej klasy wektory znajdują się w niewielkich odległościach od siebie. Jest ona trenowana na danych sparowanych (ts_p, ts_q, y_{pq}), gdzie minimalizowana jest odległość między parami przykładów o tej samej etykiecie, podczas gdy na pary z różnych klas nakładana jest kara za przyjmowanie wartości mniejszych niż margines m, a etykieta y_{pq} wskazuje, czy para

(ts_p, ts_q) pochodzi z tej samej klasy, czy nie. Funkcja straty kontrastowej (ang. *contrastive loss* [143]) L opisana jest równaniem:

$$L(\theta, ts_p, ts_q) = y_{pq} d_\theta(ts_p, ts_q)^2 - (1 - y_{pq}) \{max(0, M - d_\theta(ts_p, ts_q))\}^2 \tag{5.1}$$

$$\mathrm{d}_\theta(ts_p, ts_q) = |ts\_sim_\theta(ts_p) - ts\_sim_\theta(ts_q)| \tag{5.2}$$

gdzie:

ts_p, ts_q - szeregi czasowe

θ - parametry sieci bliźniaczej Siamese

$ts\_sim_\theta(ts_p)$ - reprezentacja wyznaczona przez sieć bliźniaczą Siamese

y_{pq} - równe 1 jeśli przykłady p i q należą do tej samej klasy i 0, gdy klasy są różne

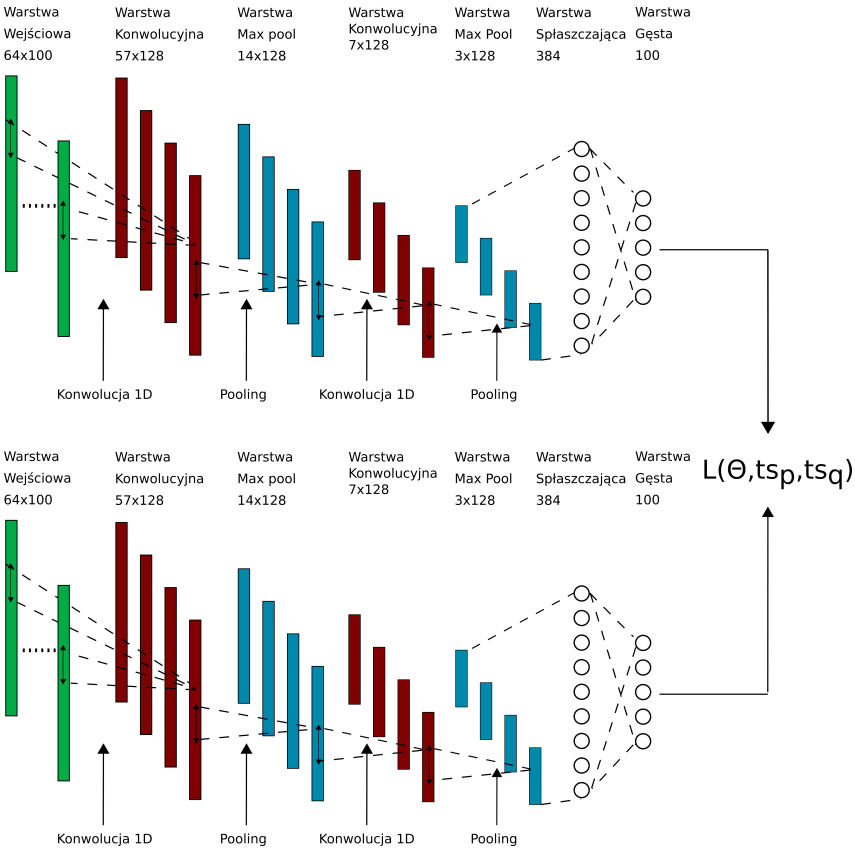
M - margines (liczba rzeczywista)

Funkcja $ts\_sim_\theta(ts_p)$ wyznacza reprezentacje przykładów ts_p jako wektor cech. Funkcja straty wyraża, jak dobrze reprezentacja wyznaczona przez sieć $ts\_sim_\theta(ts_p)$ umieszcza przykłady tej samej klasy w bliskiej odległości od siebie, a z różnych klas na dużych odległościach. Funkcja strat w przypadku par przykładów należących do tej samej klasy jest równa kwadratowi odległości euklidesowej, a w przypadku par przykładów należących do rożnych klas zero lub różnicy między marginesem M a odległością euklidesową (w zależności od tego, która liczba jest mniejsza).

W proponowanym rozwiązaniu sieć neuronowa bliźniacza Siamese operuje na szeregach czasowych cech opisujących osobę na mapie głębi i wyznacza na ich podstawie wektor cech (oznaczane jako SIM). Sieci bliźniacze zostały wykorzystane do wyznaczenia miary podobieństwa szeregów czasowych w pracy [144]. Rys. 5.4 przedstawia diagram wykorzystanej sieci neuronowej bliźniaczej Siamese. W pierwszej warstwie sieci zastosowano konwolucje 1D w wymiarze czasowym. Rozmiar danych wejściowych 64×100 wynika z długości zinterpolowanych szeregów czasowych pomnożonych przez liczbę kanałów. Pierwsza warstwa jest poprzedzona dwoma warstwami konwolucyjnymi, z których każda zawiera filtr o rozmiarze 8×1. Po nich następują warstwy max pooling o rozmiarach 4×1 i 2×1, a następnie warstwa spłaszczająca i gęsta warstwa składająca się z 100 neuronów. Sieci neuronowe bliźniacze Siamese zostały wytrenowane w 300 epokach, z batch size równym 32, regularyzacją L1 równą 0,01, przy użyciu optymalizatora Adam z współczynnikiem uczenia równym 0,00006.

5.2. Autorska metoda augmentacji szeregów czasowych

W niniejszej części pracy jako podstawowy algorytm dla zespołu klasyfikatorów przebadano wersję bardziej konwencjonalnego algorytmu zespołowego - baggingu [59]. Zaproponowano również nową metodę augmentacji danych dla wielowymiarowych szeregów czasowych - augmentację danych za pomocą deformacji w czasie (ang. *warp data augmentation*,

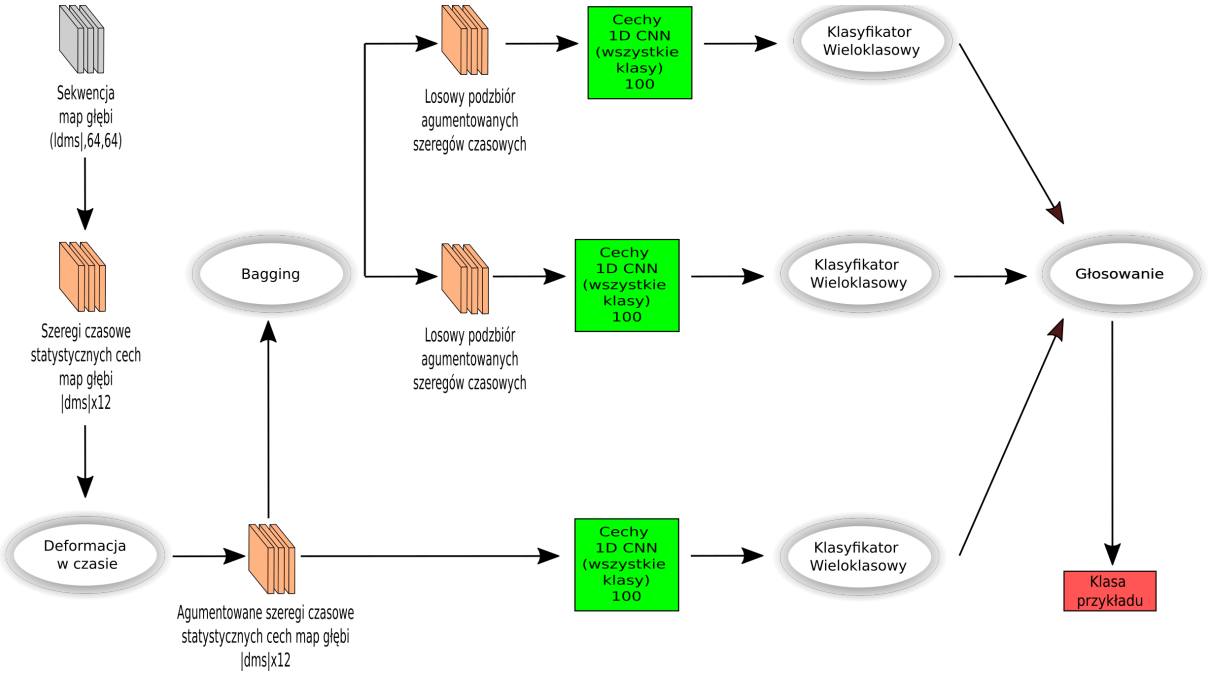


Rysunek 5.4: Sieć bliźniacza Siamese wyznaczająca cechy specyficzne dla danej klasy.

zob. 3.3.1). Motywacją dla proponowanej metody augumentacji danych jest chęć osiągnięcia zalet cech opartych o DTW w klasyfikacji akcji wykonywanych z różną prędkością i jednocześnie uniknięciem kosztu obliczeniowego DTW. Na zaugmentowanych szeregach czasowych cech trenowano wieloklasowe, wielokanałowe sieci neuronowe konwolucyjne (1D CNN) do modelowania wielowymiarowych szeregów czasowych.

Algorytm został zilustrowany na rys. 5.5 oraz opisany w pseudokodzie 4. Na wejściu znajdują się przykłady treningowe (train_samples), etykiety przykładów treningowych (train_labels), przykłady testowe (test_labels), liczba augmentowanych przykładów wygenerowanych na podstawie jednego przykładu z oryginalnego zbioru (n_augm) oraz liczba klasyfikatorów (n_clf). Najpierw (linie 2-6) dla każdego szeregu czasowego generowane jest 6 dodatkowych szeregów czasowych za pomocą funkcji GEN_WARP_AGUM. Następne (linia 7-10) w oparciu o augumentowane szeregi czasowe trenowane są klasyfikatory.

Wielokanałowe konwolucyjne sieci neuronowe 1D CNN są trenowane w oparciu o szeregi czasowe losowane z powtórzeniami. Cechy wyznaczane przez takie modele baggingowe są używane do trenowania klasyfikatorów SVM i wieloklasowych klasyfikatorów regresji logistycznej [119]. Zespół oparty na klasyfikatorach SVM i regresji logistycznej dokonuje klasyfikacji. Ostateczna decyzja jest podejmowana przez miękkie głosowanie. Pokazano, że pomimo ogra-



Rysunek 5.5: Schemat algorytmu dla klasyfikacji akcji człowieka z wykorzystaniem augmentacji wielowymiarowych szeregów czasowych.

niczonej liczby danych w zbiorze trenującym możliwe jest wytrenowanie cech szeregów czasowych o wysokiej sile dyskryminacyjnej.

W algorytmach zespołowych agregacja bootstrapowa, zwana również baggingiem, polega na budowaniu kilku instancji klasyfikatora na losowych podzbiorach oryginalnego zbioru treningowego, a następnie agregowaniu ich indywidualnych predykcji w celu podjęcia ostatecznej decyzji. Dzięki wprowadzeniu losowości do procedury treningowej i utworzeniu zespołu klasyfikatorów trenowanych na losowych podzbiorach, redukuje się wariancje predyktora [59, 145]. Bagging jest również sposobem na zmniejszenie przeuczenia, bez konieczności adaptacji podstawowego klasyfikatora [146].

5.2.1. Autorski algorytm do augmentacji szeregów czasowych

Wielokanałowe sieci konwolucyjne (1D CNN) zostały wytrenowane na szeregach czasowych po ich augmentacji. Architektura 1D CNN jest taka sama, jak opisana w podrozdziale 5.1.3, z wyjątkiem mniejszej liczby kanałów wejściowych, równych liczbie cech tworzonych ręcznie dla pojedynczych map głębi (czyli 12). Dodatkowe cztery szeregi czasowe zostały wygenerowane dla każdej sekwencji danych wejściowych przez:

1. Wybranie pierwszych szesnastu elementów szeregu czasowego, a następnie interpolację do długości 8,
2. Wybranie pierwszych szesnastu elementów szeregu czasowego, a następnie interpolację do długości 32,

Algorytm 4 Zespół bagging z zastosowaniem augmentacji przez deformacje w czasie.

```
1: function WARPBAGGING(train_samples,train_labels,test_samples,n_augm,n_clf)
2:     for sample_i ∈ train_samples do
3:         for augm_sample_j ∈ GEN_WARP_AGUM(sample_i) do
4:             augm_samples[n_augm × i + j] = augm_sample_j
5:         end for
6:     end for
7:     for k ← 1 ... n_clf do
8:         subset_of_samples_k ← RANDOM(augm_samples)
9:         bagging_models[k] ← TRAIN_MODEL(subset_of_samples_k, train_labels)
10:    end for
11:    for sample_t ∈ test_samples do
12:        predicted_labels[t] ← VOTING(sample_t, bagging_models)
13:    end for
14:    return predicted_labels
15: end function
```

3. Wybranie ostatnich szesnastu elementów szeregu czasowego, a następnie interpolację do długości 8,
4. Wybranie ostatnich szesnastu elementów szeregu czasowego, a następnie interpolację do długości 32.

Następnie takie podsekwencje są dodawane do sekwencji danych wejściowych, a ostatecznie interpolowane do długości równej 128. Interpolacja jest przeprowadzana w oparciu o algorytm kubicznych splajnów (ang. *cubic spline*) [147]. Następnie dwa dodatkowe szeregi czasowe zostały wygenerowane przez skalowanie sekwencji danych w dziedzinie czasu przez 2 i 0,5.

Motywacją do stosowania augmentacji danych za pomocą autorskiej metody jest uzyskanie przynajmniej części korzyści płynących z metod DTW bez ponoszenia kosztów obliczeniowych z nimi związanych. Akcje człowieka (oprócz stania w miejscu) w różnych przykładach zaczynają się i kończą w różnych częściach sekwencji mapy głębi. Dzięki odpowiedniej interpolacji początku i końca sekwencji, możemy przezwyciężyć ograniczenia wynikające z niedostatecznej ilości danych treningowych.

5.3. Wyniki uzyskane w oparciu o NECSCF i DTW

Zaproponowany algorytm NECSCF, opisany w pkt. 5.1, przebadano eksperymentalnie na dwóch ogólnie dostępnych benchmarkowych zbiorach danych: MSR Action3D [97] (zobacz również podrozdział 3.4.1) [92] oraz UTD-MHAD [93] (zobacz również podrozdział 3.4.2). Zbiory danych zostały wybrane ze względu na ich częste wykorzystanie przez społeczność za-

jmującą się rozpoznawaniem akcji w celu oceny i porównania różnych algorytmów. W obu zbiorach danych tło zostało już usunięte. Jedynym etapem przetwarzania wstępnego było wyodrębnienie prostokąta, który ogranicza piksele reprezentujące kształt osoby.

