



POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Kierunek studiów:

Informatyka

Rodzaj studiów:

Stacjonarne / Niestacjonarne

Praca dyplomowa

Temat pracy:

System Rozpoznawania Przemocy

Temat pracy w języku angielskim:

Violence Recognition System

Opiekun pracy:

prof. dr hab. Marek Bednarczyk

Wykonawcy:

Benedykt Kościński

Jakub Kulaszewicz

Mateusz Chodyna

Ola Piętka

Streszczenie: Celem pracy dyplomowej było stworzenie systemu umożliwiającego rozpoznawanie aktów przemocy na podstawie sekwencji składających się z kilkudziesięciu klatek obrazu wideo. System analizuje sekwencję w celu określenia prawdopodobieństwa występowania przemocy. Zapis pochodzić może z kamery, a analiza dokonywana w czasie bliskim rzeczywistemu. Większość przyrostów poświęciliśmy na naukę modeli sieci neuronowych dla późniejszej integracji modułów i otrzymania finalnego wyniku klasyfikacji przy pomocy autorskiej funkcji oceniającej prawdopodobieństwo wystąpienia przemocy. Model stworzony został z pomocą języka Python oraz framework'ów Tenserflow, Keras oraz YoloV3. System udostępnia użytkownikowi możliwość wglądu w wyniki za pomocą systemu logów.

Oświadczenie autorów pracy dyplomowej

Świadom/a odpowiedzialności prawnej oświadczam, że niniejszą pracę dyplomową w zakresie przeze mnie przedstawionym wykonałem/łam samodzielnie i nie zawiera ona treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że praca w przedstawionym przeze mnie zakresie nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu ukończenia studiów wyższych.

Oświadczam ponadto, że niniejsza wersja pracy dyplomowej jest identyczna z załączoną wersją elektroniczną.



POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Karta dyplomowego projektu inżynierskiego

Temat projektu: System Rozpoznawania Przemocy (ang. Violence Recognition System)		Akronim: VRS
		Data ustalenia tematu: 02.03.2019
Opiekun: prof. dr hab. Marek Bednarczyk	Konsultanci: 1. dr inż. Stanisław Szejko 2. dr inż. Paweł Syty 3. prof. dr hab. Konrad Wojciechowski 4. dr Bartosz Marcinkowski	
Cele projektu (co chcemy osiągnąć): System dokonujący predykcji przemocy na bazie otrzymanej serii klatek z kamery wideo.		
Rezultaty projektu (oczekiwane produkty lub usługi): <ul style="list-style-type: none">• System umożliwiający klasyfikację przemocy z kamery wideo w czasie bliskim rzeczywistemu• Udostępnienie użytkownikowi wglądu do logów z wynikami klasyfikacji• Pełna dokumentacja projektu		
Miary sukcesu: <ul style="list-style-type: none">• System w 80 na 100 przypadków poprawnie rozpoznaje czy na danym odcinku wideo znajduje się przemoc• Ukończenie pracy nad projektem w wyznaczonym terminie• Obrona pracy zakończona sukcesem		
Ograniczenia: <ul style="list-style-type: none">• Moc obliczeniowa – proces nauki oraz klasyfikacji jest bardzo wymagający i jest możliwy tylko w przypadku wykonania wielu obliczeń• Złożoność problemu – problem klasyfikacji wideo oraz przemocy jest bardzo skomplikowany, z racji zmiennego otoczenia oraz niezliczonej mnogości przypadków mogących wystąpić na danych nagraniach• Czas – ograniczony czas na ukończenie projektu oraz duże zapotrzebowanie czasowe na wytrenowanie modeli• Dataset – proces nauczania sieci neuronowych wymaga wielu nakładów danych obrazujących oraz ręcznej edycji plików w celu wyodrębnienia odpowiednich fragmentów		

Wykonawcy	Numer albumu	Specjalizacja	Tryb studiów:
<i>Benedykt Kościński</i>	<i>s17472</i>	<i>Sztuczna inteligencja</i>	<i>stacjonarne</i>
<i>Jakub Kulaszewicz</i>	<i>s17838</i>	<i>Sztuczna inteligencja</i>	<i>stacjonarne</i>
<i>Mateusz Chodyna</i>	<i>s17606</i>	<i>Aplikacje internetowe</i>	<i>niestacjonarne</i>
<i>Ola Piętka</i>	<i>s17611</i>	<i>Sztuczna inteligencja</i>	<i>stacjonarne</i>

Data ukończenia projektu: <i>25.01.2021</i>	Recenzent:
---	-------------------

Spis treści

1.	Wstęp	9
2.	Słownik pojęć	10
3.	Analiza problemu	13
3.1.	Przedstawienie problemu	13
3.2.	Rich picture	14
3.3.	Rozwiązania konkurencyjne	16
3.3.1.	Abto Software	16
3.3.2.	FightNet	17
3.4.	Propozycja rozwiązania	17
4.	Planowanie	19
4.1.	Cele projektu	19
4.2.	Produkty	19
4.3.	Udziałowcy projektu	20
4.4.	Charakterystyka klientów.....	22
4.4.1.	Klienci wewnętrzni	22
4.4.2.	Klienci zewnętrzni	23
4.5.	Przypadki użycia	23
4.5.1.	Scenariusze	25
4.6.	Komercjalizacja.....	26
4.7.	Ograniczenia	27
4.7.1.	Czas.....	27
4.7.2.	Moc obliczeniowa.....	28
4.7.3.	Złożoność problemu	28
4.7.4.	Dataset	28
4.8.	Strategia wytwarzania	29
4.8.1.	Opis planowanych przyrostów	30
5.	Analiza wymagań	31
5.1.	Wymagania ogólne i dziedzinowe	31
5.2.	Wymagania funkcjonalne.....	32
5.2.1.	Interfejs z otoczeniem.....	34
5.3.	Wymagania niefunkcjonalne.....	35
5.4.	Wymania dotyczące procesu wytwarzania	37
5.5.	Wymagania jakościowe i inne	39
6.	Projektowanie	40

