STRESZCZENIE

Stworzono inteligentny licznik, którego celem jest zliczanie osób wchodzących i wychodzących z pomieszczenia. Jego działanie polega na reidentyfikacji osób przechodzących po raz kolejny i osobnym ich zliczaniu. Proces zliczania został podzielony na trzy główne części: detekcji, śledzenia i reidentyfikacji.

Porównano różne metody detekcji postaci na obrazie. Przetestowano sposób rozpoznawania obiektów na podstawie operacji morfologicznych oraz z wykorzystaniem sieci neuronowych.

Śledzenie osób zostało oparte na porównywaniu odległości między wykrytymi postaciami na kolejnych klatkach filmu. Przyjęto założenie, że osoby liczone idą pojedynczo.

Do reidentyfikacji użyto wektorów charakteryzujących osoby, uzyskanych poprzez przetworzenie obrazów przez głęboką sieć neuronową. Sprawdzono jakie warstwy sieci MobileNet i VGG-19 pozwalają na pozyskanie takich wektorów, które będą umożliwiały reidentyfikację. Zaimplementowano różne metody transformacji wyjścia sieci w postaci tablicy trójwymiarowej na wektor jednowymiarowy. Zrealizowano dwa sposoby klasyfikacji postaci: metodę najmniejszych średnich wśród k-najbliższych oraz metodę największej ilości najbliższych wektorów.

Udało się osiągnąć średnią poprawność reidentyfikacji na poziomie 60% oraz tradycyjnego licznika na poziomie około 95%.

Projekt realizowano w grupie trzyosobowej. Zastosowano następujący podział pracy: Katarzyna Gałka – udział w rozdziałach 1, 7 oraz indywidualnie podrozdziały 2.1, 2.2, 3.1, 3.2, 4.4, 5.6; Katarzyna Retowska – udział w rozdziałach 1, 7 oraz indywidualnie rozdział 6 oraz podrozdziały 3.3, 3.4, 4.5, 5.2, 5.3, 5.5; Zofia Rytlewska – udział w rozdziałach 1, 7 oraz indywidualnie podrozdziały 2.3, 4.1, 4.2, 4.3, 5.1, 5.4.

Słowa kluczowe:
Reidentyfikacja, Głębokie sieci neuronowe, Detekcja, Śledzenie, Licznik osób

Dziedzina nauki i techniki, zgodnie z wymogiem OECD:
Nauka o komputerach, informatyka i bioinformatyka

ABSTRACT

The intelligent people counter was implemented with aim to count people entering and leaving the room. Its working principle is based on re-identification of persons passing again and their separate counting. The process of counting was divided into three main parts: detection, tracking and re-identification.

Various approaches to people detection in the image were compared. The methods of recognizing objects based on morphological operations and using neural networks were tested.

The process of tracking was based on comparing the distances between persons on consecutive frames of the movie. It was assumed that counted people were passing individually.

For the re-identification, vectors characterizing people obtained by transforming images through a deep neural network were used. The correctness of all the MobileNet and VGG-19 neural network layers for the production of this data was tested. Different methods of transforming three-dimensional output of neural network into one-dimensional vector were implemented. Two forms of person classification were compared: the method of the smallest means among the k-closest and the method of the largest number of the nearest vectors.

An average correctness about 60% was achieved for the re-identification algorithm and about 95% for a traditional counter.

The project was carried out in a three-person group. The following distribution of work was used: Katarzyna Gałka – participation in sections 1, 7 and individually subsections 2.1, 2.2, 3.1, 3.2, 4.4, 5.6; Katarzyna Retowska – participation in sections 1, 7 and individually section 6 and subsections 3.3, 3.4, 4.5, 5.2, 5.3, 5.5; Zofia Rytlewska – participation in sections 1, 7 and individually subsections 2.3, 4.1, 4.2, 4.3, 5.1, 5.4.

Keywords:

Re-identification, Deep neural networks, Detection, Tracking, People counter

Field of science and technology in accordance with OECD requirements:

Computer and information sciences

SPIS TREŚCI

WYKAZ WAŻNIEJSZYCH POJĘĆ I SKRÓTÓW

Ramka – pojedyncza klatka filmu.
Sesja zliczająca – jeden cykl działania licznika. Na początku danej sesji liczba zliczonych osób wynosi zero, a rejestr osób jest pusty.

Obraz postaci – wycięty fragment klatki przedstawiający dokładnie jednego człowieka. Obraz postaci jest rezultatem operacji detekcji.
Postać – obiekt opisujący jednego człowieka podczas jednego przejścia. Zawiera między innymi zbiór obrazów postaci uzyskanych w wyniku śledzenia.
Osoba – obiekt reprezentujący jednego człowieka podczas jednej sesji zliczającej. Człowiek, który wszedł i wyszedł jest jedną osobą oraz dwiema postaciami.
Wektor postaci – obraz postaci przetworzony przez sieć neuronową do tablicy jednowymiarowej.

Reidentyfikacja – ponowne rozpoznanie osoby, która była wcześniej rozpoznana i zapisana do rejestru.
Odległość między wektorami – norma euklidesowa różnicy dwóch wektorów.
Próg reidentyfikacji – maksymalna odległość między dwoma wektorami, przy której postać zostaje przyporządkowana do wcześniej sklasyfikowanej osoby. Powyżej tego progu postaci uznawane są za nowe osoby.

Licznik prymitywny – licznik osób zliczający każdą osobę.
Licznik inteligentny – licznik osób, zliczający tylko te osoby, które pojawiły się po raz pierwszy.

Poprawność – stosunek liczby prób zakończonych sukcesem do liczby wszystkich prób.

GPU (Graphics Processing Unit) – graficzna jednostka licząca
CPU (Central Processing Unit) – główna jednostka licząca
mAP (mean Average Precision) – jednostka określająca wartość średniej precyzji
FPS (Frames Per Second) – liczba ramek na sekundę

1. WPROWADZENIE

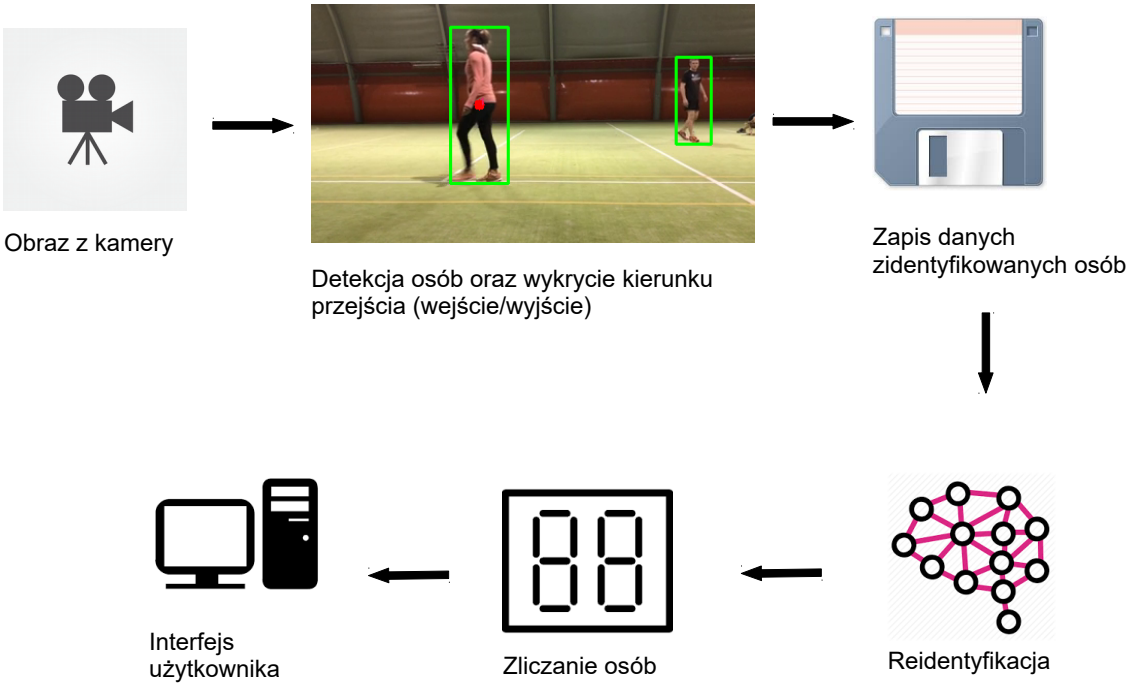
1.1. Dziedzina problemu

Aktualnie na rynku istnieją prymitywne rozwiązania liczenia osób działające zazwyczaj na podstawie sygnału odbieranego przez fotokomórkę. Zadaniem przedstawionego w tej pracy produktu jest liczenie różnych osób pojawiających się w danym miejscu. Przewidywane obszary zastosowań to miejsca publiczne takie jak sklepy, galerie handlowe, kina czy teatry. Jednym z przykładów jest sklep odzieżowy, gdzie są prowadzone statystyki osób, które odwiedzają ten sklep w stosunku do liczby klientów, którzy zrobili w nim zakupy. Dzięki reidentyfikacji osób, w statystykach nie są brani pod uwagę pracownicy tego sklepu, ponieważ są zapamiętani i przy każdym wejściu reidentyfikowani. Użycie inteligentnego licznika pozwala prowadzić bardziej precyzyjne statystyki.

1.2. Cel pracy

Celem pracy jest stworzenie inteligentnego licznika osób działającego w czasie rzeczywistym. Zliczanie osób ma bazować na obrazie z kamery oraz uwzględniać ich reidentyfikację.

1.3. Wizja rozwiązania



Obraz z kamery jest wczytywany do programu na bieżąco. Kolejne klatki filmu przetwarzane są przez sieć neuronową w celu detekcji postaci. Określany jest też kierunek jej ruchu (wejście/wyjście). Następnie dane tej postaci są porównywane z danymi wcześniej zidentyfikowanych osób w celu poszukiwania podobieństw. Jeśli osoba zostanie zreidentyfikowana, to nie jest liczona po raz drugi. Licznik inteligenty jest zwiększany tylko wtedy, gdy wejdzie nowa osoba.

1.4. Podział zadań

Podział zadań pomiędzy poszczególnymi członkami zespołu projektowego został przedstawiony w tabeli 1.

Tabela 1 Podział zadań

Imię i nazwisko	**Wykonane zadania**
Katarzyna Gałka	Implementacja algorytmu reidentyfikacji, dobranie odpowiedniego progu reidentyfikacji, dokumentacja projektu (diagramy UML)
Katarzyna Retowska	Implementacja interfejsu użytkownika, przeprowadzenie testów poprawności licznika prymitywnego oraz inteligentnego, testy wydajności,przeprowadzenie eksperymentów dotyczących porównania warstw dwóch sieci neuronowych oraz różnych algorytmów klasyfikacji
Zofia Rytlewska	Implementacja algorytmu detekcji, śledzenia i liczenia, przeprowadzenie eksperymentu dotyczącego porównania metod transformacji oraz dobrania progu poprawnego wykrycia

1.5. Układ pracy

Praca składa się z siedmiu rozdziałów. W rozdziale drugim określono główne założenia projektu. Opisano wymagania projektowe oraz funkcje inteligentnego licznika. Trzeci rozdział jest poświęcony projektowi systemu. Tam znajdują się m. in. opis architektury oraz projekt interfejsu użytkownika. W czwartym rozdziale opisano sposób implementacji systemu. W rozdziale piątym przedstawiono proces doboru odpowiednich parametrów. Testy przeprowadzone na gotowym systemie przedstawiono w rozdziale szóstym. Na końcu, w rozdziale siódmym, zaprezentowano podsumowanie i najważniejsze wnioski.

2. ZAŁOŻENIA PROJEKTOWE

Przed rozpoczęciem implementacji systemu liczącego zebrano wymagania od wszystkich udziałowców projektu, określono cele biznesowe i funkcjonalne, zidentyfikowano sytuacje wyjątkowe. Sporządzono model przypadków użycia oraz model klas z dziedziny problemu.

2.1. Wymagania

Źródło wymagań stanowili następujący udziałowcy projektu:

I. Zleceniodawca – opiekun projektu dyplomowego inżynierskiego mgr inż. Jan Cychnerski, priorytet: wysoki.
II. Potencjalni użytkownicy systemu – aktualnie zespół studencki, priorytet: średni.
III. Zespół projektowy – autorzy projektu, priorytet: średni.
IV. Ustawa o ochronie danych osobowych – typ prawny, priorytet: krytyczny.

2.1.1. Cele systemu

Celem biznesowym systemu jest ułatwienie zliczania osób wchodzących i wychodzących z pomieszczenia. Do celów funkcjonalnych należy zliczanie osób oraz ich reidentyfikacja.

2.1.2. Otoczenie systemu

Przewidziano dwóch użytkowników systemu:

I. Operator systemu – osoba obsługująca system. Jej zadaniem jest uruchomienie licznika, może być nim także eksport danych o liczbie osób do pliku.
II. Analityk statystyk – osoba, która chce zdobyć wiedzę o liczbie różnych osób, które pojawiły się w danym pomieszczeniu w konkretnym przedziale czasowym. Jej zadaniem jest tylko eksport danych o liczbie tych osób do pliku.

Jako system zewnętrzny zdiagnozowano system monitoringu, z którego obrazu przeprowadzane będzie zliczanie osób. Aktualnie system ma wykorzystywać do tego celu pojedynczą kamerę.

2.1.3. Wymagania funkcjonalne

Do najważniejszych wymagań funkcjonalnych należą:

I. Zliczanie osób - zliczanie osób przebywających obecnie w pomieszczeniu oraz wszystkich osób, które pojawiły się w przeciągu całej sesji zliczającej.
II. Detekcja postaci na obrazie z kamery - wykrycie postaci człowieka na obrazie.
III. Rozpoznanie kierunku ruchu człowieka - stwierdzenie na podstawie obrazu z kamery kierunku przemieszczania się danej osoby (wejście lub wyjście z pomieszczenia).
IV. Reidentyfikacja osób - dopasowanie obrazu osoby wychodzącej do osoby, która wcześniej weszła lub wchodzącej do osoby, która wcześniej wyszła.

V. Rozpoczęcie/zakończenie danej sesji zliczającej - możliwość zakończenia sesji zliczającej – zresetowanie ilości zliczonych osób oraz danych o osobach, które wcześniej wchodziły do danego pomieszczenia. By uniknąć ujemnego stanu licznika i niespójności systemu przyjęto założenie, że początkowo pomieszczenie jest puste. Ze względu na konieczność reidentyfikacji nie przewiduje się możliwości wyłączenia licznika, a następnie ponownego włączenia i kontynuowania tej samej sesji zliczającej.
VI. Eksport danych - możliwość wyeksportowania danych o ilości zliczonych osób do pliku tekstowego wraz z podaniem czasu rozpoczęcia i zakończenia danej sesji zliczającej.

2.1.4. Wymagania na dane

Dane przetwarzane w systemie liczącym:

I. obraz z kamery,
II. dane zliczonych osób – wektory liczb charakteryzujące konkretną osobę, przechowywane do momentu rozpoczęcia nowej sesji zliczającej,
III. liczba policzonych osób.

2.1.5. Wymagania jakościowe

Do najważniejszych wymagań jakościowych należą:

I. Zgodność liczby zliczonych osób przez system ze stanem rzeczywistym (nie biorąc pod uwagę reidentyfikacji). Dopuszczalna różnica wynosi 5%.
I. Poprawność reidentyfikacji na poziomie minimum 60%.
II. Działanie systemu w czasie rzeczywistym z dopuszczalnym opóźnieniem wynoszącym 2s. Spełnialność tego wymagania zależy od użytego sprzętu. Dokładne wymagania sprzętowe określono w rozdziale *6.1 Testy wydajności*.
III. Ochrona danych osób – zdjęcia osób nie będą przechowywane ani udostępniane wraz ze statystyką odwiedzin.
IV. Graficzny interfejs użytkownika.