W niniejszej części pracy przebadano 12 konfiguracji eksperymentalnych dla zespołowego algorytmu NECSCF z dwoma rodzajami cech wspólnych, scharakteryzowanych w tabeli 5.3 i trzema rodzajami cech specyficznych dla klasy opisanych w tabeli 5.4.

Tabela 5.3: Rodzaje cech wspólnych.

ID	Selekcja cech	Opis
-	-	Brak cech wspólnych, używane są tylko cechy specyficzne dla klasy
DTW	Nie	Cechy DTW wyznaczone na podstawie cech pojedynczych map głębi opisanych w sekcji 5.1
DTW RFE	Tak	Cechy oparte na DTW wybrane przez RFE

Tabela 5.4: Rodzaje cech specyficznych dla klasy. Wszystkie cechy specyficzne dla klasy są wyznaczone na podstawie cech opartych na sieciach neuronowych typu CNN (zob. pkt. 5.1.1).

Etykieta	Selekcja cech	Opis
-	-	Brak cech specyficznych dla klasy, używane są tylko cechy wspólne
1D-CNN	Nie	Cechy wyznaczone przez jednowymiarową sieć konwolucyjną
SIM	Nie	Cechy wyznaczone przez sieć bliźniaczą Siamese
STATS	Nie	Cztery statystyczne cechy szeregów czasowych opisane w podrozdziale 5.1.2

Przedstawiono również wyniki uzyskane w oparciu o cechy DTW bez użycia uczenia zespołowego. Dodatkowo przeprowadzono selekcję klasyfikatorów. Wygenerowano 1000 zestawów klasyfikatorów, gdzie każdy zestaw jest tworzony przez losowy podzbiór k klasyfikatorów spośród wszystkich C klasyfikatorów, przy czym k jest losowane z zakresu liczb całkowitych od 2 do C, a C oznacza liczbę klas. W celu wyboru optymalnego podzbioru klasyfikatorów, zbiór treningowy podzielono na dwie części. Jedna służy do trenowania klasyfikatorów (zbiór podstawowy), a druga do oceny podzbiorów klasyfikatorów (zbiór walidacyjny). Dla każdego podzbioru klasyfikatorów spośród 1000 zestawów klasyfikatorów (trenowanych na zbiorze podstawowym) obliczono dokładność (na zbiorze walidacyjnym) osiągniętą przez zespół składający się z podzbioru klasyfikatorów. Przykłady zostały podzielone losowo - 80% przykładów przyporządkowano do zbioru podstawowego, a 20% przykładów przypisanych jest do zbioru walidacyjnego. Podziały są identyczne dla wszystkich klasyfikatorów.

Tabela 5.5 przedstawia wyniki uzyskane na zbiorze danych MSR Action 3D. Tabela 5.6 zawiera szczegóły o liczbie i rodzajach cech użytych w eksperymentach. Najlepszy wynik jest pogrubiony, zaś drugi najlepszy wynik jest podkreślony. Porównując wyniki w wierszach A2

Tabela 5.5: Wyniki rozpoznawania akcji na zbiorze danych MSR Action 3D. Pierwsza kolumna podaje identyfikator eksperymentu. Druga kolumna podaje rodzaj cech wspólnych (zob. tabela 5.3). Trzecia kolumna podaje rodzaj cech specyficznych dla klasy (zob. tabela 5.4). Czwarta kolumna podaje liczbę klasyfikatorów w zespole.

ID	Cechy wspólne	Cechy specyficzne dla klasy	Liczba klasyfikatorów	Dokładność	Precyzja	Czułość	F1-score
A1	–	1D-CNN	20	0,7273	0,7317	0,7273	0,6926
A2	–	STATS	20	0,7818	0,8047	0,7818	0,7621
A3	–	SIM	20	0,7345	0,7265	0,7345	0,7120
B	DTW	–	1	0,8764	0,8857	0,8764	0,87
C1	DTW	1D-CNN	20	0,9091	0,9173	0,9091	0,905
C2	DTW	STATS	20	0,9164	0,9243	0,9164	0,9131
C3	DTW	SIM	20	0,9091	0,9139	0,9091	0,9034
D1	DTW RFE	1D-CNN	20	0,9091	0,9129	0,9091	0,9059
D2	DTW RFE	STATS	20	0,9309	0,9370	0,9309	0,9292
D3	DTW RFE	SIM	20	0,9200	0,9204	0,92	0,9156
E1	DTW RFE	1D-CNN	4	0,9309	0,9323	0,9309	0,9278
E2	DTW RFE	STATS	6	**0,9564**	0,9594	0,9564	0,9542
E3	DTW RFE	SIM	8	0,9455	0,9471	0,9455	0,9421

z A1 i A3, wyniki w wierszach C2 z C1 i C3, a następnie D2 z D1 i D3, oraz E2 z E1 i E3, można zauważyć, że cechy statystyczne umożliwiają uzyskanie lepszych wyników w porównaniu do cech 1D-CNN oraz cech Siamese. Cechy DTW umożliwiają uzyskanie znacznie lepszych dokładności klasyfikacji akcji w porównaniu do cech statystycznych, cech 1D-CNN oraz cech Siamese, co ilustruje zestawienie wyników w wierszu B z wynikami w wierszach A1, A2 i A3. Połączenie cech DTW z cechami Siamese, cechami 1D-CNN lub cechami statystycznymi prowadzi do znacznego wzrostu dokładności klasyfikacji. Jednak nie wszystkie cechy DTW mają dużą moc dyskryminacyjną. Zastosowanie RFE prowadzi do dalszego poprawienia dokładności klasyfikacji, co obrazują wyniki w wierszach D1-D3. Zespół zbudowany na wybranych klasyfikatorach spośród puli dwudziestu klasyfikatorów oraz cechach DTW wybranych przez RFE pozwala na dalszy wzrost dokładności klasyfikacji, na co wskazują wyniki w wierszach E1-E3. Najlepsze rezultaty osiągnięto dzięki zespołowi zbudowanemu na klasyfikatorach operujących na specyficznych dla klas cechach statystycznych, które zostały połączone z cechami DTW użytymi jako cechy wspólne, co ilustrują wyniki w wierszu E2.

Tabela 5.6: Liczba cech w zbiorze danych MSR Action 3D.

Cechy	Typ	Liczba cech	Opis
DTW	Wspólne	1168	Liczba przykładów w zbiorze treningowym × liczba podzbiorów cech
DTW RFE	Wspólne	350	Hiperparametr
1D-CNN	Specyficzne dla klasy	100	Hiperparametr
STATS	Specyficzne dla klasy	400	Liczba szeregów czasowych (100) razy cztery
SIM	Specyficzne dla klasy	100	Hiperparametr

5.3.1. UTD-MHAD

Tabela 5.7 przedstawia wyniki uzyskane na zbiorze danych UTD-MHAD. Tabela 5.8 podaje liczby cech i rodzaje cech użyte w eksperymentach. Porównując wyniki w wierszach A1, A2 i A3, można zauważyć, że cechy 1D-CNN pozwalają osiągnąć lepsze rezultaty w porównaniu do algorytmu operującego zarówno na cechach statystycznych, jak i cechach Siamese.

Cechy DTW pozwalają uzyskać lepsze wyniki w porównaniu do omawianych wcześniej cech, co obrazują wyniki w wierszu B i wierszach B2-B3, z wyjątkiem cech 1D-CNN, jak pokazują wyniki w wierszu B i wierszu B1. Połączenie cech DTW z cechami 1D-CNN lub statystycznymi, lub cechami Siamese prowadzi do znacznego wzrostu dokładności klasyfikacji. Cechy DTW wybrane przez RFE i połączone z cechami 1D-CNN lub statystycznymi, lub cechami Siamese umożliwiają uzyskanie lepszych dokładności klasyfikacji w porównaniu do dokładności klasyfikacji uzyskiwanych przez algorytmy bez RFE. Najlepszą dokładność klasyfikacji uzyskał algorytm oparty na klasyfikatorach specyficznych dla klas, trenowanych na cechach 1D-CNN, które zostały połączone z cechami DTW użytymi jako cechy wspólne. Jak można zauważyć, najlepszą dokładność klasyfikacji uzyskał zespół klasyfikatorów złożony z siedmiu klasyfikatorów wybranych spośród dwudziestu siedmiu klasyfikatorów oraz cechom DTW wybranym przez RFE. Selekcja cech wraz z selekcją klasyfikatorów prowadzi do znacznie lepszych wyników. W konfiguracji, w której algorytm był tworzony przy użyciu cech Siamese zamiast cech 1D-CNN, do osiągnięcia najlepszych wyników wystarczyło tylko pięć klasyfikatorów specyficznych dla klas, co obrazują wyniki w ostatnim wierszu. Zespół używający cech Siamese, które zostały połączone z cechami DTW uzyskał nieco gorsze wyniki od zespołu używającego cech 1D-CNN.

5.4. Wyniki prac eksperymentalnych w oparciu o deformacje w czasie

Zaproponowany algorytm zespół bagging z agumentacją przez deformacje w czasie (zob. podrodział 5.2) przebadano eksperymentalnie na tym samych zbiorach danych, na których ewaluowany był zespół NECSCF: zbiorze danych MSR Action3D [97] oraz zbiorze danych UTD-MHAD [93]. Wszystkie algorytmy klasyfikacji operują na wielowymiarowych szeregach

Tabela 5.7: Wyniki rozpoznawania akcji na zbiorze danych UTD-MHAD.

ID	Common	Binary	Classifiers	Accuracy	Precision	Recall	F1-score
A1	–	1D-CNN	27	0,7907	0,8119	0,7907	0,7843
A2	–	STATS	27	0,7302	0,7354	0,7302	0,7153
A3	–	SIM	27	0,7767	0,8021	0,7767	0,774
B	DTW	–	1	0,7860	0,8146	0,7860	0,7842
C1	DTW	1D-CNN	27	0,8628	0,8772	0,8628	0,8614
C2	DTW	STATS	27	0,8326	0,8461	0,8326	0,8301
C3	DTW	SIM	27	0,8339	0,8349	0,8552	0,8349
D1	DTW RFE	1D-CNN	27	0,8698	0,8842	0,8698	0,8694
D2	DTW RFE	STATS	27	0,8419	0,8617	0,8419	0,8386
D3	DTW RFE	SIM	27	0,8349	0,858	0,8349	0,8347
E1	DTW RFE	1D-CNN	7	**0,8814**	0,8979	0,8814	0,8804
E2	DTW RFE	STATS	13	0,8605	0,8716	0,8605	0,8578
E3	DTW RFE	SIM	5	0,8744	0,8935	0,8744	0,8766

Tabela 5.8: Liczba cech na zbiorze danych UTD-MHAD Action 3D.

Cechy	Typ	Liczba cech	Opis
DTW	Wspólne	1724	Liczba przykładów w zbiorze treningowym × liczba podzbiorów cech
DTW RFE	Wspólne	300	Hiperparametr
1D-CNN	Specyficzne dla klasy	100	Hiperparametr
STATS	Specyficzne dla klasy	400	Liczba szeregów czasowych (100) razy cztery
SIM	Specyficzne dla klasy	100	Hiperparametr

czasowych składających się z cech tworzonych ręcznie (zob. pkt. 3.2). Zbadano trzy algorytmy:

1. 1D CNN - pojedyncza wielokanałowa, konwolucyjna sieć neuronowa trenowana na szeregach czasowych cech tworzonych ręcznie (wyznaczonych na podstawie pojedynczych map głębi). Cechy map głębi są opisane w podrozdziale 3.2.
2. 1D CNN z deformacją w czasie - zespół zawiera dwa klasyfikatory różniące się rodzajem augmentacji danych. W pierwszym podejściu do augmentacji danych używamy tylko augmentacji danych przez deformację w czasie (zob. 3.2.1). W drugim algorytmie danych używamy zarówno augmentacji danych z deformacją w czasie, jak i skalowania szeregów czasowych (skalowanie przykładów w dziedzinie czasu przez 2 i 0,5).
3. Zespół bagging z deformacją w czasie - zespół składa się z siedmiu klasyfikatorów, z których każdy operuje na danych zaugmentowanych w identyczny sposób, a także trenowany na losowych podzbiorach tego samego zbioru danych.

Tabela 5.9 przedstawia wyniki eksperymentalne uzyskane przez omawiane algorytmy na zbiorze danych MSR Action 3D. Jak można zauważyć, najlepsze wyniki osiągnął zespół bagging z deformacją w czasie. Porównując wyniki eksperymentalne uzyskane przez 1D CNN z deformacją w czasie, można zauważyć, że użycie autorskiej augmentacji przez deformacje w czasie wraz z algorytmem bagging poprawiło precyzję (lecz nie dokładność) klasyfikacji akcji.

Tabela 5.10 przedstawia wyniki eksperymentalne uzyskane przez omawiane algorytmy na zbiorze danych UTD-MHAD. Wyniki uzyskane przez klasyfikator 1D CNN przedstawiono w pierwszym wierszu. W kolejnych dwóch wierszach zaprezentowano wyniki eksperymentalne uzyskane przez zespoły klasyfikatorów. Można zauważyć, że na cechach pojedynczych map głębi najlepsze wyniki osiągnęła sieć 1D CNN na danych z deformacją w czasie [148].

Tabela 5.9: Wyniki uzyskane na zbiorze danych MSR Action 3D. Kolumna „Algorytm" zawiera użyty algorytm uczenia. Kolumna „Klasyfikatory" zawiera liczbę klasyfikatorów użytych w zespole.

Algorytm	Klasyfikatory	Cechy	Accuracy	Precision	Recall	F1-score
1D CNN	1	12	0,9345	0,9419	0,9345	0,9344
1D CNN z deformacją w czasie	2	12	**0,9455**	0,9471	0,9455	0,9438
Zespół bagging z deformacją w czasie	7	12	**0,9455**	0,9517	0,9455	0,9435

Tabela 5.10: Wyniki uzyskane na zbiorze danych UTD-MHAD. Kolumna „Algorytm" zawiera użyty algorytm uczenia. Kolumna „Klasyfikatory" zawiera liczbę klasyfikatorów użytych w zespole.

Algorytm	Klasyfikatory	Cechy	Accuracy	Precision	Recall	F1-score
1D CNN	1	12	0,8349	0,8537	0,8349	0,8319
1D CNN z deformacją w czasie	2	12	**0,8651**	0,8788	0,8651	0,8624
Zespół bagging z deformacją w czasie	7	12	0,8535	0,8648	0,8535	0,8496

5.5. Podsumowanie

Zaprezentowane wyniki badań eksperymentalnych wskazują, że zarówno zaproponowany zespół NECSCF, jak i zespół bagging z deformacją w czasie, uzyskują obiecujące wyniki na zbiorach danych MSR-Action3D i UTD-MHAD i przewyższają kilka zaawansowanych algorytmów operujących na mapach głębi, które przedstawiono w tabeli 5.12 i tabeli 5.13. Najlepszy wynik jest pogrubiony, zaś drugi najlepszy wynik jest podkreślony. W tabeli 5.11 przedstawiono modalności używane przez omawiane algorytmy.