6.1.	Ogólny schemat systemu	40
6.1.1.	Architektura systemu.....	40
6.1.2.	Działanie systemu	41
6.1.3.	Funkcja finalnej klasyfikacji	45
6.2.	Wybrane technologie	46
6.2.1.	Język programowania.....	46
6.2.2.	Biblioteki i framework'i.....	46
6.2.3.	Inne narzędzia	47
6.3.	Dataset	49
6.3.1.	Dataset z filmami	49
6.3.2.	Dataset ze zdjęciami.....	55
6.3.3.	Dataset z audio	57
6.4.	Moduły.....	60
6.4.1.	Moduł FGN	60
6.4.2.	Moduł VRN.....	62
6.4.3.	Moduł DIDN	63
6.4.4.	Moduł DSDN	65
6.5.	Proces nauczania.....	67
6.5.1.	Maszyna	67
6.5.2.	Wyuczanie modułu FGN.....	67
6.5.3.	Wyuczanie modułu VRN	73
6.5.4.	Wyuczanie modułu DIDN.....	83
6.5.5.	Wyuczanie modułu DSDN.....	84
6.6.	API.....	86
6.6.1.	Napotkane problemy	86
6.6.2.	Podsumowanie	87
6.7.	TensorFlow Serving.....	87
6.8.	Porzucone pomysły.....	88
6.8.1.	Segmentacja	88
6.8.2.	Szkieletyzacja.....	89
6.8.3.	Modele 3D.....	89
6.8.4.	Usuwanie tła.....	89
7.	Realizacja projektu.....	90
7.1.	Przyrost zero - planowanie	90
7.1.1.	Założenia przyrostu	90

7.1.2.	Zadania	90
7.1.3.	Napotkane problemy	91
7.2.	Przyrost pierwszy – dataset	92
7.2.1.	Założenia przyrostu	92
7.2.2.	Zadania	92
7.2.3.	Napotkane problemy	97
7.3.	Przyrost drugi – implementacja modułów	98
7.3.1.	Założenia przyrostu	98
7.3.2.	Zadania	98
7.3.3.	Napotkane problemy	100
7.4.	Przyrost trzeci – uczenie	100
7.4.1.	Założenia przyrostu	101
7.4.2.	Zadania	101
7.4.3.	Napotkane problemy	102
7.5.	Przyrost czwarty - integracja.....	103
7.5.1.	Założenia.....	103
7.5.2.	Zadania	103
7.5.3.	Napotkane problemy	106
7.6.	Przyrost piąty – optymalizacja i walidacja.....	107
7.6.1.	Założenia.....	107
7.6.2.	Zadania	107
7.6.3.	Napotkane problemy	110
7.7.	Przyrost szósty – poprawki	110
7.7.1.	Założenia.....	110
7.7.2.	Zadania	110
7.7.3.	Napotkane problemy	111
7.8.	Podsumowanie	111
8.	Walidacja rozwiązania.....	112
8.1.	Sposób walidacji	112
8.1.1.	Błąd predykcji systemu.....	112
8.1.2.	Dokładność predykcji systemu	113
8.2.	Przeprowadzone testy.....	113
8.2.1.	Test 1	113
8.2.2.	Test 2	114
8.2.3.	Test 3	114

8.2.4.	Test 4	115
8.2.5.	Test 5	115
8.2.6.	Test 6	116
8.2.7.	Test 7	116
8.2.8.	Test 8	117
8.2.9.	Test 9	117
8.2.10.	Test 10	118
8.2.11.	Test 11	118
8.2.12.	Test 12	119
8.3.	Analiza przeprowadzonych testów	120
8.3.1.	Błąd predykcji systemu	122
8.3.2.	Dokładność predykcji systemu.....	122
8.4.	Podsumowanie	122
9.	Nakład sumaryczny	123
9.1.	Sposoby obliczeń nakładu sumarycznego	123
9.1.1.	Nakład pracy w postaci tabeli	123
9.1.2.	Nakład pracy w postaci diagramu Gantta	124
10.	Wkład własny.....	126
10.1.	Ola Piętka.....	126
10.1.1.	Prace wykonane na poczet systemu	126
10.1.2.	Prace wykonane na poczet dokumentacji.....	127
10.2.	Jakub Kulaszewicz.....	129
10.2.1.	Prace wykonane na poczet systemu	129
10.2.2.	Prace wykonane na poczet dokumentacji.....	131
10.3.	Benedykt Kościński	132
10.3.1.	Prace wykonane na poczet systemu	132
10.3.2.	Prace wykonane na poczet dokumentacji.....	134
10.4.	Mateusz Chodyna	135
10.4.1.	Prace wykonane na poczet systemu	135
10.4.2.	Prace wykonane na poczet dokumentacji.....	135
11.	Bibliografia	137
12.	Spis rysunków	141
13.	Spis tabel	144
14.	Spis równań.....	146
15.	Załączniki.....	147

1. Wstęp

W dzisiejszych czasach technologia wkracza w nowe obszary życia, ułatwiając oraz przyspieszając wykonywanie wszelakich czynności. Szczególną gałęzią technologii, przy której możemy zauważyć bardzo szybki rozwój jest dziedzina sztucznej inteligencji [1]. Dzięki jej technikom coraz częściej jesteśmy świadkami rozwiązań problemów, które przed laty były dla dużej części społeczeństwa niewyobrażalne. Jako studenci w większości specjalizujący się w tej przestrzeni chcieliśmy się podjąć wyzwania rozwiązania rzeczywistego problemu, z którym społeczeństwo może się borykać. Dlatego przyjęliśmy za zadanie jako zespół skupienie się na problemie jakim jest przemoc w miejscach publicznych oraz zakładach pracy. Uważamy to za realny problem, który może dotknąć każdego. Niekiedy przyspieszenie wykrycia sytuacji niebezpiecznej przez uprawnione do tego osoby jest w stanie uratować człowiekowi życie. Z tego powodu postanowiliśmy stworzyć system wykorzystujący współczesne techniki w celu ułatwienia lub nawet pełnego zautomatyzowania tego procesu, tworząc system klasyfikacji przemocy (ang. Violence Recognition System - VRS). Dzięki ciągłej analizie wideo z kamer, będzie on w stanie wykryć sytuację niebezpieczną, uniknąć błędów ludzkiej detekcji, wykrywać przemoc z wielu kamer jednocześnie oraz zawiadomić odpowiednie osoby o zaistniałej sytuacji co pozwoli na szybszą reakcję.

W aktualnym dokumencie można znaleźć analizę wyżej omawianego problemu w rozdziale 3, dokładny proces planowania w rozdziale 4 i analizę wymagań w rozdziale 5. Natomiast szczegóły implementacyjne oraz wytwórcze znajdują się odpowiednio w kolejnych rozdziałach, wraz z walidacją systemu opisaną w rozdziale 8. W późniejszych rozdziałach załączyliśmy opis nakładu sumarycznego oraz wkładu każdego członka zespołu.