2.1.6. Sytuacje wyjątkowe

Zdiagnozowano następujące sytuacje wyjątkowe:

I. Osoba weszła do pomieszczenia, gdy licznik był wyłączony, a następnie wychodzi przy włączonym liczniku. Zdarzenie to jest złamaniem założenia przyjętego w punkcie piątym wymagań funkcjonalnych (początkowo pomieszczenie jest puste). W powyższym przykładzie nie ma możliwości poprawnej reidentyfikacji osoby ze względu na brak danych o jej wejściu. Między innymi w tym celu wprowadzono *próg reidentyfikacji* osób, powyżej którego dana osoba wychodząca nie jest reidentyfikowana ani liczona. Mogłoby to prowadzić do ujemnego stanu licznika i niespójności systemu. Dlatego wyświetlany jest jedynie komunikat o niespodziewanym zdarzeniu.
II. Wejście/wyjście kilku osób naraz. Ze względu na ograniczoną wizję (widok 2D - obraz uzyskiwany jest tylko z pojedynczej kamery) możliwe jest, że dana osoba zostanie

niemal całkowicie zasłonięta przez inną. Taka sytuacja wprowadzi licznik w stan nieprawidłowy.

2.2. Przypadki użycia

W celu przedstawienia funkcjonalności systemu liczącego wraz z jego otoczeniem stworzono model przypadków użycia. Każde wcześniej sformułowane wymaganie funkcjonalne znalazło swoje odwzorowanie jako osobny przypadek użycia. Na rys. 2.1 przedstawiono diagram przypadków użycia.

Rys. 2.1 Diagram przypadków użycia

Zidentyfikowano trzech aktorów:

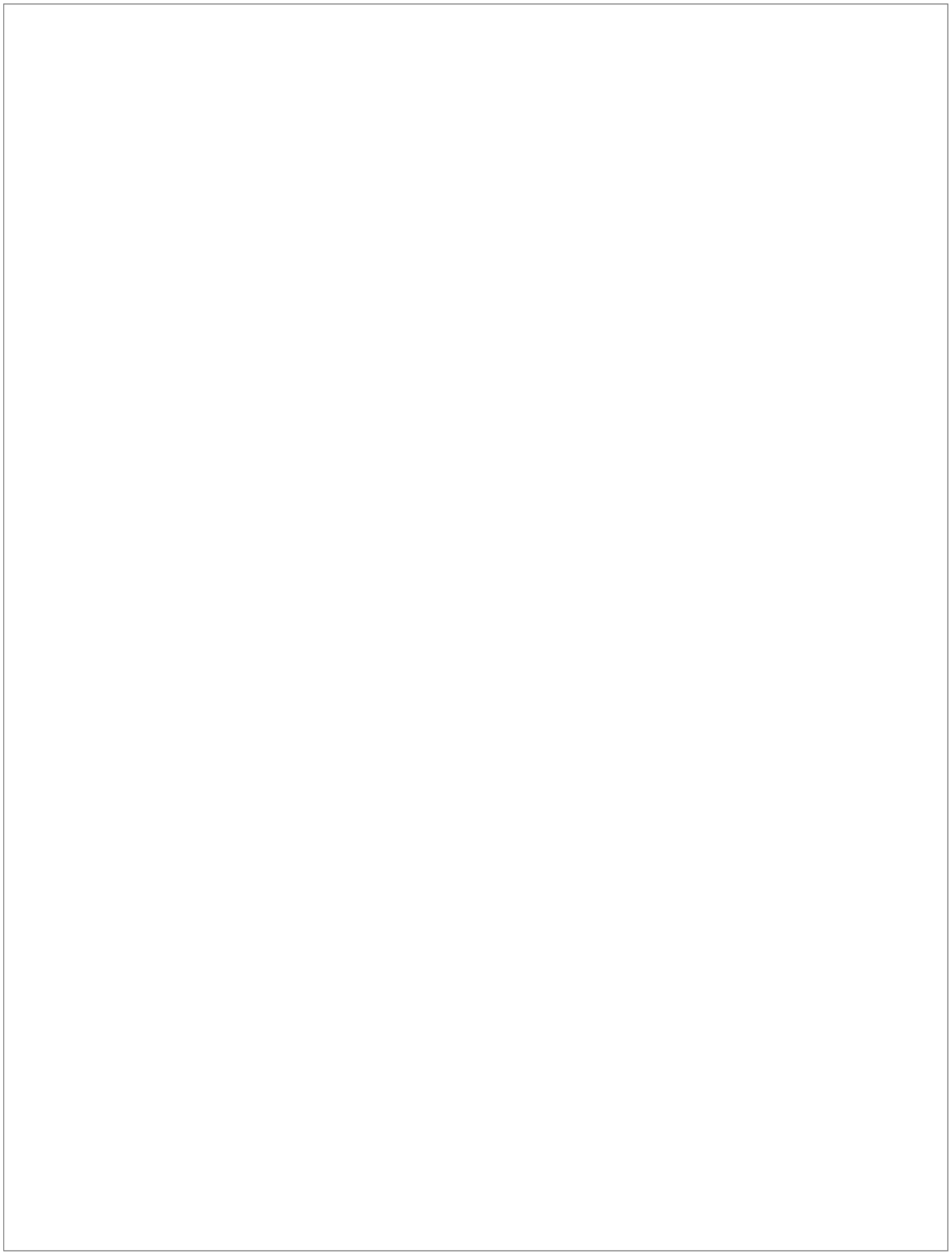
- operatora systemu,
- analityka statystyk,
- system monitoringu.

Operator systemu oraz analityk statystyk reprezentują role, w które może wcielić się użytkownik systemu, natomiast system monitoringu reprezentuje system zewnętrzny (aktualnie pojedynczą kamerę).

2.3. Struktura pojęciowa

W oparciu o wymagania na dane (*2.1.4 Wymagania na dane*) stworzono model klas. Diagram klas (rys. 2.2) przedstawia używane w projekcie pojęcia i zależności między nimi. Kamera jest narzędziem dostarczającym do programu dane wejściowe, czyli film. Film składa się z *klatek* o określonym rozmiarze. Za pomocą detekcji na przetwarzanej klatce rozpoznawane są *postaci*. Postacią jest każdy obiekt, który pojawi się na danej klatce i zostanie sklasyfikowany jako człowiek. Zapamiętywane są m. in. takie parametry jak: aktualne położenie postaci, wszystkie poprzednie położenia postaci, jej stan (typ wyliczeniowy: policzona/niepoliczona), ostatni czas detekcji. Na ich podstawie możliwe jest śledzenie osoby, dzięki temu dla jednej osoby podczas jednego przejścia tworzona jest tylko jedna instancja klasy postać.

Klasa osoba reprezentuje unikalną *osobę* – człowiek, który wszedł i wyszedł posiada jedną instancję klasy osoba oraz dwie instancje klasy postać (przeszedł dwa razy). Osoba posiada swój stan (typ wyliczeniowy: w pomieszczeniu/ poza pomieszczeniem). Podczas procesu identyfikacji ustalane jest podobieństwo postaci do wcześniej zapamiętanych osób. Jeśli podobieństwo do jednej z nich jest dostatecznie wysokie, postać ta uznawana jest za jej kolejne wystąpienie. W przeciwnym razie tworzony jest nowy obiekt osoby. Równocześnie licznik informowany jest o aktualnych zdarzeniach wejścia lub wyjścia.



Rys. 2.2 Diagram klas

2.3.1. Diagramy stanów

W celu ułatwienia zrozumienia zachowania obiektów postaci oraz osoby stworzono ich diagramy stanów. Stany postaci przedstawiono na rys. 2.3, stany osoby przedstawiono na rys. 2.4.

Rys. 2.3 Diagram stanów postaci

Rys. 2.4 Diagram stanów osoby

3. PROJEKT SYSTEMU

Po określeniu założeń projektowych zaprojektowano architekturę systemu oraz przeprowadzono analizę komponentów usługowych. Stworzono projekt interfejsu użytkownika i struktury danych.

3.1. Projekt architektury systemu

W trakcie wyboru języka programowania kierowano się głównie łatwością implementacji oraz szerokim wyborem dostępnych bibliotek używanych do operacji na sieciach neuronowych. Analizując ranking dziesięciu najczęściej wykorzystywanych frameworków do uczenia głębokiego [1] zauważono, że większość z nich wykorzystuje język Python. Dlatego zdecydowano się na użycie Pythona 3.6.

Projekt jest zrealizowany jako aplikacja desktopowa. Wszelkie potrzebne dane wczytywane są do pamięci programu. Do wczytywania filmu oraz przetwarzania obrazu użyto biblioteki OpenCV w wersji 3.3. Operacje związane z sieciami neuronowymi zrealizowano z wykorzystaniem bibliotek Keras 2.0 oraz TensorFlow 1.2.

3.1.1. Warstwy architektoniczne

System podzielono na trzy warstwy architektoniczne:

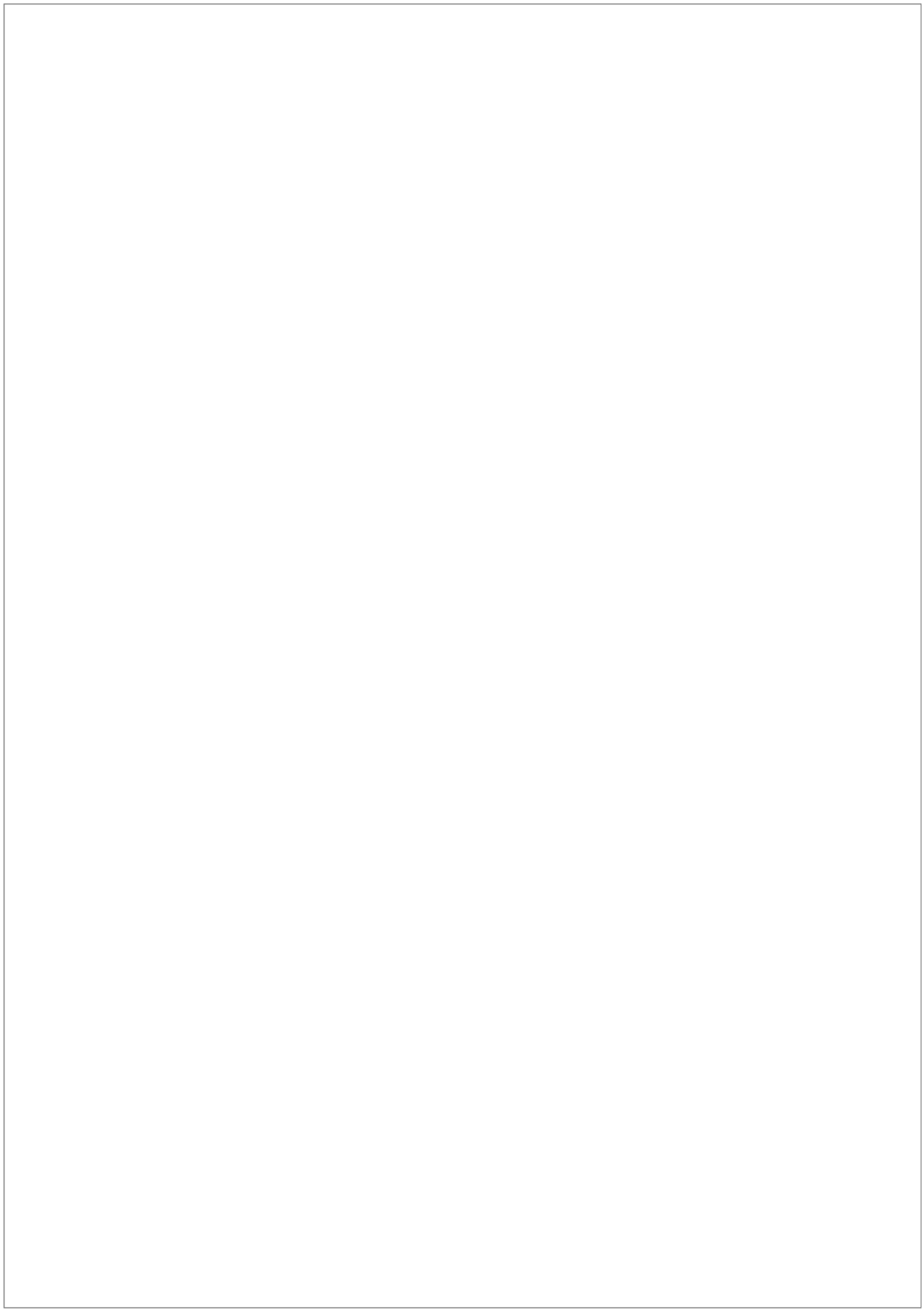
I. Warstwa prezentacji – interfejs użytkownika wyświetlający liczbę zliczonych osób, umożliwiający włączenie oraz wyłączenie licznika oraz wygenerowanie raportu.
II. Warstwa usług – pozwala na detekcję, śledzenie, klasyfikację osób (rozpoznawanie nowej osoby bądź jej reidentyfikację). Umożliwia ich zliczanie.
III. Warstwa danych – przechowuje klatki przetwarzanego filmu, parametry sieci neuronowych, dane zidentyfikowanych osób oraz wartości licznika.

3.1.2. Komponenty programowe

System składa się z czterech głównych komponentów, w skład których wchodzą komponenty warstwy prezentacji, warstwy usług oraz warstwy danych:

I. Detekcja – sieć neuronowa zajmująca się detekcją osób na filmie. Komponenty: Detekcja osób, Obraz z kamery, Sieć neuronowa do detekcji.
II. Śledzenie – śledzenie osoby na kolejnych klatkach filmu. Komponenty: Śledzenie.
III. Reidentyfikacja – klasyfikacja osób jako nowe/zreidentyfikowane z użyciem sieci neuronowej. Komponenty: Klasyfikacja, Rejestr osób, Sieć neuronowa do reidentyfikacji.
IV. Zliczanie – zliczanie osób wchodzących/wychodzących z pomieszczenia. Komponenty: Interfejs użytkownika, Zliczanie, Wartości licznika.

Strukturę programową systemu przedstawiono na rys. 3.1.

