POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Praca dyplomowa magisterska

Detekcja upadków u osób starszych z wykorzystaniem czujnika GridEye i metod inteligencji obliczeniowej na krawędzi chmury obliczeniowej

Autor: inż. Łukasz Błasiak

Kierujący pracą: dr hab. prof. Pol. Śl. Dariusz Mrozek

Konsultant: dr inż. Krzysztof Tokarz

Gliwice, Październik, 2020

Załącznik Nr 2 do Zarz. Nr 97/08/09

Oświadczenie

Wyrażam zgodę / nie wyrażam* zgody na udostępnienie mojej pracy dyplomowej / rozprawy doktorskiej*

Gliwice, dnia ……

……..……..……..……

(podpis)

……

(poświadczenie wiarygodności podpisu przez Dziekanat)

** podkreślić właściwe*

Spis treści

1. Wstęp

Rozwój technologiczny, postęp cywilizacyjny, stale wzrastające tempo życia oraz ustawiczne dążenie do poprawy komfortu życia nierzadko sprawiają, że nie zauważa się problemów i potrzeb otaczających nas ludzi. O ile niewielkie zainteresowanie losem osób młodych, zdrowych i samodzielnych raczej nie jest kwestią problematyczną, o tyle sytuacja staje się bardziej skomplikowana, gdy stan taki dotyczy ludzi w podeszłym wieku, o bezpieczeństwo których powinniśmy dbać szczególnie. I tu z pomocą przychodzą coraz bardziej zaawansowane rozwiązania z dziedziny automatyki, elektroniki i informatyki, które w znacznym stopniu pozwalają wyjść naprzeciw oczekiwaniom osób starszych często znacznie trudniej sobie radzących z wyzwaniami XXI wieku. Poszukiwanie, tworzenie i rozwijanie takich rozwiązań wydaje się być szczególnie ważne w przypadku, jak wskazują statystyki, starzejącego się społeczeństwa europejskiego.

Aktualnie na rynku dostępne są różnego rodzaju urządzenia do wykrywania upadków takie, jak: opaski, naszyjniki, inteligentne zegarki, aplikacje do smartfonów. Takie rozwiązania są jednak obarczone kilkoma istotnymi wadami. Wymagają kontaktu z ciałem, ciągłego noszenia oraz ładowania. Są zatem dość absorbujące, a niekiedy niewygodne.

Natomiast mniej jest rozwiązań z wykorzystaniem kamery, a szczególnie kamery termowizyjnej, które w połączeniu ze sztuczną inteligencją pozwalają na wykrycie obecności i upadku osoby. Rozwiązanie bazujące na układzie termowizyjnym jest bezdotykowe, nie trzeba o nim pamiętać i może być stale podłączone do zasilania. W ostatnich latach obserwuje się również dynamiczny rozwój metod inteligencji obliczeniowych oraz sztucznych sieci

neuronowych, których budowa bazuje na strukturach ludzkiego mózgu. One też stały się inspiracją podczas tworzenia projektu, którego dotyczy niniejsza praca.

W kontekście omawianej kwestii należy też zwrócić uwagę na dużą użyteczność platform komputerowych, zwłaszcza mikroprocesorów, i usług chmurowych, które znajdują zastosowanie przy tworzeniu rozwiązań z myślą o bezpieczeństwie osób starszych. Stałe dążenie do miniaturyzacji elementów elektronicznych sprzyja osiąganiu bardzo korzystnych parametrów wskazujących na oszczędność w eksploatacji dzięki malejącemu zużyciu energii potrzebnej na ich funkcjonowanie. Ponadto, przy zmniejszeniu rozmiarów urządzenia znaczącą zaletą jest mobilność produktu końcowego i możliwość jego zamontowania praktycznie w dowolnym miejscu. Należy również zauważyć, że wraz z popularyzacją wspomnianych elementów wzrasta ich dostępność, a to pociąga za sobą spadek kosztów zakupu i wytworzenia urządzeń elektronicznych.

1.1. Cel i zakres pracy

Celem niniejszej pracy jest detekcja upadków u osób starszych z wykorzystaniem czujnika Panasonic GridEye i metod uczenia maszynowego na krawędzi chmury Azure lub AWS (ang. *Amazon Web Services*). Utworzone rozwiązanie powinno cechować się następującą funkcjonalnością:

- Wykonywanie oraz analiza pomiarów termicznych,
- Detekcja upadku starszej osoby w oparciu o metody inteligencji obliczeniowej,
- Wysyłanie powiadomień do opiekunów o wykrytym upadku w postaci wiadomości typu SMS (ang. *Short Message Service*),

- Zapis nagrania upadku podopiecznego w usłudze chmurowej,
- Przegląd upadków przypisanych podopiecznych poprzez dedykowany interfejs graficzny.

Ponadto w skład planu pracy w ramach niniejszego projektu wchodzi:

1. Analiza zagadnienia;
2. Określenie funkcjonalności urządzenia/programu;
3. Przegląd istniejących rozwiązań;
4. Wybór technologii i narzędzi programistycznych;
5. Projekt i implementacja urządzenia/programu;
6. Testowanie i uruchamianie;
7. Badania klasyfikatorów umożliwiających detekcję upadków;
8. Analiza wybranych układów nadających się do realizacji zadania.

1.2. Organizacja dokumentu

Niniejszy dokument jest podzielony na 6 rozdziałów. Drugi rozdział przedstawia analizę tematu wraz z opisem wybranych algorytmów inteligencji obliczeniowej. Trzeci rozdział zawiera opis przedmiotu pracy oraz opis przyjętego rozwiązania programowego. Czwarty rozdział przedstawia wyniki badań wykorzystanych algorytmów w celu detekcji upadku osoby oraz porównanie wybranych platform cyfrowych. Piąty rozdział zawiera podsumowanie zrealizowanego zakresu prac oraz wnioski.

Ostatnią częścią dokumentu jest bibliografia, a w następnej kolejności spis skrótów i symboli, opis zawartości dołączonej płyty oraz spis rysunków oraz tabel.

2. Analiza tematu

Niniejszy rozdział zawiera przegląd istniejących rozwiązań pozwalających na detekcję upadku starszych osób. Dodatkowo przedstawione zostały wybrane metody inteligencji obliczeniowej wraz z technologiami znajdującymi zastosowanie w rozwiązywaniu problemów klasyfikacji aktywności ruchowych.

2.1. Istniejące rozwiązania

Wraz z rozwojem technologii oraz rosnącą dostępnością układów cyfrowych urządzenia dedykowane osobom starszym zyskują coraz bardziej na popularności.

Jedną z bardziej popularnych kategorii urządzeń są czujniki wymagające noszenia takie, jak inteligentne zegarki, opaski, naszyjniki. Mają one zastosowanie zarówno diagnostyczne, jak i obserwacyjne. Wykorzystywane są do zbierania informacji fizjologicznych oraz ruchu, umożliwiając w ten sposób monitorowanie stanu pacjenta. Komunikacja opiera się na bezprzewodowej transmisji danych z czujników do telefonu komórkowego lub punktu dostępu, a te następnie przekazują informację do centrum danych poprzez łącze internetowe. Zdarzenia nagłego wypadku, takie, jak na przykład upadek osoby, wykrywane są poprzez zaimplementowane oprogramowanie przetwarzania danych, które w następnym kroku informuje pogotowie ratunkowe, zapewniając tym samym natychmiastową pomoc pacjentom. Szczególnie istotne dla

zastosowań w dziedzinie detekcji upadku są postępy w technologii wytwarzania systemów MEMS (ang. *microelectromechanical system*). Powyższa technologia umożliwiła opracowanie zminiaturyzowanych czujników bezwładnościowych, które są wykorzystywane w analizie aktywności ruchowej i innych systemach monitorowania stanu zdrowia. Dzięki zastosowaniu technik produkcji seryjnej osiągnięto znaczną redukcję rozmiaru i kosztu czujników. Mikroelektronika jest również wykorzystana do integracji innych komponentów takich, jak mikroprocesory i obwody komunikacji radiowej w jednym obwodzie scalonym [17].

Układy wymagające noszenia często są łączone z czujnikami otoczenia, szczególnie w sytuacji, gdy podopieczny jest monitorowany w środowisku domowym. Takie rozwiązania cechują się większym komfortem użytkowania, obserwacją nawet do kilku osób jednocześnie oraz możliwością podłączenia urządzenia do stałego źródła zasilania. Dodatkowym atutem jest brak konieczności pamiętania o zakładaniu czujnika, co pozwala na nieprzerwaną obserwację podopiecznego, w szczególności w przypadku osób cierpiących na obniżoną sprawność umysłową, jak na przykład alzheimer lub demencja.

Jednym z przykładów zastosowań czujników otoczenia jest komercyjny system Walabot HOME firmy Vayyar Imaging Ltd. Układ, wykorzystując moduł radiowy 4D, tworzy trójwymiarową mapę obserwowanego pomieszczenia, co pozwala na detekcje aktywności ruchowej takiej, jak położenie na łóżku, siad na krześle lub upadek. Analiza wykonywana jest z użyciem algorytmów sztucznej inteligencji, gdzie dokonywana jest klasyfikacja upadku wraz z możliwością wysłania powiadomienia do wybranej osoby sprawującej opiekę nad podopiecznym [23].

Dodatkowo wartą uwagi jest technologia WBANs (*ang. wireless body area networks*) będąca specjalistyczną siecią obejmującą różne podsieci oraz urządzenia bezprzewodowe, umożliwiając zdalne monitorowanie

pacjentów w różnych środowiskach. Szereg małych czujników zostaje strategicznie rozmieszczonych na ciele lub w organizmie pacjenta w celu utworzenia sieci WBAN. Charakteryzują się one małymi rozmiarami, lekkością, niskim zużyciem energetycznym oraz wytrzymałością. Jednym z docelowych zastosowań takiego rozwiązania są środowiska medyczne, w których stan dużej liczby pacjentów jest stale monitorowany w czasie rzeczywistym. Głównym celem jest dostarczanie danych biomedycznych oraz możliwość ciągłego monitorowania parametrów zdrowotnych takich, jak temperatura ciała, tętno, ciśnienie tętnicze krwi w dyskretny i skuteczny sposób [2].

2.2. Metody inteligencji obliczeniowej

Niniejszy rozdział zawiera przegląd wybranych metod inteligencji obliczeniowej znajdujących zastosowanie w rozwiązywaniu problemu klasyfikacji obrazów. Dodatkowo omówione zostały podstawowe pojęcia procesu uczenia maszynowego.

2.2.1. Wybrane pojęcia uczenia maszynowego

- Wsad (ang. *batch*) – w przypadku bardziej złożonych projektów, zbiór treningowy może być zbyt duży, aby w jednym kroku przekazać go w całości do sieci neuronowej. W takim przypadku zbiór zostaje podzielony na mniejsze części zwane wsadem;
- Epoki (ang. *epochs*) – wielkość określająca, ile razy zbiór treningowy został przekazany od warstwy wejściowej do wyjściowej i z powrotem przez sieć neuronową;
- Iteracje (ang. *iterations*) – liczba iteracji w jednej epoce, jest to iloraz wielkości zbioru treningowego i wielkości wsadu.

Przykładowo dla zbioru o wielkości 2000 próbek i rozmiarze wsadu równym 500 liczba iteracji w jednej epoce wynosi 4;

- Strata (ang. *loss*) – miara odchylenia lub różnicy pomiędzy przewidywaną wartością a oczekiwaną;
- Overfitting – nadmierne dopasowanie modelu do danych treningowych. Najczęściej występuje w wyniku zbyt długiego procesu uczenia lub zbyt małej różnorodności zbioru treningowego. Powoduje to zmniejszenie miary straty kosztem dopasowania się do wszystkich cech zbioru, w tym szumów oraz innych wzorców, które nie powinny zostać uogólnione przez model;
- Underfitting – przeciwieństwo zjawiska *overfitting*; przypadek kiedy model nie potrafi zmodelować danych treningowych, ani sklasyfikować nowych próbek. Najczęściej występuje w sytuacji zbyt wczesnego przerwania uczenia maszynowego lub zbyt małego zbioru treningowego.

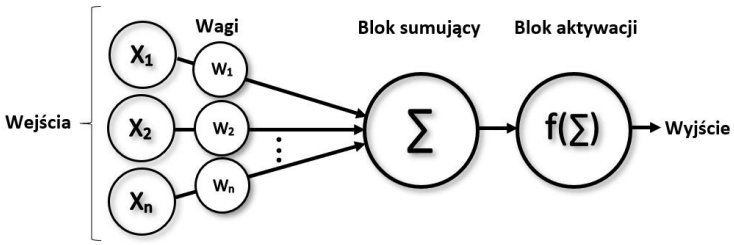
Istnieją również metryki opisujące jakość modeli utworzonych w procesie uczenia maszynowego, które przedstawione zostały w rozdziale 4.1.

2.2.2. Sieć neuronowa

Jedną z bardziej popularnych struktur do rozwiązywania problemów klasyfikacji są sztuczne sieci neuronowe. Znajdują one szerokie zastosowanie w wykrywaniu anomalii w danych, systemach decyzyjnych, rozumieniu oraz rozpoznawaniu języka naturalnego, systemach, a także w dokonywaniu detekcji i klasyfikacji obiektów na zdjęciu [28].

Podstawową jednostką sieci neuronowej jest neuron, będący prostym elementem obliczeniowym, do którego doprowadzane są sygnały z wejść

sieci lub neuronów z poprzedniej warstwy. Przykładowy neuron został przedstawiony na rysunku 2.1. Każdy sygnał wchodzący do neuronu jest mnożony przez odpowiednią wagę, które ulegają zmianom podczas procesu uczenia. W ostatnim kroku zważona suma cech jest przekazywana jako argument do funkcji aktywacji. W zależności od potrzeb, blok aktywacji może być opisany funkcją skoku, liniową lub nieliniową [15].

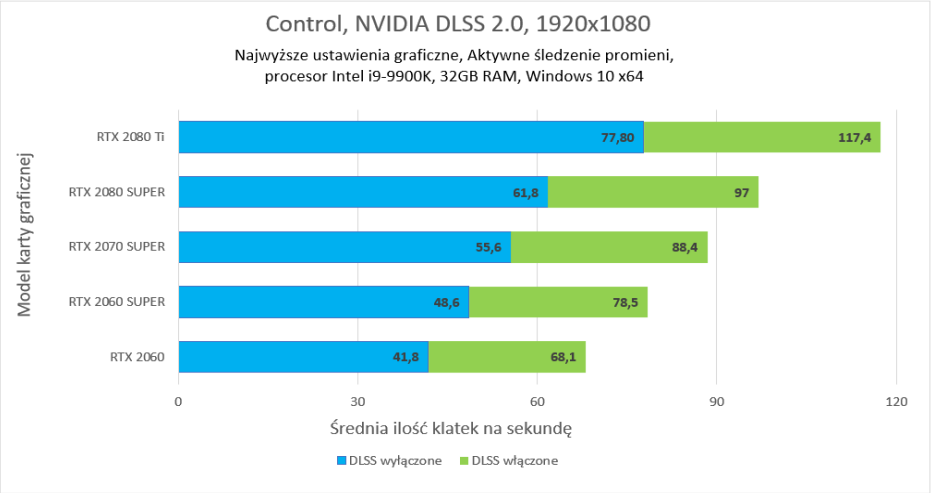


Rysunek 2.1 Model neuronu.

Sieć zbudowana jest z warstw, w której skład wchodzą pojedyncze, niepołączone ze sobą, neurony. Każda sieć posiada warstwę wejściową, odpowiedzialną za pobieranie i przekazywanie danych w głąb sieci. Dodatkowo wyróżniana jest również warstwa wyjściowa, odpowiedzialna za obliczenie wyniku końcowego. Ponadto najczęściej spotykane są wielowarstwowe struktury zawierające dodatkowo jeszcze przynajmniej jedną warstwę ukrytą, odpowiedzialną za naukę oraz łącznie sąsiednich warstw [22].

Jednym z większych oraz bardziej popularnych zastosowań sztucznej sieci neuronowej jest projekt amerykańskiego przedsiębiorstwa komputerowego Nvidia o nazwie NVIDIA DLSS. Rozwiązanie dokonuje sztucznego zwiększenia rozdzielczości obrazu w oparciu o dedykowane procesory AI układów producenta oraz sieci neuronowe z zakresu głębokiego uczenia. Model został utworzony na podstawie zbioru treningowego składającego się z dziesiątek tysięcy nagrań w różnych rozdzielczościach. Możliwość generowania animacji w niższej rozdzielczości skutkuje odciążeniem procesora karty graficznej, zmniejszeniem poboru mocy oraz zwiększeniem liczby generowanych klatek

na sekundę przy zachowaniu ostrości obrazu. Wpływ technologii na wydajność na przykładzie gry komputerowej „Control" przedstawia rysunek 2.2. Ponadto w niektórych przypadkach zaobserwowano poprawę wygładzania krawędzi oraz jakości animacji względem tradycyjnego podejścia polegającego na renderowaniu klatki od razu w docelowej rozdzielczości. Wartym uwagi jest fakt, iż od wersji 2.0 rozwiązanie wykorzystuje uogólnioną sieć neuronową, która działa w różnych grach komputerowych oraz nie wymaga przeprowadzania procesu uczenia dla każdej nowej produkcji z osobna. Ponadto projekt jest w dalszym ciągu rozwijany przez przedsiębiorstwo [16].



Rysunek 2.2 Wpływ technologii Nvidia DLSS na wydajność w grze Control.

Dodatkowo sieci neuronowe potrafią znaleźć zastosowanie również w wykrywaniu upadku starszych osób. Wartym uwagi jest artykuł naukowy „Home Camera-Based Fall Detection System for the Elderly" [6] znajdujący się w archiwum Amerykańskiego Narodowego Instytutu Zdrowia. Publikacja opisuje układ bazujący na układzie Raspberry Pi oraz dedykowanej kamerze, mający na celu detekcję upadku starszej osoby. Oprogramowanie systemu wykorzystuje konwolucyjną sieć neuronową wyuczoną na podstawie zbioru treningowego liczącego 50 nagrań upadającej

osoby. Dodatkowo zastosowane zostały pomocnicze techniki ekstrakcji cech i śledzenia obiektu takie, jak filtr Kalmana oraz usuwanie tła algorytmem MoG2 biblioteki OpenCV. Pomimo wykorzystania podstawowej kamery cechującej się niższą jakością wykonywanych zdjęć, autorzy uzyskali dokładność na poziomie 96.9% w świetle dziennym oraz w odległości do 10 metrów od modułu. Natomiast zaobserwowano również trudności detekcji upadku osoby w przypadku gorszego oświetlenia w pomieszczeniu lub w porze nocnej. Rozwiązaniem powyższego problemu może być zastąpienie lub rozszerzenie układu o moduł kamery termowizyjnej taki, jak np. Panasonic GridEye. Ponadto oprogramowanie dokonywało w przybliżeniu około 7-8 klasyfikacji klatek na sekundę na platformie Raspberry Pi w wersji 2. Wykazano również, iż jest to częstotliwość wystarczająca do detekcji upadku osoby [6].

2.2.3. Maszyna Wektorów Nośnych

Maszyna wektorów nośnych jest popularnym algorytmem metod uczenia maszynowego, cechującym się wysoką skutecznością oraz niskim zapotrzebowaniem na moc obliczeniową. Koncept znajduje zastosowanie w problemach klasyfikacji obrazu, wykrywania twarzy, kategoryzacji tekstu, rozpoznawaniu pisma ręcznego oraz w dziedzinie bioinformatyki. Proces tworzenia modelu polega na znalezieniu *n*-wymiarowej hiperpłaszczyzny, gdzie *n* jest liczbą cech zbioru, która jednoznacznie rozdziela klasyfikowane dane. W celu oddzielenia dwóch klas można wybrać wiele hiperpłaszczyzn, jednakże celem algorytmu jest znalezienie hiperpłaszczyzny z maksymalnym marginesem, czyli maksymalną odległością między próbkami zbiorów obu klas. Maksymalna odległość marginesu sprawia, iż przyszłe punkty zbioru będą mogły być klasyfikowane z większą pewnością. W algorytmie SVM (ang. *Support Vector Machine*), jeżeli wynik klasyfikacji jest większy niż 1 to jest identyfikowany z jedną klasą, natomiast w przypadku wartości równej -1 następuje klasyfikacja do drugiej klasy. W związku ze zmiennym progiem przedział z zakresu [-1,1] jest pewnego

rodzaju wzmocnieniem klasyfikacji oraz pełni rolę marginesu. W celu znalezienia jak największego marginesu pomiędzy hiperpłaszczyzną a punktami wykorzystuje się funkcję kosztu zwaną „hinge loss", którą przedstawia wzór 2.1.

$$c\big(x, y, f(x)\big) = \begin{cases} 0, & y * f(x) \geq 1 \\ 1 - y * f(x), & x < 0 \end{cases} \quad \textbf{(2.1)}$$

Koszt wynosi 0, jeżeli przewidywana oraz właściwa wartość posiadają ten sam znak. W przeciwnym wypadku wyliczana jest funkcja kosztu. Ponadto dodawany jest również tzw. parametr regulacyjny, mający na celu zrównoważenie maksymalizacji marginesu oraz kosztu [7].

Koncept został poddany analizie pod kątem wykrywania demencji starczej w artykule „Machine learning in medicine: Performance calculation of dementia prediction by support vector machines (SVM)". Autorzy wykorzystali fragment otwartego zbioru projektu OASIS (ang. *Open Access Series Of Imaging Studies*) zawierający badania ponad 1000 osób pod kątem demencji. Każda próbka zawierała podstawowe informacje o pacjencie, takie jak wiek, płeć, wykształcenie, oraz szczegóły badań klinicznych. Algorytm SVM pozwolił uzyskać wydajność klasyfikacji na poziomie 70% [5].

Ponadto maszyna wektorów nośnych znajduje również zastosowanie w wykrywaniu upadku starszych osób. W artykule „Fall Detection with the Support Vector Machine during Scripted and Continuous Unscripted Activities" autorzy poddali analizie połączenie układu akcelerometru z algorytmem SVM w celu klasyfikacji aktywności ruchowych. Do realizacji projektu utworzony został zbiór nowy zbiór treningowy liczący 1050 pomiarów przyspieszenia w trzech osiach X, Y oraz Z. Każdy z wolontariuszy wykonywał różnorodne przypadki aktywności codziennych takich, jak poślizg i schodzenie po schodach, wstanie z łóżka, upadek swobodny czy zmiana pozycji podczas snu. Wartym uwagi jest fakt, iż zbiór zawiera zarówno symulowane, jak i rzeczywiste przypadki upadków. W procesie uczenia maszynowego oraz klasyfikacji wykorzystano również

metody wyodrębnienia cech takie, jak np. obliczenie wektora sumy całkowitej $SV_{Total}(t)$. Takie podejście pozwoliło na uzyskanie dużej liczby kombinacji danych wejściowych oraz zaobserwowano osiągnięcie dokładności do 98,95% [9].

W niniejszej pracy poddano analizie maszynę wektorów nośnych pod kątem upadku z uwzględnieniem różnych funkcji jądra (ang. kernel functions) oraz rozdzielczości próbek zbioru treningowego.

2.2.4. Naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski jest probabilistycznym modelem uczenia maszynowego wykorzystywanym w zadaniach klasyfikacji. Zasada działania algorytmu bazuje na twierdzeniu Bayesa, opisanym wzorem 2.2,

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}, \tag{2.2}$$

gdzie P(C) oznacza prawdopodobieństwo a priori wystąpienia klasy C, czyli prawdopodobieństwo, że dowolny przykład należy do klasy C. Natomiast prawdopodobieństwo P(X|C) oznacza prawdopodobieństwo a-posteriori, że X należy do klasy C, natomiast P(X) oznacza prawdopodobieństwo a priori wystąpienia próbki X [14].