2. Słownik pojęć

Termin	Wyjaśnienie
Accuracy	Metryka służąca do pomiaru wydajności algorytmu w sposób możliwy do zinterpretowania. Accuracy modelu jest zwykle określane po parametrach modelu i przedstawiane w procentach
Augmentacja danych	Techniki służące do zwiększania ilości danych dzięki dodawaniu nieznacznie zmodyfikowanych kopii już istniejących danych
Batch size	Określa liczbę próbek, które będą propagowane w sieci neuronowej
Callback	Funkcja, która zostanie wykonana, gdy jakaś inna funkcja zakończy działanie
CNN	Sieć konwolucyjna (ang. Convolutional neural network) która potrafi stopniowo filtrować różne części danych uczących i wyodrębiać ważne cechy w procesie wykorzystanym do rozpoznawania lub klasyfikacji wzorców
Conda	Wieloplatformowy, niezależny od języka menedżer pakietów i system zarządzania środowiskiem
Dataset	Zbiór danych potrzebny w celu wyuczenia sieci neuronowych, zawiera dane potrzebne do rozpoznania danej klasy
FastAPI	Jest to framework webowy dla języka Python. FastAPI jest zbudowany na Starlette
Finetuning	Dotrenowanie (ang. Finetuning) już wcześniej zainicjowanych wag
Fully Connected	Splaszczona jednowymiarowa tablica poprzedniej warstwy wynikowej
Google Colaboratory	Usługa umożliwiająca pisanie i uruchamianie kodu Python bezpośrednio w przeglądarce

Termin	Wyjaśnienie
gRPC	System zdalnego wywoływania procedur (RPC) typu Open Source
Hiperparametry	Parametry, których wartości są wykorzystywane do kontrolowania procesu uczenia się sieci neuronowych
Keras	Biblioteka Open Source do tworzenia sieci neuronowych. Wykorzystuje ona TensorFlow jako backend
Klient	Synonim dla klient docelowy
Loss	Służy do optymalizacji algorytmu uczenia maszynowego. Loss jest obliczany na podstawie wyników treningu i walidacji, a jego interpretacja opiera się na tym, jak dobrze model radzi sobie w tych dwóch zestawach danych
LSTM	Długoterminowa pamięć krótkoterminowa (LSTM) – sieć rekurencyjna
Moduł	Część systemu z określonym zadaniem klasyfikującym, opartym na jednej bądź wielu sieciach neuronowych. Moduły są kluczowe dla działania systemu
OpenCV	Biblioteka funkcji wykorzystywanych podczas obróbki obrazu w czasie rzeczywistym
Optical Flow	Wzór ruchu pojedynczych pikseli na obrazie, spowodowany względnym ruchem między sąsiadującymi klatkami
Protobuf	Rozwijany przez Google sposób na binarną serializację strukturalnych danych. Z powodzeniem można go przedstawić jako jedną z alternatyw dla XML'a.
SCRUM	Jedna z najpopularniejszych metodyk zwinnych w zarządzaniu projektami IT, oparta na zasadach Agile
Segmentacja	Proces podziału obrazu na części określane jako obszary, które są jednorodne pod względem pewnych wybranych własności np. zbiory pikseli

Termin	Wyjaśnienie
Seq	System logów umożliwiający przeszukiwanie, filtrowanie, analizę oraz tworzenie alertów
Sieć neuronowa	Model składający się z neuronów, tworzących warstwy, tj. warstwa wejściowa, warstwy ukryte oraz warstwa wyjściowa
Spektrogram	Wizualne przedstawienie widma częstotliwości sygnału w czasie
Temporal Pooling	Nowa technika przetwarzania zdarzeń czasowych poprzez tworzenie deklaratywnych reprezentacji kompletnych sekwencji
TensorFlow	Bezpłatna i Open Source biblioteka python'owa do uczenia maszynowego
TensorFlow Serving	Elastyczny i wysokowydajny system obsługujący modele uczenia maszynowego, przeznaczony dla środowisk produkcyjnych
YOLOv3	Najnowsza wersja algorytmu detekcji obiektów