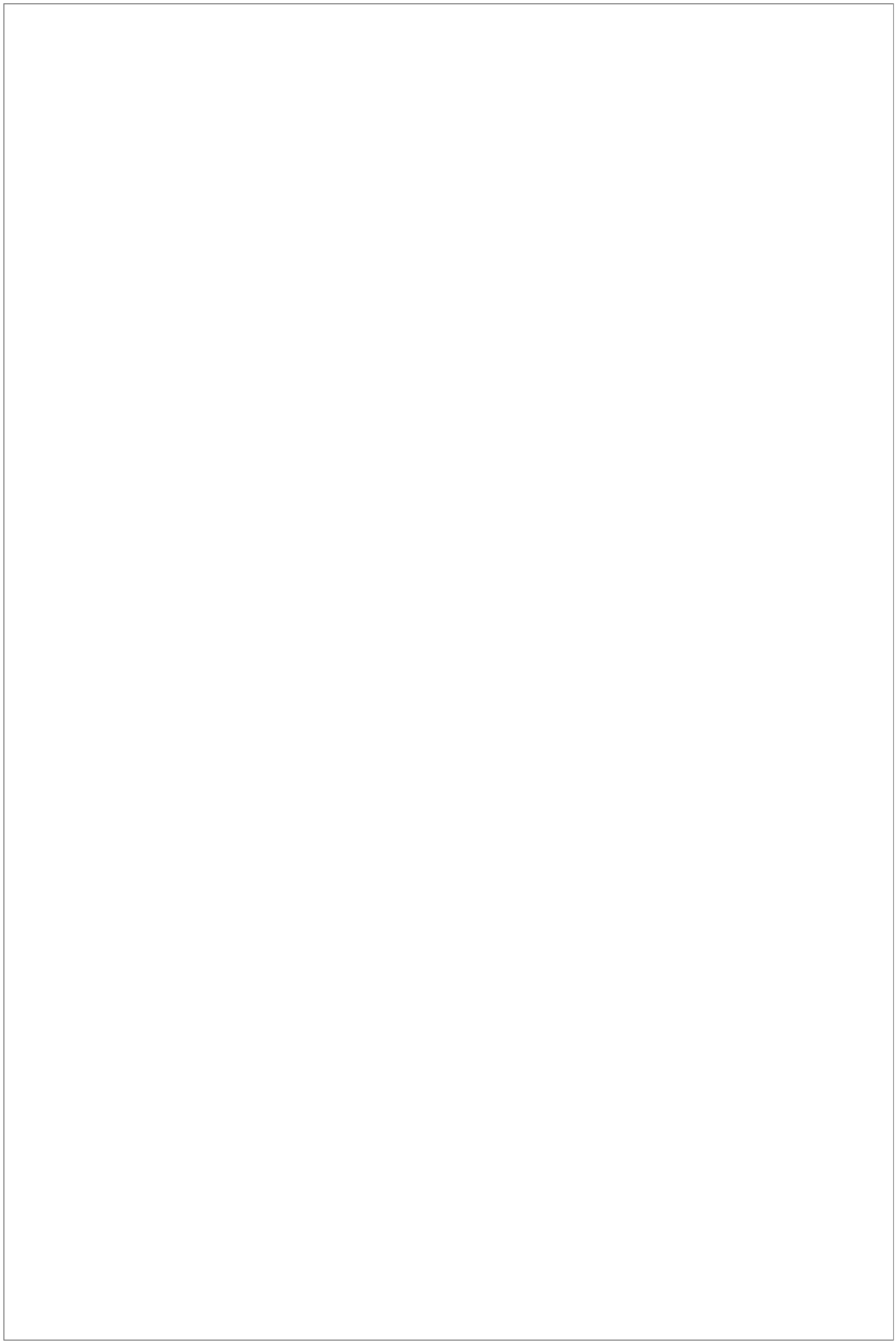


Rys. 3.1 Diagram komponentów programowych

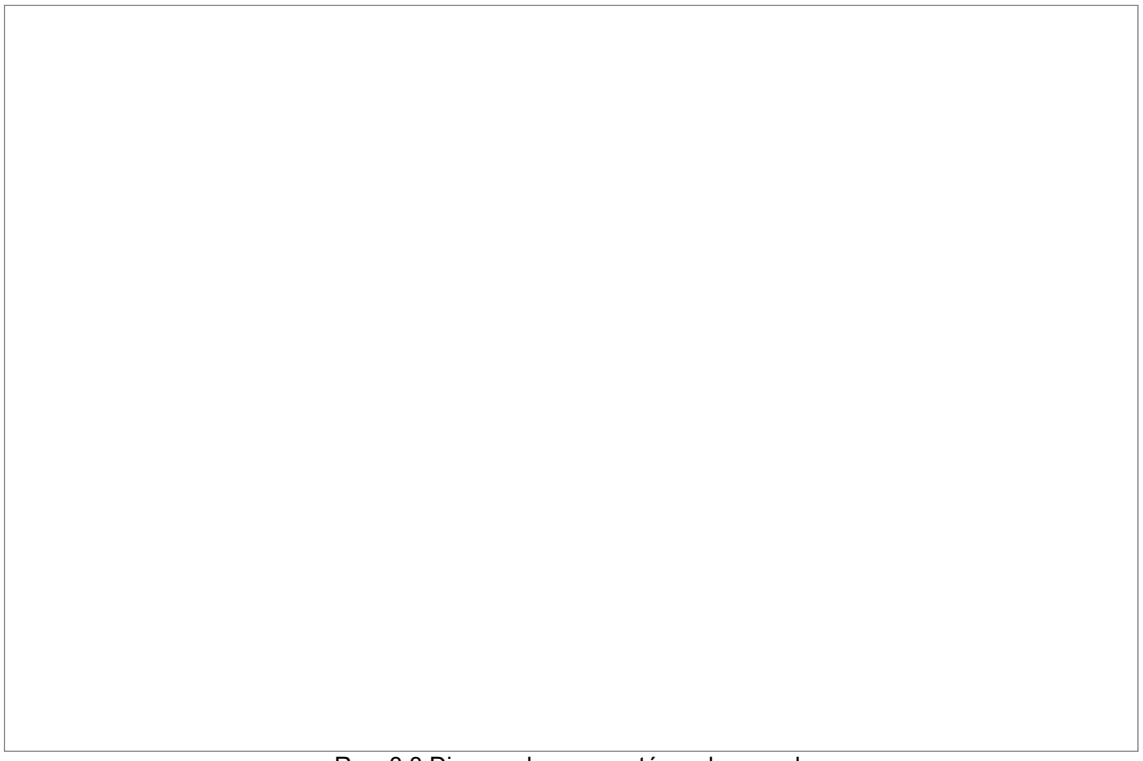
3.2. Projekt logiki biznesowej

W celu zaimplementowania warstwy usługowej dokonano powtórnej analizy klas biznesowych, tym razem z podziałem na klasy encyjne oraz klasy przetwarzające. Podział ten zilustrowano na rys. 3.2. W porównaniu do modelu klas stworzonego w momencie analizy struktury pojęciowej, klasę Identyfikacja rozdzielono na dwie nowe klasy: Transformacja oraz Klasyfikacja. Klasa Transformacja przetwarza obraz postaci na wektor ją charakteryzujący wykorzystując do tego celu sieć neuronową. Klasa Klasyfikacja podejmuje decyzję o przypisaniu postaci do konkretnej osoby.

Dla warstwy usług realizującej logikę biznesową systemu stworzono także diagram komponentów usługowych (rys. 3.3), które pośredniczą między warstwą prezentacji a warstwą danych. Poszczególne komponenty korzystają z operacji innych komponentów poprzez dostarczane interfejsy. W porównaniu do wcześniej zaprojektowanej architektury systemu z komponentu Klasyfikacja wyodrębniono komponent Transformacja.



Rys. 3.2 Diagram klas z podziałem na klasy przetwarzające i encyjne



Rys. 3.3 Diagram komponentów usługowych

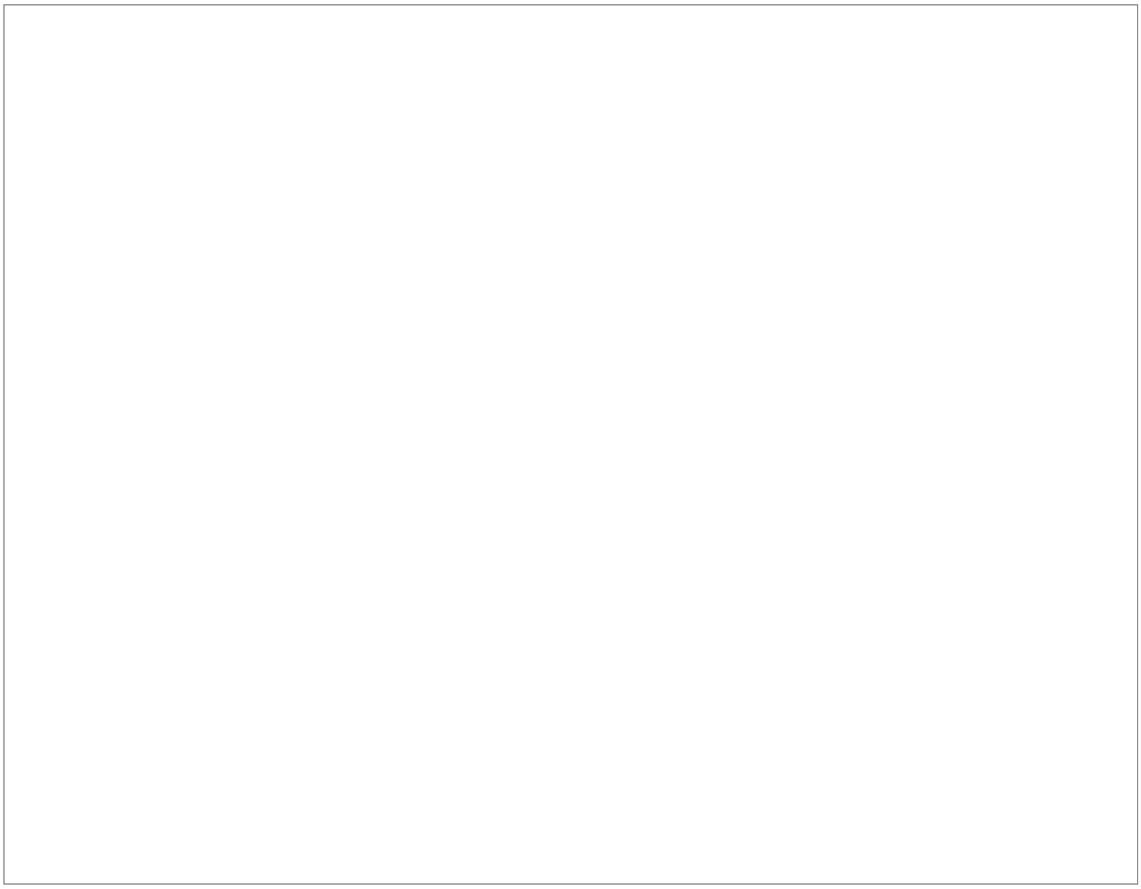
3.3. Projekt interfejsu użytkownika

Docelowym operatorem obsługującym licznik jest dorosły pracownik sklepu, na kierowniczym stanowisku. Posiada on średnie umiejętności obsługi komputera oraz preferuje proste rozwiązania i programy o małej liczbie okien. To z myślą o nim zaprojektowano graficzny interfejs użytkownika.

Panel użytkownika składa się z następujących elementów:

I. przycisk ‘Zacznij zliczanie’ - włączenie podłączonej kamerki, rozpoczęcie zliczania,
II. przycisk ‘Zatrzymaj zliczanie’ - zatrzymanie przetwarzania, wyzerowanie liczników; przycisk staje się aktywny dopiero po uruchomieniu przetwarzania,
III. przycisk ‘Eksportuj raport’ - zapis wyników przetwarzania do pliku, staje się aktywny dopiero po zakończeniu przetwarzania,
IV. obraz z kamery – przedstawia obraz aktualnie przetwarzany przez licznik,
V. grupa etykiet informacyjnych – aktualne wartości licznika tradycyjnego oraz licznika inteligentnego,
VI. panel wyboru kierunku liczenia - pozwala na wybór z której strony znajduje się wejście do pomieszczenia.

Rysunek 3.4 przedstawia widok interfejsu użytkownika po uruchomieniu programu, w czasie zliczania.



Rys. 3.4 Interfejs użytkownika

By ułatwić użytkownikowi korzystanie z programu podczas pracy blokowane są niektóre funkcje interfejsu. Oznacza to, że przed rozpoczęciem przetwarzania zablokowana jest możliwość zatrzymania przetwarzania oraz generowania raportu. Po rozpoczęciu przetwarzania obrazu blokowana jest możliwość rozpoczęcia przetwarzania oraz zmiany parametrów początkowych (liczenie w prawo oraz liczenie w lewo). Po zakończeniu przetwarzania odblokowywana jest możliwość generowania raportu z przetwarzania. W oknie interfejsu umieszczone są również aktualne wartości licznika prymitywnego i inteligentnego oraz wyjaśnienia znaczeń wyświetlanych wartości. Na rys. 3.4 przedstawiony jest stan interfejsu w trakcie przetwarzania.

3.4. Projekt struktury danych

Jedynym plikiem wejściowym programu jest obraz z kamery, który jest przetwarzany w czasie rzeczywistym. Wczytywany film może mieć maksymalną rozdzielczość równą 1920x1080 pikseli. Dopuszczalna częstość klatek jest zależna od sprzętu, na którym uruchamiany jest program. Domyślnie przyjmuje się częstość na poziomie 30 ramek na sekundę (*FPS*).

Każda *klatka* po kolei jest przetwarzana przez sieć neuronową w celu detekcji *postaci*. Dla każdej postaci zbierane są odpowiednie *wektory* z każdej ramki, na której się pojawiła. Wektory te służą do porównywania osób w celu ich reidentyfikacji.

4. IMPLEMENTACJA SYSTEMU

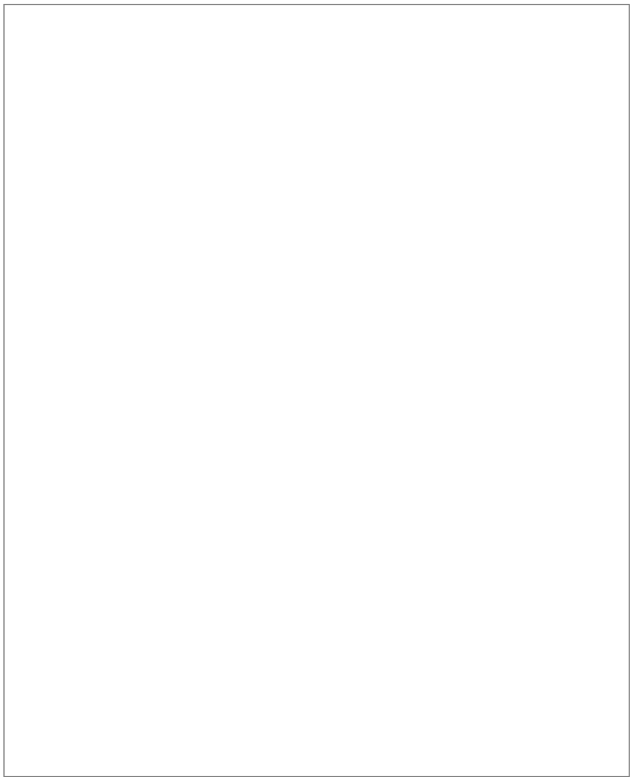
Aby poprawnie policzyć daną *osobę*, najpierw musi być ona prawidłowo rozpoznana na *klatce* filmu. Wykryta *postać* pojawia się na kolejnych klatkach w innych lokalizacjach, dlatego konieczne jest śledzenie jej w odpowiedni sposób. Dzięki temu możliwe jest zbieranie kolejnych informacji o obiekcie na podstawie jego zdjęć, wyciętych z klatek filmu. Te dane są wykorzystywane następnie do reidentyfikacji osób pojawiających się po raz kolejny.

Przedstawiony proces podzielono na cztery główne części: detekcji, śledzenia, liczenia oraz reidentyfikacji. W niniejszym rozdziale omówiono sposoby implementacji każdej z tych części.