Tabela 5.11: Modalności używane przez klasyfikatory. Metody zaproponowane w niniejszej pracy operują jedynie na mapach głębi.

Nazwa	Modalność
Szkielet	Pozycja stawów szkieletu człowieka z sensora Kinect
Mapy głębi (MG)	Mapy głębi z sensora Kinect
RGB	Zwykłe obrazy RGB

Tabela 5.12: Porównanie wyników zaproponowanej metody z ostatnimi algorytmami na zbiorze danych MSR Action 3D. Kolumna "Metoda" zawiera nazwę metody użytej w ewaluacji na zbiorze danych. Kolumna "Protokół" zawiera podział zbioru danych (zob. pkt. 3.4.1). Kolumna "Modalność" charakteryzuje modalności danych używanych w algorytmie (zob. tabela 5.11). Kolumna "Dokładność" zawiera osiągniętą dokładność klasyfikacji akcji przez algorytm na zbiorze danych.

Metoda	Protokół	Modalność	Dokładność
3DCNN [31]	Split II	Mapy głębi - MG	84,07
DMMs [96]	Split II	Mapy głębi - MG	88,73
PRNN [149]	Split II	Mapy głębi - MG	94,90
WHDMM + CNN [96]	Split I	Mapy głębi - MG	**100,00**
S DDI [93]	Split I	Mapy głębi - MG	**100,00**
3DHoT-MBC [150]	Split I	Mapy głębi - MG + RGB	95,24
Action-fusion [151]	Split I	Mapy głębi - MG	94,51
Deep Sequence Learning Framework [152]	Split I	Mapy głębi - MG	96,00
Zespół bagging z deformacją w czasie	Split I	Mapy głębi - MG	94,55
NECSCF	Split I	Mapy głębi - MG	95,64

Porównując wyniki zamieszczone w tabelach 5.12 i 5.13 zauważyć można, że choć algorytm WHDMM, który operuje jedynie na mapach głębi, osiągnął lepsze dokładności klasyfikacji niż proponowany algorytm, to jednak uzyskuje gorsze rezultaty na zbiorze danych UTD-MHAD w porównaniu do rezultatów osiągniętych przez zaproponowany algorytm. Nieco lepsze wyniki osiągnięto na zbiorze danych MSR Action 3D w pracy [152].

Spośród trzech rodzajów cech specyficznych dla klasy, cechy STATS umożliwiają uzyskanie najlepszego wyniku na zbiorze MSR, 1D-CNN pozwala na uzyskanie najlepszych wyników na zbiorze MHAD, a cechy Siamese uzyskują drugie najlepsze wyniki zarówno na zbiorze MSR, jak i MHAD. W oparciu o cechy Siamese uzyskać można wyniki tylko nieznacznie gorsze od 1D-CNN na zbiorze MHAD. Wyniki potwierdzają to, że sieci bliźniacze Siamese są obiecującą metodą dla małych zbiorów danych HAR (ponieważ dają konsekwentnie dobre wyniki na różnych zbiorach danych).

Zespół bagging z deformacją w czasie daje dobre wyniki zarówno na zbiorze MSR, jak i MHAD. Metoda baggingu nie prowadzi do poprawy, ale nowa metoda augmentacji danych zaproponowana w niniejszej pracy (augumentacja za pomocą deformacji w czasie), umożliwia

Tabela 5.13: Porównanie proponowanej metody z konkurencyjnymi algorytmami na zbiorze danych UTD-MHAD. Kolumna "Metoda" zawiera nazwę metody użytej w ewaluacji na zbiorze danych. Kolumna "Modalność" zawiera modalności danych używanych w algorytmie (zob. tabela 5.11). Kolumna "Dokładność" zawiera osiągniętą dokładność przez algorytm na tym zbiorze danych.

Metoda	Modalność	Dokładność [%]
JTM [31]	Szkielet	85,81
SOS [149]	Szkielet	86,97
Kinect i inercjalne [93]	Szkielet	79,10
Strukt. ciało [93]	Szkielet	66,05
Strukt. część [93]	Szkielet	78,70
Strukt. staw [93]	Szkielet	86,81
Strukt. S DDI [93]	Szkielet	**89,04**
3DHoT-MBC [150]	Mapy głębi - MG + RGB	84,40
WHDMMs + ConvNets [96, 93]	Mapy głębi - MG	73,95
Action-fusion [151]	Mapy głębi - MG	88,14
1D CNN z deformacją w czasie	Mapy głębi - MG	86,51
NECSCF	Mapy głębi - MG	88,14

uzyskanie znacznie lepszych wyników na obydwu zbiorach danych.

Algorytm NECSCF opracowany w niniejszej pracy osiąga lepsze wyniki niż bardziej konwencjonalny algorytm bagging na tych samych szeregach czasowych cech map głębi, przy zastosowaniu dedykowanej dla rozpoznawania akcji człowieka augmentacji danych. Wadą cech DTW używanych w tej wersji algorytmu NECSCF jest wysoki koszt obliczeniowy odległości DTW. Problem ten wynika z użycia tych konkretnych cech wspólnych i nie dotyczy ogólnie algorytmu NECSCF. Warto podkreślić, że zaproponowana metoda operuje jedynie na mapach głębi. Sprawia to, że proponowany algorytm jest bardziej uniwersalny niż algorytmy oparte na szkielecie, m.in. z uwagi na możliwość zastosowania w zadaniach interakcji człowieka z obiektami.

6. Cechy głębokie do rozpoznawania akcji na nieprzetworzonych mapach głębi

W rozdziale 5 zaproponowano wersję algorytmu NECSCF, która uzyskuje lepsze dokładności klasyfikacji niż większość alternatywnych algorytmów. Jednak wariant zaproponowany we wspomnianym rozdziale - NECSCF oparty na DTW - jest skomplikowany i charakteryzuje się wysokim kosztem obliczeniowym. W tym rozdziale proponujemy warianty NECSCF oparte na bardziej złożonych architekturach uczenia głębokiego. Założone cele to:

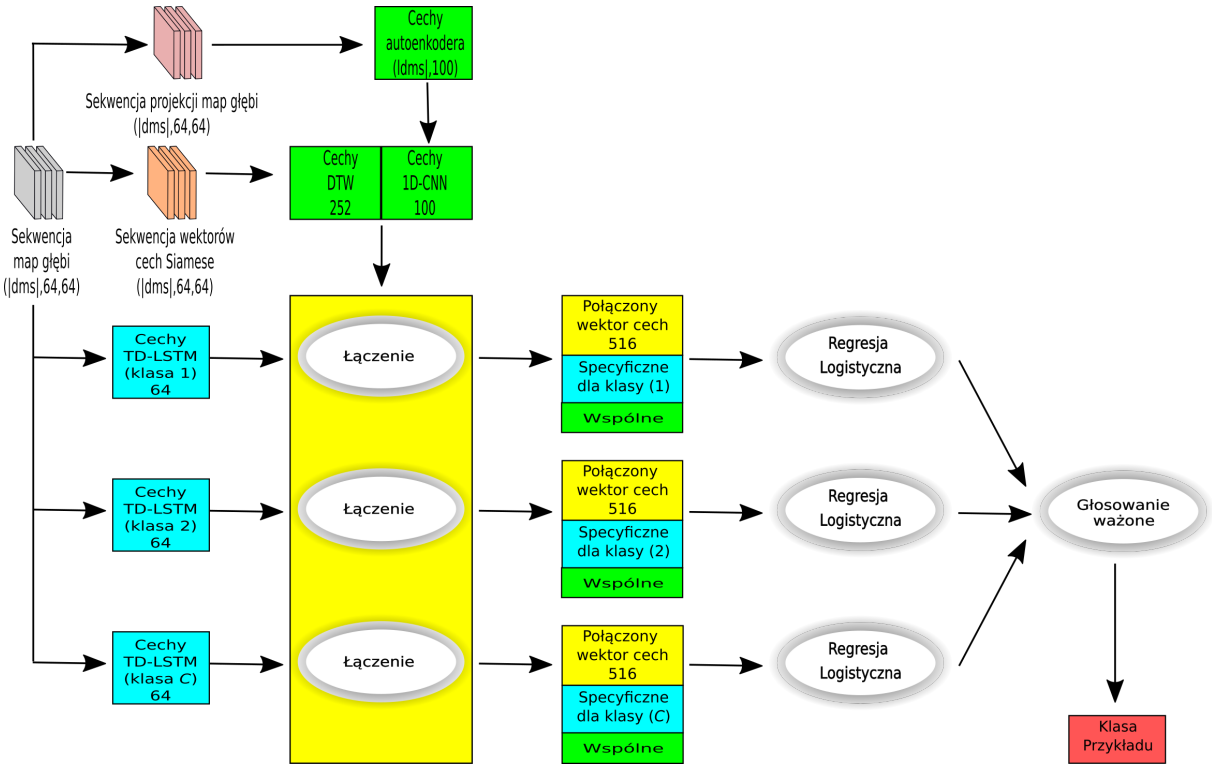
1. Usunięcie lub ograniczenie liczby cech tworzonych ręcznie używanych w algorytmie. W rozdziale 3 metody oparte na cechach tworzonych ręcznie osiągnęły słabe wyniki na zbiorze 3DHOI. Zbiór 3DHOI zawiera akcje polegające na interakcji z obiektami. Może to świadczyć, że zaproponowane cechy tworzone ręcznie nie sprawdzają się w tym problemie. Dlatego badamy bardziej elastyczną metodę wyznaczania cech przez sieć neuronową.
2. Wykorzystanie sieci neuronowych trenowanych end-to-end do wyznaczania cech specyficznych dla klasy.
3. Zbadanie potencjału sieci bliźniaczych Siamese w rozpoznawania akcji. Algorytm zaprezentowany w poprzednim rozdziale uzyskuje obiecujące rezultaty, używając jednowymiarowej sieci bliźniaczej Siamese (trenowanej na szeregach czasowych). W tym rozdziale badamy sieci bliźniacze Siamese trenowane do ekstrakcji cech reprezentujących osobę na pojedynczych mapach głębi.

W niniejszym rozdziale proponujemy wersję zespołu NECSCF opartego na sieciach bliźniaczych Siamese. Algorytm działania zilustrowano na rys. 6.1.

1. Cechy specyficzne dla klasy - wyznaczone są za pomocą sieci TD-LSTM.
2. Cechy wspólne - użyto dwóch zestawów cech wspólnych. Pierwszy opiera się na autoenkoderze konwolucyjnym (CAE) [153]. Autoenkoder wyznacza cechy reprezentujące osoby na pojedynczych mapach głębi. Na podstawie cech wyznaczonych tym sposobem trenowana jest sieć 1D-CNN. Drugi zestaw opiera się na sieciach bliźniaczych Siamese. Na podstawie cech wyznaczonych przez sieci bliźniacze Siamese wyznaczane są cechy DTW lub cechy Shapelets [154].

3. Klasyfikator wieloklasowy - jako klasyfikatora wieloklasowego użyto regresji logistycznej (zob. rysunek 6.1)
4. Agregacja wyników klasyfikatorów - zbadano kilka różnych metod agregacji wyników klasyfikatorów, zarówno podstawowe głosowanie miękkie, jak i głosowanie ważone.

Podobnej wersji algorytmu NECSCF użyto w pracy [155]. W tym algorytmie nie użyto cech opartych na sieciach Siamese, lecz na cechach tworzonych ręcznie.



Rysunek 6.1: NECSCF oparty na cechach wyznaczonych przez sieci bliźniacze, cechach wyznaczonych przez CAE (zob. 6.1.2) oraz 1D-CNN, które są następnie łączone z cechami specyficznymi dla klasy, które są wyznaczane przez sieci TimeDistributed i LSTM (TD-LSTM, zob. podrozdział 6.1.1).

6.1. Cechy użyte w zespole NECSCF opartym na sieciach bliźniaczych Siamese

W niniejszym podrozdziale zaprezentowano rodzaje cech użytych w zespole NECSCF opartym na sieciach bliźniaczych Siamese. Podstawowe informacje o wszystkich rodzajach cech użytych w zespole NECSCF opartym na sieciach bliźniaczych Siamese zawiera tabela 6.1.

6.1.1. Cechy wyznaczone przez sieci TD-LSTM

W rozdziale 5 w zaproponowano wersję algorytmu NECSCF w którym cechy specyficzne dla klasy były określane w dwóch etapach. Najpierw trenowane są sieci konwolucyjne,

Tabela 6.1: Kolumna 'Nazwa' zawiera nazwę poszczególnych rodzajów cech. Kolumna 'Typ' zawiera informacje w jaki sposób cechy wykorzystywane są w zespole NECSCF. Kolumna 'Rodzaj Danych' opisuje na jakim rodzaju danych wyznaczane są cechy. Kolumna 'Opis' zawiera podrozdział w którym opisano dane cechy.

Nazwa	Typ	Rodzaj Danych	Opis
TD-LSTM	Specyficzne dla klasy	Sekwencja map głębi	6.1.1
Siamese	Wspólne	Mapa głębi	6.1.3
Autoenkoder	Wspólne	Mapa głębi	6.1.2
1D-CNN	Wspólne	Szereg czasowy	5.1.3
Shapelets	Wspólne	Szereg czasowy	6.1.4
DTW	Wspólne	Szereg czasowy	3.3

które wyznaczają wektor cech na pojedynczych mapach głębi, a następnie na uzyskanych tym sposobem wielowymiarowych szeregach czasowych wyznaczane są cechy sekwencji (za pomocą różnych metod). W niniejszym rozdziale przedmiotem badań jest wersja algorytmu NECSCF, w której cechy specyficzne dla klasy są wyznaczane przez sieci neuronowe trenowane end-to-end. Z uwagi na powyższe, jako architektury sieci neuronowej używamy sieci TD-LSTM.

Sieć neuronowa TD-LSTM operuje na sekwencjach map głębi, każda sekwencja składa się z 30 map głębi o wymiarach 64×64. Wspomniane mapy głębi są wycinane z wejściowych map głębi w ten sposób, że zawierają one piksele należące do osoby, które następnie skalowane są do rozmiarów 64 x 64. Mapy głębi wybrano poprzez losowanie map głębi z sekwencji map głębi ze zwracaniem. W pierwszych trzech warstwach użyto wrappera TimeDistributed [156], aby zastosować tę samą warstwę Conv2D niezależnie do każdego z 30 kroków czasowych. Pierwsza warstwa TimeDistributed zawiera 32 filtry konwolucyjne o rozmiarze 5×5, z paddingiem ustawionym na 'same'. Druga warstwa TimeDistributed zawiera 32 filtry konwolucyjne o rozmiarze 5×5. Trzecia warstwa TimeDistributed zawiera warstwę max pooling o rozmiarze okna 4×4. Następnie, warstwa TimeDistributed opakowuje warstwę spłaszczającą. Kolejno, dwie warstwy TimeDistributed opakowują warstwy gęste odpowiednio z 256 i 128 neuronami. Na tym etapie, rozmiar mapy cech wynosi (None, 30, 128). Na koniec, wykorzystano warstwę LSTM o rozmiarze 64, a następnie 64 globalne filtry uśredniające. Otrzymane cechy nazywane są TD-LSTM. Sieci neuronowe zostały trenowane przy użyciu algorytmu optymalizacji adadelta, wartość współczynnika uczenia lr była równa 0,0001. Jako funkcję straty użyto kategorycznej entropii krzyżowej (ang. *categorical cross-entropy*), a modele były trenowane w podejściu jeden-przeciw-wszystkim.