Dodatkowo przyjęte jest założenie o wzajemnej niezależności zmiennych wejściowych, które jednak nie zawsze jest spełnione, co sprawia, iż potocznie klasyfikator jest określany naiwnym. Wartym uwagi jest fakt, iż pomimo powyższego założenia oraz prostoty działania klasyfikator ten często pozwala na uzyskanie lepszych wyników w porównaniu do bardziej złożonych algorytmów.

Naiwny klasyfikator bayesowski znalazł zastosowanie w detekcji upadku osoby w artykule „Application of naive Bayes classifier in fall detection systems based on infrared depth sensors” [11]. Do realizacji projektu autorzy wykorzystali dwa zsynchronizowane czujniki podczerwieni, tworząc tym

samym model przestrzenny obserwowanego pomieszczenia w czasie rzeczywistym. W następnym kroku, z wykorzystaniem autorskiego algorytmu, zostało usunięte tło, pozostawiając jedynie sylwetkę osoby. W oparciu o ustalone scenariusze upadków sporządzony został zbiór treningowy oraz przeprowadzony proces uczenia maszynowego. W utworzonym modelu zaobserwowano specyficzność na poziomie 79,2–86,1% oraz czułość w zakresie 86,1–98,6% [11].

2.2.5. Algorytm k najbliższych sąsiadów

Klasyfikator k-NN jest jednym z algorytmów regresji nieparametrycznej używanych w statystyce do prognozowania wartości pewnej zmiennej losowej. Ponadto może być również używany do klasyfikacji. Cechuje się małą czułością na punkty osobliwe oraz szum w danych treningowych. Metoda działania opiera się o dwa następujące kroki:

1. Poszukaj *k* najbliższych obiektów (sąsiadów) dla klasyfikowanego obiektu $X_{q'}$.
2. Głosuj wśród *k* najbliższych sąsiadów w celu wyznaczania klasy, do której należy $X_{q'}$.

gdzie *k* jest parametrem określającym liczbę najbliższych sąsiadów biorących udział w głosowaniu. Natomiast do wyznaczania odległości pomiędzy obiektami wykorzystuje się metryki takie, jak odległość euklidesowa, współczynnik korelacji liniowej Pearsona, odległość kątowa lub miara Gowera [14].

2.2.6. Drzewo decyzyjne

Drzewo decyzyjne to klasyfikator reprezentowany przeważnie przez drzewo binarne. Węzły drzewa opisują podział zbioru danych ze względu na wartość wybranej cechy, natomiast w liściach drzewa odbywa się przypisanie próbek do klasy. W trakcie budowania drzewa tworzone są

reguły decyzyjne, którym odpowiadają węzły. Proces tworzenia drzewa zazwyczaj zaczyna się od skonstruowania głównego węzła, a w następnej kolejności tworzone są węzły potomne. Reguły decyzyjne zostają ustalone w oparciu o takie kryteria, jak m.in. indeks Gini'ego (ang. *Gini's index*), współczynnik przyrostu informacji (ang. *gain ratio*) oraz przyrostu informacji (ang. *information gain*) [19].

2.2.7. Las losowy

Las losowy (ang. *random forest*) jest zbiorem klasyfikatorów, w którym każdy pojedynczy klasyfikator jest drzewem decyzyjnym. Każde drzewo wchodzące w skład lasu losowego jest uczone na podstawie wylosowanej próbki danych, powstałej poprzez wylosowanie *n* razy ze zwracaniem ze wszystkich *N* próbek uczących. Taka technika generowania danych określana jest jako *bagging*. Dodatkowo, w trakcie budowania drzewa decyzyjnego nie wszystkie atrybuty są brane pod uwagę przy wyznaczaniu reguły decyzyjnej w węźle. Wylosowanych zostaje część atrybutów, na podstawie których wyznaczana jest reguła decyzyjna w węźle. Taka metoda nazywana jest *feature subspace*. Obie techniki stosowane są w celu zwiększenia stabilności odpowiedzi algorytmu i jego ochrony przed nadmiernym dopasowaniem do danych uczących, co przekłada się na lepszą skuteczność działania klasyfikatora na nowych danych. W trakcie przewidywania klasy dla próbki jest ona okazywana wszystkim drzewom decyzyjnym, a przypisanie do klasy najczęściej wskazywanej przez drzewa decyzyjne w lesie traktowane jest jak końcowa decyzja o przynależności do klasy. Las losowy zapewnia lepszą stabilność działania i zazwyczaj lepsze wyniki klasyfikacji niż pojedyncze drzewo decyzyjne [19].

2.3. Przegląd wybranych technologii

Niniejszy rozdział zawiera opis wybranych technologii znajdujących zastosowanie w metodach inteligencji obliczeniowej, implementacji usług chmurowych oraz platform programistycznych.

- C++ - wieloparadygmatowy język programowania wysokiego poziomu, utworzony przez Bjarne Stroustrup'a z firmy Bell Labs na początku lat 80. W dużym stopniu zachowuje zgodność z językiem C, co pozwala na wykorzystywanie bibliotek napisanych w tym języku, bezpośrednie zarządzanie pamięcią oraz wykorzystywanie różnych paradygmatów programowania (obiektowy, proceduralny i generyczny) [21]. W niniejszym projekcie został wykorzystany do implementacji oprogramowania na platformie Arduino;
- Python - interpretowany, zorientowany obiektowo język programowania wysokiego poziomu z dynamiczną semantyką. Posiada wbudowane struktury danych wysokiego poziomu, które, w połączeniu z dynamicznym typowaniem zmiennych, sprawiają, że jest bardzo atrakcyjny dla szybkiego tworzenia aplikacji, a także do wykorzystania jako język skryptowy. Ponadto język Python zyskuje coraz większą popularność w dziedzinie sztucznej inteligencji oraz data science [26];
- TensorFlow - biblioteka typu open-source stworzona przez firmę Google do obliczeń numerycznych, wydana 9 listopada 2015 roku dla języka Python. Rozwiązanie oferuje zestaw narzędzi służących do projektowania oraz trenowania sieci neuronowych wraz z możliwością tworzenia grafów wizualizujących proces uczenia. Aplikacje z użyciem biblioteki TensorFlow mogą zostać uruchomione na szerokiej gamie urządzeń takich, jak lokalny komputer, klaster w usłudze chmurowej lub urządzenie mobilne z systemem

operacyjnym iOS lub Android. Ponadto rozwiązanie posiada również możliwość wykonania złożonych obliczeń z wykorzystaniem karty graficznej [3];

- Scikit-learn - darmowa biblioteka algorytmów z dziedziny uczenia maszynowego, napisana w języku Python. Udostępnia algorytmy nadzorowanego i nienadzorowanego uczenia maszynowego w postaci spójnego interfejsu programistycznego. Posiada wsparcie dla różnego rodzaju klasyfikatorów oraz algorytmów regresji takich jak SVM, gradient boosting, naive Bayes, random forest, k-means [4];
- AWS (ang. *Amazon Web Services*) – jedna z najbardziej wszechstronnych platform chmurowych znajdująca szerokie zastosowanie na całym świecie. Oferuje ponad 175 pełni funkcjonalnych usług z centrów serwerowych na całym świecie. Platforma jest wykorzystywana przez miliony klientów, w tym najszybciej rozwijające się projekty typu start-up, agencje rządowe oraz największe przedsiębiorstwa, w celu obniżenia kosztów, zwiększania elastyczności oraz szybszego wprowadzania innowacji [24];
- Serverless - model usług w chmurze, w którym programista lub architekt skupia się wyłącznie na tworzeniu logiki biznesowej, a nie na infrastrukturze, na której ma ona być wykonywana. Odpowiedzialność za obsługę fizycznych urządzeń oraz ich konfigurację przejmują dostawcy danej usługi serverless, do których zaliczają się m.in. Amazon, Microsoft oraz Google;
- AWS Lamba – usługa w ramach platformy chmurowej AWS, umożliwiająca uruchomienie kodu bez udostępniania lub zarządzania serwerami. Technologia wspiera popularne języki programowania takie, jak Java, Go, PowerShell, Node.js, C#, Python oraz Ruby wraz z skalowalnością kodu oraz wysoką dostępnością. Warta uwagi jest możliwość łączenia funkcji AWS Lambda napisanych w różnych

językach w ramach jednej aplikacji lub zdarzenia. Ponadto istnieje możliwość konfiguracji automatycznego uruchamiania usługi Lambda z innych usług AWS lub z dowolnej aplikacji internetowej lub mobilnej. Rozwiązanie znajduje również zastosowanie do autentykacji oraz autoryzacji użytkowników [25];

- AWS Gateway – usługa w ramach platformy chmurowej AWS ułatwiająca programistom tworzenie, publikowanie, utrzymywanie, monitorowanie i zabezpieczanie interfejsów API (ang. *Application Programming Interface*) w dowolnej skali. AWS Gateway pośredniczy pomiędzy aplikacją, a logiką biznesową lub oprogramowaniem serwera. Ponadto usługa obsługuje konteneryzację, skalowalność oraz rozwiązania typu serverless. Rozwiązanie pozwala obsłużyć wszystkie zadania związane z akceptowaniem i przetwarzaniem do setek tysięcy równoczesnych wywołań API, w tym zarządzanie ruchem, autoryzację i kontrolę dostępu [1];
- AWS DynamoDB – w pełni zarządzana usługa bazy danych NoSQL w ramach platformy AWS, która zapewnia szybką i przewidywalną wydajność oraz płynną skalowalność;
- REST (ang. *Representional State Transfer*) – styl architektoniczny zapewniający standardy pomiędzy systemami komputerowymi w sieci oraz ułatwiający komunikację między nimi. Cechuje się bezstanowością oraz separuje oprogramowanie klienta i serwera, co pozwala na dokonywanie implementacji w różnych językach programowania z wykorzystaniem modularności oraz skalowalności [27].

2.4. Wybrane zintegrowanie środowiska programistyczne

Opis wybranych zintegrowanych środowisk programistycznych znajdujących zastosowanie w implementacji rozwiązań wykorzystujących metody inteligencji obliczeniowej, usług chmurowych oraz platform komputerowych.

- JetBrains PyCharm – komercyjne środowisko programistyczne autorstwa firmy JetBrains dedykowane dla języka Python. Zapewnia inteligentne uzupełnianie i inspekcję kodu źródłowego wraz z wykrywaniem błędów na etapie implementacji. Ponadto zawiera obszerną kolekcję narzędzi do analizy działania oraz testowania uruchomionego programu wraz z rozporoszonym system kontroli wersji Git. Dodatkowo, warta uwagi jest wbudowana obsługa bibliotek naukowych takich, jak Pandas, Numpy, Matplotlib wspierająca analizę oraz wizualizację przetwarzanych struktur danych wraz z wykresami i diagramami. W niniejszym projekcie oprogramowanie zostało wykorzystane do utworzenia oraz trenowania klasyfikatorów dla platformy Raspberry Pi 3B+ oraz implementacji panelu administracyjnego;
- Atmel Studio – darmowe zintegrowanie środowisko programistyczne fimy Microchip, dedykowane dla wszystkich mikrokontrolerów z rodziny AVR produkowanych przez firmę Atmel znajdujących zastosowanie w platformie Arduino. Oprogramowanie pozwala na budowanie aplikacji w języku C/C++ wraz z możliwością symulowania mikrokontrolera. W niniejszym projekcie środowisko zostało wykorzystane do wgrywania oraz analizy skompilowanego programu na platformę Wemos d1 mini oraz Arduino.

3. Przedmiot pracy

Rozdział przedstawia zaproponowane rozwiązanie, specyfikację wewnętrzną oraz opis elementów składowych wraz z platformą sprzętową.

3.1. Projekt eksperymentalnego środowiska

Niniejszy podrozdział zawiera opis wybranych elementów składowych, schemat blokowy i ideowy rozwiązania wraz z wykorzystanymi interfejsami komunikacyjnymi. Dodatkowo, analizie poddane zostały struktury danych wraz z architekturą wewnętrzną systemu oraz punktów dostępu.

3.1.1. Wybrane elementy składowe

W celu analizy ruchu starszej osoby oraz detekcji potencjalnego upadku wykorzystany został moduł Adafruit AMG8833 wraz z matrycą termiczną IR 8x8 GridEye firmy Panasonic, który przedstawiony został na rysunku 3.1. Konstrukcja zawiera wysokoprecyzyjny układ czujników podczerwieni bazujący na zaawansowanej technologii MEMS. Poprzez promienie podczerwieni układ jest w stanie wykryć ciepło ciała ludzkiego i innych przedmiotów bez bezpośredniego kontaktu. Moduł znajduje szerokie zastosowanie w systemach bezpieczeństwa, bankomatach, automatyzacji domowej i biurowej oraz w aparaturach i placówkach medycznych. Pomiar temperatury wykonywany jest z dokładnością ± 2,5 °C z zakresu od

0 do 80 °C oraz zwraca tablicę 64 indywidualnych odczytów temperatury w postaci macierzy o rozmiarze 8x8. Układ cechuje kąt widzenia wynoszący 60° oraz częstotliwość wykonywania pomiarów w zakresie 1-10 Hz, co pozwala na detekcję człowieka w odległości do 7 metrów w zależności od warunków otoczenia. Konstrukcja posiada wbudowany stabilizator oraz dwukierunkowy konwerter poziomów logicznych, co umożliwia pracę z napięciem zarówno 3,3 V jak i 5 V. Dodatkową zaletą układu jest niski pobór prądu wynoszący 4.5 mA (tryb normalny), 0.8 mA (tryb czuwania), 0.2 mA (tryb uśpienia) oraz małe wymiary 25,8 x 25,5 x 6 mm. Komunikacja z modułem następuje z wykorzystaniem magistrali I^2C (ang. *Inter-Integrated Circuit*) opracowanej przez firmę Philips, która została opisana w rozdziale 3.1.3. Ponadto układ cechuje się dużym wsparciem ze strony producenta oraz społeczności.

Rysunek 3.1 Układ Adafruit AMG8833.

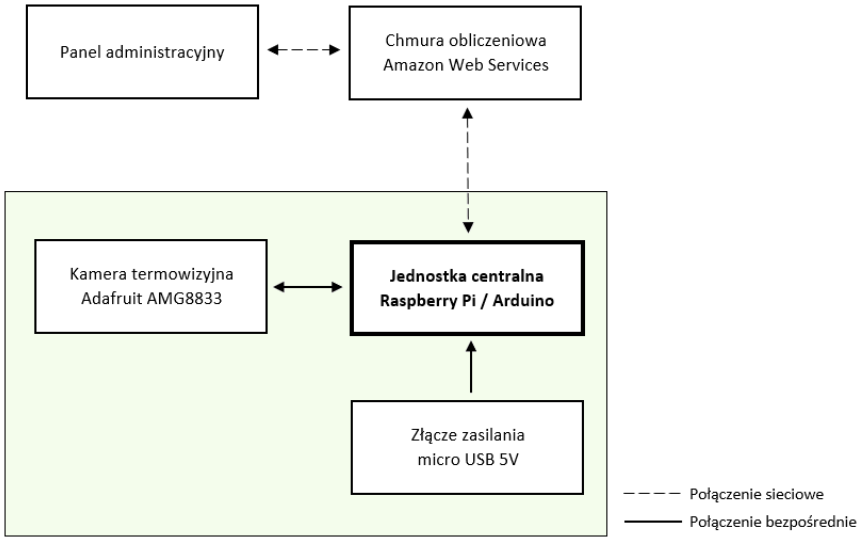
Kolejnym kluczowym elementem jest platforma sprzętowa. Jednym z układów wykorzystanych do realizacji niniejszego projektu jest Raspberry Pi w wersji 3B+. Jest tu urządzenie pierwotnie zaprojektowane do celów edukacyjnych, składające się z pojedynczego obwodu drukowanego. Dodatkowo, implementacji dokonano na platformie Arduino opartej na projekcie Open Hardware oraz w większości przypadków wyposażonej w mikrokontroler firmy Atmel. Podobnie jak w przypadku Raspberry Pi, urządzenie pierwotnie zostało zaprojektowane do celów edukacyjnych.

Szczegółowy opis oraz analiza platform sprzętowych zostały przedstawione w rozdziale 4.4.

Dodatkowo, w rozwiązaniu wykorzystane zostały pojedyncze elementy elektroniczne takie, jak rezystory oraz kondensatory tantalowe, w celu ograniczenia płynącego prądu, stabilizacji napięcia oraz redukcji tętnień.

3.1.2. Schemat blokowy i ideowy

Przed przystąpieniem do fizycznej realizacji układu oraz implementacji oprogramowania sporządzony został schemat blokowy oraz ideowy, który przedstawia rysunek 3.2 oraz 3.3.

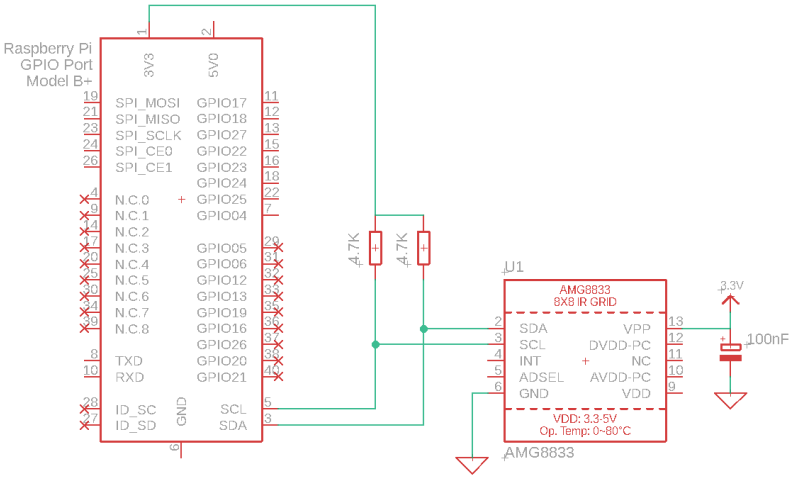


Rysunek 3.2 Schemat blokowy układu.

Przed przystąpieniem do projektowania schematu ideowego wykonana została analiza dokumentacji technicznej wykorzystywanych modułów oraz jednostki centralnej. Jednym z najważniejszych parametrów jest dobór

odpowiedniego napięcia wraz z kondensatorami filtrującymi i odsprzęgającymi napięcie zasilania. Kolejnym ważnym aspektem jest zastosowanie konwerterów stanów logicznych w przypadku różnych wartości tolerancji złączy wejściowych. Ponadto w przypadku magistrali I^2C zalecane jest użycie rezystorów podciągających o wartości 4.7 kΩ dla linii SDA oraz SCL oraz podłączenie ich pod napięcie zasilania. Dodatkowo, zgodnie z dokumentacją techniczną, dla interfejsu UART na liniach RT oraz TX powinny zostać szeregowo zamontowane rezystory o oporności 1 kΩ.

Rysunek *3.3* przedstawia schemat ideowy układu z platformą Raspberry Pi wraz z uwzględnionymi zaleceniami dokumentacji technicznej.



Rysunek 3.3 Schemat ideowy układu.

3.1.3. Wykorzystane interfejsy komunikacyjne

Jednym z istotniejszych interfejsów wykorzystanych w niniejszym projekcie jest I^2C. Jest to dwukierunkowa szeregowa magistrala komunikacyjna opracowana w 1982 roku przez firmę Philips Semiconductor, przeznaczona do komunikacji o niskiej przepustowości pomiędzy urządzeniami peryferyjnymi a jednostkami centralnymi lub

mikrokontrolerami. Standard oparty jest na dwóch liniach: SDA (Serial Data – linia odpowiedzialna za przesył danych) oraz SCL (Serial Clock – sygnał zegara). Szyna SDA umożliwia zarówno odczyt, jak i zapis danych, w przeciwieństwie do linii SCL, na której tylko jeden układ jest w stanie generować przebieg zegara. Obie linie są wspólne dla wszystkich układów podłączonych do magistrali. Ponadto magistrala wykorzystuje model Master-Slave będący modelem asymetrycznej komunikacji, w którym wyróżnić można dwa typy układów:

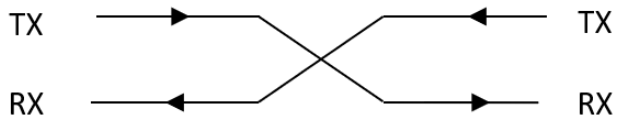
- Master – zarządzające, ustalające kierunek transmisji oraz posiadają adres docelowego urządzenia do komunikacji,
- Slave – podrzędne, posiadają swój własny adres.

W większości przypadków transmisja odbywa się w trybie standardowym o przepustowości 100 kbit/s, jednakże dodatkowo wyróżniane są tryby Fast (400 kbit/s), Fast Plus (1 Mbit/s) oraz High Speed (3.4 Mbit/s). Wartym uwagi jest fakt, iż nie jest określona dolna granica prędkości, a poszczególne fazy przebiegów mogą być dowolnie wydłużane. Wyjścia SDA i SCL wszystkich układów posiadają konfigurację otwarty kolektor lub otwarty dren. W przypadku niezajętej magistrali na liniach występuje stan wysoki, a wygenerowanie niskiego poziomu logicznego przez dowolne urządzenie peryferyjne powoduje niski stan na całej linii [20].

W niniejszym projekcie interfejs I^2C został wykorzystany do komunikacji pomiędzy modułem termowizyjnym Adafruit AMG8833 a platformą Raspberry Pi i Arduino. Zgodnie z zaleceniami dokumentacji technicznej, linie SDA oraz SCL zostały podłączone do napięcia zasilania poprzez rezystory podciągające o oporności 4.7 kΩ. Ponadto układ AMG8833, dzięki zastosowaniu konwertera poziomów logicznych, pracuje zarówno z napięciem o wartości 3.3 V jak i 5 V.

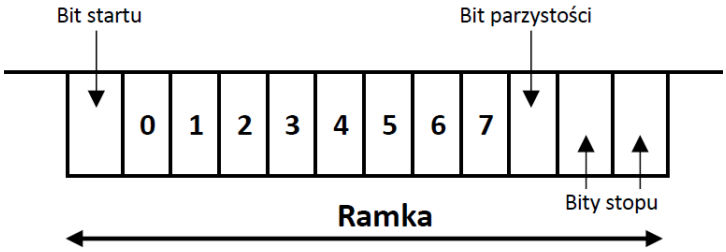
Kolejnym wykorzystanym interfejsem jest UART (ang. *Universal Asynchronous Receiver-Transmitter*), będący szeregowym kanałem

komunikacyjnym do komunikacji z urządzeniami peryferyjnymi oraz komputerami. Do przesyłu danych wykorzystuje sygnały TxD (ang. *Transmiter* – nadajnik) oraz RxD (ang. *Receiver* – odbiornik), których linie danych należy łączyć krzyżowo. Przykład prawidłowego połączenia przedstawia rysunek 3.4.



Rysunek 3.4 Przykład połączenia interfejsu UART.

Układ konwertuje wychodzące oraz przychodzące bajty danych na strumień szeregowy, w skład którego wchodzi bit początkowy (rozpoczynający szeregowy strumień), bity danych, bit parzystości (służący do kontroli błędów odbioru – jeżeli liczba wystąpień „1” w bitach danych jest parzysta wtedy przyjmuje wartość „1”, w przeciwnym wypadku „0”) oraz jeden lub dwa bity stopu (kończące słowo danych – ramkę). W zależności od konfiguracji występuje po sobie 7, 8 lub 9 bitów danych. Rysunek 3.5 przedstawia schemat przykładowej ramki.



Rysunek 3.5 Przykład ramki w interfejsie UART.