4.1. Detekcja

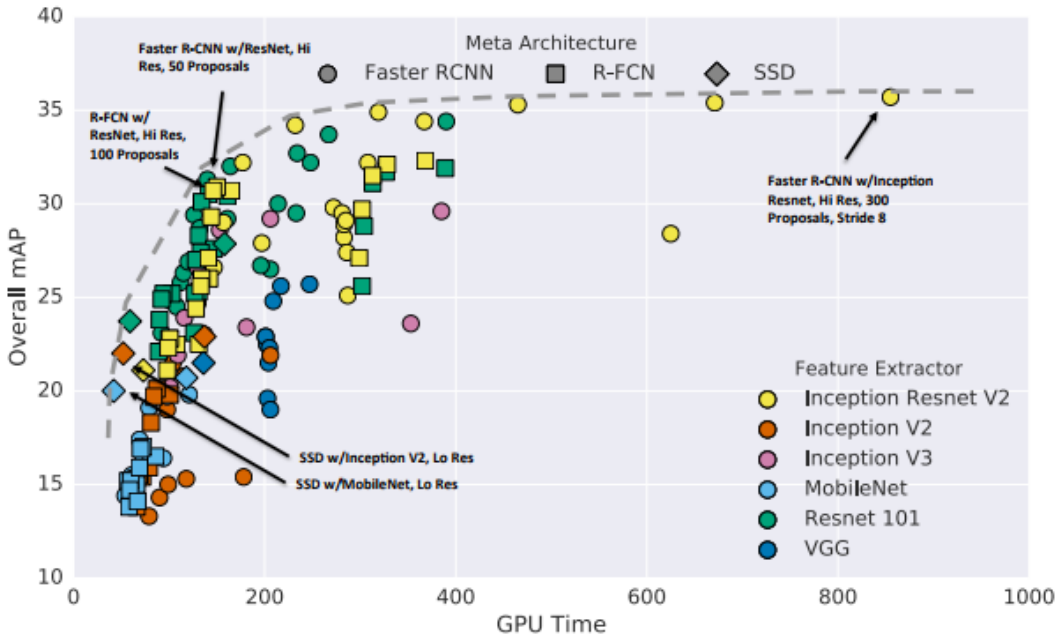
Pierwszym sposobem detekcji jaki zrealizowano było podejście oparte na operacjach morfologicznych. Proces ten polega na wyodrębnieniu ruchomych obiektów z nieruchomego tła filmu. Następnie pole powierzchni wybranego obiektu jest porównywane z określoną wartością progową i na tej podstawie jest podejmowana decyzja czy dany obiekt jest postacią człowieka czy nie. Wynik detekcji na pojedynczym zdjęciu przedstawiono na rys. 4.1. Zielonymi ramkami zaznaczono rozpoznane obiekty, a czerwone kropki w ich środkach wyznaczają środki ich mas.

Sposób detekcji oparty na operacjach morfologicznych wymaga całkowicie nieruchomego tła, co ogranicza wykorzystanie systemu w niektórych okolicznościach (np. przy ruchomych drzwiach). Dodatkowo generowanych jest stosunkowo dużo niewłaściwych detekcji, np. rozpoznanie cieni jako człowieka. Poprawność procesu zależy też od ustawienia kamery, co zmniejsza uniwersalność systemu. Po przeprowadzeniu szeregu testów manualnych ustalono, że takie rozwiązanie jest zbyt nieprzewidywalne dla zaprojektowanego licznika.



Rys. 4.1 Zrzut ekranu z aplikacji deweloperskiej przedstawiający wynik detekcji opartej na operacjach morfologicznych

Zdecydowano się na przeprowadzenie rozpoznawania postaci na obrazie z filmu z wykorzystaniem głębokiej sieci neuronowej. W tym celu należało dobrać odpowiednią sieć neuronową oraz metodę przeprowadzenia detekcji z wykorzystaniem tej sieci. Kryteriami w ich wyborze były szybkość działania oraz ilość poprawnych detekcji.



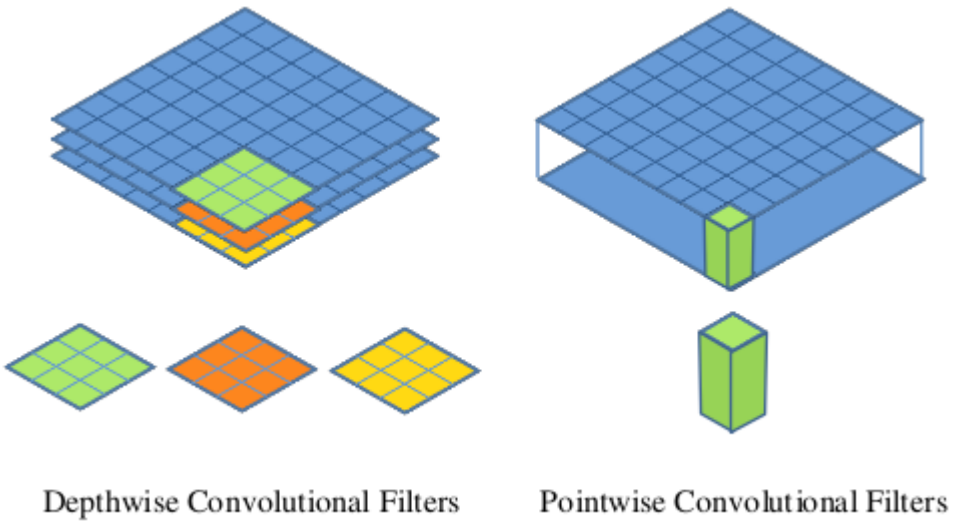
Rys. 4.2 Wykres przedstawiający stosunek średniej precyzji (*mAP*) do czasu wykonania operacji dla trzech różnych architektur oraz sześciu różnych modeli sieci [2]

Na rysunku 4.2 przedstawiono wykres zależności średniej precyzji algorytmu detekcji (*mAP*) od czasu jego wykonania na jednostce graficznej (*GPU*) dla rozwiązań wykorzystujących jedną z trzech meta-architektur [2]:

I. Faster R-CNN [3],
II. R-FCN [4],
III. Single Shot Detection (SSD) [5],

oraz jeden z sześciu modeli sieci neuronowych: Inception Resnet V2, Inception V2, Inception V3, MobileNet, Resnet 101 oraz VGG. Poza wymienionymi na wykresie (rys. 4.2) architekturami, porównano również podejście You Only Look Once (YOLO) [6], które jest szybsze niż SSD, natomiast wykazuje się mniejszą precyzją. Biorąc pod uwagę kryteria wydajności i poprawności detekcji, metodę Single Shot Detection uznano za najbardziej optymalną. Przetwarzając średnio od 22 do 46 ramek na sekundę, jest bardziej wydajna niż metoda Faster R-CNN oraz charakteryzuje się większą ilością poprawnych detekcji niż R-FCN.

Analizując wykres (rys. 4.2) przy wyborze modelu sieci, zdecydowano się na użycie stosunkowo nowej (z kwietnia 2017), wydajnej sieci konwolucyjnej MobileNet [7]. Stworzona do wykorzystywania na urządzeniach mobilnych charakteryzuje się małym rozmiarem oraz szybkim działaniem. Model MobileNet jest oparty na metodzie depthwise separable convolutions, która polega na rozłożeniu pointwise convolution na trzy poziomy i nałożeniu filtrów na każdy kanał wejściowy osobno (rys. 4.3).



Rys. 4.3 Porównanie depthwise i pointwise convolutional filters [8]

Wykorzystany w projekcie model sieci był uprzednio wytrenowany [9] na danych Common Objects in Context (COCO) oraz PASCAL VOC.

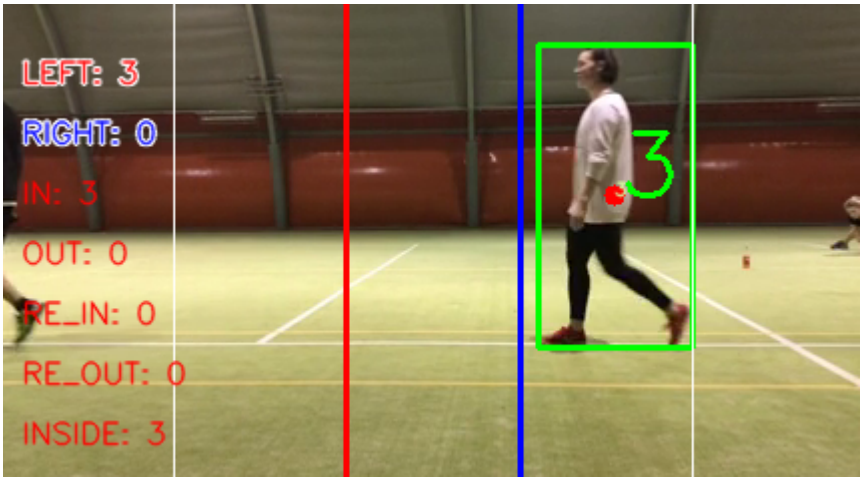
Do wczytywania danych wejściowych wykorzystano bibliotekę OpenCV 3.3, która zawiera moduł głębokich sieci neuronowych (deep neural network – dnn) umożliwiający operowanie na sieciach neuronowych. Poniżej (tekst 1) przedstawiono fragment kodu wykonujący operacje wczytania danych sieci neuronowej MobileNet oraz detekcji.

```
net =
cv2.dnn.readNetFromCaffe("../caffe/MobileNetSSD_deploy.prototxt.txt',
"../caffe/MobileNetSSD_deploy.caffemodel")

detections = net.forward()
```

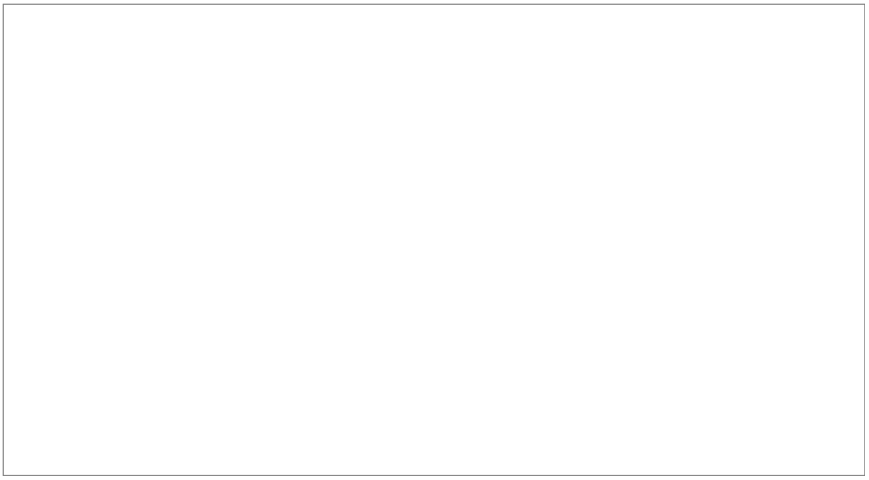
Tekst 1 Fragment kodu przedstawiający operacje na sieci neuronowej.

Operacja detekcji przeprowadzana jest na każdej ramce filmu po kolei. Funkcja forward() (tekst 1) wywołana na obiekcie reprezentującym sieć (net) zwraca listę elementów rozpoznanych na obrazie. Zwracane obiekty zawierają informację o klasie i wartości procentowej pewności (confidence) poprawnego rozpoznania obiektu. Po przeprowadzeniu testów (*5.2 Dobieranie progu pewności poprawnego wykrycia*) ustawiono próg pewności na 30%, czyli jeśli pewność poprawnej detekcji jest większa niż 30%, to rozpoznany obiekt jest uznawany za *postać*. Przy doborze zbyt niskiego progu występowały nieprawidłowe detekcje, natomiast przy zbyt wysokim progu właściwe obiekty nie były brane pod uwagę. Każdy obiekt rozpoznany jako nowa postać, jeśli pojawił się w wyznaczonych granicach, jest zapisywany do tablicy postaci (postures). Granice są zaznaczone na obrazie z filmu (rys. 4.4) białymi liniami , a ich pozycje są dostosowywane do konkretnych danych wejściowych. Przestrzeń detekcji obiektów jest ograniczona w celu niewykrywania niepożądanych postaci, np. przechodzących za szklanymi drzwiami.



Rys. 4.4 Zrzut ekranu z wersji deweloperskiej licznika przedstawiający poprawną detekcję.

Wybrana metoda detekcji jest przeprowadzana jest na dwuwymiarowym obrazie. Takie podejście wyklucza rozpoznanie obiektu, który jest całkowicie zakryty przez inny obiekt oraz nie daje pewności poprawnej detekcji jeśli obiekt jest częściowo zasłonięty.


Rys. 4.5 Zrzut ekranu z wersji deweloperskiej licznika przedstawiający niepoprawną detekcję.

Możliwe jest również nieprawidłowe rozpoznanie postaci, gdy system wykryje jeden obiekt w środku drugiego. Taki przypadek przedstawiono na rys. 4.5, gdzie postać o identyfikatorze 7 została wykryta jako dwa osobne obiekty.

Porównując metodę detekcji wykorzystującą głębokie sieci neuronowe do metody opartej na operacjach morfologicznych, ta pierwsza wymaga dużej ilości obliczeń. Z tego powodu potrzebny jest sprzęt o lepszych parametrach technicznych, na którym jest uruchamiana aplikacja. Natomiast wyniki metody używającej operacje morfologiczne są poniżej poziomu akceptacji.

4.2. Śledzenie

Głównym problemem związanym z poprawnym liczeniem osób jest śledzenie wykrytych postaci na kolejnych ramkach filmu. Każdy obiekt postaci posiada między innymi takie atrybuty, jak:

- unikalny identyfikator (id),
- współrzędne środka masy x i y,
- czas ostatniego wykrycia w formacie „minuty:sekundy” (lastTime).

Na podstawie powyższych właściwości nowa detekcja jest porównywana ze wszystkimi z poprzednich ramek. Porównania te są wykonywane na podstawie różnic przestrzennych (wartości x i y) oraz czasowych postaci. Wartość kryterium czasowego odwołuje się do czasu rzeczywistego, dlatego jest dopasowana do mocy sprzętu, na którym uruchamiany jest program. Wartości kryteriów przestrzennych są uzależnione od parametrów danych wejściowych takich jak: prędkość poruszania się osób oraz stosunek wielkości postaci na obrazie do wielkości tego obrazu. Dla filmu testowego (zbiór testowy **I**) przyjęto założenie, że detekcje tej samej postaci na kolejnych ramkach nie są oddalone od siebie o więcej niż:

- 1/5 szerokości filmu (w poziomie),
- 1/4 wysokości filmu (w pionie).

Przyjęto również, że kolejna detekcja tej samej postaci nastąpi w czasie maksymalnie 4 sekund.

Dobrane parametry nie gwarantują stuprocentowej skuteczności algorytmu śledzenia, ponieważ możliwe jest zidentyfikowanie jednej postaci jako dwa różne obiekty, czyli nadanie jej dwóch różnych identyfikatorów. Dodatkową trudnością jest niedokładność algorytmu detekcji. Jeśli obiekt jest rozpoznany prawidłowo na początku przejścia, a następnie nie jest wykryty na kilku kolejnych ramkach i znów jest rozpoznany prawidłowo, może znajdować się w dużej odległości zarówno przestrzennej, jak i czasowej od poprzedniej detekcji. Skutkiem takiego działania jest nadanie tej osobie nowego identyfikatora.

Rozważano również zrealizowanie śledzenia postaci z wykorzystaniem głębokich sieci neuronowych w sposób podobny do procesu reidentyfikacji (*4.4 Reidentyfikacja*). Stwierdzono jednak, że ilość potrzebnych do tej metody obliczeń spowodowałaby znaczny spadek wydajności licznika.

4.3. Liczenie

Liczenie osób odbywa się w obydwu kierunkach. Liczone są zarówno postacie wchodzące do pomieszczenia, jak i wychodzące z niego. Dana postać jest zliczana tylko wtedy, jeśli zostanie rozpoznana i zidentyfikowana jako ta sama przed i po przejściu przez określoną linię. Wyznaczono dwie linie (na rys. 4.4: czerwoną i niebieską): jedna do zliczania osób wchodzących, druga do zliczania osób wychodzących. Aby uniknąć nieprawidłowości w liczeniu spowodowanych niepoprawną detekcją, dodano następujący warunek - aby dana postać została policzona musi być rozpoznana na przynajmniej trzech ramkach przed przejściem przez

odpowiednią linię. Linia ta jest wybierana zgodnie z kierunkiem ruchu obiektu, w zależności czy wchodzi on do środka pomieszczenia, czy z niego wychodzi. Każdy stworzony obiekt ma przypisaną wartość stanu określającą, czy został on już policzony. Uniemożliwia to kilkukrotne policzenie tej samej postaci w czasie jednego przejścia. Licznik osób wchodzących jest zwiększany po spełnieniu następujących warunków:

- ostatnio zapisane współrzędne środka wykrytej postaci wyznaczają punkt za linią graniczną,
- przedostatnio zapisane współrzędne środka wykrytego obiektu wyznaczają punkt przed linią graniczną,
- wykryty obiekt ma stan ‘niepoliczony’,
- wykryty obiekt był rozpoznany wcześniej na przynajmniej trzech ramkach.