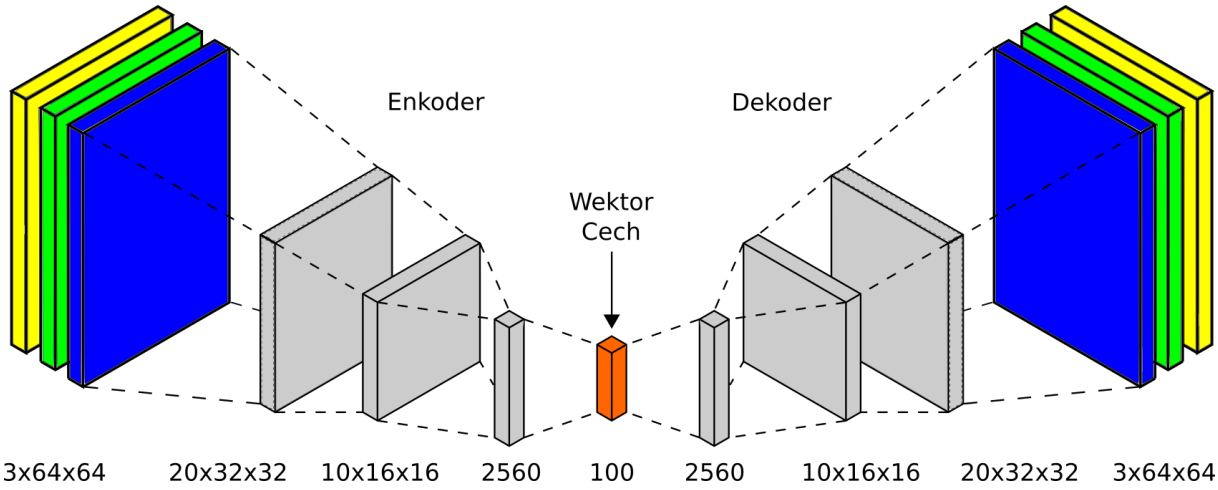
6.1.2. Cechy wyznaczone przez konwolucyjny autoenkoder

W wersji algorytmu NECSCF zaproponowanej w tym rozdziale wykorzystano cechy oparte na autoenkoderze. Są one używane jako jeden z rodzajów cech wspólnych. Użyto ich aby pokazać, że cechy wspólne mogą zostać wyznaczone przez modele nienadzorowane.

Autoenkoder jest rodzajem sieci neuronowej, która przekształca wysokowymiarowe wejście na reprezentacje o niższej wymiarowości (enkoder) i następnie odtwarza wejście przy użyciu takiej reprezentacji (dekoder) [153]. Aby to osiągnąć, autoenkoder uczy się znajdować reprezentacje dla zestawu danych wejściowych, zwykle przez kompresję (redukcję wymiarowości), uczy się ignorować mniej istotne dane. Oznacza to, że autoenkoder stara się wyznaczać ze zredukowanej w ten sposób reprezentacji danych wyjście, które jest jak najbardziej zbliżone do swojego wejścia. Gdy ukryta reprezentacja używa mniejszej liczby wymiarów niż wejście, enkoder przeprowadza redukcję wymiarowości. Autoenkoder składa się z wewnętrznej (ukrytej) warstwy, która przechowuje skompresowaną reprezentację wejścia, a także enkodera, który mapuje wejście na reprezentacje, a także dekodera, który mapuje reprezentacje na odtworzenie oryginalnego wejścia. Enkoder kompresuje wejście i generuje reprezentacje podczas, gdy dekoder odtwarza wejście w oparciu o wyznaczone tym sposobem małowymiarowe reprezentacje danych wejściowych. Uczenie replikacji wejścia na wyjściu jest osiągane dzięki wytrenowaniu podsieci odpowiedzialnych za redukcje i rekonstrukcje. Autoenkodery są uważane za technikę uczenia nienadzorowanego, ponieważ nie wymagają etykietowanych danych do uczenia sieci. Po wytrenowaniu takiej reprezentacji o zmniejszonej wymiarowości można ją wykorzystać jako dane wejściowe do algorytmu nadzorowanego

Ponieważ większość obecnie dostępnych zbiorów danych do rozpoznawania akcji nie zawiera wystarczającej liczby sekwencji, aby wytrenować głębokie modele o odpowiednich zdolnościach generalizacji, proponujemy wykorzystanie autenkodera konwolucyjnego działającego na pojedynczych mapach głębi w celu wyznaczania dyskryminatywnych cech map głębi. Duża liczba map głębi w obecnych zbiorach danych do rozpoznawania akcji umożliwia uczenie głębokich sieci neuronowych. Dla danej wejściowej sekwencji map głębi $dms = fr_1, fr_2, \ldots, fr_n$, wyznaczamy małowymiarową reprezentację każdej mapy głębi fr_i przy użyciu głębokiej sieci CAE (Convolutional Autoencoder). Każda mapa głębi jest reprezentowana jako wektor cech $ae(fr_i)$, co daje sekwencję wektorów cech $ae(dms) = ae(fr_1), ae(fr_2), ..., ae(fr_{|dms|})$. Wymiar takiej reprezentacji dla sekwencji map głębi wynosi $|dms| \times |ae(fr_i)|$, gdzie $|ae(fr_i)|$ i oznacza liczbę neuronów wewnętrznej warstwy autoenkodera.

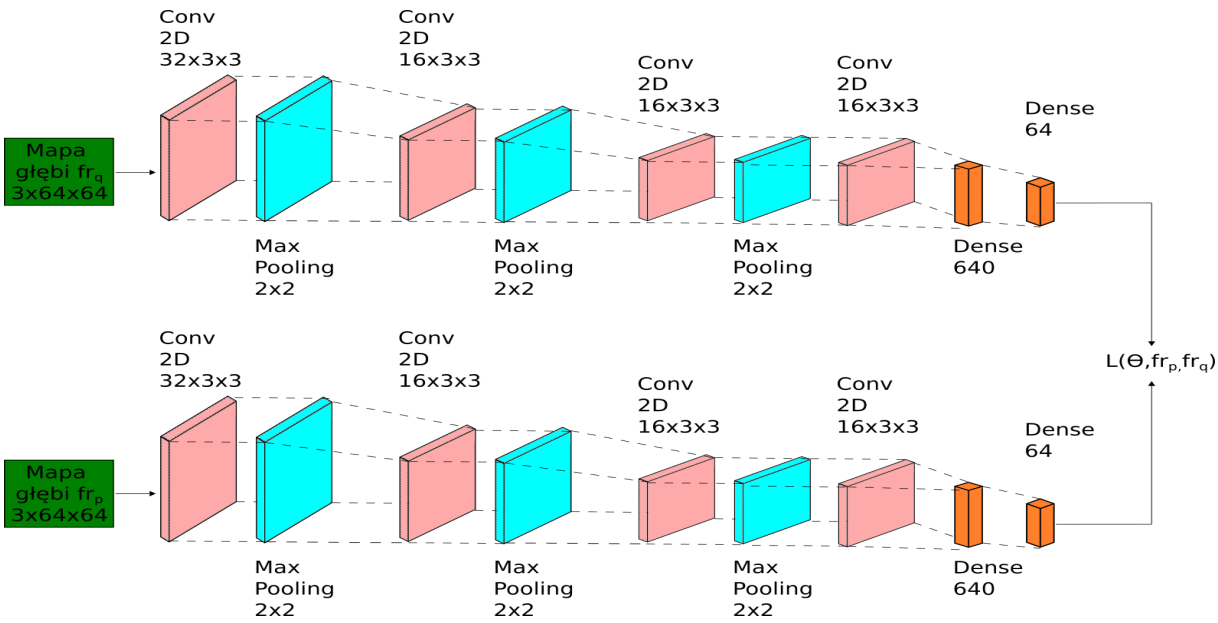
Autoenkoder został wytrenowany na wszystkich mapach głębi wchodzących w skład zbioru treningowego. Konwolucyjny autoenkoder został wytrenowany na mapach głębi o rozmiarze $3 \times 64 \times 64$. Architektura sieci CAE jest przedstawiona na rys. 6.2. Sieć składa się z dwóch warstw kodujących i dwóch odpowiadających im warstw dekodujących. Rozmiar wektora cech pojedynczej mapy głębi wyznaczonego przez konwolucyjny autoenkoder wynosi 100.



Rysunek 6.2: Architektura konwolucyjnego autoenkodera.

6.1.3. Wektor cech specyficznych dla klasy wyznaczony przez sieci bliźniacze Siamese

W rozdziałach 3 i 5 badano cechy DTW wyznaczone na podstawie cech tworzonych ręcznie. Na podstawie pojedynczych map głębi wyznaczano cechy tworzone ręcznie, a następnie na podstawie wielowymiarowego szeregu czasowego wyznaczano cechy DTW. W niniejszym podrozdziale przedmiotem badań jest alternatywna metoda wyznaczania cech na podstawie pojedynczych map głębi oparta na sieciach neuronowych. W poprzednim rozdziale użyto sieci Siamese do wyznaczania cech specyficznych dla klasy na podstawie szeregów czasowych. W niniejszym rozdziale sieć Siamese jest używana do wyznaczania cech wspólnych na podstawie pojedynczych map głębi [135, 109].



Rysunek 6.3: Diagram architektury sieci bliźniaczej Siamese.

Na rysunku 6.3 przedstawiono diagram wykorzystanej sieci bliźniaczej Siamese. W sieci

zastosowano konwolucję 2D dla pojedynczej mapy głębi. Sieć neuronowa składa się z czterech warstw konwolucyjnych. Pierwsza warstwa zawiera 32 filtry 3×3 i warstwę max pooling o rozmiarze 2×2. Druga warstwa zawiera 16 filtrów 3×3 i warstwę max pooling o rozmiarze 2×2. Trzecia warstwa zawiera 16 filtrów 3×3 i warstwę max pooling o rozmiarze 2×2. Czwarta warstwa zawiera 16 filtrów 3×3. Warstwy konwolucyjne są następnie połączone z warstwą spłaszczającą, a następnie z dwiema warstwami gęstymi o rozmiarze odpowiednio 640 i 64 neuronów. Sieć bliźniacza Siamese minimalizuje funkcję straty L, która jest podobna do funkcji opisanej w równaniu 5.1 jest jednak wyznaczana na pojedynczych mapach głębi, a nie na szeregach czasowych:

$$L(\theta, fr_p, fr_q) = y_{pq} dist_\theta(fr_p, fr_q)^2 - (1 - y_{pq}) \{max(0, M - dist_\theta(fr_p, fr_q))\}^2 \quad (6.1)$$

$$dist_\theta(fr_p, fr_q) = |frame\_sim_\theta(fr_p) - frame\_sim_\theta(fr_q)|^2 \quad (6.2)$$

gdzie:

fr_p, fr_q - pojedyncze mapy głębi

θ - parametry sieci bliźniaczej Siamese

$frame\_sim_\theta(fr)$ - reprezentacja wyznaczona przez sieć bliźniaczą Siamese

y_{pq} - równe 1 jeśli próbki p i q należą do tej samej klasy i 0, gdy klasy są różne

M - margines (liczba rzeczywista)

Sieć wytrenowano w oparciu o algorytm optymalizacji Adam z współczynnikiem uczenia równym 0,00006.

Do trenowania użyto wszystkich par sekwencji w wykorzystywanym zbiorze treningowym. W następnym etapie z każdej pary sekwencji map głębi losowo wybrano 3 pary map głębi. W każdej parze są mapy głębi z różnych sekwencji. Wylosowane pary pojedynczych map głębi są dodawane do zestawu treningowego. Rozkład, z którego losowane są mapy głębi, nie jest ani jednostajny, ani normalny. Zamiast tego prawdopodobieństwo wylosowania mapy głębi jest proporcjonalne do kwadratu odległości od bliższego końca sekwencji map głębi, co przedstawiono na rys. 6.4. Zastosowano niejednostajny rozkład, ponieważ najbardziej dyskryminacyjne mapy głębi mają tendencję do znajdowania się w środku sekwencji (sekwencje zazwyczaj zaczynają się i kończą z osobą stojącą w bezruchu).

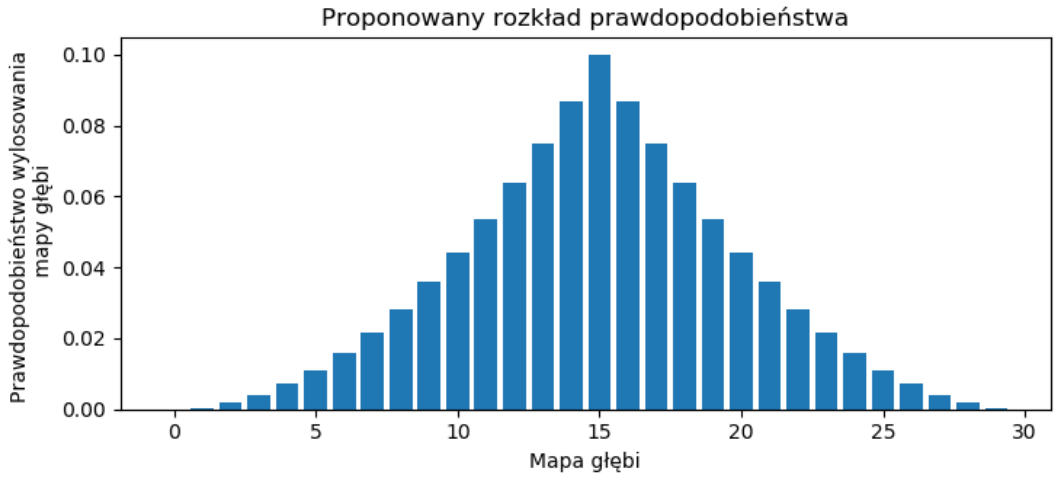
Dla sekwencji map głębokości $dms = fr_1 \dots fr_{|dms|}$ prawdopodobieństwo losowego wyboru mapy głębi fr_i jest równe:

$$p(fr_i) = NC \times min(d(fr_0, fr_i), d(fr_i, fr_{|dms|}))^2 \quad (6.3)$$

gdzie:

$$d(fr_i, fr_j) = |i - j| \quad (6.4)$$

NC to stała normalizacji.