W celu prawidłowego przebiegu transmisji danych, w obu urządzeniach musi być ustawiona taka sama prędkość transmisji danych (ang. *baud-rate*), która określi liczbę bitów przesyłanych na sekundę. Najczęściej spotykanymi wartościami są 9600 oraz 115200 [13].

Ponadto wartym uwagi jest współczynnik *baud rate error*, którego występowanie jest spowodowane różnymi częstotliwościami pracy modułu UART, a procesora lub mikrokontrolera. Wartość współczynnika można wyliczyć na podstawie wzorów znajdujących się w dokumentacji technicznej mikrokontrolera. Wzory 3.1, 3.2 oraz 3.3 przedstawiają obliczenia dla mikrokontrolera ATmega328P wykorzystywanego w niniejszym projekcie przez platformę Arduino. W pierwszym kroku należy obliczyć wartość współczynnika UBBR (ang. *UART Baud Rate Register*) według wzoru 3.1.

$$\boldsymbol{UBRR = \frac{F_{CPU}}{16 * BaudRate} - 1}, \tag{3.1}$$

gdzie:

F_{CPU} – częstotliwość bazowa mikrokontrolera,
BaudRate – teoretyczna szybkość transmisji,
UBRR – wartość rejestru Uart Baud Rate.

Otrzymany wynik należy zaokrąglić do najbliższej liczby całkowitej ze względu na to, iż omawiany rejestr może przechowywać wyłącznie liczby całkowite. W kolejnym kroku należy obliczyć rzeczywistą szybkość transmisji, korzystając ze wzoru 3.2:

$$\boldsymbol{BaudRate_{ClosestMatch} = \frac{F_{CPU}}{16 * (UBRR + 1)}}, \tag{3.2}$$

gdzie:

F_{CPU} – częstotliwość bazowa mikrokontrolera,
UBRR – wartość rejestru Uart Baud Rate Register,
$BaudRate_{ClosestMatch}$ – rzeczywista prędkość transmisji.

W ostatnim kroku należy wyliczyć współczynnik *baud rate error*, wykorzystując wzór 3.3:

$$\boldsymbol{Error}[\%] = \left(\frac{\boldsymbol{BaudRate_{ClosestMatch}}}{\boldsymbol{BaudRate}} - \mathbf{1}\right) * \mathbf{100}\%, \qquad \textbf{(3.3)}$$

gdzie:

$BaudRate_{ClosestMatch}$ – rzeczywista prędkość transmisji,

BaudRate – teoretyczna szybkość transmisji.

Tabela 3.1 przedstawia przykładowe wartości współczynnika *baud rate error* dla wybranych taktowań mikrokontrolera ATmega328P [12].

Tabela 3.1 Przykładowe wartości baud rate error dla wybranych częstotliwości mikrokontrolera ATmega328P.

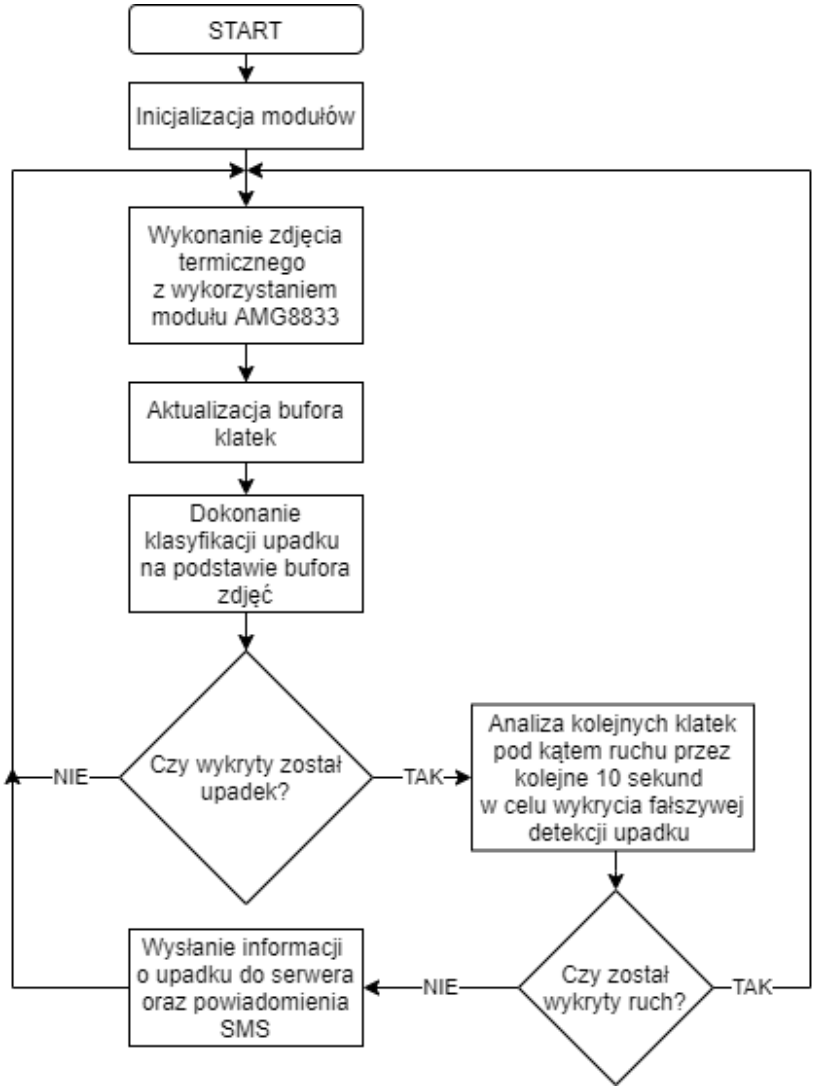
	16.0000 MHz		**18.4320 MHz**		**20.0000 MHz**	
Baud rate	**UBRR**	**Error[%]**	**UBRR**	**Error[%]**	**UBRR**	**Error[%]**
9600	103	0,2	119	0,0	129	0,2
57 600	16	2,1	19	0,0	21	1,4
115 200	8	-3,5	9	0,0	10	-1,4
230 400	3	8,5	4	0,0	4	8,5
250 000	3	0,0	4	-7,8	4	0,0

W niniejszym projekcie interfejs UART został wykorzystany do komunikacji pomiędzy platformą Arduino a komputerem osobistym w celu pobrania wykonanych zdjęć termicznych oraz utworzenia zbioru treningowego. Powyższe rozwiązanie pozwoliło na podgląd obrazu w czasie rzeczywistym oraz analizę wpływu czynników zewnętrznych i częstotliwości pracy kamery na dokonywane pomiary temperatury. Mikrokontroler ATmega328P domyślnie taktowany jest częstotliwością 16 MHz, w związku z czym na interfejsie UART została wybrana prędkość

transmisji o wartości 250 000 bit/s w celu wyeliminowania współczynnika *error baud rate*.

3.1.4. Schemat blokowy oprogramowania

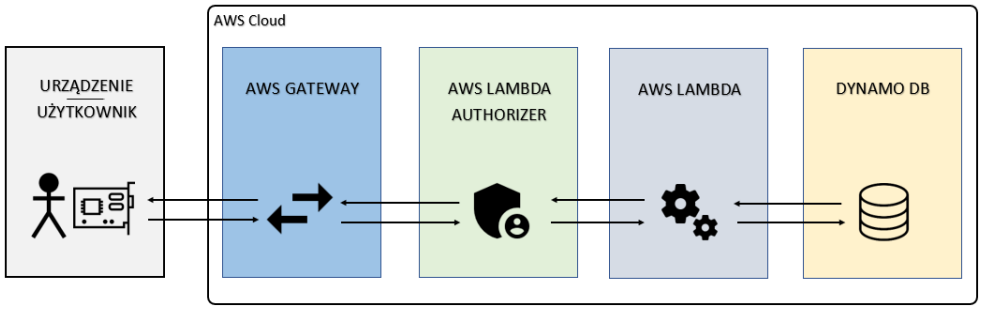
Rysunek 3.6 przedstawia schemat blokowy przyjętego oprogramowania układu do detekcji upadku, którego szczegółowa analiza została omówiona w rozdziale 3.3.



Rysunek 3.6 Schemat blokowy oprogramowania układu.

3.1.5. Architektura systemu chmurowego

W celu realizacji systemu chmurowego do zarządzania upadkami podopiecznych wykorzystana została platforma *AWS*, której architekturę przedstawiono na rysunku 3.7.



Rysunek 3.7 Architektura wewnętrzna systemu chmurowego

Do komunikacji pomiędzy urządzeniem a usługą wykorzystany został styl architektoniczny *REST API*, którego zapytania w pierwszej kolejności przetwarza moduł *AWS Gateway*. Każde żądanie wraz z parametrami zostaje przekierowane do dedykowanej implementacji, której wybór następuje na podstawie adresu URL. Przed docelową obsługą zapytania dokonywane jest uwierzytelnienie sesji przez moduł *AWS Lambda Authorizer* na podstawie poświadczeń logowania przekazanych w sekcji nagłówkowej. Podanie niepoprawnych danych skutkuje zwróceniem stosownego błędu zwrotnego poprzez moduł *AWS* Gateway. W przypadku uzyskania dostępu następuje przetworzenie żądania przez dedykowany moduł *AWS Lambda*, który posiada bezpośredni dostęp do bazy danych *Dynamo DB*. W ostatnim kroku wygenerowana odpowiedź zostaje zwrócona poprzez moduł *AWS* Gateway.

Dodatkowo architektura wykorzystuje model *serverless*, w którym odpowiedzialność za obsługę fizycznych urządzeń oraz ich konfigurację przejmuje platforma AWS. Wraz z rosnącą liczbą zapytań oraz obciążeniem następuje automatyczne skalowanie infrastruktury, zapewniając stały poziom świadczenia usług.

3.1.6. Struktury danych

Dokonanie detekcji upadków oraz przechowywanie ich historii wiąże się z koniecznością zaprojektowania i utworzenia specyficznych struktur danych. Do przechowywania informacji wykorzystana została baza danych DynamoDB, będąca częścią platformy chmurowej AWS, której opis znajduje się w rozdziale 2.3. Informacje o pacjentach przechowywane są w tabeli o nazwie „patients", a o opiekunach w „supervisors", do reprezentacji których został wykorzystany numer PESEL. Natomiast lista fizycznych urządzeń przechowywana jest tabeli o nazwie „devices", a wykryte upadki w strukturze „detections". Opis utworzonych struktur danych przedstawiają tabele 3.2, 3.3, 3.4 oraz 3.5.

Tabela 3.2 Struktura tabeli „patients".

Nazwa kolumny	Typ danych	Opis kolumny
pesel	liczba	Unikalny numer PESEL pacjenta
address	ciąg znaków	Adres pacjenta
first_name	ciąg znaków	Imię pacjenta
last_name	ciąg znaków	Nazwisko pacjenta
phone_number	liczba	Numer telefonu pacjenta
supervisors_pesel	tablica liczb	Lista numerów pesel opiekunów pacjenta

Tabela 3.3 Struktura tabeli „supervisors".

Nazwa kolumny	Typ danych	Opis kolumny
pesel	liczba	Unikalny numer PESEL opiekuna
address	ciąg znaków	Adres opiekuna
first_name	ciąg znaków	Imię opiekuna
last_name	ciąg znaków	Nazwisko opiekuna
password	ciąg znaków	Hasło do panelu administracyjnego w postaci SHA256
phone_number	liczba	Numer telefonu pacjenta
type	ciąg znaków	Typ konta (np. lekarz, członek rodziny)

Tabela 3.4 Struktura tabeli „devices”.

Nazwa kolumny	Typ danych	Opis kolumny
uuid	ciąg znaków	Unikalny numer identyfikujący urządzenie
model	ciąg znaków	Model platformy (np. Raspberry Pi 3B+)
password	ciąg znaków	Hasło dostępu do usługi chmurowej w postaci SHA256
patient_pesel	liczba	Numer PESEL pacjenta, do którego przypisane jest urządzenie
software_version	ciąg znaków	Numer wersji wgranego oprogramowania

Tabela 3.5 Struktura tabeli „detections”.

Nazwa kolumny	Typ danych	Opis kolumny
uuid	ciąg znaków	Unikalny numer identyfikujący upadek
frames	ciąg znaków	Kolejne klatki reprezentujące upadek osoby w postaci pikseli (pomiary temperatury) oddzielone znakiem średnika.
patient_pesel	liczba	Numer PESEL pacjenta, u którego wykryto upadek
timestamp	liczba	Data i godzina upadku zapisana w systemie POSIX (ang. *Portable Operating System Interface for UNIX*)

3.1.7. Architektura punktów dostępu

W celu wymiany informacji z platformą chmurową, utworzone zostały dedykowane punkty dostępu w oparciu o styl architektury oprogramowania REST. Jednym z kluczowych jest punkt o adresie */patients/{pesel}/detections* odpowiedzialny za pobieranie (metodą typu GET) oraz tworzenie (metodą typu POST) wpisów dotyczących wykrytych upadków. Wskazanie pacjenta następuje poprzez podanie numeru PESEL w adresie URL (ang. *Uniform Resource Locator*). W celu utworzenia

nowego wpisu reprezentującego wykryty upadek, należy podać w ciele zapytania kolejne klatki nagrania wykonanego z wykorzystaniem modułu AMG8833 w postaci pikseli (pomiary temperatury) oddzielonych znakiem średnika. Ponadto zgłoszenie upadku powoduje również wysłanie powiadomienia w formie wiadomości SMS do wszystkich opiekunów przypisanych do wskazanego pacjenta. Listing 3.1 przedstawia przykład zgłoszenia upadku do chmury obliczeniowej.

```
Adres URL (POST):
1  https://us-east-1.amazonaws.com/patients/9
2  5111712345

Ciało zapytania POST:
1  {
2    "frames": "… 22.25, 22.75, 22.00, 23.00 23.25;
3               … 22.75, 22.75, 23.50, 22.50, 22.50;
4               … 22.75, 22.50, 22.50, 23.25, 23.25;
5               … 23.50, 22.75, 22.50, 22.25, 22.50;
6               … 23.25, 23.50, 23.50, 23.75, 23.00;
7               … 22.50, 22.75, 23.25, 23.75, 23.50;
8               … 22.75, 23.25, 22.50, 22.50, 22.50;
9               … 23.25, 25.75, 23.75, 23.25, 24.00"
10 }

Kod odpowiedzi:
1  HTTP/1.1 200 OK
```

Listing 3.1 Przykład zapytania typu POST rejestrującego wykryty upadek w chmurze obliczeniowej.

Dodatkowo, w niniejszym projekcie utworzony został punkt dostępu o adresie */patients* (metoda typu GET) zwracający listę pacjentów. Poprzez opcjonalny parametr typu „query” o nazwie *supervisorPesel*, zasób przyjmuje numer PESEL opiekuna, co pozwala na zwrócenie jedynie podopiecznych przypisanych do wskazanej osoby. Listing 3.2 przedstawia przykład pobrania listy podopiecznych przypisanych do opiekuna o numerze PESEL 5111712345.

```
Adres URL (GET):
https://us-east-1.amazonaws.com/patients?pesel=
5111712345

Ciało odpowiedzi:
[{
  "last_name": "Kowalski",
  "first_name": "Jan",
  "address": "ul. Pszczyńska 1, 44-100 Gliwice",
},{
  "last_name": "Nowak",
  "first_name": "Marek",
  "address": "ul. Akademicka 1, 44-100 Gliwice",
}]
Kod odpowiedzi:
HTTP/1.1 200 OK
```

Listing 3.2 Przykład zapytania typu GET pobierającego listę podopiecznych przypisanych do opiekuna o numerze PESEL 5111712345.

Kluczowym, z punktu widzenia bezpieczeństwa, jest punkt dostępu o adresie */logon* (metoda typu POST) odpowiedzialny za autentykację oraz autoryzację zapytań do usługi chmurowej, którego przykład przedstawia listing 3.3. Poprzez parametr o nazwie *basic_auth* zlokalizowany w nagłówku zapytania, zasób przyjmuje dane logowania w postaci *login:hasło* zakodowane metodą base64.

```
Adres URL (GET):
https://us-east-1.amazonaws.com/logon

Nagłówek zapytania:
{
  "basic_auth": " dXNlcjpwYXNzd29yZA=="
}

Kod odpowiedzi:
HTTP/1.1 401 Unauthorized
```

Listing 3.3 Przykład zapytania logowania z użyciem niepoprawnych danych uwierzytelniających.

Dodatkowo, każde żądanie zakończone jest zwróceniem kodu odpowiedzi. W przypadku wykonania poprawnego zapytania zostaje zwrócony kod 200 OK. Podanie numeru PESEL, który nie występuje w bazie danych, powoduje zwrócenie kodu 404 Not Found, a podanie jakiegokolwiek parametru w niepoprawnym formacie lub typie (np. ciąg znaków zamiast wartości liczbowej) skutkuje otrzymaniem kodu 400 Bad Request. W przypadku punktów dostępu wymagających autoryzacji, poprawne zakończenie procesu autentykacji zwraca kod odpowiedzi 200 OK, w przeciwnym wypadku 401 Unauthorized.

3.2. Montaż elementów składowych

Ze względu na małą liczbę elementów składowych oraz zmian dokonywanych w układzie cyfrowym zrezygnowano z montażu bazującego na uniwersalnych płytkach stykowych, które cechują się niską estetyką połączeń, słabą wytrzymałością na czynniki zewnętrzne oraz występującą pojemnością pasożytniczą. Do realizacji niniejszego układu wykorzystana została płytka uniwersalna wraz z żeńskimi złączami łączącymi moduł kamery termowizyjnej z platformą Raspberry Pi lub Arduino, która została przedstawiona na rysunku 3.8.

Całość została przymocowana do podstawki o zmiennym kącie nachylenia, co umożliwia zmianę aktualnego pola widzenia kamery niezależnie od umiejscowienia układu w pomieszczeniu. Ponadto z tyłu konstrukcji znajduje się pomocnicza płytka stykowa oraz otwory montażowe co pozwala na podmianę platformy sprzętowej bez konieczności demontażu modułu AMG8833.

Rysunek 3.8 Montaż końcowy układu

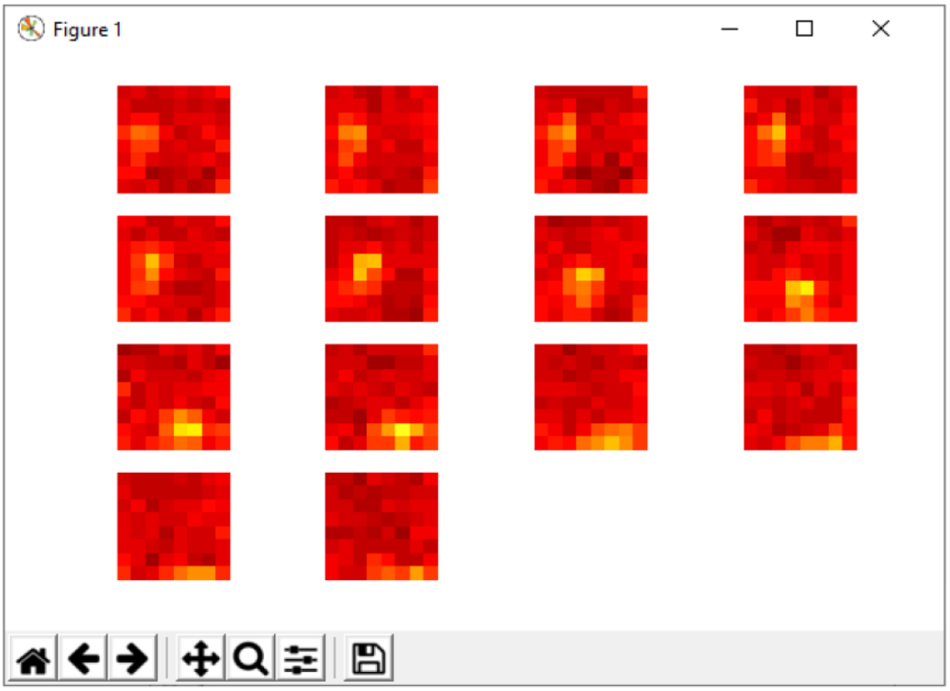
3.3. Realizacja przyjętego rozwiązania programowego

Niniejszy podrozdział zawiera opis przyjętego rozwiązania programowego układu elektronicznego oraz panelu opiekuna wraz z przykładami działania. Dodatkowo przedstawione zostały wykorzystane technologie, biblioteki oraz języki programowania.

3.3.1. Opis rozwiązania części elektronicznej

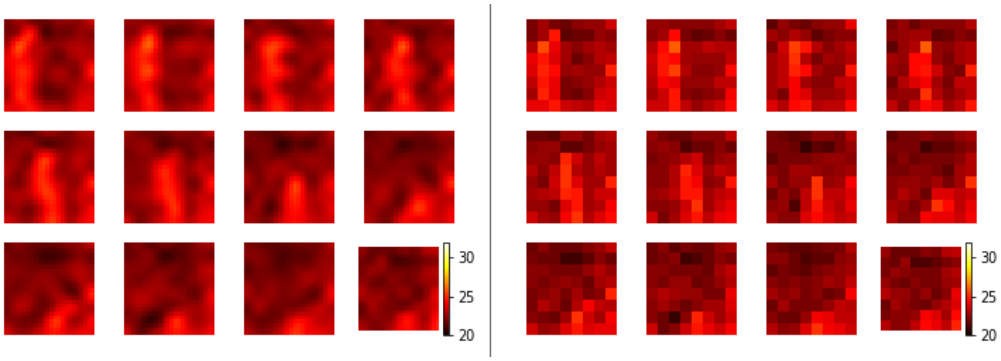
W początkowej fazie realizacji projektu konieczny okazał się podgląd aktualnego obrazu z kamery termowizyjnej w celu analizy kolejnych klatek pod kątem wykrycia upadku. Producent modułu AMG8833 nie oferuje żadnego gotowego rozwiązania, co wiąże się z koniecznością utworzenia własnej aplikacji do wizualizacji nagrań. W pierwszym etapie zaimplementowana została aplikacja odczytująca kolejne pomiary z matrycy

termicznej Panasonic GRID-EYE oraz zapisująca je do bufora w pamięci operacyjnej. Bufor posiada programistycznie ustawiony rozmiar, co pozwala na przechowywanie ustalonej liczby klatek wstecz. Ponadto aplikacja umożliwia zmienną częstotliwość odświeżania w przedziale 1-10 Hz. Przykładowy podgląd upadku osoby z buforem o rozmiarze 14 klatek oraz częstotliwością odświeżania 8 Hz przedstawia rysunek 3.9.



Rysunek 3.9 Przykład upadku osoby przy odświeżaniu 8 Hz oraz buforze o rozmiarze 14 klatek.

Wartą uwagi funkcjonalnością jest również możliwość interpolacji wyświetlanego obrazu do wskazanej rozdzielczości. Moduł Adafruit AMG8833 zwraca obraz termiczny w postaci macierzy zmierzonych temperatur o rozmiarze 8x8. Zmiana nominalnej rozdzielczości w znacznym stopniu polepsza aspekty wizualne oraz pozwala wyłuskać kontur ciała obserwowanej osoby, jednocześnie wspomagając analizę danych wejściowych. Rysunek 3.10 przedstawia różnicę w wizualizacji upadku z wykorzystaniem interpolacji do rozdzielczości 24x24 oraz bez (8x8).



Rysunek 3.10 Wpływ interpolacji na wizualizację klatek

Do wykonania interpolacji wykorzystany został moduł *ndimage* biblioteki SciPi dla języka Python. Bazuje on na interpolacji funkcjami sklejonymi, będącą metodą numeryczną polegającą na przybliżaniu nieznanej funkcji wielomianami niskiego stopnia.