3. Analiza problemu

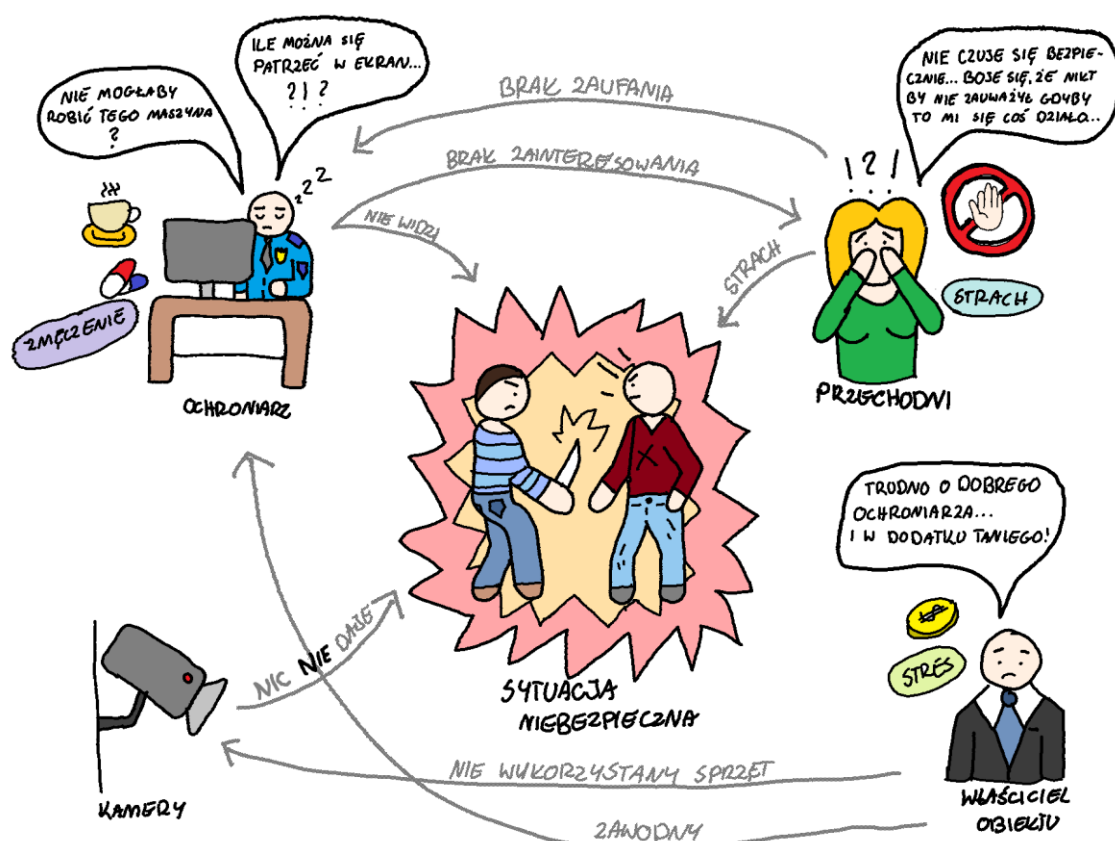
3.1. Przedstawienie problemu

Przemoc stanowi poważne zagrożenie dla bezpieczeństwa osobistego i stabilności społecznej. Jest ona obecna w miejscach publicznych, szkołach, domach jak i Internecie. Według statystyk dokonanych przez Bureau of Justice Statistics [2] w latach 2004-2008 na obszarze Stanów Zjednoczonych liczba zarejestrowanych aktów przemocy wynosiła około 5.5 mln, w tym prawie 2.0 mln wydarzyło się w miejscach publicznych, a ponad 700 tyś. w szkołach (Rysunek 3.1). Dlatego niezwykle ważne jest szybkie wykrywanie zdarzeń związanych z przemocą i odpowiednia interwencja. Niestety zatrudnienie ludzi, których zadaniem byłoby monitorowanie prawie każdego zakątka danej przestrzeni może stać się niesamowicie kosztowne. Popularnym rozwiązaniem jest montaż kamer monitorujących miejsca publiczne, zbierając dowody i odstraszać potencjalnych przestępców. Jednak monitorowanie tak dużej ilości danych wideo w czasie rzeczywistym jest narażone na wiele ludzkich błędów. Dlatego uważamy, że automatyczne wykrywanie scen przemocy w czasie rzeczywistym może stać się niezbędne w nadchodzących latach, powodując przy tym zmniejszenie kosztów monitoringu oraz zwiększenie bezpieczeństwa w danej przestrzeni.

Detailed place of occurrence for violent and property crimes, average annual 2004-2008				
	Average annual 2004-2008			
	Violent victimizations ^a		Property victimizations ^b	
	Number	Percent	Number	Percent
Total	5,507,130	100.0%	18,004,640	100.0%
Total in commercial places^c	664,700	12.1%	933,270	5.2%
Commercial restaurant, bar, nightclub	244,000	4.4	209,350	1.2
Inside office	74,900	1.4	157,720	0.9
Bank	9,100	0.2	10,820	0.1
Gas station	63,530	1.2	58,860	0.3
Other commercial building	230,210	4.2	438,840	2.4
Factory/warehouse	42,960	0.8	57,680	0.3
Total in parking lots or garages^c	401,920	7.3%	2,018,180	11.2%
Commercial parking lot/garage	110,620	2.0	413,050	2.3
Noncommercial parking lot/garage	213,540	3.9	864,190	4.8
Apartment/townhouse parking lot/garage	77,760	1.4	740,950	4.1
Total in school or on school property	714,860	13.0%	1,272,570	7.1%
Inside school building	464,540	8.4	1,004,890	5.6
On school property	250,320	4.5	267,670	1.5
Total in open areas, on street or public transportation	967,800	17.6%	748,150	4.2%
In apartment yard, park, field, playground	130,270	2.4	158,360	0.9
On street other than near own or friend/neighbor/ relative's house	787,640	14.3	457,590	2.5
On public transportation or in station	49,910	0.9	132,190	0.7
Other	396,700	7.2%	861,660	4.8%

Rysunek 3.1 Tabela udostępniona przez Bureau of Justice Statistics [2], przedstawiająca wykaz liczbowy oraz procentowy aktów przemocy z podziałem na lokalizacje w Stanach Zjednoczonych w latach 2004-2008

3.2. Rich picture



Rysunek 3.2 Rich picture – ogólne przedstawienie problemu

Na rysunku powyżej przedstawiliśmy problem z perspektywy właściciela obiektu, ochroniarza oraz ludzi. Dodatkowo pokazaliśmy połączenia między nimi i obiektami w formacie rich picture. Ten etap pozwolił nam na głębszą analizę problemu z perspektywy ochroniarza oraz wstępne wyspecyfikowanie wymagań.



Rysunek 3.3 Rich picture – perspektywa ochroniarza

Na podstawie głębszej analizy problemu pracy ochroniarza, wygenerowaliśmy kolejne rich picture w celu pokazania jego zmagania. Ten etap uzmysłowił nam jak odpowiedzialna jest to praca i z jakimi dolegliwościami fizycznymi może się zmagać człowiek.

danych użyty później w procesie uczenia. Firma Abto Software jako dataset wybrała 1,000 nagrań wideo z walk hokejowych oraz 200 klipów z różnych filmów akcji, patrz [5].

Wady rozwiązania

Abto Software nie rozważa sytuacji, w których akt przemocy nie jest całkowicie w zasięgu urządzenia. Dlatego proponujemy wykrywanie przemocy przy użyciu audio, w przypadku, gdy kąt kamery nie obejmuje całego zdarzenia. Istnieje jeszcze sytuacja, gdy na nagraniu kamer znajduje się uzbrojony człowiek. Zdecydowanie jest to skrajnie niebezpieczna sytuacja, której produkt firmy Abto Software nie jest w stanie wykryć. Dodatkowo słabą stroną jest również jednostronny dobór danych uczących.

3.3.2. FightNet

Innym wartym omówienia rozwiązaniem jest model sieci FightNet [6] opracowany przez Peipei Zhou, Qinghai Ding, Haibo Luo oraz Xinglin Hou w pracy o tytule „Violent Interaction Detection in Video Based on Deep Learning” z 2017 roku. Przy użyciu zbiorów danych [5] filmowych z meczy hokejowych, filmów oraz walk bokserskich z zastosowaniem takich technologii wizji komputerowej jak optical flow czy acceleration, udało im się osiągnąć dokładność predykcji na poziomie 97%.

Wady rozwiązania

FightNet jest wyłącznie wyuczonym modelem sieci neuronowej i nie posiada żadnej obudowy do wykorzystania go w sposób komercyjny. Tak samo jak w przypadku wyżej opisanego rozwiązania konkurencyjnego od Abto Software, słabą stroną jest również jednostronny dobór danych uczących.

3.4. Propozycja rozwiązania

Analiza rynku wykazała, że istnieje niewiele rozwiązań konkurencyjnych będących w stanie rozwiązać przedstawione problemy. Świadomi rynkowej niszy i zmotywowani faktem, że problem występuje na ogromną skalę, zdecydowaliśmy się stworzyć system umożliwiający analizę danych video w czasie bliskim rzeczywistemu w celu detekcji aktów przemocy.