Rysunek 6.4: Proponowany rozkład prawdopodobieństwa wylosowania map głębi dla sekwencji map głębi składającej się z 30 elementów.

$$NC = \sum_{i=0}^{N} min(d(fr_0, fr_i)), d(fr_i, fr_{|dsm|}))^2 \tag{6.5}$$

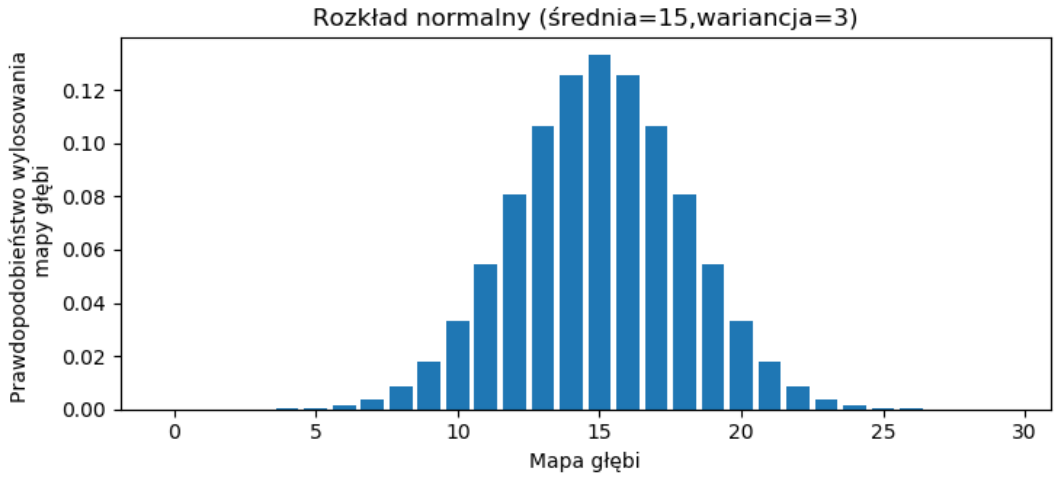
Pożądane jest, aby rozkład przyporządkowywał największe prawdopodobieństwo map głębi znajdującym się w środku, co oznacza, że rozkład powinien być unimodalny. Rozkład normalny wydaje się naturalnym wyborem, lecz wyniki próbkowania z tego rozkładu są bardzo wrażliwe na parametr sigma (odchylenie standardowe). Jeśli parametr sigma jest zbyt mały, zbyt wiele map głębi ma praktycznie zerowe prawdopodobieństwo wylosowania, co pokazano na rys. 6.5. Jeśli parametr jest zbyt duży, różnica między prawdopodobieństwami wylosowania poszczególnych map głębi jest zbyt mała, co zaprezentowano na rys. 6.6. Próbkowanie map głębi oparto o algorytm multinomialnego próbkowania [157], który jest zaimplementowany w bibliotece numpy [158].

6.1.4. Cechy wyznaczone przez algorytm Shapelets

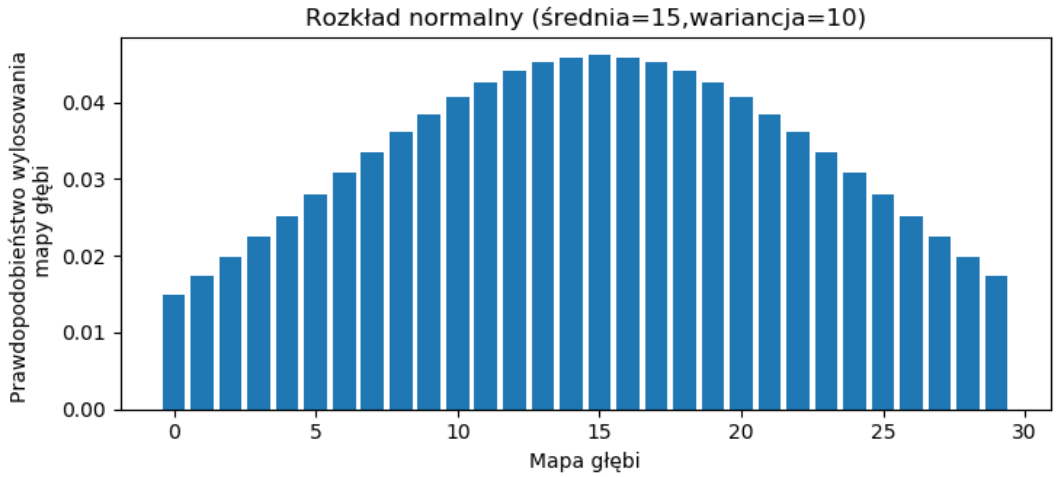
W algorytmie zaprezentowanym w rozdziale 5 jako cechy wspólne wykorzystano cechy DTW. Zespoły NECSCF operujące na tych cechach osiągnęły obiecujące rezultaty, lecz ich wadą jest wysoki koszt obliczeniowy. W niniejszym podrozdziale przebadano alternatywną metodę wyznaczania cech reprezentujących szeregi czasowe. Metody oparte na algorytmie Shapelets osiągnęły konkurencyjne wyniki klasyfikacji na powszechnie stosowanych zbiorach do ewaluacji algorytmow rozpoznawania akcji [159]. Dlatego w pracy [109] zostały one wybrane do modelowania działań człowieka jako alternatywa dla DTW.

W analizie algorytmu Shapelets ważne jest rozróżnienie następujących pojęć:

1. Podciąg Shapelet - podciąg szeregu czasowego wybrany tak aby jego występowanie umożliwiało dyskryminacje poszczególnych klas szeregów czasowych. Shapelet s jest



Rysunek 6.5: Rozkład normalny (o średniej równej 15 i wariancji równej 3) prawdopodobieństwa wylosowania map głębi dla sekwencji map głębi składającej się z 30 elementów.



Rysunek 6.6: Rozkład normalny (o średniej równej 15 i wariancji równej 10) prawdopodobieństwa wylosowania map głębi dla sekwencji map głębi składającej się z 30 elementów.

dyskryminujący, jeśli występuje on w większości szeregów czasowych jednej klasy oraz jest nieobecny w szeregach czasowych innych klas [154]. Wybór podciągu odbywa się na podstawie kryterium maksymalizacji przyrostu informacji (ang. *information gain*) [154]. Idea przyrostu informacji jest również używana w innych algorytmach jak na przykład tworzeniu drzew decyzyjnych. Podciąg Shapelet może rozpoczynać się w różnych momentach czasu w różnych szeregach czasowych (jest niezależny od fazy szeregu czasowego). Dlatego podciągi Shapelet są zdolne do odkrywania reprezentatywnych wzorców niezależnie od momentu w czasie, w którym występują.

2. Cechy Shapelet - szereg czasowy jest reprezentowany przez wektor cech wyznaczonych

na podstawie podciągu Shapelet. Każda z cech to odległość wektora od jednego z podciągów Shapelet. Odległość jest obliczana na podstawie zależności:

$$d(l, s) = min_t \left|\left|l_{t \to t+|s|} - s\right|\right|_2 \tag{6.6}$$

gdzie:

s - podciąg Shapelet

l - szereg czasowy

$|s|$ - długość podciągu Shapelet s

$l_{t \to t+|s|}$ - podciąg szeregu czasowego l

Wyznaczanie odległości między szeregiem czasowy a podciągiem Shaplet jest zilustrowane na rysunkach 6.7 i 6.8.

3. Algorytm Shapelets - algorytm wyznaczający podciągi Shapelets na podstawie zbioru szeregów czasowych wraz z przypisanymi im klasami.

Złożoność obliczeniowa algorytmu wyznaczania podciągu Shapelet wynosi $O(|s||l|)$, gdzie $|l|$ to długość szeregu czasowego, zaś $|s|$ to długość podciągu Shapelet. Wykorzystano implementację [160], aby wyznaczyć szeregi Shapelet, zgodnie z podejsciem zaproponowanym w pracy [161]. Liczba cech opartych o algorytm Shapelets (liczba wyznaczanych pociągów Shapelet) w pracach eksperymentalnych zrealizowanych w pracach prezentowanych w niniejszym rozdziale wynosi 14.

Rysunek 6.7: Przykładowy jednowymiarowy szereg czasowy i podciąg Shapelet. Podciąg Shapelet rozpoczyna się w punkcie 5 i kończy się w punkcie 14.

Rysunek 6.8: Wykres odległości między podciągiem Shapelet pokazanym na rysunku 6.7 a poszczególnymi podciągami szeregu czasowego (również pokazanego na rysunku 6.7). Odległość jest obliczana między stałym podciągiem Shaplet, a podciągiem rozpoczynającym się w danym punkcie czasu.

6.2. Metody agregacji wyników klasyfikatorów

Uczenie zespołowe wymaga metody agregacji predykcji z poszczególnych klasyfikatorów. Najprostszymi metodami są głosowanie twarde (wybieranie klasy wybranej przez większość klasyfikatorów) i głosowanie miękkie (dodawanie rozkładów prawdopodobieństwa wyznaczonych przez klasyfikatory i wybieranie klasy o najwyższym prawdopodobieństwie) [162].

W niniejszym rozdziale przedmiotem zainteresowania było głosowanie ważone. W głosowaniu ważonym każdemu klasyfikatorowi przyporządkowana jest waga. Rozkłady prawdopodobieństwa wyznaczone przez poszczególne klasyfikatory są mnożone przez wagi tych klasyfikatorów, a następnie dodawane. Ostatecznie wybierana jest klasa, dla której suma prawdopodobieństw przynależności do tej klasy jest największa. Aby znaleźć wagi do głosowania ważonego, wymagana jest funkcja straty i algorytm optymalizacji. Wszystkie wagi badane w niniejszym rozdziale są optymalizowane przy użyciu algorytmu Differential Evolution (DE) [163]. Użyto dwóch funkcji strat:

1. W pierwszej funkcji strat optymalizujemy wagi ze względu na dokładność klasyfikacji wyznaczonej na danych ze zbioru treningowego. W każdym wywołaniu optymalizowanej funkcji przeprowadzamy głosowanie ważone (z optymalizowanymi wagami) na zbiorze klasyfikatorów, a następnie obliczamy dokładność klasyfikacji na zbiorze walidującym.

2. Drugą badaną funkcją jest GASEN [110]. Algorytm ten przyporządkowuje największe wagi klasyfikatorom nieskorelowanym z pozostałymi klasyfikatorami. Metoda ta może również zostać użyta do selekcji klasyfikatorów. Minimalizowana jest funkcja kosztu,

która zależy od wag:

$$w = argmin \sum_{i=1}^{C} \sum_{j=1}^{C} w_i w_j G_{ij} \tag{6.7}$$

$$G_{ij}^{train} = \frac{\sum_{v \in train}(f_i(v) - y(v))(f_j(v) - y(v))}{|train|} \tag{6.8}$$

które powinny spełniać:

$$0 < w_i < 1 \tag{6.9}$$

$$\sum_{i=1}^{C} w_i = 1 \tag{6.10}$$

gdzie:

w_i - wagi klasyfikatora i.

v - próbka (wektor cech).

C - liczba klas.

$f_i(v)$ - klasyfikator w zespole.

$y(v)$ - prawdziwa etykieta próbki v.

Klasyfikatory, których wagi są poniżej pewnego progu, są odrzucane. Umożliwia to selekcję klasyfikatorów i obniżenie kosztów obliczeniowych klasyfikacji przez zespół klasyfikatorów.

6.3. Wyniki prac eksperymentalnych

Ewaluacje wariantów algorytmu NECSCF zrealizowano na zbiorze danych 3DHOI [94]. Ewaluacje na zbiorze 3DHOI przeprowadzono dla dwóch protokołów ewaluacji: setting 1, setting 2. Dokładny opis tych protokołów znajduje się w podrozdziale 3.4.3.

Tabela 6.2 zawiera wyniki uzyskane dla setting-1, zaś tabela 6.3 przedstawia wyniki dla setting-2. Przebadano cztery metody głosowania (agregacji wyników klasyfikatorów) oraz dwa różne zestawy cech wspólnych. W sumie przebadano osiem wariantów dla każdego podziału zbioru danych. Użyte metody agregacji wyników klasyfikatorów są opisane w tabeli 6.4, zaś w tabeli 6.5 opisano użyte typy cech.

Użyto dwóch typów cech wyznaczonych przez DTW (oznaczane jako DTW(SIM)) oraz wyznaczone przez Shapelets (oznaczane jako SHAPE(SIM)). Cztery metody głosowania to głosowanie miękkie (oznaczane jako SOFT), głosowanie twarde (oznaczane jako HARD) oraz dwa warianty głosowania ważonego: z wagami optymalizującymi accuracy (oznaczane jako WEIGHTED(ACCURACY)) i wagami optymalizującymi funkcję GASEN (oznaczane jako

Tabela 6.2: Wyniki na zbiorze danych SYSU 3DHOI (setting I, zob. 3.4.3). Kolumna 'Głosowanie' zawiera metody agregacji klasyfikatórów. Kolumna 'N_clf' podaje liczbę klasyfikatorów w zespole. Użyte oznaczenia wyjaśniono w tabeli 6.4.

Głosowanie	Wspólne	Specyficzne dla klasy	N_clf	Accuracy	Precision	Recall	F1-score
HARD	SHAPE(SIM) + 1D-CNN(CAE)	TD-LSTM	12	0,9167	0,9217	0,9167	0,9171
SOFT	SHAPE(SIM) + 1D-CNN(CAE)	TD-LSTM	12	0,9079	0,9102	0,9079	0,9071
WEIGHTED (ACCURACY)	SHAPE(SIM) + 1D-CNN(CAE)	TD-LSTM	12	0,9079	0,9110	0,9079	0,9073
WEIGHTED (GASEN)	SHAPE(SIM) + 1D-CNN(CAE)	TD-LSTM	6	**0,9254**	0,9271	0,9254	0,9246
HARD	DTW(SIM) + 1D-CNN(CAE)	TD-LSTM	12	**0,9341**	0,9298	0,9341	0,9298
SOFT	DTW(SIM) + 1D-CNN(CAE)	TD-LSTM	12	**0,9341**	0,9298	0,9341	0,9298
WEIGHTED (ACCURACY)	DTW(SIM) + 1D-CNN(CAE)	TD-LSTM	11	0,9266	0,9211	0,9266	0,9211
WEIGHTED (GASEN)	DTW(SIM) + 1D-CNN(CAE)	TD-LSTM	7	**0,9340**	0,9298	0,9340	0,9298

WEIGHTED(GASEN)). Obydwa typy cech są wyznaczane na podstawie cech Siamese. Dodatkowo w każdym przebadanym przypadku używamy jako cech wspólnych cech wyznaczonych przez konwolucyjny autoenkoder oraz sieć 1D-CNN. Cechy specyficzne dla klasy są zawsze wyznaczane przez TD-LSTM. Jako klasyfikatora wieloklasowego zawsze używamy Regresji Logistycznej.