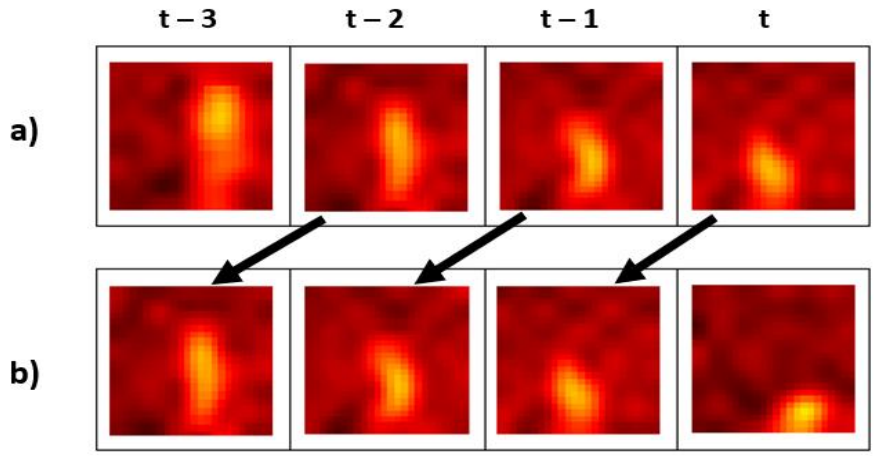
Ponadto zaobserwowany został wpływ interpolacji kolejnych klatek na proces uczenia klasyfikatorów, którego badania zawiera rozdział 4.3.

Dodatkowo, w celu zarządzania utworzonym zbiorem treningowym zaimplementowana została kolejna aplikacja z możliwością wyświetlenia wskazanego pliku o dowolnej liczbie zapisanych klatek wraz z interpolacją do wybranej rozdzielczości.

W następnej fazie realizacji projektu rozpoczęty został proces trenowania klasyfikatorów w oparciu o utworzony zbiór treningowy. W pierwszej kolejności wykorzystana została sieć neuronowa zbudowana z użyciem biblioteki programistycznej TensorFlow dla języka Python. Na potrzeby projektu zbudowane zostały sieci o różnych liczbach warstw ukrytych oraz dla różnych rozdzielczości obrazów termicznych. W celu analizy procesu uczenia wykorzystane zostało narzędzie TensorBoard, będące częścią biblioteki TensorFlow, służące do wizualizacji przebiegu treningu w postaci interaktywnych wykresów dla funkcji straty, dokładności oraz wielu innych metryk. Ponadto, w oparciu o bibliotekę *scikit-learn* dla języka Python, wykorzystany został również klasyfikator Bayesa, maszyna wektorów

nośnych SVM, drzewo decyzyjne, las losowy oraz kNN. Szczegółowy opis klasyfikatorów wraz z analizą i porównaniem zawiera rozdział 2.2 oraz 4.3.

W kolejnej fazie realizacji projektu została wykonana implementacja oprogramowania dla platformy Raspberry Pi oraz integracja z wyuczonymi modelami klasyfikatorów. Po uruchomieniu programu w pierwszej kolejności następuje nawiązanie komunikacji z modułem Adafruit AMG8833 poprzez interfejs I^2C. Po poprawnej inicjalizacji kamery zostaje wywołana nieskończona pętla programowa, rozpoczynająca się od wykonania zdjęcia termicznego oraz jego zapisu w pamięci operacyjnej. Aktualizacja bufora polega na nadpisaniu najstarszej zapisanej klatki poprzez przesunięcie wszystkich zdjęć w kierunku końca bufora oraz umieszczeniu najnowszego obrazu termicznego na samym początku. Przykład operacji aktualizacji przedstawia rysunek 3.11.



Rysunek 3.11 Przykład aktualizacji bufora w chwili t przechowującego 4 zdjęcia w kolejności chronologicznej od lewej strony. Ilustracja przedstawia stan bufora przed aktualizacją (a) oraz po przesunięciu (b).

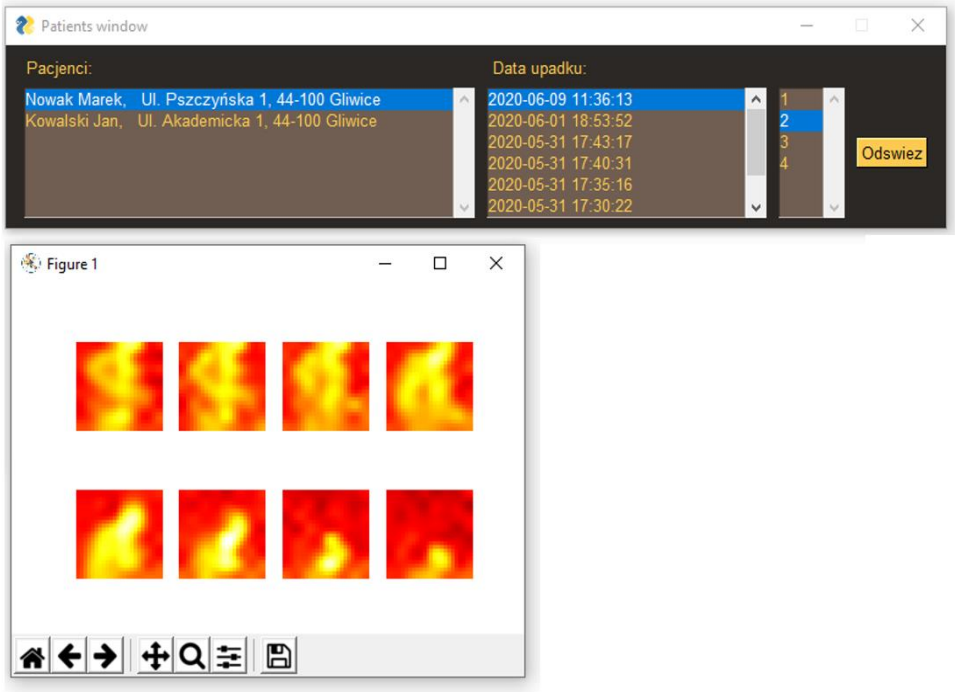
W kolejnym kroku wykonywana jest klasyfikacja upadku na podstawie obrazów termicznych znajdujących się w buforze. Wykryty potencjalny upadek zostaje zweryfikowany poprzez analizę pod kątem ruchu osoby, która upadła przez ustalony okres czasu. Takie rozwiązanie pozwala na

rozróżnienie gwałtownych ruchów w dół takich, jak schylenie się lub położenie na łóżku. Wykrycie ruchu skutkuje przerwaniem obsługi upadku oraz powrotem do analizy obrazów z kamery termowizyjnej. Natomiast w przypadku potwierdzenia upadku następuje zapis informacji o zdarzeniu w usłudze chmurowej wraz z klatkami znajdującymi się w buforze oraz wysłanie powiadomienia w postaci wiadomości SMS do opiekunów osoby podopiecznej. Szczegółowy schemat blokowy oprogramowania został przedstawiony w rozdziale 3.1.4.

Implementacja oprogramowania układu elektronicznego została zrealizowana w oparciu o zbiór pięciu reguł dobrych praktyk architektonicznych SOLID. Jedną z bardziej kluczowych zasad jest „Liskov substitution" mówiącą, iż w miejscu klasy bazowej można użyć dowolnej klasy pochodnej [10]. W niniejszym projekcie każda klasa modułu detekcji wykorzystuje jedynie jeden wybrany klasyfikator oraz implementuje metodę o nazwie „classify_frames". Takie rozwiązanie pozwala na dodanie nowego lub zmianę aktualnie używanego algorytmu inteligencji obliczeniowej bez konieczności modyfikacji kodu źródłowego.

3.3.2. Panel opiekuna

W celu podglądu, weryfikacji oraz podjęcia działania w związku z wykrytym upadkiem utworzony został panel do przeglądania upadów podopiecznych, nad którymi użytkownik sprawuje opiekę. Po poprawnym zalogowaniu z użyciem numeru PESEL oraz hasła wyświetlana zostaje lista podopiecznych wraz z datami ich upadków w porządku malejącym. Rysunek 3.12 przedstawia przykładowe użycie aplikacji. Wybranie pacjenta, daty upadku oraz numeru fragmentu skutkuje wyświetleniem nagrania, które może zostać wykorzystane do weryfikacji zdarzenia.



Rysunek 3.12 Przykład działania panelu opiekuna.

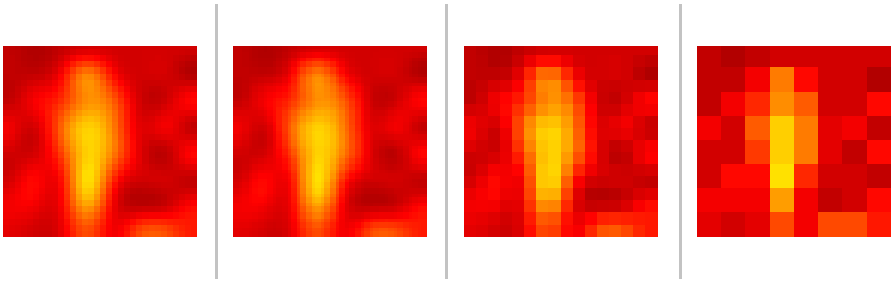
4. Badania

Rozdział przedstawia przeprowadzone badania wykorzystanych metod inteligencji obliczeniowej, wpływu danych wejściowych na proces nauki klasyfikatorów oraz porównanie użytych platform komputerowych.

4.1. Metodyka badań

W celu uzyskania klasyfikatora cechującego się wysoką skutecznością predykcji utworzony został zbiór treningowy z zachowaniem dużej różnorodności nagrań, którego szczegółowy opis zawiera rozdział 4.2.

Dodatkowo, zbadano wpływ interpolacji rozdzielczości zdjęć na jakość utworzonego modelu. Wstępna analiza wykazała, iż interpolacja do rozdzielczości powyżej 24x24 nie przynosi poprawy wizualnej, a w niektórych przypadkach powoduje deformację obrazu, co zostało przedstawione na rysunku 4.1.

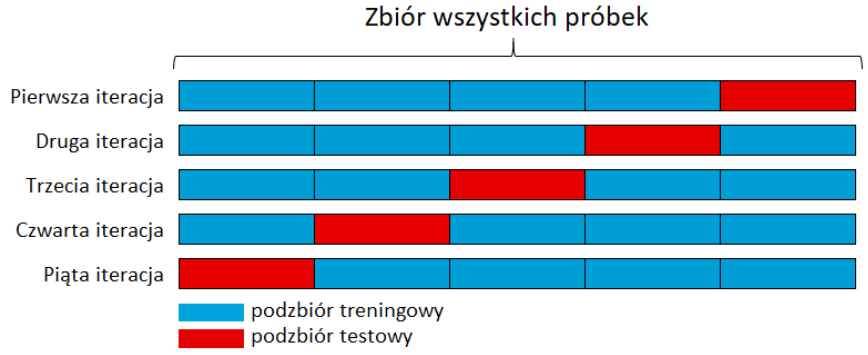


Rysunek 4.1 Przykład interpolacji obrazu termicznego do różnych rozdzielczości. Od lewej: 32x32, 24x24, 16x16 oraz 8x8.

W związku z powyższym w niniejszych badaniach poddane badaniom zostały kolejno rozdzielczości 24x24, 16x16 oraz 8x8.

Ponadto zastosowane zostały różne podejścia detekcji upadku, polegające na zmiennej liczbie przechowywanych klatek w ramach pojedynczego nagrania.

Zbiór nagrań został podzielony na dwa podzbiory: podzbiór treningowy, stanowiący 80% wszystkich próbek, oraz pozostała reszta będąca podzbiorem testowym. Dodatkowo, dla każdego klasyfikatora przeprowadzony został sprawdzian krzyżowy (ang. *cross-validation*) mający na celu zapobieganie zjawisku niedopasowania lub przetrenowania modelu. Polega on na powtórzeniu *k*-krotnie procesu uczenia maszynowego, gdzie *k* jest liczbą naturalną określającą liczbę równomiernych podziałów zbioru próbek. Dla każdego podzbioru został utworzony osobny model oraz oszacowana jest skuteczność. Ostateczny wynik został przyjęty jako średnia wszystkich uzyskanych wyników. Przykład podziału przedstawia rysunek 4.2. W niniejszych badaniach ustalona została wartość parametru *k=12*.



Rysunek 4.2 Przykładowy podział zbioru sprawdzianu krzyżowego dla parametru k = 5. Kolor niebieski reprezentuje podzbiór treningowy, czerwony testowy.

Dla binarnych klasyfikacji, do których należy również detekcja upadku, każdy wynik zaliczany jest do jednej spośród czterech kategorii przedstawionych w tabeli 4.1.

Tabela 4.1 Kategorie wyniku binarnej klasyfikacji.

		Klasa predykowana	
		pozytywna	**negatywna**
Klasa rzeczywista	**pozytywna**	prawdziwie pozytywna (TP)	fałszywie negatywna (FN)
	negatywna	fałszywie pozytywna (FP)	Prawdziwie negatywna (TN)

Dodatkowo, w celu opisu jakości utworzonego klasyfikatora, dla każdego utworzonego modelu wykonane zostały cztery miary testu diagnostycznego [8]:

- precyzja – określona wzorem 4.1, stosunek poprawnie sklasyfikowanych próbek pozytywnych do sumy wszystkich próbek sklasyfikowanych jako pozytywne:

$$precyzja\ klasyfikacji\ pozytywnej = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}}, \qquad \textbf{(4.1)}$$

- czułość – określona wzorem 4.2, stosunek poprawnie klasyfikowanych próbek pozytywnych do wszystkich próbek pozytywnych:

$$czułość = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}}, \qquad \textbf{(4.2)}$$

- specyficzność - określona wzorem 4.3, przedstawia stosunek poprawnie klasyfikowanych próbek negatywnych do wszystkich próbek negatywnych:

$$specyficzność = \frac{\boldsymbol{TN}}{\boldsymbol{TN + FP}}, \qquad \textbf{(4.3)}$$

- f1 score – określony wzorem 4.4, określa balans pomiędzy precyzją, a czułością:

$$f1 = \mathbf{2} * \frac{\boldsymbol{precyzja} * \boldsymbol{czułość}}{\boldsymbol{precyzja} + \boldsymbol{czułość}}. \quad \textbf{(4.4)}$$

W celu oceny utworzonego modelu utworzona została również krzywa ROC, będąca ilustracją związku pomiędzy specyficznością a czułością. Na osi odciętych umieszczone zostały wartości: (1 – specyficzność), a na osi rzędnych czułość. Dla każdego z możliwych punktów odcięcia obliczona została czułość oraz specyficzność. Uzyskane punkty umieszczone są na wykresie oraz zostały połączone linią. Im więcej różnych wartości badanego wskaźnika, tym gładsza jest uzyskana krzywa. W niniejszych badaniach przyjęte zostały równe koszty błędnych klasyfikacji, co sprawia, iż optymalnym punktem odcięcia jest punkt krzywej ROC znajdujący się najbliżej punktu o współrzędnych (0,1). Warte uwagi jest również podejście polegające na wyliczaniu pola pod wykresem krzywej ROC, oznaczanego jako AUC (ang. *area under curve*), i traktowanie go jako miarę dobroci i trafności danego modelu. Wartość wskaźnika AUC przyjmuje wartości z przedziału [0,1], im większa, tym wyższa jakość modelu [8]. W niniejszych badaniach dla każdej krzywej ROC został wyznaczony wskaźnik AUC.

W celu dodatkowej oceny klasyfikatorów, dla każdego modelu został obliczony współczynnik korelacji Matthews opisany wzorem 4.5, będący miarą jakości dla klasyfikacji binarnych. Przyjmuje on wartości z zakresu od -1 do 1, gdzie im większa wartość, tym wyższa jakość klasyfikacji:

$$MCC = \frac{\boldsymbol{TP} * \boldsymbol{TN} - \boldsymbol{FP} * \boldsymbol{FN}}{\sqrt{(\boldsymbol{TP} + \boldsymbol{FP}) * (\boldsymbol{FN} + \boldsymbol{TN}) * (\boldsymbol{FP} + \boldsymbol{TN}) * (\boldsymbol{TP} + \boldsymbol{FN})}}, \quad \textbf{(4.5)}$$

Poddana badaniom została również wydajność wykorzystanych algorytmów w celu analizy wybranych platform komputerowych takich, jak Raspberry Pi oraz Arduino. Dla każdego klasyfikatora został zmierzony czas

wymagany na dokonanie detekcji, zużycie pamięci operacyjnej oraz rozmiar zapisanego modelu.

4.2. Zbiory danych

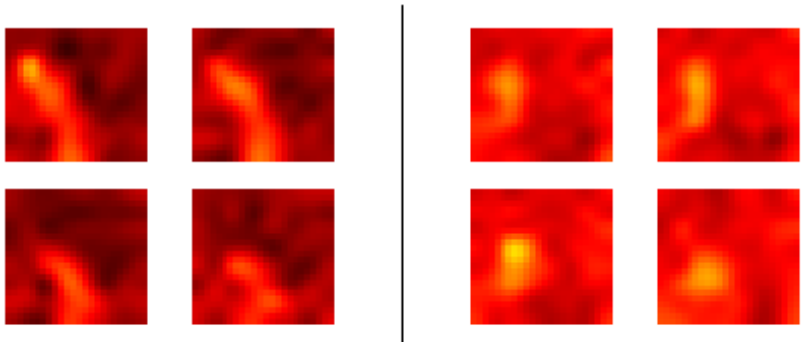
Moduł Adafruit AMG8833 wykonuje zdjęcia termiczne o rozdzielczości 8x8 pikseli. W sieci internetowej znaleźć można portale zawierające otwarte zbiory danych do celów uczenia maszynowego. Jednakże na dzień pisania niniejszej pracy żaden dostępny zasób nie zawierał nagrań upadków osoby z wykorzystaniem układu Panasonic GridEye. Na potrzeby projektu utworzone zostały 3 zbiory treningowe, każdy liczący 240 próbek, w tym 120 upadków oraz 120 zwyczajnych aktywności dziennych. Zbiory różnią się liczbą zapisanych klatek w ramach pojedynczego nagrania, których opis przedstawia tabela 4.2.

Tabela 4.2 Opis sposobów zapisu upadku osoby.

Liczba klatek	Opis sposobu zapisu upadku osoby
35	Zawiera zarówno moment upadku osoby oraz krótki okres czasu przed i po upadku.
14	Zawiera pełny moment upadku osoby oraz 3 klatki przed i po upadku.
8	Zawiera jedynie moment upadku.

W celu uzyskania wysokiej skuteczności klasyfikacji, proces tworzenia zbioru treningowego został przeprowadzony z zachowaniem dużej różnorodności nagrań. Uwzględniona została pora dnia, lokalizacja modułu termowizyjnego, oddalenie od kamery oraz ewentualne przeszkody zasłaniające częściowo upadek. Ponadto rozróżniony został typ upadku, jak np. upadek swobodny czy zsunięcie się z łóżka lub krzesła. Wartym uwagi jest wstępna analiza, która wykazała, iż najbliższe otoczenie modułu może

mieć bezpośredni wpływ na pomiary temperatur, nawet jeżeli obiekty te nie wchodzą bezpośrednio w kadr. Taką sytuację obrazuje rysunek 4.3. W związku z powyższym oba przypadki zostały uwzględnione podczas tworzenia zbioru treningowego.



Rysunek 4.3 Wpływ otoczenia na pomiary temperatury. Po prawej zdjęcie wykonane z obecnością monitora komputerowego w odległości 40cm od układu. Po lewej zdjęcie wykonane bez otoczenia urządzeń elektronicznych.

Moduł Panasonic GridEye dokonuje pomiarów z wykorzystaniem macierzy termicznej o rozdzielczości 8x8. W zależności od odległości od układu, ciało osoby ma przełożenie na różną liczbę pikseli. Wstępna analiza wykazała, iż praktycznie niezauważalny jest upadek osoby z odległości powyżej 4 metrów, gdyż w skrajnym przypadku jest to przesunięcie jedynie jednego piksela. Wynika to z ograniczeń sprzętowych oraz niedoskonałości modułu, ubioru osoby oraz temperatury pokojowej, która w większości przypadków jest zbliżona do temperatury zmierzonej przez ubranie. Dodatkowo, błąd pomiarowy wynoszący ± 2 stopnie Celsjusza jeszcze bardziej utrudnia analizę obrazu. W związku z powyższym utworzony zbiór treningowy zawiera jedynie nagrania upadków zarejestrowanych do 4 metrów od kamery termowizyjnej.

Każde pojedyncze nagranie upadku zlokalizowane jest w osobnym pliku tekstowym, którego nazwa zawiera informacje na temat rozdzielczości, liczbie zapisanych klatek, częstotliwości pracy kamery, klasy oraz daty nagrania. Każda klatka zapisana jest w osobnym wierszu, a pojedyncze

piksele, będące pomiarem temperatury w postaci liczby zmiennoprzecinkowej, oddzielone są znakiem średnika.

4.3. Wyniki

Niniejszy rozdział zawiera prezentację otrzymanych wyników dla modeli klasyfikatorów utworzonych w procesie uczenia maszynowego wraz z miarami jakości klasyfikacji. Dodatkowo, przedstawione zostało porównanie wybranych platform komputerowych pod kątem wydajności oraz dokonywania klasyfikacji upadku.

4.3.1. Maszyna wektorów nośnych

W pierwszej kolejności poddane analizie zostały następujące funkcje decyzyjne: liniowa, wielomianowa oraz Gaussa. Dla każdej funkcji został utworzony indywidualny klasyfikator z wykorzystaniem danych treningowych zawierających różną liczbę klatek w pojedynczym nagraniu. Łącznie analizie poddano 9 kombinacji, których wyniki zaobserwowanych miar modelu zostały przedstawione w tabelach 4.3, 4.4 oraz 4.5. Wartości występujące w nawiasach przedstawiają różnicę względem zbioru liczącego 8 klatek w ramach tej samej funkcji decyzyjnej.

Tabela 4.3 Miary modelu z wykorzystaniem funkcji liniowej oraz różnych zbiorów treningowych dla rozdzielczości 8x8.

	35 klatek	14 klatek	8 klatek
Precyzja	*0,90 (-0,02)*	*0,90 (-0,02)*	*0,92*
Czułość	*0,93 (-0,01)*	*0,93 (+0,01)*	*0,92*
Specyficzność	*0,88 (-0,03)*	*0,88 (-0,03)*	*0,91*
F1 score	*0,89 (-0,02)*	*0,89 (-0,02)*	*0,91*
MCC	*0,80 (-0,03)*	*0,82 (-0,01)*	*0,83*

Tabela 4.4 Miary modelu z wykorzystaniem funkcji wielomianowej oraz różnych zbiorów treningowych.

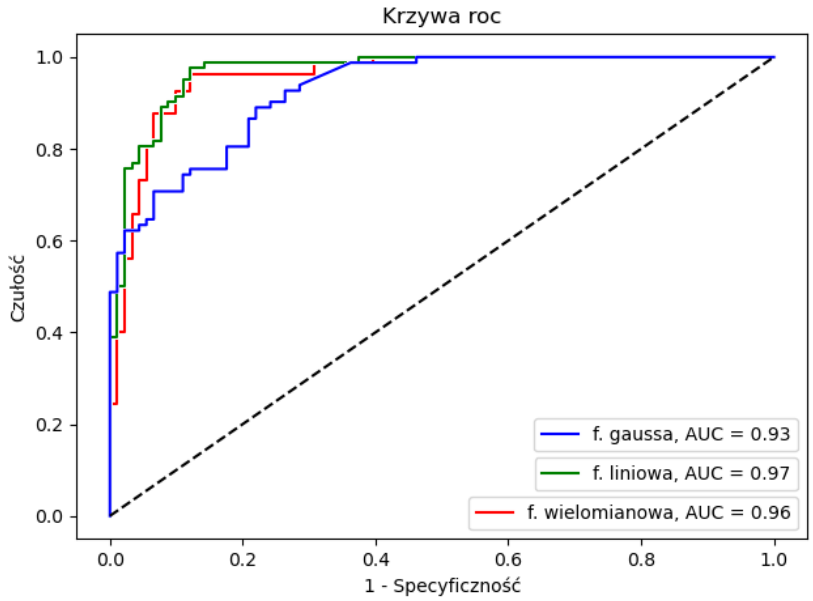
	35 klatek	14 klatek	8 klatek
Precyzja	*0,91 (-0,02)*	*0,92 (-0,01)*	*0,93*
Czułość	*0,94 (-0,03)*	*0,94 (+0,03)*	*0,91*
Specyficzność	*0,88 (-0,06)*	*0,90 (-0,04)*	*0,94*
F1 score	*0,90 (-0,03)*	*0,92 (-0,01)*	*0,93*
MCC	*0,82 (-0,04)*	*0,83 (-0,03)*	*0,86*

Tabela 4.5 Miary modelu z wykorzystaniem funkcji gausa oraz różnych zbiorów treningowych.