Program jest wyposażony również w licznik osób wychodzących, który działa analogicznie do przedstawionego powyżej.

Możliwym, lecz niepożądanym działaniem algorytmu śledzenia jest nadanie tej samej osobie nowej tożsamości (nowego identyfikatora). Jeśli stanie się to w okolicy linii, przy której wykonuje się liczenie osób, to taka postać nie zostanie policzona, ponieważ nie była rozpoznana na trzech ramkach.

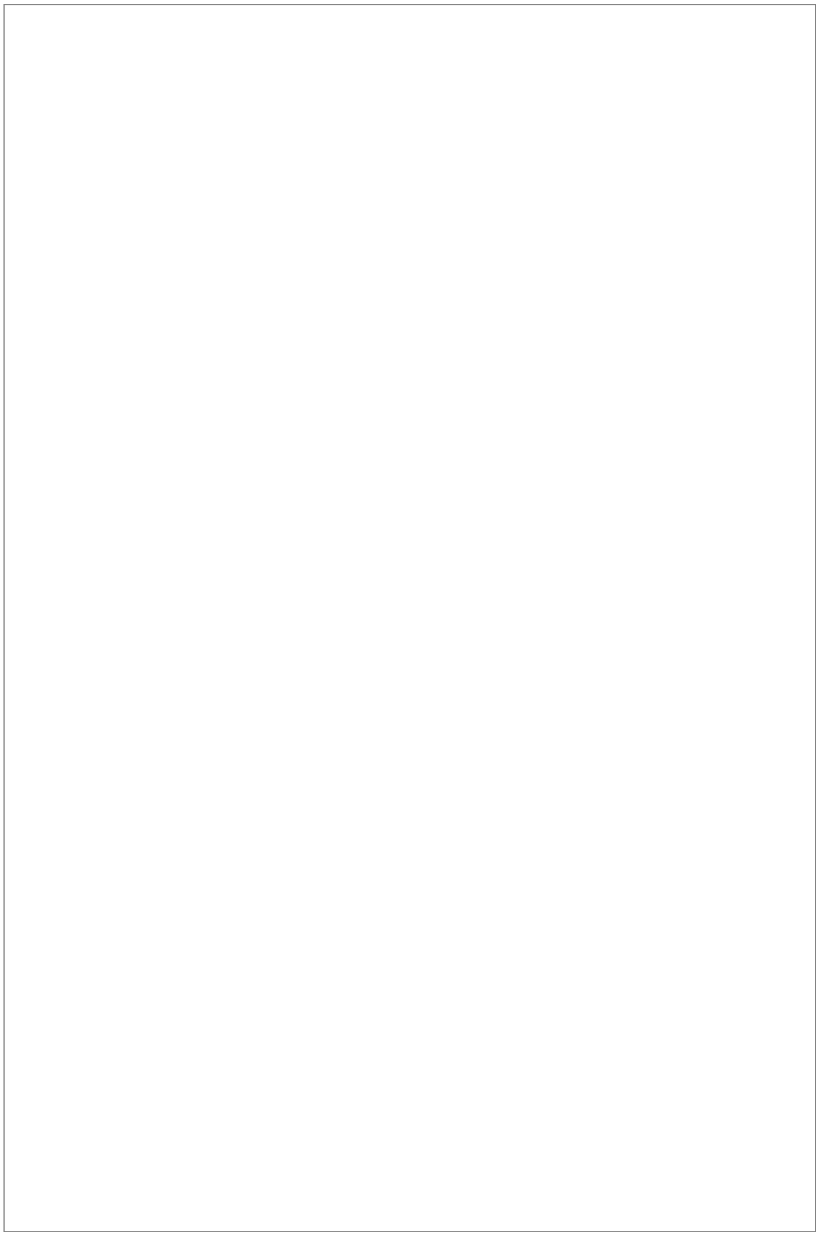
4.4. Reidentyfikacja

Reidentyfikacja polega na przyporządkowaniu obrazu wchodzącej lub wychodzącej aktualnie postaci do rejestru wcześniej rozpoznanych osób. Do tego celu wykorzystywane są *obrazy postaci* uzyskane w procesie detekcji i śledzenia. Aby ułatwić zadanie reidentyfikacji każda osoba ma zapisywany swój aktualny stan: *w pomieszczeniu* lub *poza pomieszczeniem*.

W momencie rozpoznania akcji *wejście* możliwe jest dopasowanie aktualnie identyfikowanej postaci jako nowa lub któraś z osób, które wyszły z pomieszczenia w przeciągu trwającej sesji zliczającej. Przyjęto założenie, że na początku pomieszczenie jest puste. W związku z tym, w momencie rozpoznania akcji *wyjście* postać zawsze powinna zostać zreidentyfikowana jako jedna z osób aktualnie posiadających stan *w pomieszczeniu*. Jednakże ze względu na sytuację wyjątkową I opisaną w rozdziale *2.1.6 Sytuacje wyjątkowe*, konieczne jest także umożliwienie klasyfikacji postaci wychodzącej jako nowa osoba (gdy postać nie jest wystarczająco podobna do żadnej z zapisanych osób).

Reidentyfikacja składa się z dwóch kolejnych procesów: transformacji oraz klasyfikacji.

4.4.1. Transformacja z użyciem sieci neuronowej



Rys. 4.6 Schemat warstw sieci VGG-19 [10]

Każde ze zdjęć uzyskanych w procesie śledzenia ulega transformacji na wektor jednowymiarowy. W tym celu skorzystano z modelu sieci neuronowej VGG-19 wbudowanego w bibliotece Keras. Każde zdjęcie podawane jest osobno na wejście sieci neuronowej, następnie pobierane jest wyjście jednej z końcowych warstw sieci: conv5_2. Schemat warstw sieci został przedstawiony na rys. 4.6. Wybór sieci oraz warstwy został poprzedzony testowaniem otrzymanych wyników pod względem wydajności oraz poprawności. Dokładny opis testów znajduje się w rozdziale *5.3 Porównanie warstw sieci VGG-19 oraz MobileNet*.

Wyjście z sieci neuronowej ma postać wielowymiarowej tablicy (o rozmiarach 14x14x512 dla warstwy conv5_2). Testowano trzy różne sposoby transformacji tej tablicy na wektor jednowymiarowy. Po przeprowadzeniu testów skuteczności, opisanych w rozdziale

5.4 Porównanie metod transformacji, zdecydowano się na metodę uwzględniającą podział pierwszego wymiaru tablicy na trzy części. Metoda ta została schematycznie przedstawiona na rys. 5.5. Poniżej (tekst 2) przedstawiono fragment kodu wykonujący operację transformacji wyjścia sieci neuronowej na wektor jednowymiarowy.

```
def transform(self, imgT):
        x = image.img_to_array(imgT)
        x = np.expand_dims(x, axis=0)
        x = preprocess_input(x)

        a = self.model.predict(x)   //model - neural network

        up = (a[0][0:4].mean(axis=0)).mean(axis=0)
        mid = (a[0][4:10].mean(axis=0)).mean(axis=0)
        down = (a[0][10:14].mean(axis=0)).mean(axis=0)
        ret = np.concatenate((up, mid, down), axis=0)

        return ret
```

Tekst 2 Implementacja transformacji wyjścia sieci neuronowej na wektor jednowymiarowy

4.4.2. Podobieństwo wektorów

Podczas przechodzenia każda z osób zostaje wykryta na ok. 10 klatkach. Oznacza to, że do zadania dopasowania postaci do konkretnej osoby mamy do dyspozycji ok. 10 wektorów. Wektory te porównywane są do wektorów wcześniej zidentyfikowanych postaci, a po podjęciu decyzji dopisywane do listy wektorów konkretnej osoby.

Jako kryterium porównawcze dwóch wektorów przyjęto normę euklidesową ich różnicy. W celu przyspieszenia procesu wyszukania najbardziej podobnych wektorów użyto implementacji kd-drzewa biblioteki SciPy. Drzewo to jest przebudowywane w momencie podejmowania decyzji dla każdej nowej postaci.

W zależności od zdarzenia wejście/wyjście zbiór wektorów, do których porównywana jest aktualnie przetwarzana postać jest różny. Tak jak wcześniej wspomniano osoby wchodzące mogą zostać zreidentyfikowane – przyporządkowane do osób, które wyszły. Natomiast postaci wychodzące są przyporządkowywane do osób, które aktualnie znajdują się w pomieszczeniu. W związku z tym w rzeczywistości budowane są dwa osobne drzewa – jedno dla wektorów osób w pomieszczeniu, drugie dla wektorów osób poza pomieszczeniem.

4.4.3. Klasyfikacja

Oprócz znanej odległości między wektorami, istotnym elementem algorytmu jest również podejmowanie decyzji na podstawie uzyskanych wartości. Dla każdego z 10 klasyfikowanych wektorów otrzymujemy bowiem *k* najbliższych sąsiadów (wektorów) wraz z informacją o przynależności danych wektorów do konkretnych osób. Przykładowo załóżmy, że dla jednej postaci wykryto jej 3 detekcje, więc po transformacji uzyskaliśmy 3 wektory (a, b, c). Wartości zwrócone przez kd-drzewo(k=4) mogą mieć następującą postać:

a: [500 (id=0), 530 (id=0), 550 (id=1), 800 (id=2)]

b: [800 (id=0), 820 (id=1), 890 (id=1), 910 (id=0)]

c: [300 (id=1), 310 (id=0), 340 (id=0), 800 (id=2)]

Oznacza to, że najbliższym wektorem dla wektora a jest wektor przypisany do osoby o id równym 0, odległość między tymi wektorami wynosi 500, dla wektora b wektor osoby o id 0, odległość wynosi 800, dla wektora c wektor osoby o id 1 i odległości 300.

Łatwo zauważyć mnogość możliwych algorytmów podjęcia decyzji, np.:

I. wybór osoby, której najwięcej wektorów zostało sklasyfikowanych jako najbliższe – w tym wypadku id = 0 (mostFrequentNearest),
II. wybór osoby, której najwięcej wektorów zostało sklasyfikowanych wśród pierwszych 4 najbliższych – w tym wypadku id = 0 (id=0 – 6 wystąpień, id=1 – 4 wystąpienia, id=2 – 2 wystąpienia),
III. wybór osoby, której wektor uzyskał najbliższą odległość – w tym wypadku id = 1,
IV. wybór osoby, której wektory uzyskały sumarycznie najmniejszą średnią odległość wśród pierwszych czterech odległości – w tym wypadku id = 0 (565) (id=1 – 640, id=2 – 800) (kMultiplyDistance).

Metody II i III nie zostały w ogóle zaimplementowane, ze względu na to, że przy metodzie drugiej bardzo łatwo o zły wynik, gdyż ani odległość ani kolejność wektora na liście nie są brane pod uwagę. Natomiast w metodzie trzeciej w rzeczywistości rezygnuje się z wielu uzyskanych danych, a pojedynczy błąd może przesądzić o złym sklasyfikowaniu osoby. Dlatego też podczas implementacji porównano metodę I – mostFrequentNearest oraz metodę IV – kMultiplyDistance. Na podstawie przeprowadzonych testów (*5.5 Różne algorytmy klasyfikacji*) ostatecznie zdecydowano się na przyjęcie metody I – mostFrequentNearest.

4.4.4. Możliwe problemy

W wyniku ewentualnego nieprawidłowego przypisania wektorów do innej osoby, całe późniejsze działanie aplikacji naznaczone jest błędem. Wynika to z faktu, że przy późniejszym porównywaniu wektory, które mogą zostać uznane za najbliższe, tak naprawdę powinny być przypisane do innej osoby. Skutkiem błędu identyfikacji, oprócz nieprawidłowego przypisania wektorów, jest również nieprawidłowo określony stan osób (*w pomieszczeniu / poza pomieszczeniem*). Wówczas system może nie mieć możliwości prawidłowego sklasyfikowania kolejnej osoby. Próbą rozwiązania tego problemu było ustalenie *progu reidentyfikacji* opisane w rozdziale *5.6 Dobieranie progu reidentyfikacji.*

4.5. Interfejs użytkownika

Interfejs użytkownika został zaimplementowany w osobnym wątku by umożliwić jednoczesną komunikację użytkownika z programem, wyświetlenie oraz przetworzenie obrazu. Sygnał rozpoczęcia, zatrzymania oraz częściowe wyniki przetwarzania przekazywane są do

wątku interfejsu za pomocą kolejek. W implementacji użyte zostały klasy wątku oraz kolejki ze standardowej biblioteki Pythona.

Do implementacji interfejsu została użyta biblioteka graficzna PyQt5. Główne okno programu dziedziczy od QMainWindow, a obraz wyświetlany jest jako QWidget, który jest zmienną w zaimplementowanym oknie głównym. W implementacji interfejsu użyte zostały również przyciski oraz etykiety. Wygląd okna został stworzony poprzez użycie narzędzia QtDesigner oraz przechowywany jest w osobnym pliku.

Przykładowy raport końcowy przedstawiono w tekście 3.

```
Czas rozpoczęcia sesji zliczającej: Mon Dec  4 20:39:36 2017
ID: 0 - wejscie - Mon Dec  4 20:39:36 2017
ID: 1 - wejscie - Mon Dec  4 20:39:57 2017
ID: 2 - wejscie - Mon Dec  4 20:40:19 2017
ID: 3 - wejscie - Mon Dec  4 20:40:37 2017
ID: 0 - wyjscie - Mon Dec  4 20:40:54 2017
ID: 2 - wyjscie - Mon Dec  4 20:41:09 2017
ID: 3 - wyjscie - Mon Dec  4 20:41:27 2017
ID: 1 - wyjscie - Mon Dec  4 20:41:43 2017

Zakończenie sesji zliczajacej: Mon Dec  4 20:43:43 2017
Końcowy stan licznika:
Tradycyjny licznik osób:
      4 osób weszło oraz 4 osób wyszło.
Inteligentny licznik osób:
      0 osób zostało zreidentyfikowanych wchodząc oraz 4 osób zostało
zreidentyfikowanych wychodząc.
Obecnie w środku znajduje się 0 osób.
```

Tekst 3 Przykładowy raport końcowy

5. EKSPERYMENTY

W niniejszym rozdziale opisano przeprowadzone eksperymenty, wykonane w celu dobrania parametrów dla procesów detekcji oraz reidentyfikacji.

Skuteczność metody reidentyfikacji zależy od warstwy sieci, metody transformacji jej wyjścia na *wektory postaci*, metody klasyfikacji, a także doboru progu reidentyfikacji. Zaprezentowano wyniki testów poprawności programu dla różnych wartości powyższych parametrów. Na początku zbadane zostały warstwy sieci neuronowych, a następnie, na najbardziej efektywnych z nich, przeprowadzono testy różnych metod transformacji. Przedstawiono również test służący dobraniu najbardziej optymalnego progu reidentyfikacji.