W tabeli 6.2 zaprezentowano wyniki uzyskane przez proponowane algorytmy dla protokołu ewaluacji setting 1. Jak można zaobserwować algorytmy operujące na cechach opartych na DTW osiągają lepsze wyniki dla każdej metody głosowania, przy czym najmniejsza różnica między dokładnością klasyfikacji osiąganą przez algorytmy operujące na cechach DTW i algorytmy operujące na cechach Shapelets jest osiągana przez metodę WEIGHTED(GASEN). Metoda agregacji wyników klasyfikatorów WEIGHTED(GASEN) osiąga najlepszą dokładność klasyfikacji zarówno na cechach opartych na DTW jak i na cechach opartych na Shapelets. W przypadku cech opartych na DTW metody SOFT i HARD osiągają praktycznie identyczne wyniki, jednak metoda WEIGHTED(GASEN) wykorzystuje mniejszą liczbę klasyfikatorów. WEIGHTED(GASEN) umożliwia znaczną redukcję liczby klasyfikatorów zarówno dla cech opartych na Shapelets jak i cech opartych na DTW (z 12 do 6 w przypadku cech opartych na

Tabela 6.3: Wyniki na zbiorze danych SYSU 3DHOI (setting II, zob. sekcja 3.4.3). Kolumna 'Voting' zawiera metody agregacji klasyfikatórów. Kolumna 'N_clf' podaje liczbę klasyfikatorów w zespole. Użyte oznaczenia są wyjaśnione w Tabeli 6.4.

Głosowanie	Wspólne	Specyficzne dla klasy	N_clf	Accuracy	Precision	Recall	F1-score
HARD	SHAPE(SIM) + 1D-CNN(CAE)	TD-LSTM	12	0,8991	0,9079	0,8991	0,8990
SOFT	SHAPE(SIM) + 1D-CNN(CAE)	TD-LSTM	12	0,9035	0,9098	0,9035	0,9036
WEIGHTED (ACCURACY)	SHAPE(SIM) + 1D-CNN(CAE)	TD-LSTM	12	0,9123	0,9175	0,9123	0,9119
WEIGHTED (GASEN)	SHAPE(SIM) + 1D-CNN(CAE)	TD-LSTM	6	**0,9211**	0,9259	0,9211	0,9209
HARD	DTW(SIM) + 1D-CNN(CAE)	TD-LSTM	12	0,9230	0,9211	0,9230	0,9211
SOFT	DTW(SIM) + 1D-CNN(CAE)	TD-LSTM	12	0,9230	0,9211	0,9230	0,9211
WEIGHTED (ACCURACY)	DTW(SIM) + 1D-CNN(CAE)	TD-LSTM	11	0,9204	0,9167	0,9204	0,9167
WEIGHTED (GASEN)	DTW(SIM) + 1D-CNN(CAE)	TD-LSTM	3	**0,9291**	0,9254	0,9291	0,9254

Shapelets i z 12 do 7 w przypadku cech opartych na DTW).

W tabeli 6.3 znajdują się wyniki uzyskane dla setting 2. Jak można zaobserwować również w tym przypadku algorytm NECSCF operujący na cechach opartych na DTW osiąga lepsze wyniki dla każdej metody głosowania. Metoda agregacji predykcji klasyfikatorów WEIGHTED(GASEN) osiąga najlepszą dokładność klasyfikacji zarówno wraz z cechami opartymi na DTW jak i z cechami opartymi na Shapelets. Ponadto WEIGHTED(GASEN) umożliwia znaczną redukcję liczby klasyfikatorów (z 12 do 6 w przypadku cech opartych na Shapelets i z 12 do 7 w przypadku cech opartych na DTW).

Podsumowując WEIGHTED(GASEN) osiąga najlepsze wyniki na wszystkich czterech permutacjach podziału zbiorów i zestawach cech wspólnych. Metoda WEIGHTED(GASEN) umożliwia znaczną redukcję liczby używanych klasyfikatorów. Celem metody GASEN jest wyszukanie nieskorelowanych klasyfikatorów. Wspomniana metoda umożliwia uzyskanie znacznie lepszych wyników niż podstawowe metody głosowania takie jak głosowanie twarde i miękkie. Metoda GASEN osiąga również lepsze wyniki niż głosowanie ważone, gdzie wagi optymalizują proste funkcje celu takie jak dokładność klasyfikacji.

We wszystkich wersjach zespołu klasyfikatorów algorytmy operujące na cechach opartych

Tabela 6.4: Tabela przedstawia użyte metody agregacja wyników klasyfikatorów. Kolumna 'Symbol' zawiera identyfikator za pomocą którego metoda agregacji jest oznaczona w innych tabelach. W przypadku głosowania ważonego podajemy funkcje celu, w oparciu o które to optymalizowano wagi.

Symbol	Funkcja celu	Opis
HARD	-	Hard voting
SOFT	-	Soft voting
WEIGHTED(ACCURACY)	Accuracy	Głosowanie ważone
WEIGHTED(GASEN)	GASEN	Głosowanie ważone

Tabela 6.5: Rodzaje zestawów cech używanych w zespole opartym na TD-LSTM. Niektóre cechy są wyznaczane bezpośrednio na podstawie sekwencji mapy głębi (jak TD-LSTM). Inne są wyznaczane w dwóch etapach. Najpierw wyznaczane są cechy opisujące poszczególne mapy głębi. Na podstawie każdej sekwencji mapy głębi wyznaczany jest wielowymiarowy szereg czasowy cech map głębi wyznaczonych w pierwszym etapie. Na takim szeregu czasowym określane są ostateczne cechy opisujące całą sekwencję map głębi.

Symbol	Typ	Mapa głębi	Szereg czasowy
DTW(SIM)	Wspólny	Cechy Siamese (6.1.3)	Cechy DTW (3.3)
SHAPELETS(SIM)	Wspólny	Cechy Siamese (6.1.3)	Cechy Shapelets (6.1.4)
1D-CNN(CAE)	Wspólny	Cechy autoenkodera (6.1.2)	Cechy 1D CNN (5.1.3)
TD-LSTM	Specyficzne dla klasy	Cechy TD-LSTM (6.1.1)	-

na DTW osiągają lepsze wyniki niż algorytmy operujące na cechach opartych na Shapelets. Możliwe jest jednak zmniejszenie różnicy w dokładności klasyfikacji między algorytmami używającymi cech opartych na DTW, a algorytmami używającymi cech opartych na Shapelets dzięki zastosowaniu odpowiedniej metody - głosowania ważonego z wagami wyznaczonymi przez GASEN. Cechy oparte na Shapelets wciąż mogą być użyteczne z uwagi na mniejszą złożoność obliczeniową w porównaniu do cech opartych na DTW.

W tabeli 6.6 zaprezentowano wyniki uzyskane przez zespół NECSCF oparty na sieciach bliźniaczych na zbiorze MHAD. Zamieszczono również wyniki uzyskane na zbiorze MHAD z poprzedniego rozdziału. Jak można zaobserwować, wersja algorytmu NECSCF nie wykorzystująca cech tworzonych ręcznie uzyskuje lepsze wyniki pomimo znacznego uproszczenia algorytmu.

6.4. Podsumowanie

Tabela 6.7 przedstawia wyniki uzyskane przez konkurencyjne algorytmy na zbiorze danych 3DHOI w porównaniu do wyników osiągniętych przez proponowany algorytm. Pokazano eksperymentalnie, że zaproponowany algorytm uzyskuje znacznie lepsze wyniki na wymaga-

Tabela 6.6: Wyniki na zbiorze danych MHAD (zob. sekcja 3.4.2).

Specyficzne dla klasy	Głosowanie	Wspólne	Accuracy	Precision	Recall	F1-score
CNN	DTW (HANDCRAFTED)	SOFT	0,8814	0,8979	0,8814	0,8804
TD-LSTM	DTW(SIM) + 1D-CNN(CAE)	SOFT	0,8814	0,8844	0,8814	0,8747
TD-LSTM	DTW(SIM) + 1D-CNN(CAE)	**WEIGHTED (ACCURACY)**	**0,8907**	0,8919	0,8907	0,8833

jącym zbiorze danych SYSU 3DHOI niż najnowsze zaawansowane algorytmy oparte na mapach głębi [164, 31, 165, 166]. Warto podkreślić, że proponowany algorytm uzyskuje lepsze wyniki od większości najnowszych metod opartych na szkielecie, które zazwyczaj osiągają lepsze wyniki w porównaniu do metod opartych wyłącznie na mapach głębi.

Tabela 6.7: Porównanie z wynikami uzyskiwanymi przez konkurencyjne algorytmy na zbiorze 3DHOI.

Metoda	Modalność	Protokół	Accuracy[%]
MSRNN [164]	depth+RGB+skel.	-	79,58
PTS [167]	depth+skeleton	-	87,92
bidirect. rank p. [168]	depth	I	76,25
LGN [165]	skel.	II	83,33
SGN [166]	skel.	II	86,90
LAFF [136]	depth+RGB	II	80,00
TD-LSTM (Shapelet)	depth	I	92,54
TD-LSTM (Shapelet)	depth	II	90,35
TD-LSTM (DTW)	depth	I	**93,41**
TD-LSTM (DTW)	depth	II	92,91

Zaproponowana wersja algorytmu NECSCF nie wykorzystuje cech tworzonych ręcznie. Cechy oparte na DTW oraz na Shapelets są wyznaczane na podstawie cech opartych na sieciach bliźniaczych Siamese. Warto podkreślić, że w rozdziale 3 algorytmy operujące na cechach opartych na DTW nie osiągnęły dobrych wyników na zbiorze SYSU 3DHOI. Połączenie cech opartych na DTW z modelami głębokimi typu Siamese pozwala na uzyskanie wysokiej dokładności na zbiorach przedstawiających interakcje człowieka z obiektami. Algorytmy operujące na cechach opartych na DTW (o wyższym koszcie obliczeniowym) wciąż uzyskują lepsze wyniki, niż algorytmy operujące na cechach opartych na Shapelets ale głosowanie ważone oparte na GASEN znacznie zmniejsza różnice. Cechy specyficzne dla klasy są wyznaczane przez modele trenowane end-to-end.

Użycie głosowania ważonego opartego na funkcji celu GASEN prowadzi nie tylko do

najlepszych wyników, ale też do redukcji liczby klasyfikatorów. Potwierdza to wnioski sformułowane w podrozdziale 4.3 o możliwości znaczącej poprawy wyników uzyskiwanych przez algorytm NECSCF poprzez odpowiednie dobranie metody agregacji predykcji klasyfikatorów.

7. Podsumowanie najważniejszych osiągnięć

W niniejszej rozprawie doktorskiej zaproponowano nowe metody rozpoznawania akcji człowieka. Wszystkie zaproponowane algorytmy zostały przebadane na ogólnie dostępnych zbiorach danych i osiągnęły na nich konkurencyjne wyniki. W badaniach skupiono się na rozpoznawaniu akcji człowieka na mapach głębi, ale praktycznie wszystkie metody, z niewielkimi modyfikacjami, można zastosować do klasyfikacji dowolnego rodzaju sekwencji obrazów. Większość algorytmów, z wyjątkiem tych opartych na szeregach czasowych (DTW, shapelets), można dostosować do rozpoznawania akcji na pojedynczych obrazach/mapach głębi.

7.1. Weryfikacja postawionych celów badawczych

Głównym celem niniejszej pracy (sformułowanym w rozdziale 1) było opracowanie nowych metod rozpoznawania akcji na małych zbiorach sekwencji map głębi. Główne wyzwanie dotyczyło klasyfikacji na ograniczonej liczbie przykładów. Zostało ono rozwiązane w rozdziale 4. dzięki zaproponowaniu nowej metody uczenia zespołowego - NECSCF. W eksperymentach na danych tabelarycznych wykazano, że NECSCF osiąga dobre wyniki na zbiorach z małymi liczbami przykładów. Aby zastosować NECSCF do klasyfikacji akcji ludzkich należy wyznaczyć zawczasu odpowiednie cechy wspólne oraz wytrenować sieci neuronowe odpowiedzialne za wyznaczenie cech specyficznych dla klas akcji.

Drugie wyzwanie dotyczyło zmiennej prędkości wykonywania akcji na różnych częściach sekwencji map głębi z akcjami ludzi. Wspomniany problem rozwiązano w rozdziale 3. poprzez zaproponowanie nowych cech szeregów czasowych opartych na DTW. Aby użyć cech opartych na DTW, należy wyodrębnić cechy reprezentujące poszczególne mapy głębi. W rozdziale 3. przebadano zespoły klasyfikatorów operujące na ręcznie tworzonych cechach map głębi. Przebadano zarówno algorytmy działające wyłącznie na cechach opartych na DTW jak i zespół NECSCF wykorzystujący cechy oparte na DTW jako cechy wspólne (zob. rozdział 5). Motywacją dla proponowanej metody augumentacji danych była umotywowana potrzeba wykorzystania zalet DTW w klasyfikacji akcji wykonywanych z różną prędkością bez kosztu obliczeniowego DTW.

Trzecim problemem jest dynamiczne tło i inne problemy związane z interakcją osoby z

otoczeniem (zob. rozdział 2.1). W rozdziale 3 cechy oparte na DTW, wyznaczone na ręcznie tworzonych cechach map głębi, uzyskały słabsze wyniki na zbiorze 3DHOI. Jest to przypadek, gdzie akcje są wykonywane na bardziej złożonym tle i obejmują interakcje z obiektami.

W rozdziale 6. użyto cech wyznaczonych przez sieci Siamese jako alternatywy dla cech tworzonych ręcznie. Utworzony zespół NECSCF osiąga lepsze wyniki na zbiorze MHAD niż algorytm używający cech tworzonych ręcznie oraz dobre wyniki na zbiorze 3DHOI.

7.2. Wkład własny i ograniczenia rozprawy doktorskiej

Głównym wkładem tej pracy jest nowa metoda uczenia zespołowego oparta na głębokich sieciach neuronowych - NECSCF. Zaproponowano trzy wersje algorytmu NECSCF. Pierwsza wersja jest przeznaczona dla danych tabelarycznych. Opisano ją w rozdziale 4. Ewaluowano proponowany algorytm w sumie na 20 zbiorach danych i osiągnięto na nich konkurencyjne wyniki, a ponadto przeprowadzono porównanie z algorytmem RF na 14 zbiorach danych UCI. Uzyskano statystycznie istotną poprawę w stosunku do algorytmu RF dla trzech zbiorów danych.

Użycie algorytmu NECSCF do klasyfikacji akcji wymaga opracowania odpowiednich cech wspólnych. W rozdziale 3 opracowano kilka zestawów cech tworzonych ręcznie dla pojedynczych map głębi. Następnie zaproponowano nowy zestaw cech opartych na odległości DTW dla szeregów czasowych. Nowe cechy zostały przebadane na trzech ogólnodostępnych zbiorach danych (MSR, MHAD, 3DHOI). Cechy oparte na DTW osiągają lepsze wyniki niż tradycyjna metoda 1-NN DTW. Co więcej łączenie cech opartych na DTW wyznaczonych na różnych zestawach cech tworzonych ręcznie powoduje dalszą poprawę wyników, co pokazano eksperymentalnie w rozdziale 3.