Stworzenie takiego produktu jest w stanie rozwiązać wiele problemów takich jak na przykład:

- brak odpowiednio szybkiej interwencji spowodowany przez brak informacji o zdarzeniu niebezpiecznym,
- błędy w ocenie spowodowane przez człowieka,
- presja i stres spoczywający na barkach ochroniarzy, spowodowany ogromną odpowiedzialnością,
- małe poczucie bezpieczeństwa społeczeństwa, nawet w pobliżu kamer.

Nasz system wyróżniałby się na tle konkurencji modułarną budową umożliwiającą dostosowanie procesu detekcji do danego klienta. Dodatkowo chcemy zastosować rozwiązania, których według nas brakuje konkurencji (jak opisano w punkcie 3.3), co skutkowałoby jeszcze trafniejszymi wynikami analizy wideo, bardziej wydajnym wykorzystaniem zasobów, a co za to idzie - mniejszymi kosztami. Będąc świadomi szerokiej grupy odbiorczej, chcemy udostępnić użytkownikom prosty i przejrzysty interfejs, który pozwoli na odczytywanie tylko najważniejszych informacji. Dodatkowo system alertów jeszcze skuteczniej może zadbać o powiadomienie użytkownika o sytuacji niebezpiecznej oraz ułatwić pracę wielu osób.

4. Planowanie

4.1. Cele projektu

Cele naszego projektu zostały jasno określone przez nas już na początku planowania systemu. Niżej w formie listy wypunktowaliśmy najważniejsze dla nas cele:

- wyuczenie sieci neuronowych klasyfikujących przemoc z dokładnością przynajmniej 80%, co oznacza, poprawnie rozpoznana przemoc w 8 na 10 przypadków,
- budowa modułarna systemu wykorzystująca różne techniki detekcji przemocy,
- intuicyjny interfejs pozwalający użytkownikowi na szybką weryfikację sytuacji,
- produkcja dokumentacji projektowej, która stanowić będzie część pracy dyplomowej,
- przedstawienie w pełni działającego systemu.

4.2. Produkty

Spodziewanym produktem naszego projektu jest:

- wielomodułowy system umożliwiający detekcję przemocy z kamer, przy pomocy sieci neuronowych i technologii wizji komputerowej,
- przedstawienie wyników klasyfikacji w intuicyjnym systemie logów, umożliwiającym użytkownikowi filtrowanie informacji oraz konfigurację systemu alertów,
- możliwość konfiguracji systemu przez użytkownika (adres kamery, włączone moduły),
- dokumentacja projektu, zawierająca informacje na temat stworzonego systemu.

4.3. Udziałowcy projektu

KARTA UDZIAŁOWCA	
Identyfikator:	UOB 01
Nazwa:	Twórca systemu
Opis:	Osoba zajmująca się wytwarzaniem systemu, która ma bezpośredni wpływ na jego kształt i proces wytwórczy. Twórca systemu należy do zespołu projektowego
Typ udziałowca:	Ożywiony bezpośredni
Punkt widzenia:	Techniczny
Ograniczenia:	Brak
Wymagania:	WO 01, WO 02, WF 01, WF 02, WF 03, WF 04, WI 01, WI 02, NF 01, NF 02, NF 03, NF 04, NF 05, NF 06, PW 01, PW 02, PW 03, PW 04, PW 05, PW 06, PW 07, PW 08, PW 09

KARTA UDZIAŁOWCA	
Identyfikator:	UOB 02
Nazwa:	Promotor
Opis:	Osoba nadzorująca przebieg pracy wytwarzania systemu
Typ udziałowca:	Ożywiony bezpośredni
Punkt widzenia:	Oceniający
Ograniczenia:	Nie ma bezpośredniego wpływu na proces wytwórczy projektu oraz sposób jego wytwarzania i wyboru technologii
Wymagania:	PW 01

KARTA UDZIAŁOWCA	
Identyfikator:	UOB 03
Nazwa:	Klient
Opis:	Osoba, która nabyła prawa do korzystania z systemu poprzez uprzednie wykupienie go
Typ udziałowca:	Ożywiony bezpośredni
Punkt widzenia:	Ekonomiczny
Ograniczenia:	Nie ma bezpośredniego wpływu na proces wytwórczy projektu oraz sposób jego wytwarzania i wyboru technologii
Wymagania:	WO 01, WO 02, WF 01, WF 02, WF 03, WF 04, WI 01

KARTA UDZIAŁOWCA	
Identyfikator:	UOP 01
Nazwa:	Społeczeństwo
Opis:	Osoby będące uczestnikami (świadomie i nieświadomie) procesu działania systemu
Typ udziałowca:	Ożywiony pośredni
Punkt widzenia:	Prawny
Ograniczenia:	Nie ma bezpośredniego wpływu na proces wytwórczy projektu oraz sposób jego wytwarzania i wyboru technologii
Wymagania:	WO 01, WO 02

KARTA UDZIAŁOWCA	
Identyfikator:	UNB 01
Nazwa:	Dataset
Opis:	Zbiór danych uczących
Typ udziałowca:	Nieożywiony bezpośredni
Punkt widzenia:	Techniczny
Ograniczenia:	Pamięciowe
Wymagania:	NF 05, PW 09

KARTA UDZIAŁOWCA	
Identyfikator:	UNB 02
Nazwa:	Moc obliczeniowa
Opis:	Sprzęt komputerowy wymagany do wytworzenia systemu
Typ udziałowca:	Nieożywiony bezpośredni
Punkt widzenia:	Techniczny
Ograniczenia:	Ekonomiczne
Wymagania:	WF 01, NF 01, NF 05

KARTA UDZIAŁOWCA	
Identyfikator:	UNP 01
Nazwa:	Źródło danych
Opis:	Kamera IP generująca materiał wideo
Typ udziałowca:	Nieożywiony pośredni
Punkt widzenia:	Jakościowy
Ograniczenia:	Brak
Wymagania:	WF 01, WF 03, WI 01, WI 02, NF 01, NF 02

KARTA UDZIAŁOWCA	
Identyfikator:	UNP 02
Nazwa:	System logów
Opis:	Narzędzie wykorzystywane do zapisywania wyników klasyfikacji w formie logów
Typ udziałowca:	Nieożywiony pośredni
Punkt widzenia:	Techniczny
Ograniczenia:	Szybkość oraz ilość zapytań na sekundę
Wymagania:	WF 02, WI 02