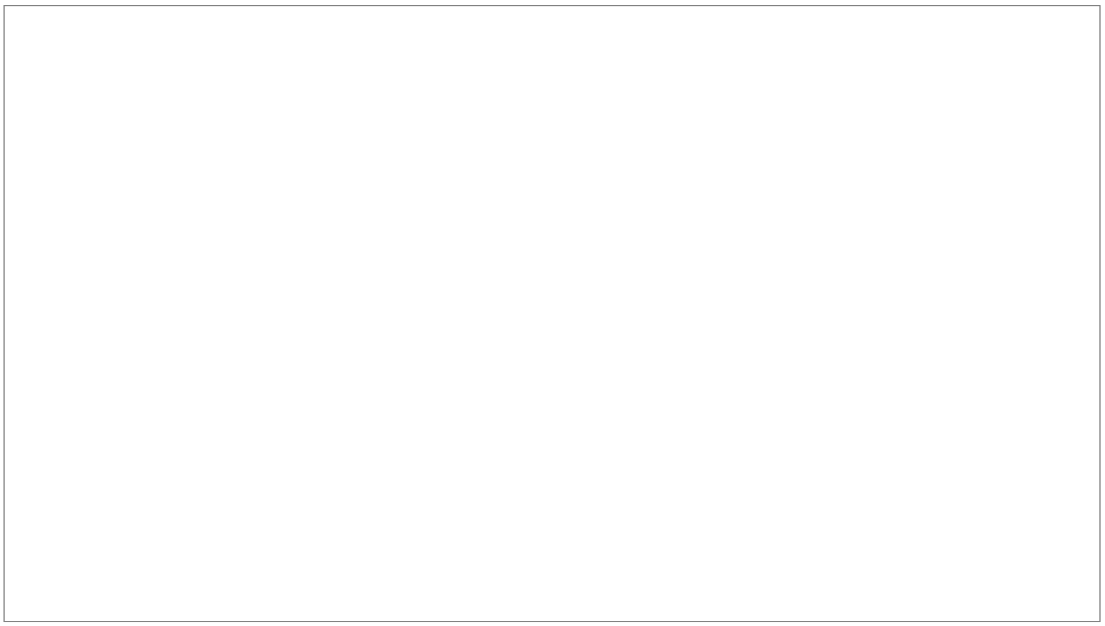
5.1. Zbiór danych testowych

Eksperymenty i testy przeprowadzono na następujących zbiorach testowych:

I. Zbiór testowy A jest krótkim filmem przedstawiającym przejście dwunastu osób z prawej strony na lewą oraz powrót tych samych osób, w tej samej kolejności z lewej strony na prawą.
II. Zbiór testowy B jest krótkim filmem przedstawiającym przejście dwóch osób w lewą stronę. Następnie te same osoby, w tej samej kolejności idą w prawą stronę, jeszcze raz w lewą i jeszcze raz w prawą.

5.2. Dobieranie progu pewności poprawnego wykrycia

Przetestowano cztery różne wartości progów pewności poprawnego rozpoznania obiektu. Porównano ilość błędnych detekcji w stosunku do wszystkich detekcji. Na rys. 5.1 przedstawiono wyniki eksperymentu.



Rys. 5.1 Wykres przedstawiający ilość błędnych detekcji w stosunku do wszystkich detekcji dla różnych wartości pewności sieci neuronowej

Dla progu równego 10% ilość błędnych detekcji jest dwukrotnie większa niż dla progu równego 30%. Ustawienie progu o wartości 90% eliminuje błędy detekcji, jednakże takie rozwiązanie skutkuje mniejszą liczbą detekcji jednej postaci. W następstwie takiego działania produkowanych jest zbyt mało danych do poprawnego działania algorytmu śledzenia, liczenia i reidentyfikacji.

5.3. Porównanie warstw sieci VGG-19 oraz MobileNet

Przy wyborze sieci kierowano się łatwością implementacji oraz szybkością działania. Z powodu ograniczonego czasu trwania projektu zdecydowano się porównać dwie sieci neuronowe. Pierwszą z nich była MobileNet, którą użyto już wcześniej. Drugim wyborem była sieć VGG-19, której przykład znaleziono w dokumentacji Keras [11]. Przy testowaniu obu sieci korzystano z implementacji sieci dostępnych w bibliotece Keras.

W celu uzyskania jak najlepszych rezultatów dokonano porównania wyników otrzymanych poprzez przetwarzanie różnych warstw sieci neuronowych. Zaimplementowaną aplikację uruchomiono dostarczając te same dane testowe oraz parametry do każdej z powyższych sieci. Danymi testowymi było krótkie nagranie na którym powinno dojść do 6 poprawnych reidentyfikacji (zbiór testowy II). Poprawność określono jako procentowy stosunek poprawnych reidentyfikacji do wszystkich możliwych.

Rys. 5.2 Porównanie rezultatów uzyskanych korzystając z sieci MobileNet

Rys. 5.3 Porównanie rezultatów uzyskanych korzystając z sieci VGG-19

Przedstawione testy pokazują, że nie ma prostej zasady na której warstwie osiągnięte zostaną najlepsze wyniki. Dla sieci VGG-19 najlepsze wyniki osiągnięto dla warstw conv1_1, conv1_2, pool_1 oraz conv5_2. Są to warstwy początkowe oraz jedna z końcowych. Dla sieci MobileNet najlepsze wyniki osiągnięto dla warstw conv_dw_4_bn oraz conv_dw_5_bn, które są warstwami środkowymi.

Najlepsze warstwy ponownie przetestowano na większym zbiorze testowym I. Uzyskane wyniki przedstawiono w tabeli 2.

Tabela 2 Poprawność reidentyfikacji na najlepszych warstwach sieci VGG-19 oraz MobileNet

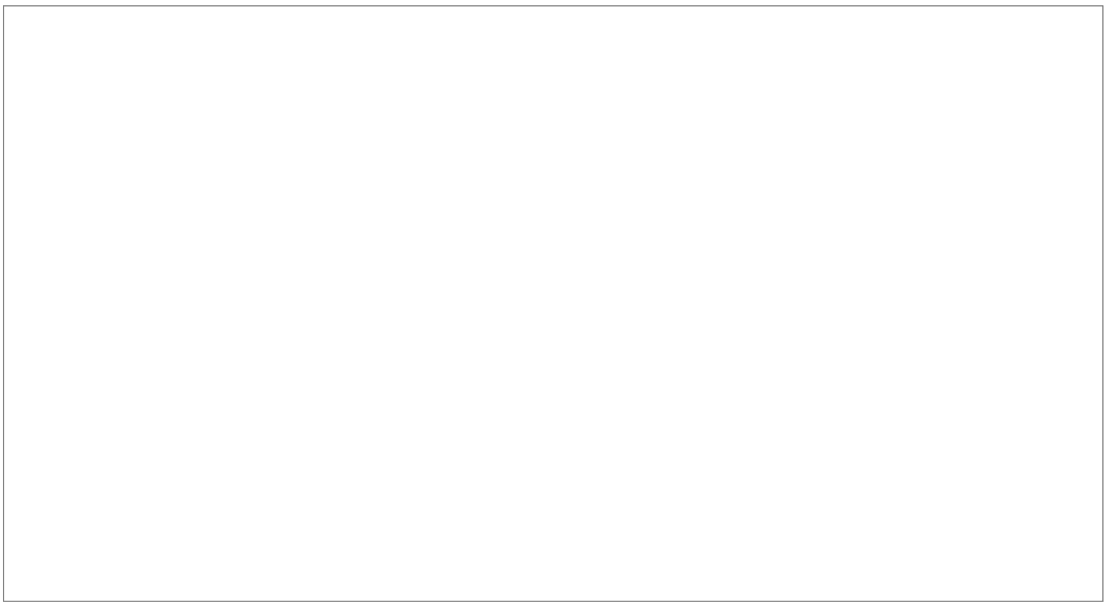
Nazwa sieci	Nazwa warstwy	Liczba poprawnych reidentyfikacji		Liczba możliwych reidentyfikacji	Średnia poprawność
		pierwsza próba	druga próba		
MobileNet	conv_dw_4_bn	2	1	12	12,50%
MobileNet	conv_dw_5_bn	2	5	12	25,00%
VGG-19	conv1_1	2	2	12	16,67%
VGG-19	conv1_2	3	1	12	16,67%
VGG-19	pool_1	0	2	12	8,33%
VGG-19	conv5_2	8	7	12	62,5%

Łatwo zauważyć, że średnia poprawność jest wysoka tylko na jednej wśród wcześniej wybranych warstw. Została ona wybrana do dalszych testów.

5.4. Porównanie metod transformacji

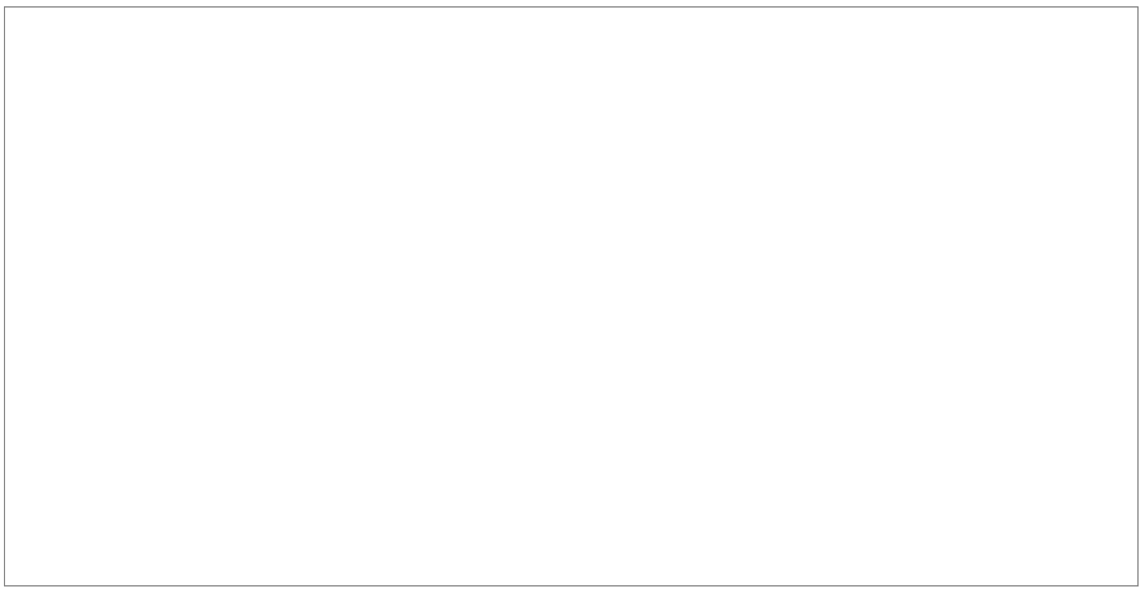
Wyjście warstwy sieci neuronowej conv5_2 używanej do reidentyfikacji ma postać trójwymiarowej tablicy (14x14x512). Z tablicy tej tworzony jest jednowymiarowy wektor, który porównywany jest z wektorami innych osób (*4.4.1 Transformacja z użyciem sieci neuronowej*). Zaproponowano trzy rozwiązania kompresji trzech wymiarów do jednowymiarowego wektora.

Pierwsze podejście (metoda 1), przedstawione na rys. 5.4 polega na wyliczeniu średniej arytmetycznej każdej warstwy 14x14, a następnie stworzeniu z uzyskanych wyników wektora jednowymiarowego.



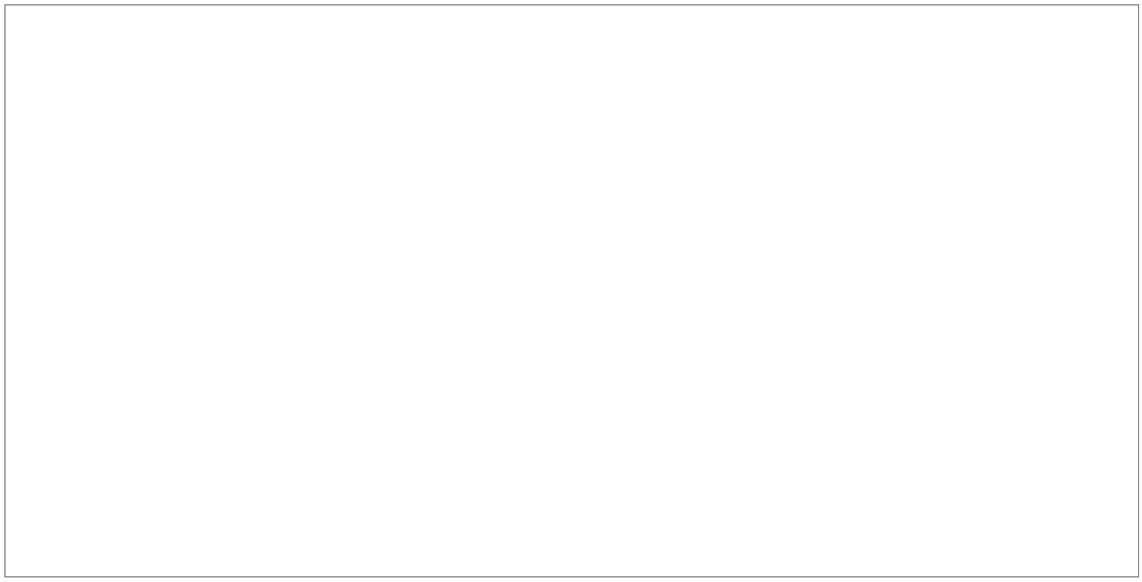
Rys. 5.4 Metoda 1. transformacji wyjścia warstwy sieci conv5_2 na wektor

Ponieważ reidentyfikacja jest wykonywana na podstawie zdjęcia całej sylwetki, przeprowadzono eksperyment sprawdzający, czy rozdzielenie trójwymiarowej tablicy na 3 części poprawi uzyskiwane wyniki. Dzięki takiemu podziałowi porównanie ma się odbywać osobno dla górnych części zdjęcia (głowa człowieka), środkowych (tułów) i dolnych (nogi). Dlatego też drugi sposób (metoda 2, rys. 5.5) polegał na podzieleniu pierwszego wymiaru tablicy (o rozmiarze 14) na trzy części, odpowiednio: [0:3], [4:9], [10:13], a następnie analogicznie jak w sposobie pierwszym wyliczeniu średnich arytmetycznych.



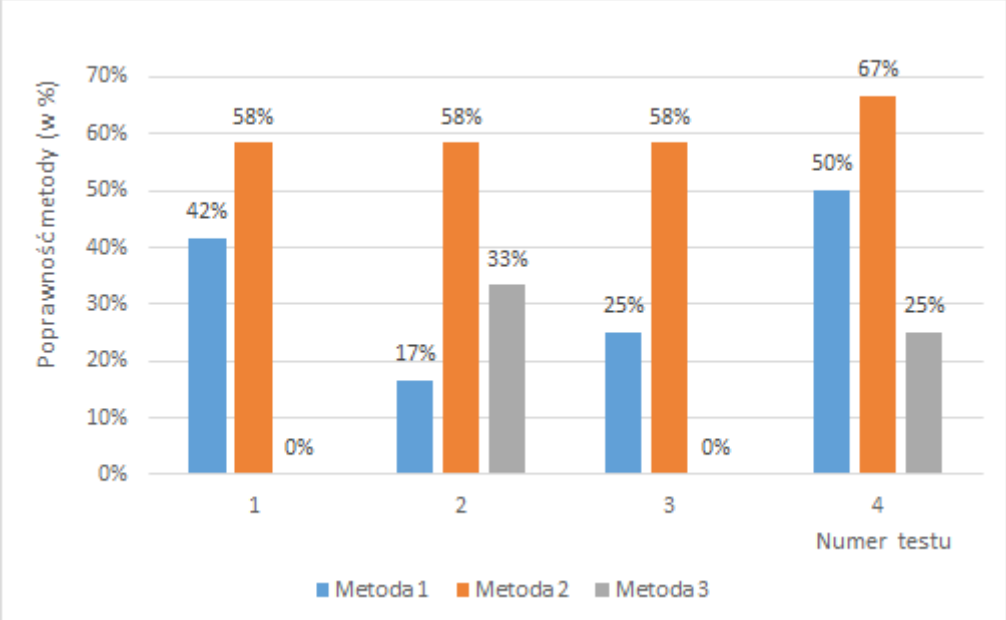
Rys. 5.5 Metoda 2. transformacji wyjścia warstwy sieci conv5_2 na wektor.

Rozwiązanie trzecie (metoda 3, rys. 5.6) jest analogiczne do rozwiązania drugiego. Jedyną różnicą jest podział na trzy części wymiaru drugiego zamiast pierwszego.



Rys. 5.6 Metoda 3. transformacji wyjścia warstwy sieci conv5_2 na wektor.