	35 klatek	14 klatek	8 klatek
Precyzja	*0,82 (-0,07)*	*0,82 (-0,07)*	*0,89*
Czułość	***0,59** (-0,21)*	***0,40** (-0,40)*	*0,80*
Specyficzność	***0,96** (+0,00)*	***1** (+0,04)*	*0,96*
F1 score	***0,71** (-0,16)*	***0,57** (-0,30)*	*0,87*
MCC	***0,58** (-0,20)*	***0,51** (-0,27)*	*0,78*

Wyniki wstępnie wykazały, iż funkcja liniowa oraz wielomianowa poprawnie rozwiązuje problem klasyfikacji upadku, pozwalając sklasyfikować poprawnie średnio 93% upadków oraz 90% pozostałych aktywności ruchowych. Natomiast wartym uwagi są miary czułości i specyficzności dla funkcji gausa, które wykazują bardzo dobrą klasyfikację normalnej aktywności ruchowej (wysoka specyficzność) oraz gorszą jakość detekcji upadku (niska czułość), w szczególności dla nagrań liczących 35 oraz 14 klatek. Dodatkowo analiza współczynnika MCC oraz f1 score potwierdza niską jakość wyuczonego modelu. Ponadto analiza krzywej ROC oraz wskaźnika AUC, które zostały przedstawione na rysunku 4.4, również potwierdza powyższe wnioski. Największe pole pod wykresem można

zaobserwować dla funkcji liniowej i wielomianowej, kolejno 0,97 oraz 0,96, natomiast najmniejsze dla gausa 0,93.



Rysunek 4.4 Krzywa ROC modelu SVM dla różnych funkcji decyzyjnych.

Wyniki zbiorów treningowych dla funkcji liniowej oraz wielomianowej są bardzo zbliżone, a różnice na poziomie ± 0,05 mogą być częściowo spowodowane każdorazowym mieszaniem zbioru treningowego. Taki zabieg ma na celu zapewnienie jeszcze większej różnorodności próbek w procesie uczenia maszynowego.

Do dalszych badań wykorzystany został zbiór składający się z nagrań liczących 8 klatek ze względu na niższą złożoność obliczeniową mikroprocesora oraz zajętość pamięci układu przy zachowaniu takiej samej jakości predykcji. W następnym kroku zbadany został wpływ interpolacji zdjęć termicznych na proces uczenia, których wyniki przedstawia tabela 4.6, 4.7 oraz 4.8. Poddane analizie zostały kolejno rozdzielczości 24x24, 16x16 oraz 8x8.

Tabela 4.6 Miary modelu SVM z wykorzystaniem funkcji liniowej oraz interpolacji.

	24x24	16x16	8x8
Precyzja	*0,92 (+0,00)*	*0,92 (+0,00)*	*0,92*
Czułość	*0,91 (-0,01)*	*0,90 (-0,02)*	*0,92*
Specyficzność	*0,91 (+0,00)*	*0,92 (+0,01)*	*0,91*
F1 score	*0,91 (+0,00)*	*0,91 (+0,00)*	*0,91*
MCC	*0,83 (+0,00)*	*0,83 (+0,00)*	*0,83*

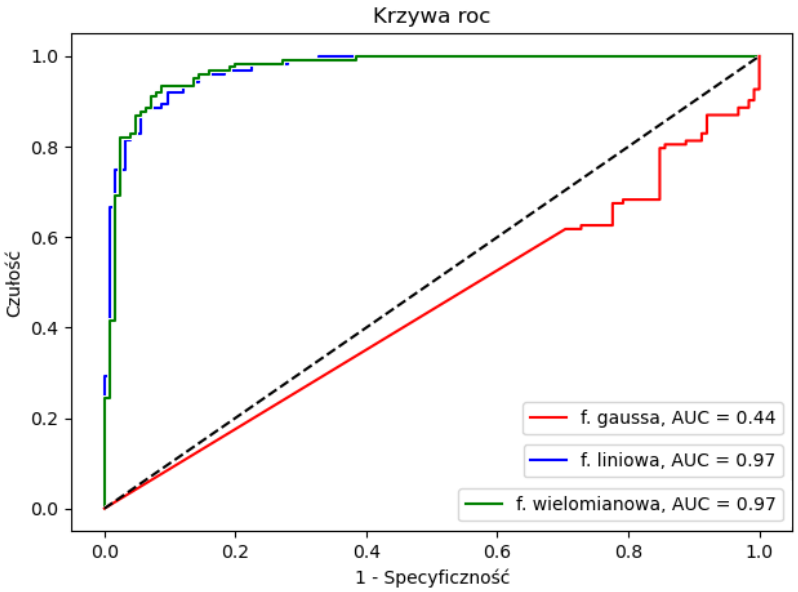
Tabela 4.7 Miary modelu SVM z wykorzystaniem funkcji wielomianowej oraz interpolacji.

	24x24	16x16	8x8
Precyzja	*0,92 (-0,01)*	*0,92 (-0,01)*	*0,93*
Czułość	*0,91 (+0,00)*	*0,92 (+0,01)*	*0,91*
Specyficzność	*0,92 (-0,02)*	*0,91 (-0,03)*	*0,94*
F1 score	*0,92 (-0,01)*	*0,91 (-0,02)*	*0,93*
MCC	*0,84 (-0,02)*	*0,83 (-0,03)*	*0,86*

Tabela 4.8 Miary modelu SVM z wykorzystaniem funkcji Gausa oraz interpolacji.

	24x24	16x16	8x8
Precyzja	***0,25** (+0,02)*	***0,78** (-0,11)*	*0,89*
Czułość	***0** (-0,80)*	***0,21** (-0,59)*	*0,80*
Specyficzność	***1** (+0,04)*	***1** (+0,04)*	*0,96*
F1 score	***0** (-0,87)*	***0,35** (-0,52)*	*0,87*
MCC	***0** (-0,78)*	***0,34** (-0,44)*	*0,78*

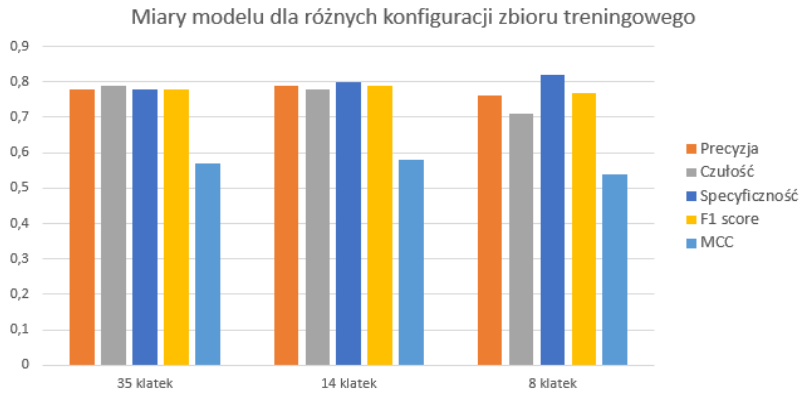
Dla funkcji liniowej oraz wielomianowej nie zaobserwowano wpływu interpolacji na jakość wyuczonego modelu. Natomiast wyniki funkcji Gausa wykazały pogorszenie modelu klasyfikatora, w szczególności dla rozdzielczości 24x24, gdzie czułość przyjmuje wartość 0. Powodem takiego zjawiska może być zbyt duże dopasowanie do jednej klasy, co oznacza, iż model nie jest w stanie poprawnie wykryć upadku. Ponadto analiza krzywej ROC, którą przedstawia rysunek 4.5, wykazuje, iż współczynnik AUC dla funkcji Gausa wynosi w przybliżeniu 0.5. Oznacza to, iż każda dokonywana predykcja jest losowa, a model nie potrafi dokonać podziału pomiędzy klasami.



Rysunek 4.5 Krzywa ROC modelu SVM dla różnych funkcji decyzyjnych i rozdzielczości 24x24.

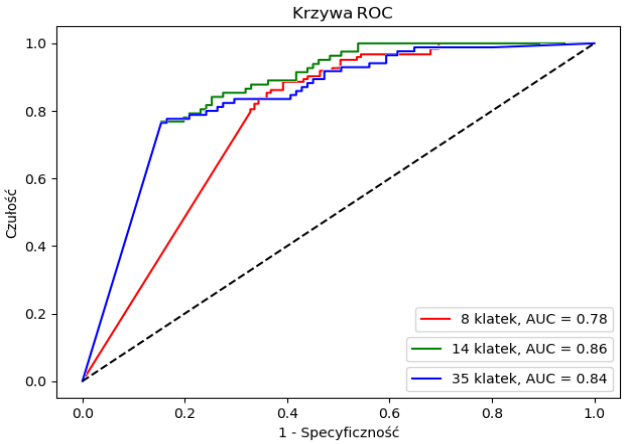
4.3.2. Naiwny klasyfikator Bayesowski

W pierwszej kolejności poddany analizie został wpływ liczby klatek w pojedynczym nagraniu na jakość wyuczonego modelu. Utworzone zostały trzy modele dla liczby klatek kolejno: 35, 14 oraz 8 oraz rozdzielczości 8x8. Uzyskane wyniki przedstawia rysunek 4.6.



Rysunek 4.6 Miary modelu klasyfikatora naiwnego Bayesa dla różnych konfiguracji zbioru treningowego o rozdzielczości 8x8 pikseli.

Wyniki wykazały, iż klasyfikator Bayesowski pozwala na rozwiązywanie problemów klasyfikacji upadku, jednakże obarczony jest zauważalnym marginesem błędu. Uzyskane wyniki różnią się między sobą w niewielkim stopniu, co może być spowodowane każdorazowym wymieszaniem zbioru treningowego. Jednakże niezależnie od przyjętej liczby klatek, w ramach eksperymentów uzyskano detekcję upadków na poziomie w zakresie 71-79%, co wykazują miary czułości. Dodatkowo, wskaźnik MCC oraz f1 score potwierdza powyższe wnioski. Utworzona została również krzywa ROC, przedstawiona na rysunku 4.7, której pole pod wykresem mieści się w przedziale od 0,78 do 0,86 w zależności od przyjętej liczby klatek.



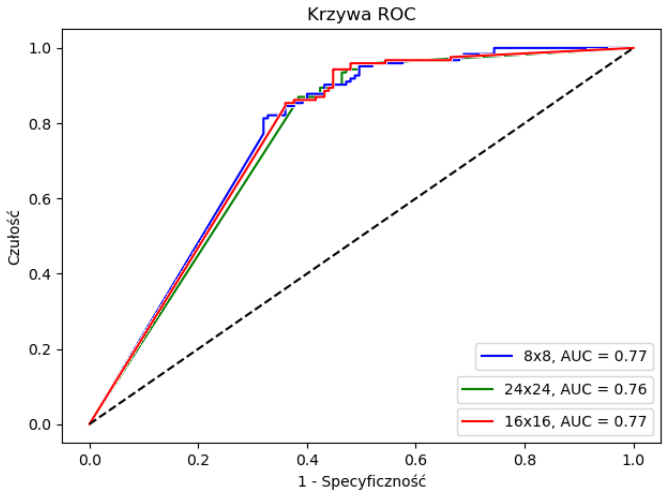
Rysunek 4.7 Krzywa ROC dla modeli klasyfikatora Bayesa dla różnej liczby klatek o rozdzielczości 8x8 pikseli.

W następnym kroku zbadany został wpływ interpolacji na jakość utworzonego modelu. Wykorzystany został zbiór składający się z nagrań liczących 8 klatek ze względu na niższą złożoność obliczeniową mikroprocesora oraz zajętość pamięci układu przy zachowaniu takiej samej jakości predykcji. Wyniki badań przedstawia tabela 4.9.

Tabela 4.9 Miary modelu klasyfikatora naiwnego Bayesa dla różnych rozdzielczości interpolacji.

	24x24	16x16	8x8
Precyzja	*0,73 (-0,03)*	*0,75 (-0,01)*	*0,76*
Czułość	*0,61 (-0,06)*	*0,62 (-0,05)*	*0,67*
Specyficzność	*0,84 (-0,01)*	*0,85 (+0,00)*	*0,85*
F1 score	*0,75 (-0,02)*	*0,76 (-0,01)*	*0,77*
MCC	*0,46 (-0,05)*	*0,49 (-0,02)*	*0,51*

Wyniki analizy nie wykazały znaczącego wpływu interpolacji na jakość wyuczonego modelu. Potwierdza to również krzywa ROC przedstawiona na rysunku 4.8, której pole pod wykresem AUC oscyluje w przedziale od 0.76 do 0.77.

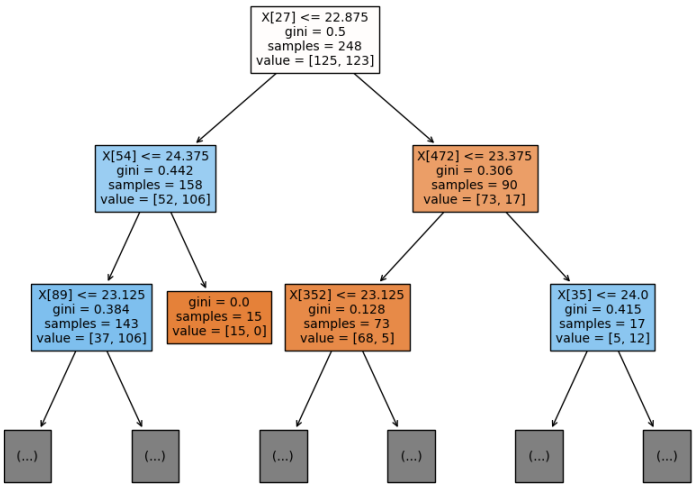


Rysunek 4.8 Krzywa ROC dla modeli klasyfikatora Bayesa dla różnych rozdzielczości interpolacji.

Algorytm naiwnego klasyfikatora Bayesa pozwala częściowo rozwiązać problem klasyfikacji upadku osoby, jednakże niezależnie od wybranej konfiguracji cechuje się zauważalnym marginesem błędu. W szczególności wartym uwagi jest miara czułości, która wykazała mniejszą zdolność modelu do poprawnej klasyfikacji upadku. W praktycznym zastosowaniu może prowadzić to do fałszywej detekcji zwyczajnej aktywności ruchowej, a tym samym braku udzielenia pomocy osobie, która upadła.

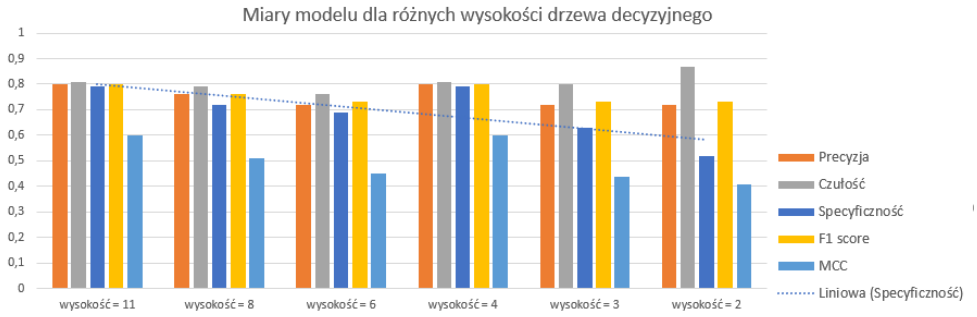
4.3.3. Drzewo decyzyjne

W pierwszym kroku poddany analizie został wpływ wysokości drzewa na jakość utworzonego modelu. Do przeprowadzenia uczenia maszynowego został wykorzystany algorytm CART (ang. *classification and regression trees*) cechujący się wyższą skutecznością i wydajnością w porównaniu do starszych wersji algorytmu. Wspiera dane ciągłe jak i dyskretne oraz potrafi obsłużyć brak danych cechy poprzez przewidywanie na podstawie innych cech. Ponadto w stosunku do algorytmu ID3 posiada mechanizmy zapobiegające zjawisku nadmiernego dopasowania oraz ogranicza zużycie pamięci, co jest kluczowym aspektem w przypadku platformy Arduino lub mikrokontrolerów [18]. Jako kryterium oceny jakości zmiennej losowej wybrany został współczynnik Giniego, który jest wykorzystywany przez algorytm CART. W początkowej iteracji nie została ustalona maksymalna głębokość drzewa, w wyniku czego utworzony model cechuje się dwudziestoma dwoma węzłami oraz jedenastoma poziomami, których fragment przedstawia rysunek 4.9.



Rysunek 4.9 Fragment utworzonej struktury drzewa decyzyjnego o wysokości równej 11.

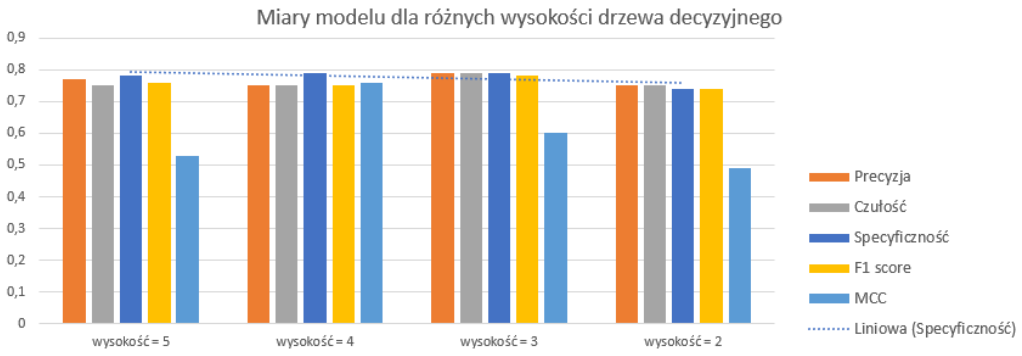
Każdy pojedynczy węzeł odpowiada za jeden odpowiadający piksel nagrania, który został wybrany poprzez ocenę jakości wskaźnikiem Giniego. Rysunek 4.10 przedstawia wybrane miary modelu dla różnych konfiguracji wysokości drzewa decyzyjnego.



Rysunek 4.10 Miary drzewa decyzyjnego modelu dla różnych wysokości oraz zbioru treningowego liczącego 8 klatek na nagranie o rozdzielczości 8x8 pikseli.

Wyniki badań pozwalają zauważyć, iż wraz z coraz większym ograniczeniem wysokości drzewa, następuje w niewielkim stopniu polepszenie zdolności detekcji upadku osoby kosztem znacznego pogorszenia klasyfikacji pozostałej aktywności ruchowej. Może być to

spowodowane metodą działania algorytmu, który w każdym węźle dokonuje decyzji na podstawie wybranego piksela. Ze względu na niską rozdzielczość 8x8 istnieje wysokie prawdopodobieństwo, iż któryś piksel będzie w większości upadków wykazywać wyższą temperaturę, gdyż w jego miejscu będzie występować ciało osoby (np. w dolnych fragmentach zdjęć termicznych). W przypadku wyboru takich pikseli w wyższych warstwach struktury może dochodzić do fałszywych detekcji upadku, gdyż decydujące są jedynie wybrane piksele zamiast całego nagrania. Wniosek potwierdza również brak znaczącego wzrostu liczby węzłów przy większej wysokości drzewa oraz brak możliwości utworzenia struktury wyższej niż 11 poziomów. Ponadto analiza wizualizacji utworzonych struktur drzew potwierdza, iż w węzłach początkowych poziomów za każdym razem decyzja podejmowana jest na podstawie tych samych pikseli, niezależnie od wysokości drzewa. Dodatkowo, wysunięty wniosek potwierdzają miary modeli, przedstawione na rysunku 4.11, które zostały utworzone na podstawie zbioru treningowego liczącego 35 klatek o rozdzielczości 8x8. Każde nagranie zawiera dodatkowe klatki zlokalizowane zaraz przed i po upadku.



Rysunek 4.11 Miary modelu drzewa decyzyjnego różnych wysokości oraz zbioru treningowego liczącego 35 klatek na nagranie o rozdzielczości 8x8 pikseli.

Pomimo dużo większej liczby klatek, w procesie uczenia maszynowego nie udało się uzyskać drzewa o wysokości większej niż 5 poziomów oraz 12 liściach. Natomiast część dodatkowych klatek została wykorzystana

w górnych poziomach drzewa, co poskutkowało zwiększeniem zdolności poprawnej klasyfikacji obu klas. Dodatkowo, zbadano wpływ interpolacji na jakość utworzonego modelu, której wyniki przedstawia tabela 4.10.

Tabela 4.10 Miary modelu drzewa decyzyjnego dla interpolacji do rozdzielczości 24x24.

	Wysokość			
	5 poziomów	**4 poziomy**	**3 poziomy**	**2 poziomy**
Precyzja	*0,80 (+0,04)*	*0,76 (-0,04)*	*0,77 (+0,05)*	*0,74 (+0,02)*
Czułość	*0,82 (+0,01)*	*0,77 (-0,04)*	*0,82 (+0,02)*	*0,82 (+0,05)*
Specyficzność	*0,77 (+0,06)*	*0,74 (-0,05)*	*0,70 (+0,07)*	*0,65 (+0,13)*
F1 score	*0,80 (+0,04)*	*0,76 (-0,04)*	*0,77 (+0,04)*	*0,75 (+0,02)*
MCC	*0,59 (+0,07)*	*0,52 (-0,08)*	*0,53 (+0,09)*	*0,48 (+0,07)*

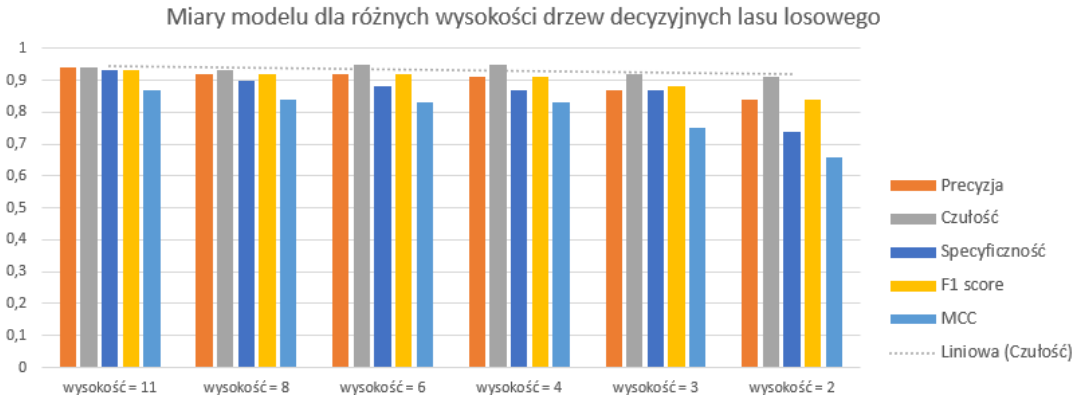
Po zastosowaniu interpolacji do wyższej rozdzielczości zaobserwowano nieznaczną poprawę miar jakości modelu. Podobnie jak w przypadku różnych konfiguracji zbiorów treningowych, nie udało się utworzyć drzewa o wysokości większej niż 5 warstw oraz 15 liściach. Proces uczenia maszynowego wykonano również dla rozdzielczości 16x16, jednakże zaobserwowane zostały zbliżone miary modelu.