4.4. Charakterystyka klientów

4.4.1. Klienci wewnętrzni

Nazwa klienta	Charakterystyka klienta	Potrzeby klienta
Zespół projektowy	Osoba uczestnicząca w implementacji i utrzymywaniu systemu oraz odpowiedzialna za wytwarzanie dokumentacji	Odpowiedni sprzęt do celów implementacyjnych
		Jak najwięcej czasu na wytworzenie systemu
		Osoby w zespole o wysokich kwalifikacjach w zakresie różnych dziedzin
Promotor	Osoba recenzująca i nadzorująca proces wytwarzania systemu oraz dokumentacji	Pilnowanie terminów przez zespół projektowy
		Dobra komunikacja z zespołem
Uczelnia	Osoba reprezentująca uczelnie w celu wydania oceny odnośnie do końcowej pracy	Dostarczenie pracy w terminie
		Trzymanie się wyznaczonych standardów akademickich

Tabela 4.1 Charakterystyka klientów wewnętrznych oraz ich potrzeb

4.4.2. Klienci zewnętrzni

Nazwa klienta	Charakterystyka klienta	Potrzeby klienta
Klient docelowy	Osoba posiadająca prawa do korzystania z systemu poprzez	Wysoka efektywność działania systemu
	ówczesne jego zakupienie	Relatywnie niski koszt zakupu oraz utrzymania systemu

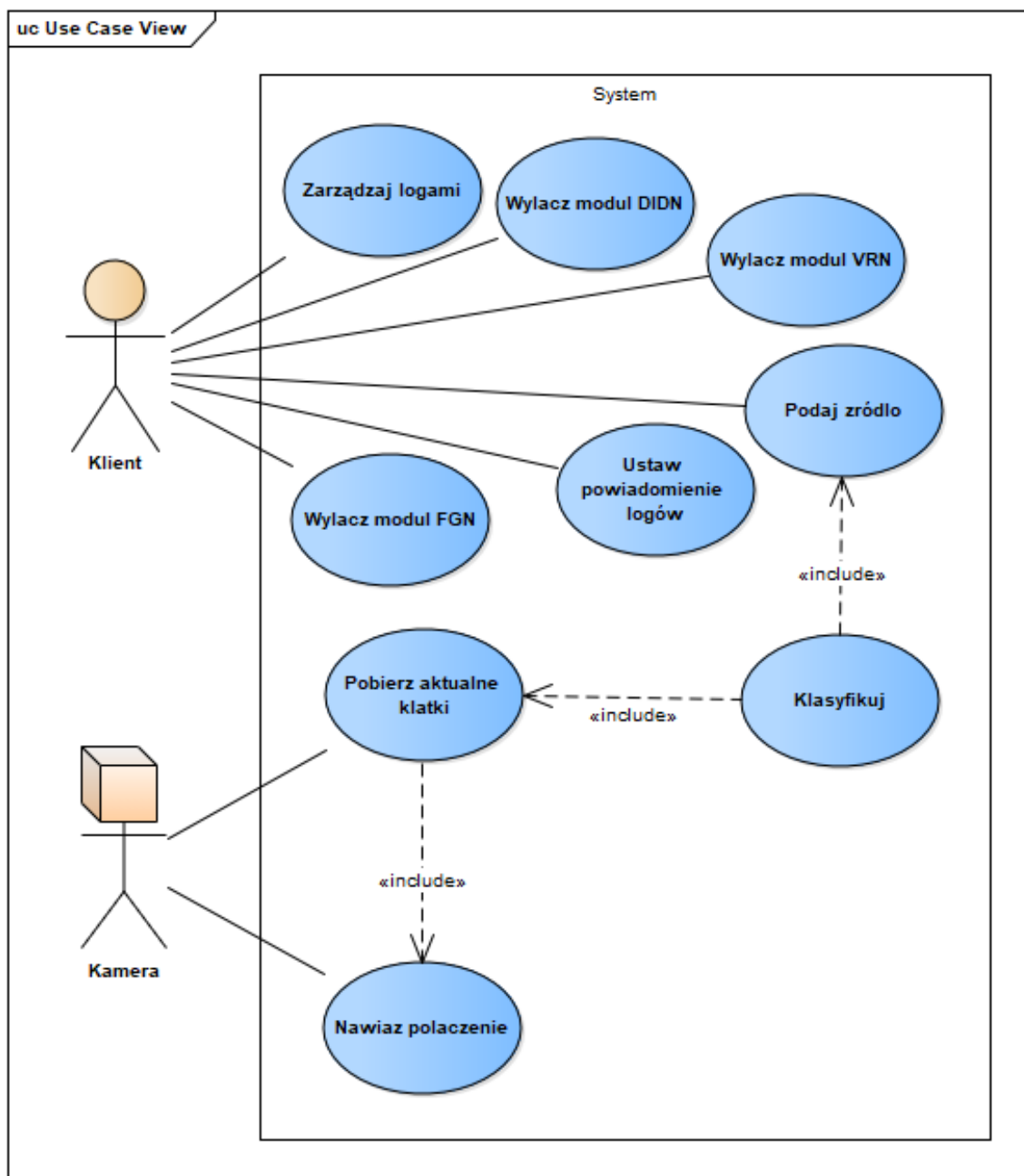
Tabela 4.2 Charakterystyka klientów zewnętrznych oraz ich potrzeb

4.5. Przypadki użycia

W naszym systemie występują następujący aktorzy:

- Klient – aktor ten posiada możliwość korzystania z systemu. Jest on w stanie podać źródło, z którego system ma czerpać informacje wideo oraz odczytać wyniki klasyfikacji dokonanej przez system.
- Kamera – źródło, które generuje dane wideo w celu późniejszej analizy.

Oraz system jako zbiór modułów mający na celu klasyfikację danych dostarczanych przez kamerę oraz zapis wyników do systemu logów.



Rysunek 4.1 Diagram przypadków użycia przedstawiający schemat wykorzystania systemu

4.5.1. Scenariusze

Niżej opisano w formie tabeli przykładowe scenariusze używania naszego systemu.