Przeprowadzono cztery testy na zbiorze testowym I wykorzystując każdą metodę po kolei. Oceniano stosunek liczby osób poprawnie zreidentyfikowanych do liczby wszystkich osób, które pojawiły się na filmie (poprawność metody). Na wykresie (rys. 5.7) przedstawiono wyniki każdego z czterech testów dla każdej metody. Metoda 3, ze średnią poprawnością na poziomie 15% okazała się zdecydowanie najmniej korzystna. Najlepsze wyniki są osiągane za pomocą metody 2 (średnia poprawność na poziomie 60%), która polega na podzieleniu pierwszego wymiaru tablicy na trzy części.



Rys. 5.7 Wykres przedstawiający porównanie poprawności trzech metod kompresji trzech wymiarów do jednego.

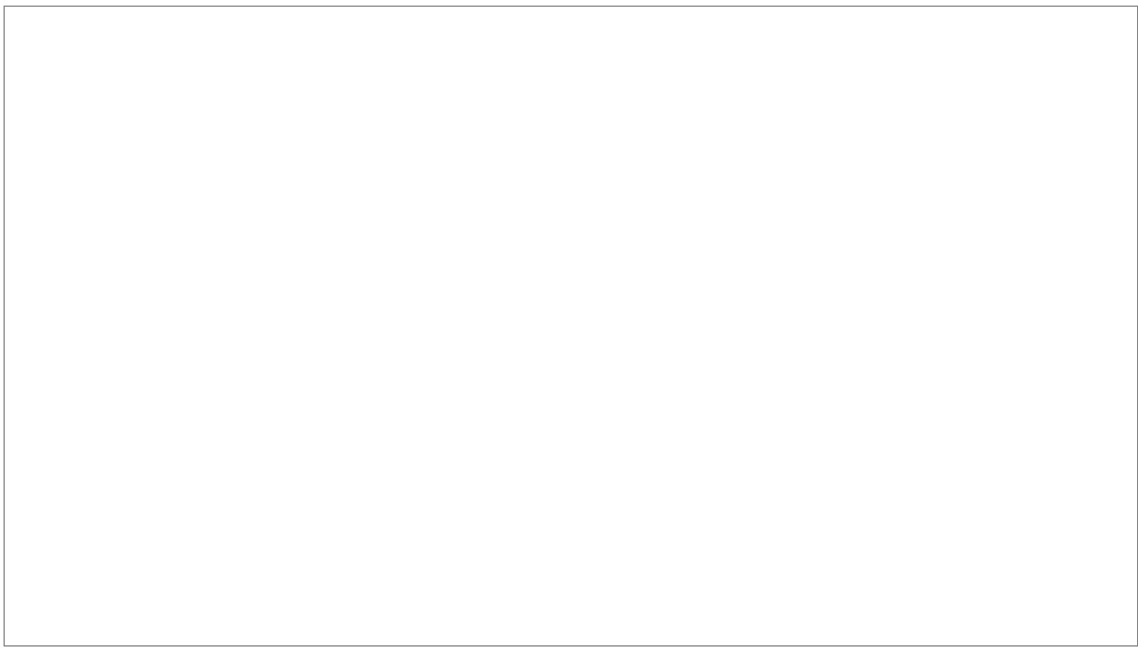
5.5. Różne algorytmy klasyfikacji

W celu porównania różnych metod klasyfikacji wektorów zaimplementowano dwa algorytmy:

I. kMultiplyDistance - wybór osoby, której wektory uzyskały sumarycznie najmniejszą średnią odległość wśród pierwszych k odległości
II. mostFrequentNearest - wybór osoby, której najwięcej wektorów zostało sklasyfikowanych jako najbliższe

Poniżej przedstawiono wyniki testów obu algorytmów przeprowadzonych na zbiorze testowym I. Pierwszy z nich okazał się zdecydowanie mniej korzystny osiągając średnią poprawność na poziomie około 27%. Drugi z nich osiągnął poprawność zdecydowanie lepszą osiągając średnią poprawność na poziomie około 59%. Druga metoda jest również bardziej przewidywalna, ponieważ odchylenie standardowe przeprowadzonych testów wynosi około 0,17 w porównaniu do około 0,23 dla metody pierwszej.

Na rysunku 5.8 przedstawiono wyniki dla poszczególnych prób.



Rys. 5.8 Porównanie dwóch algorytmów klasyfikacji

W celu zdiagnozowania problemów z reidentyfikacją dokonano analizy przypadków, dla których oba algorytmy mają trudności ze zreidentyfikowaniem postaci. W tabeli 3 oraz tabeli 4 przedstawiono oczekiwane oraz otrzymane wyniki, a także poprawność reidentyfikacji wyrażoną jako stosunek poprawnych reidentyfikacji do wszystkich możliwych.

Tabela 3 Poprawność metody kMultiplyDistance

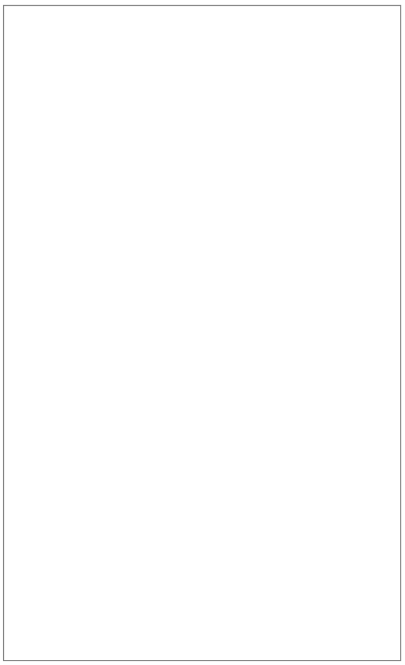
Poprawne rozwiązanie	0	1	2	3	4	5	6	7	8	9	10	11	Poprawność reidentyfikacji
I próba	11	2	5	3	4	10	6	7	8	9	1	0	6/12
II próba	10	11	0	2	6	5	3	1	4	8	9	7	1/12
III próba	10	4	11	2	3	0	5	1	7	8	9	6	0/12
IV próba	11	2	5	3	4	10	6	7	8	9	1	0	6/12

Tabela 4 Poprawność metody mostFrequentNearest

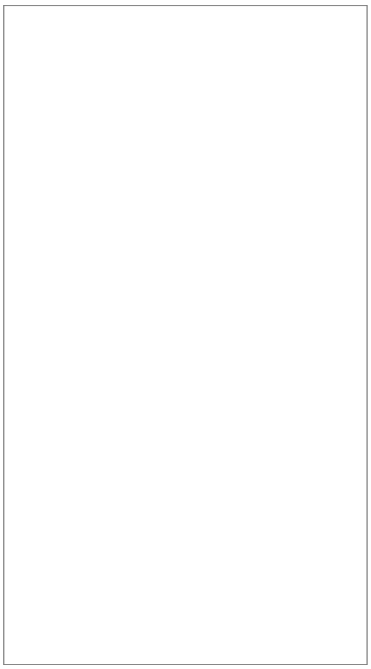
Poprawne rozwiązanie	0	1	2	3	4	5	6	7	8	9	10	11	Poprawność reidentyfikacji
I próba	3	1	2	0	4	5	6	7	8	9	10	7	7/12
II próba	3	1	2	0	4	5	11	6	8	9	10	7	7/12
III próba	3	2	6	0	4	5	9	11	8	10	1	7	3/12
IV próba	3	1	2	0	4	5	6	7	8	9	10	11	10/12

Łatwo zauważyć, że istnieją postaci, które każda z metod zawsze reidentyfikuje błędnie. Dobrym przykładem są osoby: Osoba A o id 0 oraz Osoba B o id 3 w metodzie mostFrequentNearest. Osoba A w każdej próbie została rozpoznana jako osoba B, natomiast osoba B została uznana za osobę A.

Na rys. 5.9 oraz rys.5.10 przedstawiono zdjęcia osób A i B.



Rys. 5.9 Osoba A



Rys. 5.10 Osoba B

Przedstawione postaci mają wiele podobieństw. Są to kobiety podobnego wzrostu oraz postury. Obie kobiety są ubrane w podobne sportowe spodnie oraz jasne bluzy. Nie dziwi więc fakt, że algorytmy mają problemy z odróżnieniem powyższych postaci.

Na podstawie testów stwierdzono, że spośród dwóch zaimplementowanych sposobów podejmowania decyzji algorytm mostFrequentNearest jest dużo lepszym rozwiązaniem. Pomimo trudności z klasyfikacją niektórych postaci osiąga dobre wyniki. Dlatego też został wybrany jako algorytm podejmowania decyzji dla licznika.

5.6. Dobieranie progu reidentyfikacji

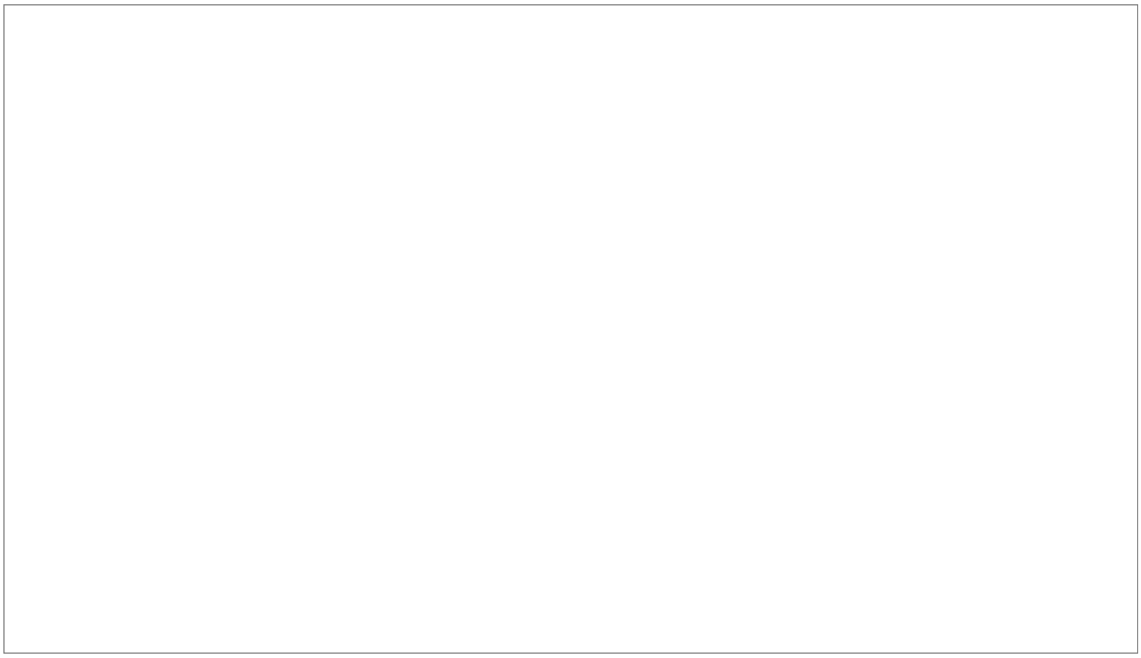
Jednym z najtrudniejszych zadań było ustalenie *progu reidentyfikacji* (odległości między *wektorami postaci* i najbliższej osoby), powyżej którego postać zostaje uznana za nową osobę. Wartości wektorów charakteryzujących postaci zależą od sieci neuronowej wybranej do transformacji, warstwy tej sieci a także sposobu podziału wektora (*5.4 Porównanie metod transformacji*). Uzyskiwane odległości między wektorami zależą bezpośrednio od wartości samych wektorów, a więc jednocześnie od wszystkich wcześniej wymienionych parametrów. Dlatego też dobranie progu możliwe jest dopiero po przeprowadzeniu testu dla konkretnie wybranej konfiguracji (sieci, jej warstwy oraz sposobu podziału wektora).

Próg reidentyfikacji ma za zadanie:

- stwierdzić, że podjęta przez algorytm decyzja była poprawna i pozwolić na reidentyfikację osoby,
- stwierdzić, że podjęta przez algorytm decyzja była błędna - osoba A nie powinna zostać przypisana jako osoba B i uniemożliwić przypisanie wektorów osoby A do osoby B. W tej sytuacji osoba A zostanie sklasyfikowana jako nowa osoba.

Ustalenie idealnego progu, który pozwalałby zawsze na reidentyfikację tej samej osoby i odrzucał błędne decyzje algorytmu nie jest możliwe. Zdarza się, że odległości pomiędzy wektorami dwóch różnych osób są mniejsze niż pomiędzy wektorami tej samej osoby.

Taka sytuacja miała miejsce podczas przeprowadzanych testów na zbiorze testowym I. Na rysunku 5.11 przedstawiono uzyskane odległości między wektorami (wraz z informacją czy reidentyfikacja powinna zostać zaakceptowana czy nie) oraz próbę dobrania progu reidentyfikacji.



Rys. 5.11 Próba dobrania progu reidentyfikacji

Na podstawie powyższego przykładu można stwierdzić, że niemożliwe jest takie dobranie progu, który zawsze spełniłby swoje zadanie. Wektory osoby o identyfikatorze 1, pomimo wprowadzenia progu reidentyfikacji, zostaną błędnie przypisane do innej osoby. Jednakże samo wprowadzenie progu pozwala zmniejszyć ryzyko nieprawidłowego przypisania wektorów do innej osoby, a więc w konsekwencji zmniejszyć ryzyko ponownych błędów.

6. TESTY

W celu sprawdzenia jakości zaimplementowanego licznika przeprowadzono testy jego wydajności oraz poprawności w dwóch wariantach: prymitywnym i inteligentnym (z reidentyfikacją).

6.1. Testy wydajności

Wydajność programu mierzona jest jako stosunek ilości przetworzonych ramek wyświetlonych użytkownikowi do czasu przetwarzania wyrażonego w sekundach (FPS – frames per second). Jako odniesienie zmierzono wydajność sczytywania oraz wyświetlania obrazu z kamery internetowej Tracer GAMMA CAM (0,3M PIXELS), która według dokumentacji posiada szybkości transmisji na poziomie 30 FPS. Pomiar prędkości przetwarzania kamerki został wykonany pięciokrotnie. Uzyskane wyniki przedstawiono w tabeli 5.

Tabela 5 Szybkość przetwarzania obrazu z kamery internetowej Tracer GAMMA

Lp	Liczba ramek	Czas przetwarzania	FPS
1	909	109.58	8.3
2	909	109.62	8.29
3	909	110.16	8.25
4	909	109.08	8.33
5	909	109.83	8.28

W trakcie wykonywania testów kamera osiągnęła szybkość średniej transmisji na poziomie około 8,29 FPS. Wartość ta została potraktowana jako prędkość przetwarzania potrzebna do pracy w czasie rzeczywistym.

6.1.1. Wydajność programu uruchomionego wyłącznie na CPU

Wydajność zaimplementowanego rozwiązania została zmierzona na maszynie Lenovo ThinkPad E440 o procesorze Intel(R) Core(TM) i5-4210M CPU @ 2.60GHz 2.59GHz działającym pod systemem operacyjnym Windows 10. Maszyna posiada zintegrowaną kartę graficzną, co uniemożliwia wykonywanie obliczeń bezpośrednio na niej. Test prędkości przetwarzania został wykonany pięciokrotnie, a wyniki przedstawiono w tabeli 6.

Tabela 6 Szybkość przetwarzania licznika osób uruchomionego wyłącznie na CPU

Lp	Liczba ramek	Czas przetwarzania	FPS
1	954	188,17	5,07
2	954	196,00	4,87
3	954	194,95	4,89
4	954	200,96	4,75
5	954	211,61	4,51

W trakcie wykonywania testów, wykorzystując jedynie pracę procesora, program osiągnął szybkość transmisji około 4,817 FPS.