Druga wersja algorytmu NECSCF jest przeznaczona dla klasyfikacji akcji ludzkich. Opisano ją w rozdziale 5. Wykorzystuje ona cechy oparte na DTW jako cechy wspólne. Opracowano również cechy specyficzne dla klasy. Cechy specyficzne dla klasy są wyznaczane w dwóch etapach. Najpierw celem wyznaczenia cech reprezentujących pojedyncze mapy głębi użyto sieci konwolucyjnych na mapach głębi. Następnie na wyznaczonych tym sposobach wielowymiarowych szeregach czasowych wyznaczane są cechy za pomocą jednej z trzech metod: sieci konwolucyjnej 1D, sieci Siamese 1D lub cech statystycznych. Algorytm osiągnął dobre wyniki na ogólnodostępnych zbiorach MHAD i MSR.

W rozdziale 4 zaproponowano również autorską metodę augmentacji danych. Proponowana metoda augumentacji została opracowana celem poprawienia dokładności klasyfikacji na sekwencjach z akcjami wykonywanymi z różną prędkością. W rezultacie uzyskano podobne rezultaty do DTW, ale przy mniejszych nakładach obliczeniowych. Wraz z nią przebadano bardziej konwencjonalny algorytm uczenia zespołowego - bagging. Metoda oparta o bagging uzyskała znacznie gorsze wyniki niż algorytm NECSCF.

Trzecia wersja algorytmu NECSCF została opisana w rozdziale 6. Jest ona również przeznaczona do klasyfikacji akcji ludzkich, lecz nie użyto w niej cech tworzonych ręcznie. Zastosowano również bardziej zaawansowane architektury sieci neuronowych. Cechy specyficzne dla klasy zostały wyznaczone przez sieci TD-LSTM trenowane w jednym etapie (end-to-end). Stosowane są dwa rodzaje cech wspólnych: oparte na uczeniu nienadzorowanym (konwolucyjnym autoenkoderze) i cechy oparte na DTW. Cechy DTW nie są wyznaczane na cechach tworzonych ręcznie, lecz na cechach wyznaczonych przez sieć Siamese. Dodatkowo przebadano cechy oparte na Shapelets jako alternatywy dla cech opartych na DTW. Zaproponowany algorytm uzyskał dobre wyniki na zbiorze 3DHOI. Cechy oparte na Shapelets osiągają nieco gorsze wyniki, jednak koszty obliczeniowe związane z ich wyznaczeniem są mniejsze niż cech opartych na DTW. Przebadano również różne wersje głosowania ważonego. Najlepsze wyniki uzyskały zespoły klasyfikatorów z wagami wyznaczonymi przez algorytm GASEN.

Podsumowując, proponowana rodzina algorytmów NECSCF osiągnęła konkurencyjne wyniki zarówno w przypadku danych tabelarycznych, jak i w rozpoznawaniu akcji. Proponowane algorytmy umożliwiają łączenie zalet różnych architektur uczenia głębokiego oraz pozwalają na wykorzystanie algorytmu DTW w połączeniu z sieciami neuronowymi.

Proponowane metody nie są jednak pozbawione ograniczeń. Algorytmy oparte na programowaniu dynamicznym, takie jak używane przy wyznaczaniu cech DTW, mają wysokie koszty obliczeniowe (kwadratowe względem długości szeregów czasowych) i nie są odpowiednie dla dużych zbiorów danych. Zespoły NECSCF mogą być stosowane na zbiorach danych o znacznej liczbie przykładów, ale w przypadku zbiorów danych z dużą liczbą klas, trenowanie dużej liczby modeli może być niepraktyczne.

7.3. Dalsze kierunki badań i rozwoju

Istnieje kilka możliwych kierunków dalszych badań związanych zagadnieniami podjętymi w niniejszej rozprawie. Wadą cech opartych na DTW jest ich koszt obliczeniowy. Jednym z możliwych kierunków byłoby zastosowanie algorytmu aproksymującego odległość DTW. Alternatywnym rozwiązaniem problemu wysokiego kosztu obliczeniowego odległości DTW jest wykorzystanie głębokich sieci neuronowych do przybliżenia odległości DTW.

Przebadano zespoły NECSCF z jednym zestawem cech wspólnych (czyli nieopartych na sieciach "jeden przeciw wszystkim"). Obiecującym podejściem byłoby zastosowanie kilku zestawów cech wspólnych w tym samym zespole. Interesujące byłoby użycie różnych algorytmów klasyfikacji w tym samym zespole (np. regresja logistyczna i SVM w tym samym zespole). Ponadto warto byłoby zbadać cechy specyficzne dla klasy wyznaczone przez inne architektury głębokich sieci neuronowych, zwłaszcza dedykowanych uczeniu na małych zbiorach danych (np. Matching Networks).

Możliwe jest udoskonalenie NECSCF dla danych tabelarycznych poprzez zastosowanie głosowania ważonego. Ostatecznie, warto byłoby dostosować algorytm NECSCF do innych typów danych audiowizualnych.

Literatura

[1] B. Liang and L. Zheng, "A Survey on Human Action Recognition Using Depth Sensors," in *Int. Conf. on Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–8, 2015.

[2] B. Ren, M. Liu, R. Ding, and H. Liu, "A Survey on 3D Skeleton-Based Action Recognition Using Learning Method," 2024.

[3] B. Kwolek, "Visual system for tracking and interpreting selected human actions," *Journal of WSCG*, vol. 11, no. 1, 2003.

[4] O. AlShorman, B. Alshorman, and M. S. Masadeh, "A review of physical human activity recognition chain using sensors," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 8, no. 3, pp. 560–573, 2020.

[5] F. Malawski and B. Kwolek, "Recognition of action dynamics in fencing using multimodal cues," *Image and Vision Computing*, vol. 75, pp. 1–10, 2018.

[6] M. Kepski and B. Kwolek, "Unobtrusive Fall Detection at Home Using Kinect Sensor," in *Computer Analysis of Images and Patterns*, pp. 457–464, 2013.

[7] L. Wang, D. Q. Huynh, and P. Koniusz, "A Comparative Review of Recent Kinect-Based Action Recognition Algorithms," *IEEE Transactions on Image Processing*, vol. 29, pp. 15–28, 2020.

[8] S. Majumder and N. Kehtarnavaz, "Vision and Inertial Sensing Fusion for Human Action Recognition : A Review," 2020.

[9] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine Recognition of Human Activities: A Survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1473–1488, 2008.

[10] X. Wang, A. Farhadi, and A. Gupta, "Actions ~ Transformations," in *CVPR*, 2016.

[11] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *Image and Vision Computing*, vol. 60, pp. 4–21, 2017.

[12] D. Weinland, M. Özuysal, and P. Fua, "Making Action Recognition Robust to Occlusions and Viewpoint Changes," in *Computer Vision – ECCV 2010*, pp. 635–648, 2010.

[13] Y. Kong and Y. Fu, "Human action recognition and prediction: A survey," *Int. Journal of Computer Vision*, vol. 130, no. 5, pp. 1366–1401, 2022.

[14] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal Multiplier Networks for Video Action Recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 7445–7454, 2017.

[15] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal Segment Networks: Towards Good Practices for Deep Action Recognition," in *Computer Vision – ECCV 2016*, pp. 20–36, 2016.

[16] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 28, no. 6, pp. 976–990, 2010.

[17] K. Schindler and L. van Gool, "Action snippets: How many frames does human action recognition require?," in *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[18] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2684–2701, 2020.

[19] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, 2012.

[20] L. Chen, H. Wei, and J. Ferryman, "A survey of human motion analysis using depth imagery," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 1995–2006, 2013.

[21] J. Wu, L. Sun, and R. Jafari, "A Wearable System for Recognizing American Sign Language in Real-Time Using IMU and Surface EMG Sensors," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 5, pp. 1281–1290, 2016.

[22] H. Wei and N. Kehtarnavaz, "Simultaneous Utilization of Inertial and Video Sensing for Action Detection and Recognition in Continuous Action Streams," *IEEE Sensors Journal*, vol. 20, no. 11, pp. 6055–6063, 2020.

[23] H. Wei, R. Jafari, and N. Kehtarnavaz, "Fusion of Video and Inertial Sensing for Deep Learning–Based Human Action Recognition," *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[24] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake, "Efficient Human Pose Estimation from

Single Depth Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2821–2840, 2013.

[25] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys*, vol. 43, no. 3, 2011.

[26] M. Ziaeefard and R. Bergevin, "Semantic human activity recognition: A literature review," *Pattern Recognition*, vol. 48, no. 8, pp. 2329–2345, 2015.

[27] F. Zhu, L. Shao, J. Xie, and Y. Fang, "From handcrafted to learned representations for human action recognition: A survey," *Image and Vision Computing*, vol. 55, pp. 42–52, 2016.

[28] M. A. R. Ahad, "Motion History Images for Action Recognition and Understanding," in *Springer Briefs in Computer Science*, 2012.

[29] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, 2001.

[30] G. Chéron, I. Laptev, and C. Schmid, "P-CNN: Pose-Based CNN Features for Action Recognition," in *IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 3218–3226, 2015.

[31] P. Wang, W. Li, C. Li, and Y. Hou, "Action recognition based on joint trajectory maps with convolutional neural networks," *Knowledge-Based Systems*, vol. 158, pp. 43–53, 2018.

[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," in *IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 4489–4497, 2015.

[33] L. Sun, K. Jia, K. Chen, D. Y. Yeung, B. E. Shi, and S. Savarese, "Lattice Long Short-Term Memory for Human Action Recognition," in *IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 2166–2175, 2017.

[34] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features," *IEEE Access*, vol. 6, pp. 1155–1166, 2018.

[35] J.-Y. He, X. Wu, Z.-Q. Cheng, Z. Yuan, and Y.-G. Jiang, "DB-LSTM: Densely-connected Bi-directional LSTM for human action recognition," *Neurocomputing*, vol. 444, pp. 319–331, 2021.

[36] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893, 2005.

[37] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions," in *Proc. of 12th Int. Conf. on Pattern Recognition*, vol. 1, pp. 582–585, 1994.

[38] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[39] T. K. Ho, "Random decision forests," in *Proc. of 3rd Int. Conf. on Document Analysis and Recognition*, vol. 1, pp. 278–282, 1995.

[40] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. of the Seventh IEEE Int. Conf. on Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[41] M. Younsi, S. Yesli, and M. Diaf, "Depth-based human action recognition using histogram of templates," *Multimedia Tools and Applications*, pp. 1–35, 2023.

[42] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," in *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[43] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. of the 27th Int. Conf. on Neural Information Processing Systems*, p. 568–576, 2014.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[45] G. Bertasius, H. Wang, and L. Torresani, "Is Space-Time Attention All You Need for Video Understanding?," in *Proc. of the 38th Int. Conf. on Machine Learning*, vol. 139, pp. 813–824, 2021.

[46] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," in *Proc. of the IEEE/CVF Int. Conf. on Comp. Vision*, pp. 6836–6846, 2021.

[47] S. Yan, X. Xiong, A. Arnab, Z. Lu, M. Zhang, C. Sun, and C. Schmid, "Multiview Transformers for Video Recognition," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3323–3333, 2022.

[48] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The Kinetics Human Action Video Dataset," 2017.

[49] M. Monfort, A. Andonian, B. Zhou, K. Ramakrishnan, S. A. Bargal, T. Yan, L. Brown, Q. Fan, D. Gutfreund, C. Vondrick, and A. Oliva, "Moments in Time Dataset: One Million Videos for Event Understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 502–508, 2020.

[50] D. Damen, H. Doughty, G. M. Farinella, A. Furnari, E. Kazakos, J. Ma, D. Moltisanti, J. Munro, T. Perrett, W. Price, *et al.*, "Rescaling Egocentric Vision: Collection, Pipeline and Challenges for EPIC-KITCHENS-100," *Int. Journal of Computer Vision*, pp. 1–23, 2022.

[51] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic, "The "Something Something" Video Database for Learning and Evaluating Visual Common Sense," in *IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 5843–5851, 2017.

[52] S. Yan, Y. Xiong, and D. Lin, "Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition," *Proc. of the AAAI Conf. on Artificial Intelligence*, vol. 32, no. 1, 2018.

[53] G. Brown, J. L. Wyatt, P. Tino, and Y. Bengio, "Managing Diversity in Regression Ensembles," *Journal of Machine Learning Research*, vol. 6, no. 55, pp. 1621–1650, 2005.

[54] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.

[55] G. Biau, "Analysis of a Random Forests Model," *Journal of Machine Learning Research*, vol. 13, no. 38, pp. 1063–1095, 2012.

[56] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[57] M. Woźniak, M. Graña, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol. 16, pp. 3–17, 2014.

[58] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546, 2005.

[59] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[60] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: a highly efficient gradient boosting decision tree," in *Proc. of the 31st Int. Conf. on Neural Information Processing Systems*, p. 3149–3157, 2017.

[61] S. R. Lipsitz and G. Fitzmaurice, "An Extension of Yule's Q to Multivariate Binary Data," *Biometrics*, vol. 50, no. 3, pp. 847–852, 1994.

[62] G. Giacinto and F. Roli, "Design of effective neural network ensembles for image classification purposes," *Image and Vision Computing*, vol. 19, no. 9, pp. 699–707, 2001.

[63] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine Learning*, vol. 51, pp. 181–207, 2003.

[64] K. G. Larsen, "Bagging is an Optimal PAC Learner," in *The Thirty Sixth Annual Conf. on Learning Theory*, vol. 195, pp. 450–468, 2023.

[65] A. Mohammed and R. Kora, "A comprehensive review on ensemble deep learning: Opportunities and challenges," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 2, pp. 757–774, 2023.

[66] P. Guo, Z. Xue, Z. Mtema, K. Yeates, O. Ginsburg, M. Demarco, L. R. Long, M. Schiffman, and S. Antani, "Ensemble Deep Learning for Cervix Image Selection toward Improving Reliability in Automated Cervical Precancer Screening," *Diagnostics*, vol. 10, no. 7, 2020.

[67] G. Haralabopoulos, I. Anagnostopoulos, and D. McAuley, "Ensemble Deep Learning for Multilabel Binary Classification of User-Generated Content," *Algorithms*, vol. 13, no. 4, 2020.

[68] B. Wang, B. Xue, and M. Zhang, "Particle Swarm optimisation for Evolving Deep Neural Networks for Image Classification by Evolving and Stacking Transferable Blocks," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.

[69] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1096–1104, 2016.

[70] E. Walach and L. Wolf, "Learning to Count with CNN Boosting," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proc., Part II 14*, pp. 660–676, 2016.

[71] I. E. Livieris, L. Iliadis, and P. Pintelas, "On ensemble techniques of weight-constrained neural networks," *Evolving Systems*, vol. 12, pp. 155–167, 2021.

[72] F. Ali, S. El-Sappagh, S. R. Islam, D. Kwak, A. Ali, M. Imran, and K.-S. Kwak, "A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion," *Information Fusion*, vol. 63, pp. 208–222, 2020.

[73] Y.-H. Cao, J. Wu, H. Wang, and J. Lasenby, "Neural random subspace," *Pattern Recognition*, vol. 112, p. 107801, 2021.

[74] K. Hara, D. Saitoh, and H. Shouno, "Analysis of Dropout Learning Regarded as Ensemble Learning," in *Artificial Neural Networks and Machine Learning – ICANN 2016*, pp. 72–79, 2016.