SCENARIUSZ			
Identyfikator:	SC 01	Nazwa:	Wykrycie przemocy
Aktorzy:	Klient		
Warunki początkowe	Akt przemocy widziany na kamerze		
Scenariusz	<ol style="list-style-type: none">1. Użytkownik widzi w systemie logów wysokie prawdopodobieństwo występowania przemocy2. Filtruje wyniki w systemie logów w celu dokładniejszej weryfikacji		
Warunki końcowe	Opanowanie sytuacji niebezpiecznej		
Wymagania powiązane	WF 01, WF 02, WI 02, NF 01		

SCENARIUSZ			
Identyfikator:	SC 02	Nazwa:	Konfiguracja – moduły
Aktorzy:	Klient		
Warunki początkowe	Klient nie potrzebuje oceny aktu przemocy na podstawie detekcji broni		
Scenariusz	<ol style="list-style-type: none">1. Klient w pliku konfiguracyjnym wyłącza moduł odpowiedzialny za detekcję broni2. Uruchamia system, który nie korzysta z wykluczonego modułu		
Warunki końcowe	Klasyfikacja bez modułu zajmującego się detekcją broni		
Wymagania powiązane	WF 04		

SCENARIUSZ			
Identyfikator:	SC 02	Nazwa:	Konfiguracja – wybór źródła
Aktorzy:	Klient		
Warunki początkowe	Klient posiada kamerę z adresem dostępnym dla serwera		
Scenariusz	<ol style="list-style-type: none"> 1. Klient w pliku konfiguracyjnym uzupełnia adres kamery <ol style="list-style-type: none"> a) Jeżeli adres NIE JEST poprawny, użytkownik dostaje komunikat błędu b) Jeżeli adres JEST poprawny, użytkownik dostaje komunikat potwierdzający połączenie 2. Nawiązanie połączenia (w przypadku wariantu 1b) 		
Warunki końcowe	Poprawna konfiguracja źródła i połączenia kamera		
Wymagania powiązane	WF 03, WF 04, WI 01		

4.6. Komercjalizacja

Analiza rynku

Poszukując rozwiązań konkurencyjnych do naszego systemu, natrafiłiśmy jedynie na dwa rozwiązania (pkt 3.3) warte wspomnienia. Dlatego łatwo można zauważyć, że zautomatyzowana detekcja przemocy z kamer znajduje się w niszy rynkowej. Dzięki czemu nasz projekt jest jednym z niewielu rozwiązujących problem relatywnie szybkiej klasyfikacji przemocy. Nasze rozwiązanie daje klientowi dowolność w kwestii zastosowania oraz zapewnia i daje komfort przy detekcji i szybkiej reakcji na zaistniałą, możliwie niebezpieczną, sytuację.

Model biznesowy

Proponowany przez nas model biznesowy dla tego systemu opierałby się na bazie rocznych licencji z możliwością przedłużenia o kolejne lata. Licencja byłaby również zależna od ilości urządzeń docelowych na których system zostałby zainstalowany.

Poza tym dodatkowymi ścieżkami zarobku byłyby odpłatne szkolenia z korzystania z systemu oraz dodatkowe wsparcie, które obejmowałoby:

- pomoc przy instalacji systemu na środowiskach docelowych,
- ustawianie powiadomień dostosowanych do potrzeb klienta.

Rozważaną przez nas również możliwością czerpania zysku z systemu byłoby dołączenie do programu resellerskiego, gdzie nabywca mógłby wykupić licencje naszego systemu, opcjonalnie zmodyfikować sposób jego działania i czerpać z niego zyski swoimi kanałami.

Pozyskiwanie klientów

Ważnym również aspektem komercjalizacji systemu są planowane kanały dotarcia do potencjalnych klientów. Niestety nasz model biznesowy powoduje, że tradycyjne ścieżki takie jak reklamy typu Google Ad Service [7] czy jego odpowiedniki dla mediów społecznościowych mogą się okazać nieskuteczne, jako że grupa naszych potencjalnych klientów jest bardzo wąska. Dodatkowo istnieje małe prawdopodobieństwo dotarcia do takowych klientów, nie mówiąc o pozyskaniu zainteresowania tymi ścieżkami. Naszą propozycją jest zatrudnienie przedstawiciela handlowego, którego zadaniem byłoby dotarcie do ów potencjalnych klientów, gdzie prezentowałby personalną ofertę naszego produktu dostosowaną do ich potrzeb.

4.7. Ograniczenia

W poniższych podpunktach opisane są ograniczenia zweryfikowane w procesie planowania. Każdy podpunkt zawiera opis danego ograniczenia oraz sposób w jaki planujemy go rozwiązać lub jak chcemy mu przeciwdziałać.

4.7.1. Czas

Jednym z największych, o ile nie największym problemem, któremu musieliśmy stawić czoła jest czas. Problem ten pojawił się pod różnymi postaciami takimi jak:

- Ograniczony czas na ukończenie projektu – z racji faktu, iż projekt ten wykonany został na potrzeby dyplomu, jesteśmy zobligowani, aby ukończyć jego proces tworzenia w odpowiednim terminie.
- Duża ilość czasu potrzebna na wytrenowanie modeli – z racji wykorzystania technologii sieci neuronowych, elementem nieuniknionym jest odpowiednie wyuczenie modeli w celu poprawnej klasyfikacji spełniającej nasze wymagania. Niestety proces ten jest długotrwały i często wymaga wielu prób zanim uzyskany efekt będzie spełniał nasze wymagania.

Z racji powyższych problemów naszą pracę postanowiliśmy zacząć o wiele wcześniej niż sugerowałby to harmonogram nauczania na studiach. Jako że, wyuczanie modeli jest jednym z najbardziej czasochłonnych elementów naszego projektu, z góry podzieliliśmy naszą pracę nad modułami co pozwoliło nam na wcześniejsze przygotowanie prototypów oraz zgłębienie wiedzy w ścisłej dziedzinie. W celu przyspieszenia procesu nauczania, zwróciliśmy się również do Pana Radosława Nieleka w celu uzyskania dostępu do maszyny CloudLab2 (opisane w punkcie 6.5.1).

4.7.2. Moc obliczeniowa

Kolejnym problemem dostrzeżonym we wczesnych etapach planowania było bardzo duże zapotrzebowanie mocy obliczeniowej wymaganej do poprawnego wytworzenia naszego systemu. Proces nauki oraz klasyfikacji jest bardzo wymagający i jest możliwy tylko w przypadku wykonania wielu obliczeń.

Z tego powodu zwróciliśmy się o dostęp do wyżej wspomnianego CloudLab2, umożliwiło to sprawne przeprowadzenie testów, a co za tym idzie - weryfikację postawionych założeń.

4.7.3. Złożoność problemu

Problem klasyfikacji wideo oraz przemocy jest bardzo skomplikowany, z racji bardzo zmiennego otoczenia oraz niezliczonej mnogości przypadków mogących wystąpić na danych nagraniach, musieliśmy podzielić nasz problem na mniejsze, łatwiejsze w interpretacji części.

Z tego powodu zdecydowaliśmy się podzielić nasz system na mniejsze moduły, dzięki temu ograniczamy ilość informacji poszukiwanych przez dane moduły co znacząco uprości problem.