Algorytm drzew decyzyjnych pozwala rozwiązać problem klasyfikacji upadku osoby, jednakże w wybranych konfiguracjach utworzony model cechował się zauważalnym marginesem błędu. W szczególności warta uwagi jest miara specyficzności, która wykazała mniejszą zdolność modelu do poprawnej klasyfikacji zwykłej aktywności ruchowej. W praktycznym zastosowaniu może prowadzić to do fałszywego wykrywania upadków.

4.3.4. Las losowy

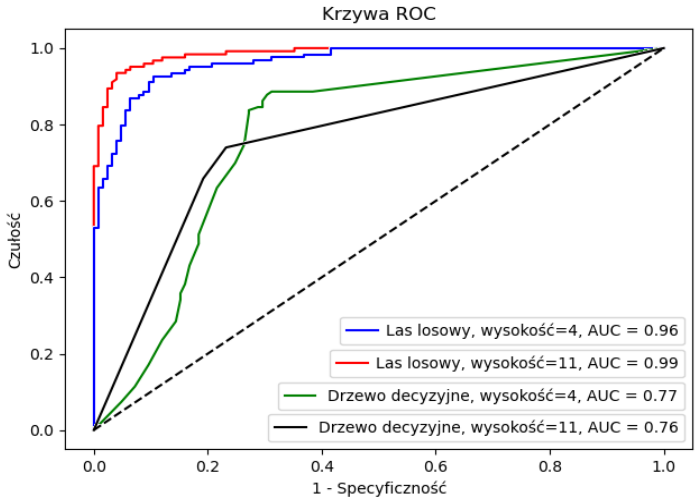
Las losowy jest zbiorem klasyfikatorów, w którym każdy pojedynczy klasyfikator jest drzewem decyzyjnym, a klasyfikacja następuje w drodze głosowania. W związku z powyższym w pierwszej kolejności

przeprowadzona została seria badań dla takich samych konfiguracji jak w rozdziale 4.3.3, których wyniki przedstawia rysunek 4.12.



Rysunek 4.12 Miary modelu lasu losowego liczącego 100 klasyfikatorów dla różnych wysokości oraz zbioru treningowego liczącego 8 klatek na nagranie o rozdzielczości 8x8 pikseli.

W procesie uczenia maszynowego wykorzystany został zbiór drzew decyzyjnych liczący 100 klasyfikatorów. Wyniki badań pozwalają zaobserwować ciekawą zależność polegającą na niższej jakości klasyfikacji w przypadku wykorzystania pojedynczego klasyfikatora oraz wysokich miarach jakości modelu dla tego samego algorytmu, lecz użytego w postaci zbioru klasyfikatorów oraz z użyciem tego samego zbioru treningowego. Przeprowadzenie uczenia maszynowego bez ograniczenia głębokości skutkowało utworzeniem modelu zawierającego drzewa decyzyjne składające się z 11 poziomów oraz 25 liści. Dla pozostałych konfiguracji liczba liści również była zbliżona do struktur utworzonych w rozdziale 4.3.3. Oznacza to, iż podejmowanie decyzji na drodze głosowania pozwala w znaczący sposób poprawić zdolność rozwiązywania tego samego problemu klasyfikacji przy zachowaniu zbliżonej struktury. Ponadto sporządzone zostały krzywe ROC dla wybranych modeli, które przedstawia rysunek 4.13.



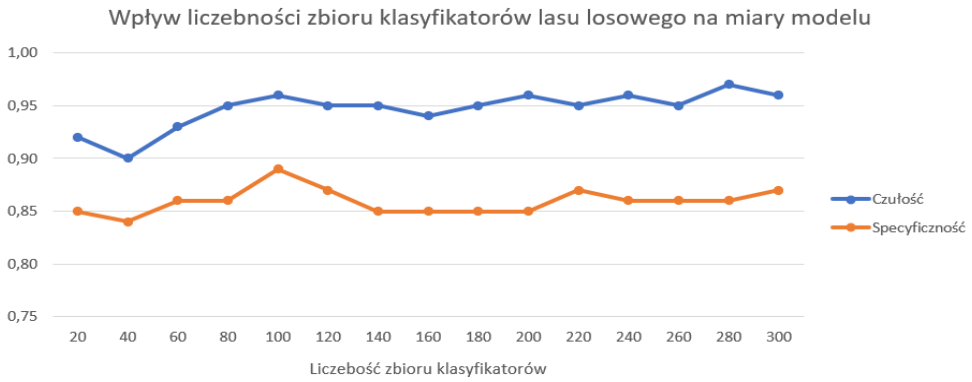
Rysunek 4.13 Krzywa ROC dla różnych modeli lasu losowego oraz drzew decyzyjnych o rozdzielczości 8x8 pikseli.

Krzywe ROC pozawalają zaobserwować znaczącą różnicę w zdolności do wykrywania upadku osoby pomiędzy drzewem decyzyjnym a lasem losowym. Dodatkowo został zbadany wpływ interpolacji na jakość dokonywanych klasyfikacji. Tabela 4.11 przedstawia porównanie miar rozdzielczości 24x24 dla lasu losowego oraz drzewa decyzyjnego. Wartości występujące w nawiasach przedstawiają różnice względem modeli drzew decyzyjnych dla takiej samej konfiguracji procesu uczenia maszynowego.

Tabela 4.11 Miary modelu lasu losowego dla zbioru treningowego liczącego 8 klatek na nagranie oraz interpolacji do rozdzielczości 24x24 pikseli. Wartości występujące w nawiasach przedstawiają różnice względem modeli drzew decyzyjnych.

	Wysokość			
	5 poziomów	**4 poziomy**	**3 poziomy**	**2 poziomy**
Precyzja	*0,93 (+0,13)*	*0,92 (+0,16)*	*0,90 (+0,13)*	*0,85 (+0,11)*
Czułość	*0,96 (+0,14)*	*0,96 (+0,19)*	*0,96 (+0,14)*	*0,93 (+0,11)*
Specyficzność	*0,90 (+0,13)*	*0,87 (+0,13)*	*0,84 (+0,14)*	*0,74 (+0,09)*
F1 score	*0,93 (+0,13)*	*0,92 (+0,16)*	*0,90 (+0,13)*	*0,85 (+0,10)*
MCC	*0,86 (+0,27)*	*0,83 (+0,31)*	*0,80 (+0,27)*	*0,80 (+0,20)*

Badania nie wykazały znaczącego wpływu interpolacji na jakość utworzonego modelu lasu losowego. Natomiast wyniki pozwoliły zaobserwować znaczącą poprawę względem algorytmu drzew decyzyjnych przy zachowaniu takiej samej konfiguracji zbioru treningowego oraz struktury drzewa. Potwierdza to wniosek, iż dokonywanie wyboru na drodze głosowania pozwala w znaczący sposób poprawić zdolność klasyfikacji. Dodatkowo zbadany został wpływ liczebności zbioru klasyfikatorów na jakość dokonywanych detekcji, której wybrane wyniki przedstawia rysunek 4.14.

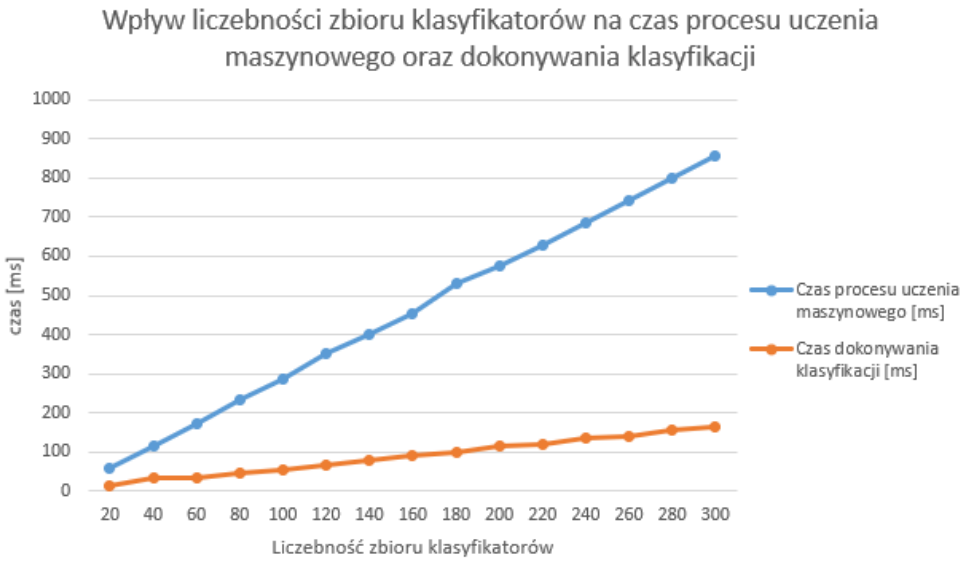


Rysunek 4.14 Wpływ liczebności zbioru klasyfikatorów lasu losowego na miary modelu dla zbioru treningowego liczącego 8 klatek na nagranie o rozdzielczości 8x8 pikseli.

Badania wykazały, iż najlepsze miary utworzonych modeli otrzymano dla zbioru klasyfikatorów o liczebności większej niż 100. Ponadto proces uczenia maszynowego został przeprowadzony dla pozostałych konfiguracji zbioru treningowego oraz interpolacji, jednakże uzyskane wyniki pozwoliły zaobserwować identyczną tendencję.

Dodatkowo, analizie poddano wpływ liczebności zbioru klasyfikatorów na czas procesu uczenia maszynowego oraz dokonywania pojedynczej klasyfikacji, której wyniki zostały przedstawione na rysunku 4.15. Proces uczenia przeprowadzono na procesorze AMD Ryzen 2700X pracującym z częstotliwością 4.2 GHz, natomiast klasyfikację na docelowej platformie Raspberry Pi 3B+ z procesorem ARM Cortex-A53 1.4 GHz. Dla obu

kategorii odnotowano liniowy wzrost czasu względem liczebności zbioru klasyfikatorów. Ponadto zaobserwowano niekorzystny wpływ współbieżności spowodowany małą liczebnością zbioru treningowego oraz niską rozdzielczością zdjęć, co w rezultacie przekłada się na dłuższy czas inicjalizacji nowego wątku niż proces uczenia lub klasyfikacji. W związku z powyższym za każdym razem wykorzystany został jedynie jeden rdzeń procesora. Dodatkowo, dla liczebności zbioru powyżej 260 zaobserwowano czas klasyfikacji na poziomie powyżej 150ms, co uniemożliwia obserwację z częstotliwością 8 Hz, która została ustalona w niniejszym projekcie.



Rysunek 4.15 Wpływ liczebności zbioru klasyfikatorów na czas procesu uczenia maszynowego oraz dokonywania klasyfikacji z wykorzystaniem jednego rdzenia procesora oraz zbioru treningowego liczącego 8 klatek na nagranie o rozdzielczości 8x8 pikseli.

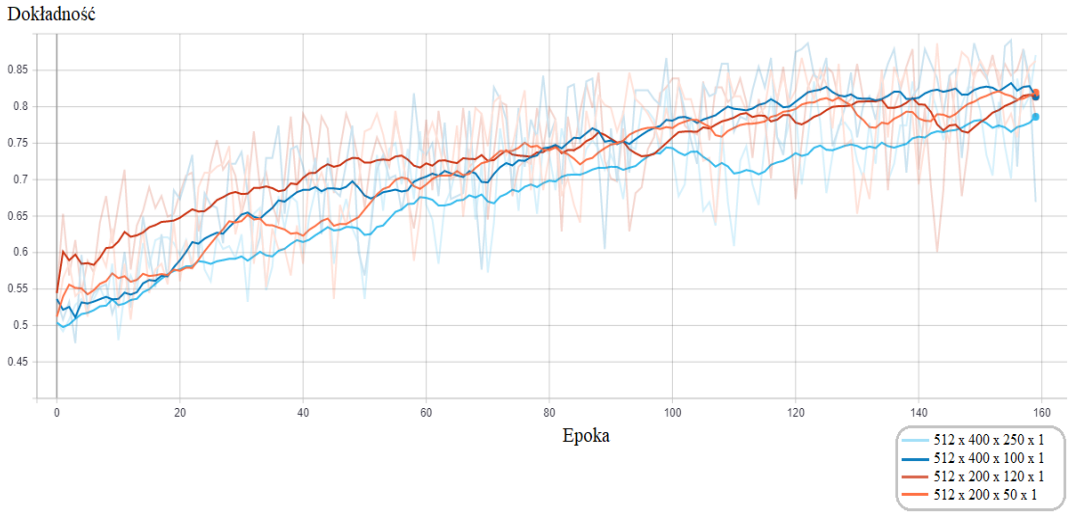
4.3.5. Sztuczne sieci neuronowe

Ze względu na duża liczbę kombinacji parametrów sztucznych sieci neuronowych, niniejsze badania zostały podzielone na dwie części. W pierwszej kolejności poddany analizie został wpływ warstw ukrytych

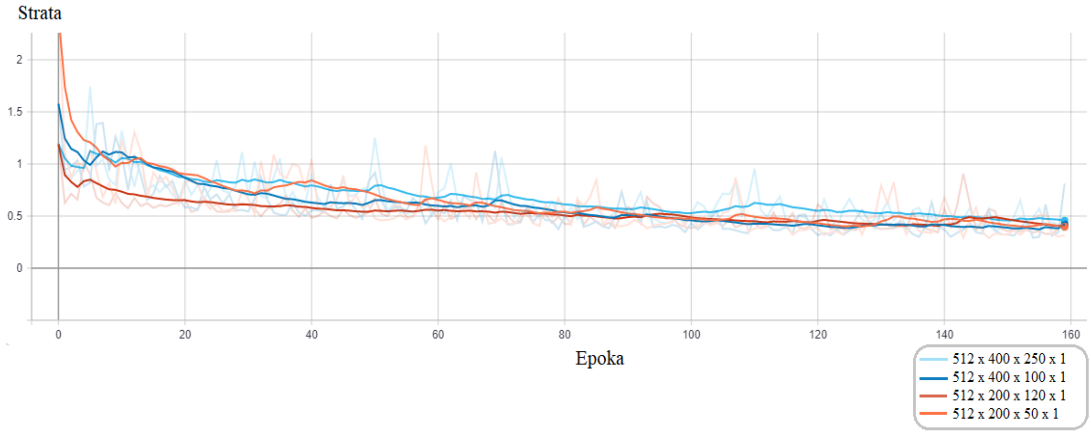
wraz z różnymi liczbami neuronów. Uczenie zostało przeprowadzone dla zbioru, w którym przypada 8 klatek na nagranie, co jest równoznaczne z liczbą 512 neuronów (8 klatek x 8 pikseli x 8 pikseli) w warstwie wejściowej. Rysunek 4.16 oraz 4.17 przedstawia miarę dokładności oraz funkcji straty dla sieci z dwiema ukrytymi warstwami dla konfiguracji przedstawionej w tabeli 4.12. Niniejsze wartości zostały ustalone w celu przeprowadzenia procesu uczenia maszynowego na zróżnicowanych strukturach sieci oraz zbadania ich wpływu na jakość utworzonego modelu.

Tabela 4.12 Konfiguracja uczenia maszynowego dla dwóch warstw ukrytych.

Numer eksperymentu	Liczba neuronów pierwszej warstwy ukrytej	Liczba neuronów drugiej warstwy ukrytej
1	400	250
2	400	100
3	200	120
4	200	50

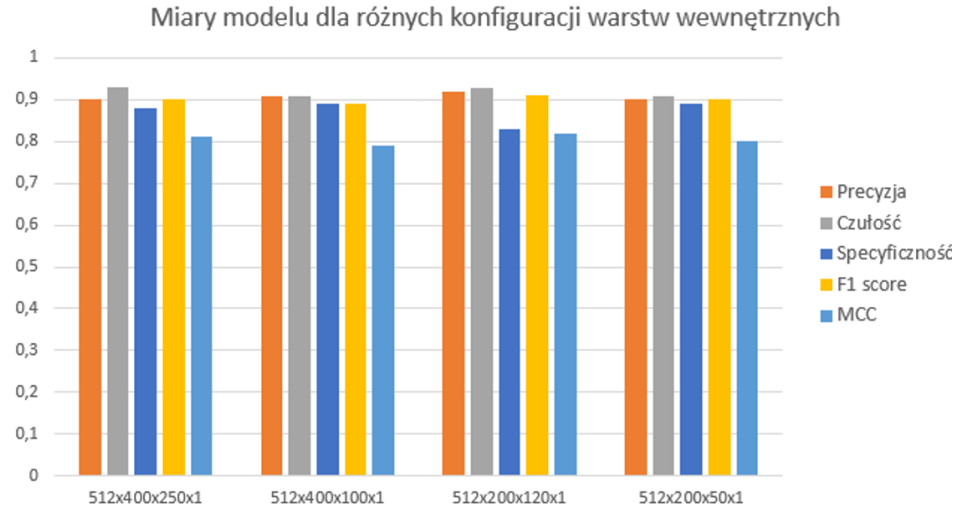


Rysunek 4.16 Przebieg uczenia maszynowego dla dwóch warstw ukrytych.



Rysunek 4.17 Przebieg funkcji straty uczenia maszynowego dla dwóch warstw ukrytych.

Proces uczenia maszynowego został przerwany po zakończeniu 160 epoki w celu uniknięcia zjawiska nadmiernego dopasowania. Wykres miary dokładności oraz funkcji straty pozwala stwierdzić, iż sieci neuronowe pozwalają rozwiązać problem detekcji upadku. Wartym uwagi jest przypadek z największą liczbą neuronów, w którym zaobserwowano niższą dokładność oraz większe wartości funkcji straty. Natomiast najbardziej stabilną konfiguracją w procesie uczenia jest struktura 512x400x100x1, w której oscylacje miar są wyraźnie najmniejsze. W celu dodatkowej oceny jakości utworzonego modelu wyznaczone zostały metryki, które przedstawia rysunek 4.18. Wyniki dla różnych konfiguracji są zbliżone, a różnice mogą być spowodowane stochastycznym procesem uczenia maszynowego ze względu na losowe wartości wag w momencie rozpoczęcia uczenia maszynowego. Dodatkowo, na wykresie miary dokładności oraz funkcji straty można zaobserwować wahania wartości, w związku z czym numer epoki, na którym został zakończony proces uczenia, również może mieć częściowy wpływ na jakość utworzonego modelu.



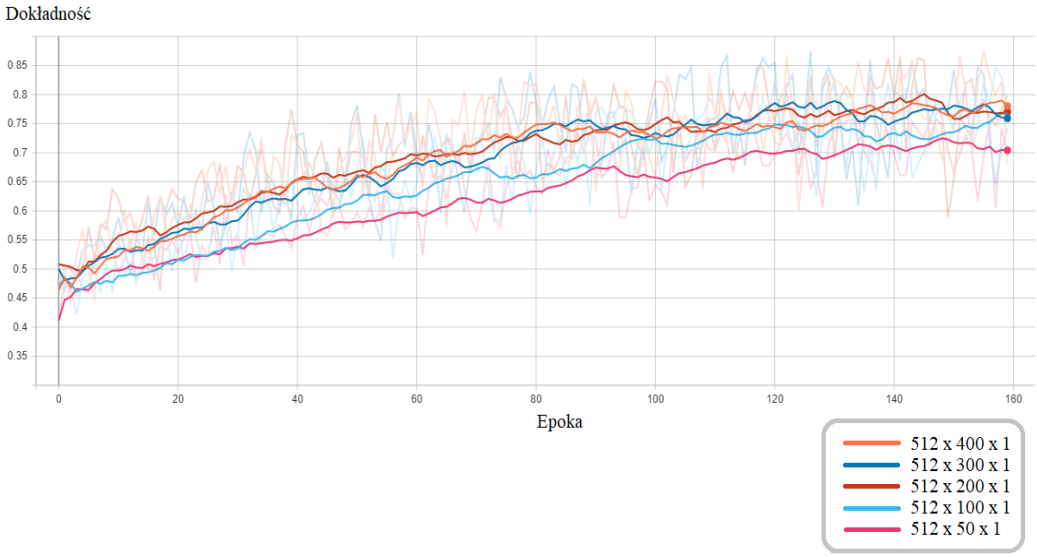
Rysunek 4.18 Wykres miar modelu sieci sztucznych neuronowych dla różnych konfiguracji warstw wewnętrznych.

Dodatkowo w tabeli 4.13 przedstawione zostały numeryczne wartości utworzonych struktur w procesie uczenia maszynowego.

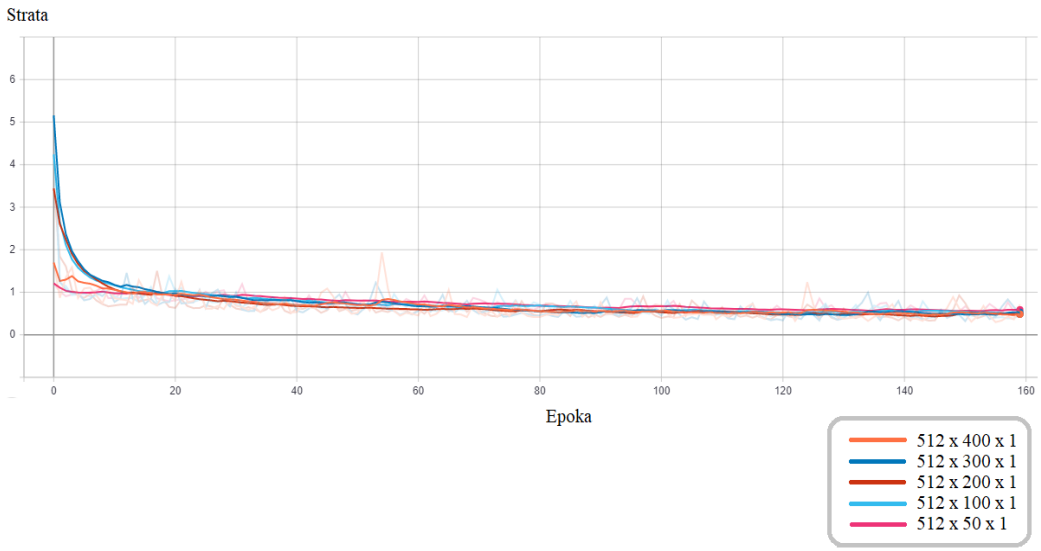
Tabela 4.13 *Miary modelu sieci sztucznych neuronowych dla różnych konfiguracji warstw wewnętrznych.*

	512x400x250x1	512x400x100x1	512x200x120x1	512x400x50x1
Precyzja	*0,9*	*0,91*	0,92	*0,90*
Czułość	*0,93*	*0,91*	0,93	*0,91*
Specyficzność	*0,88*	*0,89*	0,83	*0,89*
F1 score	*0,90*	*0,89*	0,91	*0,90*
MCC	*0,81*	*0,79*	0,82	*0,80*

W następnym kroku poddana analizie została sieć z jedną warstwą ukrytą o liczbie neuronów kolejno 400, 300, 200, 100 oraz 50. Uczenie ponownie zostało przeprowadzone dla zbioru, w którym przypada 8 klatek na nagranie o rozdzielczości 8x8 pikseli, co jest równoznaczne liczbą 512 neuronów w warstwie wejściowej. Wykresy dokładności oraz funkcji straty dla uczenia maszynowego przedstawia rysunek 4.19 oraz 4.20.