6.1.2. Wydajność programu uruchomionego z wykorzystaniem GPU

Wydajność programu została również przetestowana na komputerze z systemem operacyjnym Ubuntu 16.04 LTS, korzystając z karty graficznej GeForce GTX 1060 6GB o taktowaniu 1,759 GHz. Test prędkości przetwarzania został wykonany pięciokrotnie, a wyniki przedstawiono w tabeli 7.

Tabela 7 Szybkość przetwarzania licznika osób uruchomionego z wykorzystaniem GPU

Lp	Liczba ramek	Czas przetwarzania	FPS
1	954	107,36	8,89
2	954	122,48	7,79
3	954	118,11	8,08
4	954	125,56	7,60
5	954	120,86	7,89

W przeprowadzonych testach zaimplementowane rozwiązanie osiągnęło średnią prędkość przetwarzania około 8,04 FPS.

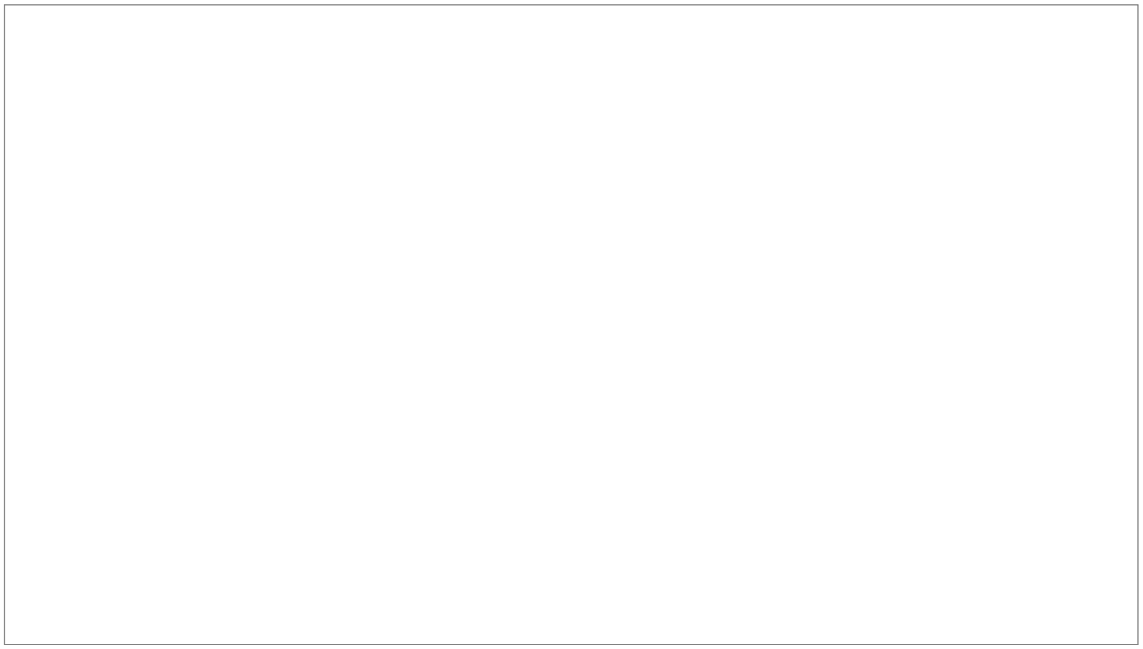
6.1.3. Podsumowanie testów wydajności

Na podstawie przeprowadzonych testów widać, że w trakcie pracy licznika wykonywanych jest wiele zasobochłonnych obliczeń. W związku z tym, wydajne działanie licznika, korzystającego tylko z procesora, nie jest możliwe w czasie rzeczywistym. Do wydajnego działania licznika potrzebna jest więc maszyna z kartą graficzną o podobnych parametrach do GeForce GTX 1060 6GB.

Za pomocą utworzonego licznika analizowane mogą być również wcześniej wykonane nagrania, w trybie offline. Przetwarzanie ich, choć niewydajne, jest możliwe nawet na maszynie ze zintegrowaną kartą graficzną.

6.2. Testy poprawności licznika inteligentnego

Po przeprowadzeniu eksperymentów (*5 Eksperymenty*) dobrano parametry, dzięki którym uzyskano największą skuteczność licznika inteligentnego. Do detekcji postaci wykorzystano sieć MobileNet, natomiast do reidentyfikacji użyto warstwę conv5_2 sieci VGG-19. Porównywanie wektorów przeprowadzono według algorytmu mostFrequentNearest. Do transformacji wyjścia sieci neuronowej na jednowymiarowy wektor wykorzystano metodę 2 (rys. 5.5). Niniejsze testy przeprowadzono na filmie zawierającym przejście dwunastu osób w jedną i drugą stronę (zbiór testowy I).

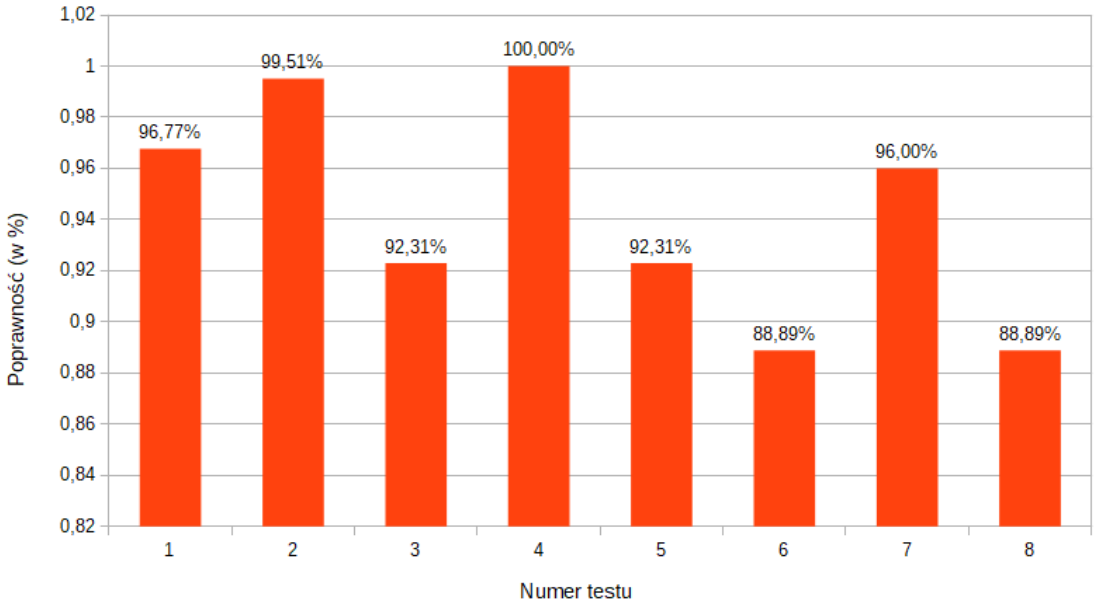


Rys. 6.1 Wykres przedstawiający wyniki testów poprawności licznika inteligentnego

Na rys. 6.1 przedstawiono wyniki sześciu testów poprawności działania licznika inteligentnego. Poprawność oznacza stosunek prawidłowo zliczonych osób do liczby wszystkich osób, które powinny być zliczone. Średnia poprawność na poziomie 59,72% jest wynikiem zadowalającym, jednak odchylenie standardowe wynoszące 17% zdecydowanie obniża poziom wiarygodności systemu. Najniższy wynik (poprawność równa 25% dla testu 5) spowodowany jest niepoprawną detekcją, która w następstwie powoduje nieprawidłową reidentyfikację.

6.3. Testy poprawności licznika prymitywnego

Do przeprowadzenia testu poprawności licznika prymitywnego użyto tych samych parametrów, które wykorzystano w teście licznika inteligentnego (*6.2 Testy poprawności licznika inteligentnego*). Tym razem jednak przeprowadzono testy na bardziej zróżnicowanym zestawie danych, wykorzystano oba zbiory testowe I i II.



Rys. 6.2 Wykres przedstawiający wyniki testów poprawności licznika prymitywnego

Na rys. 6.2 przedstawiono wyniki ośmiu testów poprawności licznika. Osiągając średnią poprawność na poziomie 94,3% oraz odchylenie standardowe równe 4,1%, system można uznać za wiarygodny.

7. PODSUMOWANIE

Celem pracy było stworzenie inteligentnego licznika osób działającego w czasie rzeczywistym. Zliczanie osób bazowało na obrazie z kamery oraz uwzględniało ich reidentyfikację.

Testowano wiele różnych rozwiązań, nie tylko opartych na sieciach neuronowych. Początkowo zaimplementowano detekcję osób z wykorzystaniem operacji morfologicznych, jednak ze względu na podatność metody na błędy ostatecznie podjęto decyzję o wykorzystaniu sieci neuronowej. Negatywnym skutkiem tego rozwiązania okazała się potrzeba uruchamiania systemu na komputerach o lepszych parametrach technicznych.

Na podstawie testów przeprowadzonych z wykorzystaniem przykładowych filmów dobrano ostateczne parametry wykorzystywane w procesie śledzenia. Określono odległość między wystąpieniem postaci na dwóch różnych klatkach, która pozwala uznać, że jest to ta sama osoba. Pod uwagę wzięto także czas jej ostatniej detekcji.

Sprawdzono jakie sieci neuronowe i jakie ich warstwy pozwalają na pozyskanie takich wektorów, które będą umożliwiały reidentyfikację. Zaimplementowano różne metody transformacji wyjść sieci neuronowych na jednowymiarowe wektory, a także dwa możliwe scenariusze podejmowania ostatecznej decyzji o reidentyfikacji osób.

Zaimplementowano przejrzysty interfejs użytkownika, pozwalający w łatwy sposób na odczytanie wartości licznika prymitywnego oraz inteligentnego. Umożliwiono również wygenerowanie raportu po zakończeniu liczenia, w którym zawarty jest stan licznika po zakończeniu liczenia oraz daty i godziny przejść osób.

Określono minimalną wydajność systemu komputerowego niezbędną do poprawnego działania. Przeprowadzono testy poprawności licznika w dwóch wariantach: prymitywnym oraz inteligentnym (pozwalającym na reidentyfikację osób).

Podział zadań wśród członków zespołu został przedstawiony w tabeli 8.

Tabela 8 Podział zadań wśród członków zespołu

Imię i nazwisko	**Wykonane zadania**
Katarzyna Gałka	Implementacja algorytmu reidentyfikacji, dobranie odpowiedniego progu reidentyfikacji, dokumentacja projektu (diagramy UML)
Katarzyna Retowska	Implementacja interfejsu użytkownika, przeprowadzenie testów poprawności licznika prymitywnego oraz inteligentnego, testy wydajności,przeprowadzenie eksperymentów dotyczących porównania warstw dwóch sieci neuronowych oraz różnych algorytmów klasyfikacji
Zofia Rytlewska	Implementacja algorytmu detekcji, śledzenia i liczenia, przeprowadzenie eksperymentu dotyczącego porównania metod transformacji oraz dobrania progu poprawnego wykrycia

7.1. Osiągnięcia

Wyniki przeprowadzonych eksperymentów pozwoliły na dobranie odpowiednich parametrów dla algorytmów detekcji i śledzenia. Dzięki temu, stworzony licznik prymitywny uzyskał średnią poprawność zliczania na poziomie 94,3%.

System w trybie reidentyfikacji działa ze średnią poprawnością wynoszącą 60%, co biorąc pod uwagę specyfikę zagadnienia jest wynikiem zadowalającym. Udało się osiągnąć założony wynik.

7.2. Porażki

Nie udało ustrzec się od pojedynczych błędów nawet przy tradycyjnym trybie licznika. Spowodowane były one wykorzystaniem obrazu z pojedynczej kamery - w momencie, gdy jedna osoba zostaje częściowo zasłonięta przez inną, detekcja nie działała poprawnie. Dobrane parametry śledzenia również nie zawsze pozwalały na poprawne działanie aplikacji.

Nawet pojedyncze błędy w tradycyjnym liczeniu powodowały kaskadowe narastanie błędów reidentyfikacji. Testy licznika inteligentnego wykazały, że w niektórych przypadkach możliwe było uzyskanie bardzo wysokiej poprawności – nawet na poziomie 80%, jednakże zdarzały się również testy podczas których poprawność wyniosła zaledwie ok. 25%.

7.3. Wnioski

By zaimplementowany licznik działał wydajnie z wysoką poprawnością musi być spełnionych wiele warunków:

I. Nagrywane przejście nie powinno być zbyt uczęszczane. Ważne jest by postaci nie zasłaniały się wzajemnie ponieważ w liczniku wykorzystywana jest tylko jedna kamera.
II. Grupa osób powinna być różnorodna pod względem ubioru oraz postur. Istnieje wówczas mniejsze ryzyko błędnej reidentyfikacji.
III. Przy obecnych parametrach liczone osoby powinny poruszać się z prędkością spokojnego marszu.

Na śledzenie postaci ma także wpływ szybkość przetwarzania obrazu przez maszynę, na której uruchomiony jest licznik. Na podstawie przeprowadzonych testów wydajnościowych stwierdzono, że stworzone rozwiązanie osiąga szybkość potrzebną do działania w czasie rzeczywistym jeśli zostanie uruchomione na sprzęcie o odpowiednich parametrach.

Uzyskana poprawność systemu nie jest wystarczająca do wykorzystania licznika inteligentnego w celach biznesowych.

7.4. Perspektywy rozwojowe

By zwiększyć wiarygodność rozwiązania należałoby przeprowadzić szersze testy poprawności. Możliwości kombinacji rodzajów sieci, ich warstw, sposobów podziału wektorów

postaci oraz sposobów decyzji o reidentyfikacji są praktycznie nieograniczone – bardzo prawdopodobne jest więc, iż istnieją takie kombinacje, dla których poprawność systemu byłaby dużo wyższa.

W celu umożliwienia wykrywania ludzi w grupach o większej gęstości, należałoby rozważyć zastosowanie większej liczby kamer.

Po zwiększeniu poprawności można by poszerzyć liczbę parametrów zbieranych przez licznik. W przemyśle odzieżowym poza liczeniem klientów coraz częściej analizuje się ich cechy, aby jak najlepiej dopasować charakter sklepu. Można rozpoznawać nie tylko takie właściwości jak płeć i wiek, ale także odnotowywać informacje z kim chodzą do sklepu, jak się ubierają, czy jaki noszą rozmiar. Licznik można również poszerzyć o moduł analizy zbieranych danych oraz ich wizualizacji, co pozwoliłoby na eliminację potrzeby integrowania systemu z zewnętrznymi narzędziami.

WYKAZ LITERATURY

1. Amey Varangaonkar, Top 10 Deep Learning Frameworks, https://datahub.packtpub.com/deep-learning/top-10-deep-learning-frameworks/, dostęp: 4.12.2017
2. Jonathan Huang et al., Speed/accuracy trade-offs for modern convolutional object detectors, 2016, CoRR
3. Shaoqing Ren et al., Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2015, CoRR
4. Jifeng Dai et al., R-FCN: Object Detection via Region-based Fully Convolutional Networks, 2016, CoRR
5. Wei Liu et al., SSD: Single Shot MultiBox Detector, 2015, CoRR
6. Joseph Redmon et al., You Only Look Once: Unified, Real-Time Object Detection, 2015, CoRR
7. Andrew G. Howard et al., MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017, CoRR
8. Zehaos, MobileNet, https://github.com/Zehaos/MobileNet, dostęp: 4.12.2017
9. chuanqi305, MobileNet-SSD, https://github.com/chuanqi305/MobileNet-SSD, dostęp: 4.12.2017
10. Manjeet Singh et al., Artistic Style Transfer with Convolutional Neural Network, https://medium.com/data-science-group-iitr/artistic-style-transfer-with-convolutional-neural-network-7ce2476039fd, dostęp: 4.12.2017
11. François Chollet, Keras, https://keras.io/applications/#vgg19, dostęp: 6.12.2017

WYKAZ RYSUNKÓW

WYKAZ TABEL

WYKAZ FRAGMENTÓW KODU