[75] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining and Knowledge Discovery*, vol. 29, pp. 565–592, 2015.

[76] K. Paliwal, A. Agarwal, and S. S. Sinha, "A modification over Sakoe and Chiba's dynamic time warping algorithm for isolated word recognition," *Signal Processing*, vol. 4, no. 4, pp. 329–333, 1982.

[77] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 31, pp. 606–660, 2017.

[78] R. Bellman, "On the Theory of Dynamic Programming," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 38, no. 8, pp. 716–719, 1952.

[79] A. Lahreche and B. Boucheham, "A Comparison Study of Dynamic Time Warping's Variants for Time Series Classification," *Int. Journal of Informatics and Applied Mathematics*, vol. 4, no. 1, p. 56–71, 2021.

[80] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh, "Generalizing DTW to the multi-dimensional case requires an adaptive approach," *Data mining and knowledge discovery*, vol. 31, no. 1, p. 1—31, 2017.

[81] M. Shokoohi-Yekta, J. Wang, and E. Keogh, "On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case," in *Proc. of the 2015 SIAM Int. Conf. on Data Mining (SDM)*, pp. 289–297, 2015.

[82] J. Treliński and B. Kwolek, "Convolutional neural network-based action recognition on depth maps," in *Int. Conf. ICCVG, Warsaw*, pp. 209–221, 2018.

[83] W. Meert, "DTAIDistance. Time series distances: Dynamic Time Warping (DTW)," *Zenodo*, 2018.

[84] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, pp. 389–422, 2002.

[85] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[86] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[87] H. Mohammadzade, S. Hosseini, M. R. Rezaei-Dastjerdehei, and M. Tabejamaat, "Dynamic Time Warping-Based Features With Class-Specific Joint Importance Maps for Action Recognition Using Kinect Depth Sensor," *IEEE Sensors Journal*, vol. 21, no. 7, pp. 9300–9313, 2021.

[88] G. S. Nambissan, P. Mahajan, S. Sharma, and N. Gupta, "The Variegated Applications of Deep Learning Techniques in Human Activity Recognition," in *Proc. of the 2021 Thirteenth Int. Conf. on Contemporary Computing*, p. 223–233, 2021.

[89] M. Masnad, G. M. MukitHasan, K. M. Iftekhar, and M. S. Rahman, "Human Activity Recognition using DTW Algorithm," in *2019 IEEE Region 10 Symposium (TENSYMP)*, pp. 39–43, 2019.

[90] M. Wöllmer, M. Al-Hames, F. Eyben, B. Schuller, and G. Rigoll, "A multidimensional dynamic time warping algorithm for efficient multimodal fusion of asynchronous data streams," *Neurocomputing*, vol. 73, no. 1, pp. 366–380, 2009.

[91] H. Świtonski, Adam and Jośinski and K. Wojciechowski, "Dynamic time warping in classification and selection of motion capture data," *Multidimensional Systems and Signal Processing*, vol. 30, pp. 1437–1468, 2019.

[92] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3D points," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition - Workshops*, pp. 9–14, 2010.

[93] C. Chen, R. Jafari, and N. Kehtarnavaz, "UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor," in *IEEE Int. Conf. on Image Processing (ICIP)*, pp. 168–172, 2015.

[94] J.-F. Hu, W.-S. Zheng, J. Lai, and J. Zhang, "Jointly learning heterogeneous features for RGB-D activity recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5344–5352, 2015.

[95] L. Xia and J. Aggarwal, "Spatio-temporal Depth Cuboid Similarity Feature for Activity Recognition Using Depth Camera," in *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 2834–2841, 2013.

[96] P. Wang, W. Li, Z. Gao, J. Zhang, C. Tang, and P. O. Ogunbona, "Action Recognition From Depth Maps Using Deep Convolutional Neural Networks," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 4, pp. 498–509, 2016.

[97] J. Treliński and B. Kwolek, "CNN-based and DTW features for human activity recognition on depth maps," *Neural Computing and Applications*, vol. 33, no. 21, pp. 14551–14563, 2021.

[98] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[99] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?," in *Thirty-sixth Conf. on Neural Information Processing Systems Datasets and Benchmarks Track*, vol. 35, pp. 507–520, 2022.

[100] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, p. 785–794, 2016.

[101] M. Minsky and S. A. Papert, *Perceptrons: An introduction to computational geometry*. The MIT Press, 2017.

[102] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[103] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022.

[104] D. Dua and C. Graff, "UCI Machine Learning Repository," 2017.

[105] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: networked science in machine learning," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 2, p. 49–60, 2014.

[106] A. M. Mohammed, E. Onieva, M. Woźniak, and G. Martínez-Muñoz, "An analysis of heuristic metrics for classifier ensemble pruning based on ordered aggregation," *Pattern Recognition*, vol. 124, p. 108493, 2022.

[107] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, 1st ed., 2000.

[108] S. W. A. Sherazi, J.-W. Bae, and J. Y. Lee, "A soft voting ensemble classifier for early prediction and diagnosis of occurrences of major adverse cardiovascular events for STEMI and NSTEMI during 2-year follow-up in patients with acute coronary syndrome," *PLOS ONE*, vol. 16, no. 6, pp. 1–20, 2021.

[109] J. Treliński and B. Kwolek, "Human Action Recognition on Raw Depth Maps," in *2021 Int. Conf.on Visual Communications and Image Processing (VCIP)*, pp. 1–4, 2021.

[110] Z. Wu and Y. Chen, "Genetic algorithm based selective neural network ensemble," in *IJCAI-01: Proc. of the Seventeenth Int. Joint Conf. on Artificial Intelligence, Seattle, Washington*, 2001.

[111] E. Zalta, "The Stanford Encyclopedia of Philosophy, Metaphysics Research Lab– Stanford University," 2016.

[112] J. Treliński and B. Kwolek, "Enhancing Decision Combination in Classifier Committee via Positional Voting," in *Computational Science – ICCS 2022*, pp. 598–609, 2022.

[113] J. Treliński and B. Kwolek, "Decision Combination in Classifier Committee Built on Deep Embedding Features," in *Computational Collective Intelligence*, pp. 480–493, 2021.

[114] P. Skowron, P. Faliszewski, and A. Slinko, "Achieving fully proportional representation: Approximability results," *Artificial Intelligence*, vol. 222, pp. 67–103, 2015.

[115] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[116] C. R. Shalizi, "Advanced Data Analysis from an Elementary Point of View." https://www.stat.cmu.edu/ cshalizi/ADAfaEPoV/, 2024.

[117] R. E. Kass, U. T. Eden, E. N. Brown, *et al.*, *Analysis of neural data*, vol. 491. Springer, 2014.

[118] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2003.

[119] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[120] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 4th ed., 2008.

[121] T. Head, MechCoder, G. Louppe, I. Shcherbatyi, fcharras, Z. Vinícius, cmmalone, C. Schröder, nel215, N. Campos, T. Young, S. Cereda, T. Fan, rene rex, K. K. Shi, J. Schwabedal, carlosdanielcsantos, Hvass-Labs, M. Pak, SoManyUsernamesTaken, F. Callaway, L. Estève, L. Besson, M. Cherti, K. Pfannschmidt, F. Linzberger, C. Cauet, A. Gut, A. Mueller, and A. Fabisch, "scikit-optimize/scikit-optimize: v0.5.2 (v0.5.2)," 2018.

[122] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd Int. Conf. on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conf. Track Proc.*, 2015.

[123] Soniya, S. Paul, and L. Singh, "Application and Need-Based Architecture Design of Deep Neural Networks," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 13, p. 2052014, 2020.

[124] A. Kadra, M. Lindauer, F. Hutter, and J. Grabocka, "Well-tuned Simple Nets Excel on Tabular Datasets," 2021.

[125] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How Does Batch Normalization Help Optimization?," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[126] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proc. of the 32nd Int. Conf. on Int. Conf. on Machine Learning*, vol. 37, p. 448–456, 2015.

[127] M. Jamshidian, "T-distribution modeling using the available statistical software," *Computational Statistics & Data Analysis*, vol. 25, no. 2, pp. 181–206, 1997.

[128] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.

[129] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[130] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[131] Z. Lu, X. Wu, X. Zhu, and J. Bongard, "Ensemble pruning via individual contribution ordering," in *Proc. of the 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, p. 871–880, 2010.

[132] G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez, "An Analysis of Ensemble Pruning Techniques Based on Ordered Aggregation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 245–259, 2009.

[133] J. Cao, W. Li, C. Ma, and Z. Tao, "Optimizing multi-sensor deployment via ensemble pruning for wearable activity recognition," *Information Fusion*, vol. 41, pp. 68–79, 2018.

[134] D. D. Margineantu and T. G. Dietterich, "Pruning Adaptive Boosting," in *Int. Conf. on Machine Learning*, vol. 97, pp. 211–218, 1997.

[135] J. Treliński and B. Kwolek, "Embedded Features for 1D CNN-based Action Recognition on Depth Maps," in *VISIGRAPP (4: VISAPP)*, pp. 536–543, 2021.

[136] J.-F. Hu, W.-S. Zheng, L. Ma, G. Wang, and J. Lai, "Real-Time RGB-D Activity Prediction by Soft Regression," in *Computer Vision – ECCV 2016*, pp. 280–296, 2016.

[137] J. Treliński and B. Kwolek, "Ensemble of classifiers using CNN and hand-crafted features for depth-based action recognition," in *Artificial Intelligence and Soft Computing: 18th Int. Conference, ICAISC 2019, Zakopane, Proc., Part II 18*, pp. 91–103, 2019.

[138] R. Rifkin and A. Klautau, "In Defense of One-Vs-All Classification," *The Journal of Machine Learning Research*, vol. 5, pp. 101–141, 2004.

[139] J. Treliński and B. Kwolek, "Multi-channels CNN temporal features for depth-based action recognition," in *Twelfth Int. Conf. on Machine Vision (ICMV 2019)*, vol. 11433, p. 114330U, 2020.

[140] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mechanical Systems and Signal Processing*, vol. 151, p. 107398, 2021.

[141] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality Reduction by Learning an Invariant Mapping," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, 2006.

[142] G. Koch, R. Zemel, R. Salakhutdinov, *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, 2015.

[143] B. Ghojogh, M. Sikaroudi, S. Shafiei, H. Tizhoosh, F. Karray, and M. Crowley, "Fisher Discriminant Triplet and Contrastive Losses for Training Siamese Networks," in *Int. Joint Conf. on Neural Networks (IJCNN)*, pp. 1–7, 2020.

[144] W. Pei, D. M. J. Tax, and L. van der Maaten, "Modeling Time Series Similarity with Siamese Recurrent Networks," 2016.

[145] P. Bühlmann and T. Hothorn, "Boosting: A statistical perspective," *The Annals of Statistics*, 2006.

[146] S. B. Kotsiantis, "Bagging and boosting variants for handling classifications problems: a survey," *The Knowledge Engineering Review*, vol. 29, no. 1, p. 78–100, 2014.

[147] C. A. Hall and W. Meyer, "Optimal error bounds for cubic spline interpolation," *Journal of Approximation Theory*, vol. 16, no. 2, pp. 105–122, 1976.

[148] J. Treliński and B. Kwolek, "Ensemble of Multi-channel CNNs for Multi-class Time-Series Classification. Depth-Based Human Activity Recognition," in *Intelligent Information and Database Systems*, pp. 455–466, 2020.

[149] Y. Hou, Z. Li, P. Wang, and W. Li, "Skeleton Optical Spectra-Based Action Recognition Using Convolutional Neural Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 3, pp. 807–811, 2018.

[150] B. Zhang, Y. Yang, C. Chen, L. Yang, J. Han, and L. Shao, "Action Recognition Using 3D Histograms of Texture and A Multi-Class Boosting Classifier," *IEEE Transactions on Image Processing*, vol. 26, no. 10, pp. 4648–4660, 2017.

[151] A. Kamel, B. Sheng, P. Yang, P. Li, R. Shen, and D. D. Feng, "Deep Convolutional Neural Networks for Human Action Recognition Using Depth Maps and Postures," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 9, pp. 1806–1819, 2019.

[152] M. F. Bulbul, A. Ullah, H. Ali, and D. Kim, "A Deep Sequence Learning Framework for Action Recognition in Small-Scale Depth Video Dataset," *Sensors*, vol. 22, no. 18, 2022.

[153] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming Auto-Encoders," in *Artificial Neural Networks and Machine Learning – ICANN 2011*, pp. 44–51, 2011.

[154] L. Ye and E. Keogh, "Time series shapelets: a novel technique that allows accurate, interpretable and fast classification," *Data Mining and Knowledge Discovery*, vol. 22, pp. 149–182, 2011.

[155] J. Treliński and B. Kwolek, "Deep Embedding Features for Action Recognition on Raw Depth Maps," in *Computational Science – ICCS 2021*, pp. 95–108, 2021.

[156] J. Zhou, X. Huang, Q. Chen, Q. Hu, T. Wang, and L. He, "Deep learning for aspect-level sentiment classification: Survey, vision and challenges," *IEEE Access*, vol. PP, pp. 1–1, 05 2019.

[157] J. D. Hol, T. B. Schon, and F. Gustafsson, "On Resampling Algorithms for Particle Filters," in *IEEE Nonlinear Statistical Signal Processing Workshop*, pp. 79–82, 2006.

[158] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[159] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The UCR Time Series Classification Archive," 2015.

[160] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, "Tslearn, A Machine Learning Toolkit for Time Series Data," *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020.

[161] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 392–401, 2014.

[162] M. A. Fauzi and P. Bours, "Ensemble Method for Sexual Predators Identification in Online Chats," in *2020 8th Int. Workshop on Biometrics and Forensics (IWBF)*, pp. 1–6, 2020.

[163] S. Luke, *Essentials of Metaheuristics*. Lulu, second ed., 2012.

[164] J.-F. Hu, W.-S. Zheng, L. Ma, G. Wang, J. Lai, and J. Zhang, "Early Action Prediction by Soft Regression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2568–2583, 2019.

[165] Q. Ke, M. Bennamoun, H. Rahmani, S. An, F. Sohel, and F. Boussaid, "Learning Latent Global Network for Skeleton-Based Action Prediction," *IEEE Transactions on Image Processing*, vol. 29, pp. 959–970, 2020.

[166] P. Zhang, C. Lan, W. Zeng, J. Xing, J. Xue, and N. Zheng, "Semantics-guided neural networks for efficient skeleton-based human action recognition," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pp. 1112–1121, 2020.

[167] X. Wang, J.-F. Hu, J.-H. Lai, J. Zhang, and W.-S. Zheng, "Progressive Teacher-Student Learning for Early Action Prediction," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3551–3560, 2019.

[168] C. Liang, D. Liu, L. Qi, and L. Guan, "Multi-Modal Human Action Recognition With Sub-Action Exploiting and Class-Privacy Preserved Collaborative Representation Learning," *IEEE Access*, vol. 8, pp. 39920–39933, 2020.