Rysunek 4.19 Przebieg uczenia maszynowego dla jednej warstwy ukrytej.

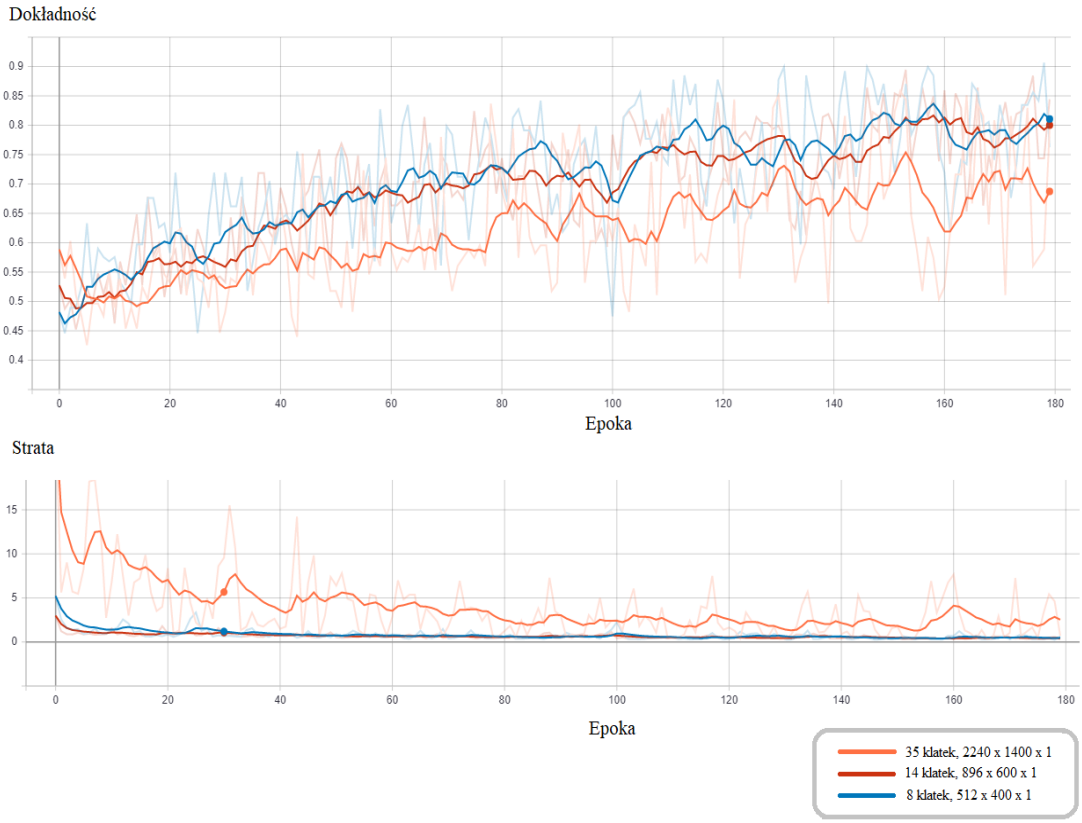


Rysunek 4.20 Przebieg funkcji straty uczenia maszynowego dla jednej warstwy ukrytej.

Podobnie jak w przypadku sieci o strukturze wielowarstwowej proces uczenia maszynowego został przerwany po 160 epoce, gdyż zaobserwowano brak wyraźnej poprawy dokładności oraz funkcji straty. Utworzony model poprawnie rozwiązuje problem detekcji upadku, jednakże najlepsze miary dokładności zaobserwowano dla konfiguracji z większą liczbą neuronów

w warstwie ukrytej. Przykładowo dla struktury zawierającej 50 oraz 100 neuronów można zaobserwować niższą dokładność uzyskaną w procesie uczenia maszynowego.

W następnym kroku poddane analizie zostały różne konfiguracje zbioru treningowego zawierające kolejno 35, 14 oraz 8 klatek na nagranie o rozdzielczości 8x8 pikseli. Ze względu na dużą liczbę kombinacji parametrów wejściowych oraz znikome różnice miar jakości pomiędzy utworzonymi modelami omówione zostały jedynie wybrane kombinacje. Rysunek 4.21 przedstawia przebieg uczenia maszynowego dla struktury jednowarstwowej. W każdym procesie uczenia maszynowego zaobserwowano wahania oraz brak znaczącej poprawy miar modelu po przekroczeniu 120 epoki.



Rysunek 4.21 Przebieg uczenia maszynowego dla struktury jednowarstwowej oraz różnych konfiguracji zbioru treningowego.

Analiza wykazała, iż liczba klatek przypadających na pojedyncze nagranie upadku ma wpływ na jakość utworzonego modelu. Dla zbioru składającego się z 35 klatek zaobserwowano niższą wartość dokładności oraz większą miarę funkcji straty. Obserwacje pozwalają stwierdzić, iż im większa liczba klatek tuż przed i zaraz po upadku, tym niższą jakością cechuje się utworzony model. Proces uczenia maszynowego został przeprowadzony dla różnych liczb neuronów w warstwie ukrytej, jednakże metryki dla każdego modelu są zbliżone. Dodatkowo przeprowadzone zostały badania dla struktury składającej się z dwóch warstw ukrytych, w których zaobserwowano podobne zjawisko. Najlepsze wyniki zostały uzyskane dla konfiguracji 8 klatek, w której pojedyncze nagranie zawiera jedynie sam moment upadku. Obserwacje potwierdzają również zmierzone metryki oceny jakości modelu, przedstawione w tabeli 4.14. Ponadto po przekroczeniu 150 epoki zaobserwować można brak wyraźniej poprawy, co może skutkować wystąpieniem zjawiska nadmiernego dopasowania.

Tabela 4.14 Miary modelu sztucznych sieci neuronowych dla różnych konfiguracji zbioru treningowego oraz jednej warstwy ukrytej.

	2240x1440x1 (35 klatek)	896x600x1 (14 klatek)	512x400x1 (8 klatek)
Precyzja	*0,74 (-0,19)*	*0,86 (-0,07)*	*0,93*
Czułość	***1*** *(+0,03)*	*0,89 (-0,08)*	*0,97*
Specyficzność	***0,14*** *(-0,74)*	*0,72 (-0,14)*	*0,86*
F1 score	***0,24*** *(-0,66)*	*0,81 (-0,09)*	*0,90*
MCC	***0,29*** *(-0,55)*	*0,71 (-0,13)*	*0,84*

Dodatkowo, dla zbioru liczącego 8 klatek na nagranie, zbadany został wpływ interpolacji na proces uczenia. Przykładowe wyniki dla sieci z jedną warstwą ukrytą przedstawia tabela 4.15.

Tabela 4.15 Miary modelu sztucznych sieci neuronowych dla różnych rozdzielczości interpolacji oraz jeden warstwy ukrytej. Wartości umieszczone w nawiasach przedstawiają liczbę neuronów w poszczególnych warstwach.

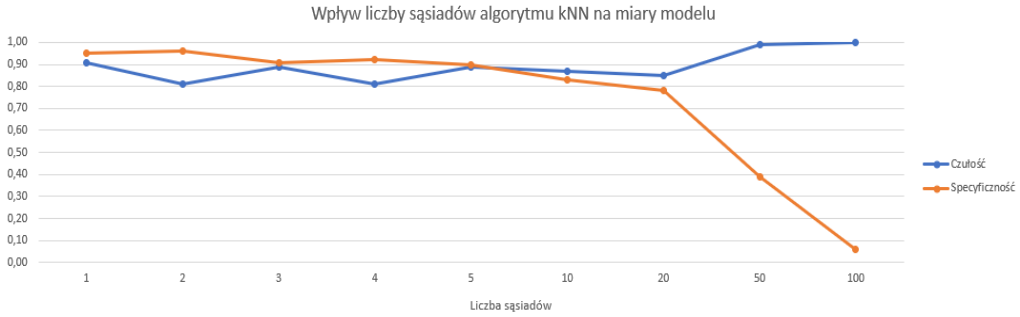
	24x24 (4608x2000x1)	16x16 (2048x1000x1)	8x8 (512x400x1)
Precyzja	*0,8 (-0,13)*	*0,79 (-0,14)*	*0,93*
Czułość	*0,29 (-0,68)*	*0,98 (+0,01)*	*0,97*
Specyficzność	*1 (+0,14)*	*0,45 (-0,41)*	*0,86*
F1 score	*0,45 (-0,45)*	*0,76 (-0,14)*	*0,90*
MCC	*0,42 (-0,42)*	*0,50 (-0,34)*	*0,84*

Wyniki badań wykazały, iż wpływ interpolacji ma negatywny wpływ na proces uczenia maszynowego sieci neuronowych. Wraz ze sztucznym wzrostem rozdzielczości maleje zdolność modelu do poprawnej klasyfikacji upadku osoby. Eksperyment został przeprowadzony dla różnej liczby warstw ukrytych oraz zawartych w nich neuronów, jednakże w każdym przypadku zaobserwowano podobną tendencję. Interpolacja do rozdzielczości 16x16 spowodowała zmniejszenie liczby poprawnych klasyfikacji zwykłych aktywności ruchowych. Natomiast najgorsze rezultaty uzyskano dla rozdzielczości 24x24, przy której model stracił znacząco zdolność do prawidłowej klasyfikacji upadku. Dodatkowo zaobserwowane zostały naprzemienne wahania miary czułości oraz specyficzności, co może wskazywać na zbyt mocne dopasowanie do wybranej klasy na skutek sztucznego zwiększenia rozdzielczości.

4.3.6. Algorytm k najbliższych sąsiadów

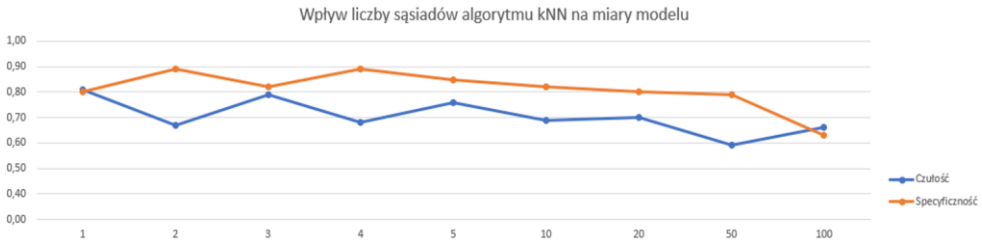
W pierwszej części badań zbadany został wpływ liczby sąsiadów na jakość utworzonego modelu. Proces uczenia maszynowego, którego wyniki przedstawia rysunek 4.22, został przeprowadzony dla zbioru treningowego

liczącego 8 klatek przypadających na pojedyncze nagranie o rozdzielczości 8x8 pikseli.



Rysunek 4.22 Wpływ liczby sąsiadów algorytmu kNN na miary modelu dla zbioru liczącego 8 klatek na nagranie o rozdzielczości 8x8 pikseli.

Wyniki badań pozawalają zaobserwować, iż większa liczba sąsiadów negatywnie wpłynęła na jakość utworzonego modelu, powodując zbyt silne dopasowanie do jednej klasy. Natomiast najlepsze rezultaty osiągnięto dla wartości w zakresie od 1 do 4, zachowując zbliżony stosunek liczby poprawnych detekcji upadków do pozostałych aktywności ruchowych. W następnym kroku wykonany został proces uczenia maszynowego dla zbioru treningowego liczącego 35 klatek na pojedyncze nagranie, którego wyniki przedstawia rysunek 4.23. Zwiększenie liczby pikseli pozwoliło otrzymać mniejsze wahania uzyskanych miar dla większej liczby sąsiadów kosztem pogorszenia jakości dokonywanych klasyfikacji. Utworzony model cechuje się zauważalnym marginesem błędu, w szczególności miary czułości. Ponownie najlepsze miary zaobserwowano dla mniejszej liczby sąsiadów.



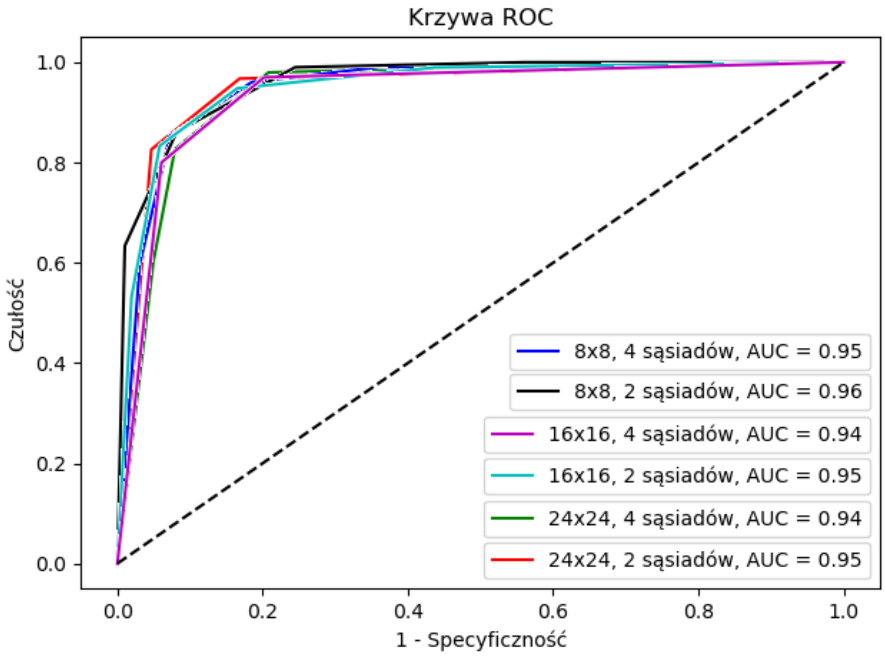
Rysunek 4.23 Wpływ liczby sąsiadów algorytmu kNN na miary modelu dla zbioru liczącego 8 klatek na nagranie.

W następnym kroku poddano analizie wpływ interpolacji na proces uczenia maszynowego, której wyniki zostały zamieszczone w tabeli 4.16. Wartości występujące w nawiasach przedstawiają różnicę względem wyników uzyskanych dla rozdzielczości 8x8 pikseli i parametru *k* z przedziału od 1 do 4. Niezależnie od zastosowanej konfiguracji, nie zaobserwowano znaczącego wpływu interpolacji na utworzony model.

Tabela 4.16 Miary modelu algorytmu kNN dla interpolacji do rozdzielczości 24x24. Wartości występujące w nawiasach przedstawiają różnice względem rozdzielczości 8x8 i parametru k z przedziału od 1 do 4.

	4 sąsiadów	**3 sąsiadów**	**2 sąsiadów**	**1 sąsiad**
Precyzja	*0,91 (+0,02)*	*0,89 (+0,03)*	*0,92 (+0,01)*	*0,91 (-0,03)*
Czułość	*0,94 (+0,08)*	*0,91 (+0,04)*	*0,84 (+0,02)*	*0,89 (-0,08)*
Specyficzność	*0,87 (-0,05)*	*0,85 (+0,00)*	*0,98 (+0,01)*	*0,93 (+0,02)*
F1 score	*0,91 (+0,02)*	*0,89 (+0,02)*	*0,92 (+0,03)*	*0,90 (-0,04)*
MCC	*0,81 (+0,03)*	*0,77 (+0,04)*	*0,83 (+0,02)*	*0,82 (-0,06)*

Dodatkowo sporządzona została krzywa ROC dla wybranych modeli, przedstawiona na rysunku 4.24, która potwierdza brak znaczącego wpływu interpolacji na proces uczenia maszynowego. Niezależnie od zastosowanej konfiguracji uzyskano zbliżone pole pod wykresem.



Rysunek 4.24 Krzywa ROC dla wybranych modeli algorytmu kNN.

4.3.7. Porównanie najlepszych wyników

W niniejszym rozdziale przedstawione zostały miary najlepszych modeli utworzonych w procesie uczenia maszynowego. Dla każdego algorytmu została wybrana konfiguracja pozwalająca na uzyskanie najlepszych rezultatów w ramach poszczególnych algorytmów. Listę ustalonych parametrów przedstawia tabela 4.17.

Dla każdego utworzonego modelu wyznaczone zostały miary jakości, które zostały przedstawione w tabeli 4.18. Wyniki pozwalają zauważyć, iż rozdzielczość 8x8 pikseli pozwoliła uzyskać najlepsze rezultaty niezależnie od wybranego algorytmu. Dodatkowo, w większości algorytmów model utworzony z wykorzystaniem zbioru treningowego liczącego 8 klatek na nagranie cechuje się najlepszymi miarami jakości klasyfikacji.

Tabela 4.17 Konfiguracje modeli oraz procesu uczenia maszynowego dla poszczególnych algorytmów.

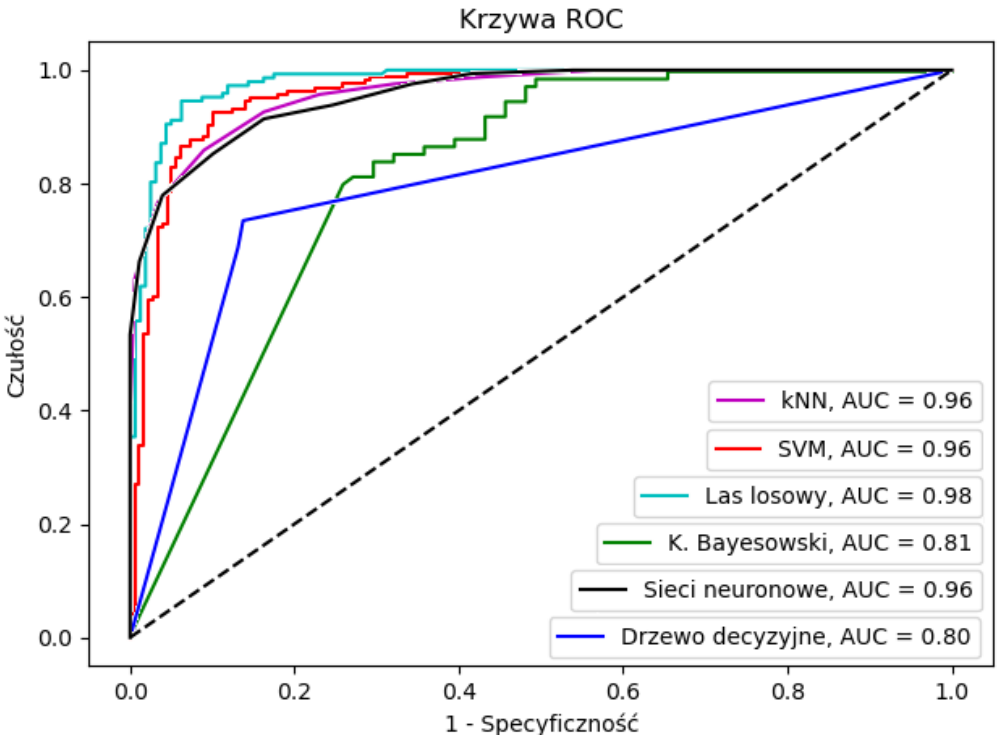
	Liczba klatek na nagranie	Rozdzielczość pojedynczej klatki	Pozostałe parametry
SVM	*8*	*8x8*	*Funkcja wielomianowa,*
Klasyfikator Bayesowski	*14*	*8x8*	*Brak*
Drzewo decyzyjne	*8*	*8x8*	*Wysokość drzewa: 11, Brak ograniczenia liczby liści*
Las losowy	*8*	*8x8*	*Wysokość drzewa: 11, Liczba klasyfikatorów: 100, Brak ograniczenia liczby liści*
Sztuczne sieci neuronowe	*8*	*8x8*	*Liczba epok: 160, Liczna warstw ukrytych: 2, Struktura: 512x400x100x1,*
Algorytm kNN	*8*	*8x8*	*Liczba sąsiadów: 3*

Na podstawie uzyskanych miar można stwierdzić, iż algorytm SVM oraz las losowy pozwalają najlepiej rozwiązać problem klasyfikacji upadku starszej osoby z wykorzystaniem nagrań termicznych. Natomiast model drzewa decyzyjnego oraz klasyfikatora Bayesa obarczony jest zauważalnym marginesem błędu, który prowadzi do występowania fałszywych detekcji.

Tabela 4.18 Najlepsze uzyskane miary dla poszczególnych algorytmów

	Las losowy	SVM	Sieci neuronowe	Algorytm kNN	Drzewo decyzyjne	Klasyfikator Bayesowski
Precyzja	***0,94***	*0,92*	*0,91*	*0,91*	*0,80*	*0,79*
Czułość	***0,94***	*0,92*	*0,91*	*0,89*	*0,81*	*0,78*
Specyficzność	***0,93***	*0,91*	*0,89*	*0,91*	*0,79*	*0,80*
F1 score	***0,93***	*0,91*	*0,89*	*0,90*	*0,80*	*0,79*
MCC	***0,87***	*0,83*	*0,79*	*0,82*	*0,52*	*0,58*

Ponadto wyznaczone krzywe ROC, przedstawione na rysunku 4.25, potwierdzają powyższe wnioski. Największe pole pod wykresem osiągnięto dla algorytmu SVM oraz lasu losowego, natomiast najmniejsze dla klasyfikatora Bayesowskiego oraz drzew decyzyjnych. Wartymi uwagi są również sieci neuronowe oraz algorytm kNN, które również poprawnie rozwiązują problem klasyfikacji upadku, osiągając skuteczność detekcji na poziomie około 90%.



Rysunek 4.25 Krzywe ROC dla najlepszych modeli poszczególnych algorytmów

4.4. Porównanie platform cyfrowych

W celu rejestracji nagrań termicznych oraz dokonywania detekcji upadku z użyciem metod inteligencji obliczeniowej kluczowy jest wybór odpowiedniej platformy cyfrowej. Parametry układu muszą pozwolić na użycie metod inteligencji obliczeniowych oraz przechowywanie modelu klasyfikatora wraz z określoną liczbą klatek wstecz. W szczególności istotnym parametrem jest pamięć flash, przechowująca kod programu, oraz pamięć operacyjna RAM (ang. *random access memory*). Ponadto kluczowym aspektem jest interfejs sieciowy pozwalający na komunikację z usługą chmurową, a tym samym wezwanie pomocy dla osoby potrzebującej.

Jednymi z bardziej popularnych rozwiązań są układy z podstawowej serii Arduino, pierwotnie zaprojektowane do celów edukacyjnych. Wyposażone są one w mikrokontroler firmy Atmel oraz cechują się łatwością programowania, dużym wsparciem społeczności, szeroką gamą dostępnych modułów oraz niskim zużyciem energetycznym przy zachowaniu korzystnego stosunku ceny do wydajności. Ponadto wartym uwagi jest również układ WeMos D1 Mini będący platformą z mikrokontrolerem ESP8266 zawierającym wbudowany moduł WiFi. Rozwiązanie jest kompatybilne z platformą Arduino, cechuje się niskim zużyciem mocy oraz małymi rozmiarami, co sprawia, iż może zostać dyskretnie umiejscowiony w niemalże dowolnym miejscu.

Ponadto na rynku znaleźć można konkurencyjną konstrukcję Raspberry Pi, pierwotnie również zaprojektowaną do celów edukacyjnych. Platforma cechuje się rozbudowaną gamą interfejsów, wydajnym wielordzeniowym procesorem z rodziny ARM oraz dużym wsparciem społeczności z zachowaniem stosunkowo niskich rozmiarów zbliżonych wielkością do karty kredytowej.

W pierwszej kolejności poddane analizie zostały podstawowe parametry wybranych konstrukcji, których porównanie przedstawia tabela 4.19.

Tabela 4.19 Porównanie wybranych platform cyfrowych.

Parametr	Arduino Uno	WeMos D1 mini	Raspberry Pi 3B+
Taktowanie procesora	do 20 MHz	80 MHz	4 x 1.4 GHz
Pamięć flash	do 32 kB	4 MB	do 32 GB
Pamięć RAM	**do 2 kB**	**64 kB**	**1 GB**
Interfejs WiFi	brak	802.11 b/g/n	802.11 b/g/n
Interfejs Ethernet	brak	brak	kategoria 5E
Interfejs I^2C	tak	tak	tak
Liczba złącz	23	11	40
System operacyjny	nie	nie	tak
Wymiary [mm]	68,6 x 53,3	34,2 x 25,6	85 x 56
Napięcia złącz	5 V	5 V	3,3 V
Napięcie zasilania	5 V	5 V	5 V

Porównanie ukazuje, iż układy Arduino z serii podstawowej cechują się małym rozmiarem pamięci, co może skutkować brakiem możliwości załadowania modelu oraz jego dalszego użytkowania. Obserwacje potwierdzają również miary zapotrzebowania pamięciowego przez bufor kolejnych klatek nagrania, które zostały przedstawione w tabeli 4.20.

Tabela 4.20 Zapotrzebowanie na pamięć dla różnych konfiguracji bufora nagrań. Kolumny przedstawiają rozdzielczość zdjęć termicznych, natomiast wiersze liczbę klatek przypadających na nagranie.

	24 x 24	16 x 16	8 x 8
35 klatek	*78,750 kB*	*35 kB*	*8,75 kB*
14 klatek	*31,5 kB*	*14 kB*	*3,5 kB*
8 klatek	*18 kB*	*8 kB*	*2 kB*

Analiza wykazała, iż dla układu Arduino jedyną możliwą konfiguracją jest przechowywanie 8 klatek o rozdzielczości 8x8. Dodatkowo wartym uwagi jest fakt, iż model klasyfikatora, zmienne pomocnicze oraz oprogramowanie modułu kamery również wykazują zapotrzebowanie na pamięć.

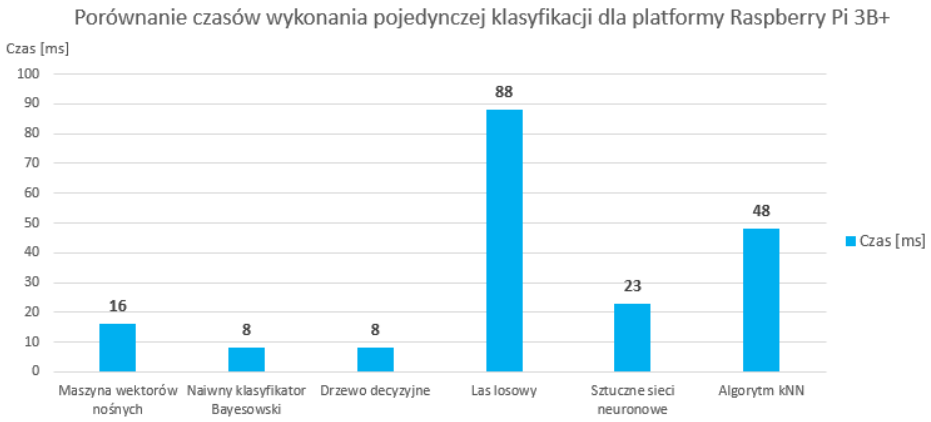
W następnym kroku wykonana została analiza rozmiaru modelu utworzonego w procesie uczenia maszynowego, której wybrane wyniki dla zbioru treningowego liczącego 8 klatek na nagranie przedstawia tabela 4.21.

Tabela 4.21 Rozmiar modeli algorytmów dla wybranych konfiguracji.

	24 x 24	16 x 16	8 x 8
Sieci neuronowe	*3 145 kB*	*2 640 kB*	*1 973 kB*
Maszyna wektorów nośnych	*81 kB*	*75 kB*	*69 kB*
Klasyfikator naiwnego Bayesa	*65 kB*	*65 kB*	*64 kB*
Drzewo decyzyjne	*45 kB*	*41 kB*	*50 kB*
Las losowy	*653 kB*	*651 kB*	*679 kB*
K najbliższych sąsiadów	*14 836 kB*	*6 596 kB*	*1 652 kB*

Wyniki badań pozwalają stwierdzić, iż podstawowa platforma Arduino oraz WeMos D1 mini nie pozwalają na dokonanie klasyfikacji ze względu na ograniczenia sprzętowe w postaci niewystarczającego rozmiaru pamięci.

W kolejnym kroku wykonane zostały pomiary czasowe wykonania pojedynczej klasyfikacji dla platformy Raspberry Pi 3B+, których miary przedstawia rysunek 4.26. Badania zostały wykonane dla rozdzielczości 24x24 oraz zbioru treningowego liczącego 35 klatek na pojedyncze nagranie w celu sprawdzenia najbardziej złożonej konfiguracji z punktu widzenia obciążenia procesora. Dodatkowo, ze względu na dużą liczbę pikseli, dla algorytmów wspierających współbieżność (sieci neuronowe, kNN oraz las losowy), wykorzystane zostały wszystkie cztery dostępne rdzenie jednostki centralnej.



Rysunek 4.26 Porównanie czasów wykonania pojedynczej klasyfikacji dla platformy Raspberry Pi 3B+.

Wartości pomiarów pozwalają stwierdzić, iż niezależnie od wybranego algorytmu platforma Raspberry Pi 3B+ jest w stanie dokonać klasyfikacji nagrania w czasie mniejszym niż 100ms. Oznacza to, iż wydajność układu pozwala na wykorzystanie pełnego zakresu częstotliwości pracy kamery, która wynosi 10 Hz. Ponadto wykorzystana została najbardziej obciążająca procesor konfiguracja, która w rozdziale 4.3 osiągała najgorsze miary jakości utworzonego modelu.

5. Podsumowanie

W ramach implementacji oprogramowania detekcji upadku starszej osoby zostały w pełni zrealizowane wszystkie założenia oraz wymagania projektowe. Układ poprawnie rejestruje oraz przetwarza zdjęcia termiczne z kamery termowizyjnej Adafruit AMG8833. Każde nagranie zostaje poddane klasyfikacji pod kątem upadku z wykorzystaniem wybranych klasyfikatorów. Dodatkowo, rozwiązanie poprawnie nawiązuje komunikację z usługą chmurową w celu przesłania wybranych nagrań podopiecznych. W przypadku wykrycia upadku następuje wysłanie powiadomienia w postaci wiadomości SMS do wszystkich przypisanych opiekunów. Ponadto w celu weryfikacji zdarzenia utworzony został panel do przeglądania nagrań zapisanych w usłudze chmurowej.

Analiza wybranych algorytmów inteligencji obliczeniowej wykazała zdolność rozwiązywania problemów klasyfikacji pod kątem upadku osoby z wykorzystaniem nagrań termicznych. Zaobserwowany został wpływ liczby klatek przypadającej na nagranie oraz interpolacji do wyższych rozdzielczości. Ponadto każdy algorytm cechuje się unikalnym zestawem parametrów uczenia maszynowego, których odpowiedni dobór może okazać się kluczowy do uzyskania modelu wyróżniającego się wysoką jakością dokonywanych klasyfikacji. W niniejszym projekcie najlepsze miary modelu uzyskano dla algorytmu lasu losowego z 94% precyzją, 93% specyficznością oraz 94% czułością. Różnica uzyskanej precyzji względem wybranych istniejących rozwiązań wynosi kolejno -3,9 [6], 24 [5], -4,9 [9] oraz 4,8 [11] punktów procentowych.

Ważnym aspektem jest również wybór platformy komputerowej, która w zależności od parametrów sprzętowych potrafi przeprowadzić obserwację

pomieszczenia pod kątem upadku. Analiza wykazała, iż jednym z kluczowych elementów jest pamięć układu, która cechuje się większym zapotrzebowaniem w zastosowaniach z inteligencją obliczeniową. Z tego względu nie udało się przeprowadzić klasyfikacji na platformie Arduino oraz WeMos D1 mini.

W porównaniu do istniejących rozwiązań bazujących na tradycyjnej kamerze, niniejsze rozwiązanie pozwala zachować prywatność podopiecznego poprzez pomiar wyłącznie temperatury w niskiej rozdzielczości. Obrazy termalne nie pozwalają na identyfikację osoby, miejsca zamieszkania, a nawet wykonywanej czynności, co znacząco wpływa na komfort użytkowania. Urządzenie może zostać umiejscowione zarówno w pokoju dziennym, jak i również sypialni, a nawet łazience. Ponadto w porównaniu do opasek, wisiorków oraz inteligentnych zegarków rozwiązanie nie wymaga bezpośredniej interakcji z podopiecznym oraz noszenia, może zostać podłączone do stałego źródła zasilania oraz pracować 24 godziny na dobę. Dodatkową zaletą względem wielu istniejących rozwiązań jest komunikacja z chmurą obliczeniową pozwalająca na podgląd oraz weryfikację nagrania upadku przez opiekunów po otrzymanym powiadomieniu SMS przy zachowaniu prywatności osoby podopiecznej. Natomiast do wad zaliczają się niedoskonałości modułu AMG8833 takie, jak odległość detekcji do 4 metrów, błąd pomiarowy na poziomie ± 2°C oraz duża wrażliwość pobliskich urządzeń takich, jak monitor komputerowy lub kaloryfer. Dodatkowo, w zależności od ubioru osoby oraz odległości układ dokonuje pomiarów ludzkiego ciała w zakresie 26 - 30°C, co w przypadku wyższej temperatury otoczenia uniemożliwia detekcję upadku. Ponadto rozwiązanie jest przeznaczone głównie do wewnętrznych pomieszczeń oraz nie jest możliwe jego użycie np. podczas spacerów.

Niniejszy projekt jest również otwarty na dalszy rozwój oraz modyfikację. Na rynku dostępne są również inne rozwiązania z rodziny układów termowizyjnych. Jednym z przykładów jest moduł Adafruit MLX90640 pracujący z częstotliwością odświeżania w zakresie 0,5-64 Hz oraz matrycą termiczną o rozdzielczości 32x24 pikseli. Większa liczba

pikseli może pozwolić na lepsze rozróżnienie podstawowych aktywności ruchowych, jak na przykład schylenie się w celu zawiązania sznurowadeł, od upadku. Dodatkowo, w przypadku modułu Adafruit AMG8833 zaobserwowano trudności rejestracji osoby w odległości powyżej 4 metrów, czego rozwiązaniem może okazać się wyższa rozdzielczość nagrań. Ponadto w platformie obliczeniowej Raspberry Pi 3B+ wykorzystane zostały jedynie 4 z 40 złącz wejścia/wyjścia, co pozwala na podłączenie dodatkowych modułów, jak np. czujnik ultradźwiękowy, mikrofon lub tradycyjna kamera.

Kolejnym potencjalnym kierunkiem rozwoju jest panel administracyjny oraz usługa chmurowa. Oprogramowanie może zostać rozszerzone o integrację z zewnętrznym systemem medycznym, numerem alarmowym lub dodatkową usługą chmurową. Platforma AWS cechuje się wysoką automatyczną skalowalnością w przypadku większego obciążenia, co pozwala na implementację dodatkowych funkcjonalności.

Bibliografia

[1] Amazon API Gateway. aws.amazon.com/api-gateway. [Data uzyskania dostępu: 2020 Lipiec 2020].

[2] Ashraf, D.; Aboul, E. H. *Wearable and implantable wireless sensor network solutions for healthcare monitoring.* Sensors 2011, 11, 5561–5595.

[3] Atienza, R. *Advanced Deep Learning with TensorFlow 2 and Keras - Second Edition*. Wielka Brytania, Packt Publishing, 2020.

[4] Aurélien, G. *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow*. USA, O'Reilly Media, 2019.

[5] Battineni, G.; Chintalapudi, N.; Amenta, F. *Machine learning in medicine: Performance calculation of dementia prediction by support vector machines (SVM).* Informatics in Medicine Unlocked 2019, 16, 4-6.

[6] De Miguel, K.; Brunete, A.; Hernando, M.; Gambao, E. *Home Camera-Based Fall Detection System for the Elderly*. Sensors 2017, 17, 2864.

[7] Gandhi, R. *Support Vector Machine—Introduction to Machine Learning Algorithms*. towardsdatascience.com /support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Data uzyskania dostępu: 10 Sierpień 2020].

[8] Harańczyk, G. *Krzywe ROC, czyli ocena jakości klasyfikatora i poszukiwanie optymalnego punktu odcięcia*. media.statsoft.pl/_old_dnn/downloads/krzywe_roc_czyli_ocena_jakosci.pdf. [Data uzyskania dostępu: 8 Sierpień 2020].

[9] Liu, S. H.; Cheng, W. C. *Fall Detection with the Support Vector Machine during Scripted and Continuous Unscripted Activities*. Sensors 2012, 12, 12301-12316.

[10] Martin, R. *Clean code*. Wielka Brytania, Pearson, 2009.

[11] Mazurek, P.; i Morawski, R. Z. *Application of naïve Bayes classifier in fall detection systems based on infrared depth sensors.* 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) 2015, 1, 717-721.

[12] Microchip ATmega328P official datashet. microchip .com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. [Data uzyskania dostępu: 26 Czerwiec 2020].

[13] Mielczarek, W. *Szeregowe Interfejsy Cyfrowe*. Helion, Gliwice, 1997.

[14] Nowak-Brzezińska, A. *Klasyfikatory: k-NN oraz naiwny Bayesa, Uniwersytet Śląski*. zsi.tech.us.edu.pl /~nowak/si/w3a.pdf. [Data uzyskania dostępu: 10 Sierpień 2020].

[15] Nowak-Brzezińska, A. *Sieci neuronowe - wprowadzenie, Uniwersytet Śląski*. zsi.tech.us.edu.pl/~nowak/psiise /psiise_7.pdf. [Data uzyskania dostępu: 09 Sierpień 2020].

[16] NVIDIA DLSS 2.0 - duży postęp w renderowaniu opartym na SI. nvidia.com/pl-pl/geforce/news/nvidia-dlss-2-0-a-big-leap-in-ai-rendering. [Data uzyskania dostępu: 02 Sierpień 2020].

[17] Patel, S.; Park, H.; Bonato, P.; Chan, L.; Rodgers, M. *A review of wearable sensors and systems with application in rehabilitation*. Journal of NeuroEngineering and Rehabilitation 2012, 9, 1-7.

[18] Płodowski, M.; Suszko, M.; Markuszewski, K. *Politechnika Wrocławska, Porównanie algorytmów tworzenia drzew decyzyjnych.* kajtsweb.webd.pl/pwr/kajt/weka/uczenie_maszyn_projekt.pdf. [Data uzyskania dostępu: 02 Październik 2020].

[19] Płoński, P. *Zastosowanie wybranych metod przekształcania i selekcji danych oraz konstrukcji cech w zadaniach klasyfikacji i klasteryzacji. Politechnika Warszawska*. repo.pw.edu.pl/docstore/download/WUTc13952609c664d15ab730db3b39e20ae/P%C5%82o%C5%84ski-final.pdf. [Data uzyskania dostępu: 12 Sierpień 2020].

[20] Politechnika Śląska. Laboratorium mikroinformatyki. Szeregowe magistrale synchroniczne. db.zmitac.aei.polsl.pl/ui/instrukcje/SzeregoweMagistraleSynchroniczne.pdf. [Data uzyskania dostępu: 13 Lipiec 2020].

[21] Prata, S. *Język C++. Szkoła programowania*. Gliwice, Helion, 2012.

[22] Silaparasetty, V. *Deep Learning Projects Using TensorFlow 2. Neural Network Development with Python and Keras*. USA, Apress, 2020.

[23] Walabot Official Site. walabot.com. [Data uzyskania dostępu: 2 Sierpień 2020].

[24] What is AWS. aws.amazon.com/what-is-aws. [Data uzyskania dostępu: 7 Lipiec 2020].

[25] What is AWS Lambda. aws.amazon.com/lambda. [Data uzyskania dostępu: 8 Lipiec 2020].

[26] What is Python? python.org/doc/essays/blurb. [Data uzyskania dostępu: 04 Lipiec 2020].

[27] What is REST? restfulapi.net. [Data uzyskania dostępu: 5 Lipiec 2020].

[28] Zeiler, M. D.; Fergus, R. *Visualizing and Understanding Convolutional Networks*. Computer Vision – ECCV 2014, 8689, 818-833.

Dodatki

Dokumentacja techniczna

1. Instalacja zależności

W celu uruchomienia aplikacji dokonującej detekcji upadku, zarządzającej nagraniami kamery, panelu administratora lub narzędzia do uruchomienia procesu uczenia maszynowego należy zainstalować język skryptowy python w wersji 3.7. Dodatkowo oprogramowanie wymaga zainstalowania następujących bibliotek:

- Scikit-learn 0.23.2,
- Keras 2.4.3,
- Tensorflow 2.3.0,
- numpy 1.19.1,
- matplotlib 3.3.1,
- joblib 0.15.1,
- scipy 1.5.1,
- PySimpleGUI 4.29.0

Listing 1.1 przedstawia przykład instalacji wymaganych zależności dla systemu operacyjnego Ubuntu.

```
sudo apt update
sudo apt upgrade
sudo apt-get install python3.7
sudo apt install python3-pip
sudo pip install scikit-learn
sudo pip install --upgrade tensorflow
sudo pip install keras
sudo pip install numpy
sudo pip install matplotlib
sudo pip install joblib
sudo pip install scipy
sudo pip install PySimpleGUI
```

Listing 1.1 Instalacja wymaganych zależności dla systemu operacyjnego Ubuntu

2. Uruchomienie aplikacji

2.1 Aplikacja do zarządzania nagraniami

W celu uruchomienia aplikacji do zarządzania nagraniami kamery termowizyjnej AMG8833 należy udać się do katalogu zawierającego kod źródłowy oraz otworzyć edytorem tekstowym plik *main.py*. W kolejnym kroku należy zmodyfikować linię 23 zawierającą zmienną o nazwie *base_path*. Jako wartość należy podać ścieżkę do katalogu w którym zostaną zapisane/odczytane nagrania modułu kamery termowizyjnej. Na końcu należy zapisać plik, uruchomić w katalogu terminal oraz wywołać polecenie przedstawione w listingu 2.1.

```
python3 ./main.py
```

Listing 2.1 Polecenie uruchamiające narzędzie do zarządzania nagraniami

2.2 Narzędzie do wykonywania procesu uczenia maszynowego

W celu uruchomienia narzędzia do wykonywania procesu uczenia maszynowego należy udać się do katalogu zawierającego kod źródłowy oraz otworzyć edytorem tekstowym plik *dataset_util.py*. W kolejnym kroku należy zmodyfikować linię 13 zawierającą zmienną o nazwie *dataset_path*. Jako wartość należy podać ścieżkę do katalogu w którym znajdują się zapisane nagrania modułu kamery termowizyjnej. Dodatkowo zmienna *max_buffer_size* w linii 14 wskazuje liczbę klatek na nagranie, a *frame_size* w linii 15 rozdzielczość interpolacji. Na końcu należy zapisać plik, uruchomić w katalogu terminal oraz wywołać wybrane polecenie przedstawione w tabeli 2.1.

Tabela 2.1 Polecenia rozpoczynające proces uczenia maszynowego

Polecenie	**Algorytm**
`python3 ./decision_trees.py`	`Drzewo decyzyjne`
`python3 ./knn.py`	`Algorytm kNN`
`python3 ./naive_bayes.py`	`Klasyfikator naiwnego Bayesa`
`python3 ./neural_network.py`	`Sieci neuronowe`
`python3 ./random_forest.py`	`Las losowy`
`python3 ./svm.py`	`Maszyna wektorów nośnych`

2.3 Aplikacja dokonująca detekcji upadku

W celu uruchomienia aplikacji dokonującej detekcji upadku należy udać się do katalogu zawierającego kod źródłowy oraz otworzyć edytorem tekstowym plik *main.py*. W kolejnym kroku należy zmodyfikować linię 16 zawierającą import wybranego klasyfikatora, których lista przedstawiona została w tabeli 2.2 oraz zapisać plik. Ponadto każdy plik zawiera zmienną *model_path* zlokalizowaną w linii numer 4, która zawiera pełną ścieżkę do pliku zawierającego model. Wspomnianą zmienna należy zmodyfikować wyłącznie w przypadku chęci zmiany modelu, gdyż domyślna konfiguracja

zawiera już wyuczoną strukturę algorytmu. Na końcu należy zapisać plik, uruchomić w katalogu terminal oraz wywołać wybrane polecenie przedstawione w listingu 2.2.

Tabela 2.2 Lista plików skryptowych algorytmów inteligencji obliczeniowej

Nazwa pliku	**Algorytm**
decision_trees.py	Drzewo decyzyjne
knn.py	Algorytm kNN
naive_bayes.py	Klasyfikator naiwnego Bayesa
neural_network.py	Sieci neuronowe
random_forest.py	Las losowy
svm.py	Maszyna wektorów nośnych

```
python3 ./main.py
```

Listing 2.2 Polecenie uruchamiające apliakcję dokonującą detekcję upadku

2.4 Panel administracyjny

W celu uruchomienia panelu administracyjnego należy udać się do katalogu zawierającego kod źródłowy, uruchomić terminal oraz wywołać wybrane polecenie przedstawione w listingu 2.3.

```
python3 ./main.py
```

Listing 2.3 Polecenie uruchamiające panel administracyjny

Spis skrótów i symboli

AWS	usługa chmurowa (ang. *Amazon Web Services*)
AUC	pole pod krzywą ROC (ang. *Area Under Curve*)
FN	fałszywie negatywna (ang. *False Negative*)
FP	fałszywie pozytywna (ang. *False Positive*)
I^2C	interfejs komunikacyjny (ang. *Inter-Integrated Circuit*)
kNN	algorytm k najbliższych sąsiadów (ang. *k Nearest Neighbours*)
MCC	współczynnik korelacji Matthewsa (ang. *Matthews Correlation Coefficient*)
MEMS	mikroukład elektromechaniczny (ang. *microelectromechanical system*)
OASIS	zbiór danych neurologicznych (ang. *Open Access Series Of Imaging Studies*)
SMS	wiadomość tekstowa (ang. *Short Message Service*)
WBAN	sieć łącząca moduły elektroniczne (ang. *wireless body area networks*)
REST	styl architektoniczny (ang. *Representional State Transfer*)
SDA	linia danych (ang. *Serial Data*)
SCL	linia zegara (ang. *Serial Clock*)
SVM	maszyna wektorów nośnych (ang. *Support Vector Machine*)
TN	prawdziwie negatywna (ang. *True Negative*)

TP prawdziwie pozytywna (ang. *True Positive*)

UART interfejs komunikacyjny (ang. *Universal Asynchronous Receiver-Transmitter*)

URL format adresowania(ang. *Uniform Resource Locator*)

Zawartość dołączonej płyty

- /dataset – katalog zawierający zbiory danych użyte w eksperymentach,
- /doc – katalog zawierający pracę dyplomową,
- /doc/wersja_pełna – katalog zawierający pełną wersję pracy dyplomowej,
- /doc/wersja_skrócona – katalog zawierający skróconą wersją pracy dyplomowej,
- */results* – katalog zawierający arkusze kalkulacyjne z uzyskanymi wynikami badań,
- /src – katalog zawierający kod źródłowy programów
- /src/admin_panel – katalog zawierający kod źródłowy panelu administracyjnego,
- /src/dataset_util – katalog zawierający kod źródłowy aplikacji do podglądu obrazu z kamery oraz zapisu nagrania do pliku tekstowego,
- /src/training_util – katalog zawierający kod źródłowy aplikacji do wykonywania procesu uczenia maszynowego,
- /src/classifier – katalog zawierający kod źródłowy aplikacji dokonujący detekcji upadku dla platformy Raspberry Pi 3B+,

Spis rysunków

Spis tablic