4.7.4. Dataset

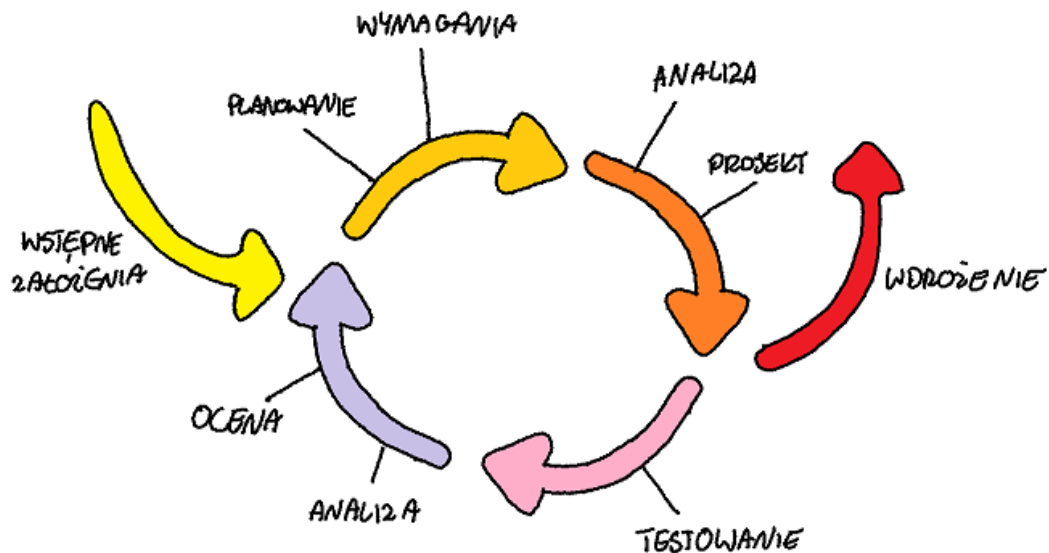
Proces nauczania sieci neuronowych wymaga wielu nakładów danych obrazujących najróżniejsze sytuacje mogące się wydarzyć. Początkowo nie zdawaliśmy sobie sprawy jak wielkim problemem będzie uzyskanie odpowiedniej ilości danych, które będą spełniać nasze wymagania. Następnym ograniczeniem, któremu musieliśmy stawić czoła był ogrom pracy wymagany przy ręcznej edycji plików w celu wyodrębnienia

odpowiednich fragmentów stanowiących dobry materiał do nauczania sieci neuronowych.

W celu rozwiązania tego problemu zdecydowaliśmy, aby zacząć zbierać materiały dużo wcześniej niż wymagał od nas tego proces dyplomowania.

4.8. Strategia wytwarzania

W celu realizacji projektu postanowiliśmy wytwarzać oprogramowanie w modelu przyrostowym z elementami SCRUM. W procesie dyskusji doszliśmy do wniosku, że ta metodyka najbardziej pasuje do systemu jaki chcemy wytwarzać, ze względu na ciągłą potrzebę analizy i testowania modułów.



Rysunek 4.2 Diagram przyjętego przez nas modelu wytwarzania systemu

Dodatkowo zdecydowaliśmy się na dwu, a w późniejszej fazie jednotygodniowe, iteracje monitorowane za pomocą narzędzia organizacji pracy, zakończone spotkaniem oraz dyskusją na temat dokonanych prac i planach na kolejne części. Podział większych przyrostów na mniejsze, pozwala na szybsze dostrzeganie wad oraz zapewnia komfortową elastyczność w dokonywaniu zmian, w celu uzyskania jak najlepszego rezultatu. Więcej na temat metodyki SCRUM możemy przeczytać w Scrum and Team Effectiveness: Theory and Practice, patrz [8].

4.8.1. Opis planowanych przyrostów

Przyrost zero – skupienie się na dyskusji odnośnie naszych pomysłów wykorzystując technikę burzy mózgów, w celu określenia wymagań i analizy projektu. Dodatkowo ustalenie schematu projektu oraz wybranie narzędzi, które posłużą nam w osiągnięciu naszego celu.

Przyrost pierwszy – rozpoczęcie prac nad budową potrzebnych zbiorów danych.

Przyrost drugi – implementacja poszczególnych niezależnych elementów systemu, zwanych w projekcie modułami.

Przyrost trzeci – skupienie się na wyuczeniu wcześniej zaimplementowanych modułów, wykorzystując przy tym przygotowane w pierwszym przyroście datasety.

Przyrost czwarty – integracja wszystkich modułów oraz budowa pliku konfiguracyjnego w celu uzyskania gotowej architektury systemu.

Przyrost piąty – optymalizacja i walidacja zintegrowanego systemu oraz parametrów.

Przyrost szósty – czas na poprawki systemu oraz dokumentacji.

5. Analiza wymagań

5.1. Wymagania ogólne i dziedzinowe

KARTA WYMAGANIA			
Identyfikator:	WO 01	Priorytet:	M – must (musi być)
Nazwa	Bezpieczeństwo danych		
Opis	System nigdzie nie zapisuje/przechowuje zebranych klatek z kamery (jedynie wynik klasyfikacji)		
Udziałowiec	UOB 01, UOB 03, UOP 01		
Wymagania powiązane	WF 01		

KARTA WYMAGANIA			
Identyfikator:	WO 02	Priorytet:	S – should (powinno być)
Nazwa	Klienci		
Opis	System przeznaczony jest dla szeroko pojętych firm (np. muzea, kina, stacje benzynowe, szkoły) oraz organizacji Państwowych potrzebujących systemu analizy przemocy		
Udziałowiec	UOB 01, UOB 03, UOP 01		
Wymagania powiązane			

5.2. Wymagania funkcjonalne

KARTA WYMAGANIA			
Identyfikator:	WF 01	Priorytet:	M – must (musi być)
Nazwa	Klasyfikacja		
Opis	System musi na podstawie danej serii klatek dokonać klasyfikacji przemocy		
Kryteria akceptacji	Poprawna klasyfikacja w 80 na 100 przypadków		
Dane wejściowe	Zestaw klatek		
Warunki początkowe	Poprawne formatowanie klatek		
Warunki końcowe	Wynik klasyfikacji		
Sytuacje wyjątkowe	Niepoprawna liczba klatek, problemy na drodze przesyłu danych		
Szczegóły implementacji	<ol style="list-style-type: none"> 1. Podłączenie źródła danych 2. Zebranie aktualnego zestawu klatek wideo ze źródła danych (pkt 6.1.2.1) 3. Wysłanie dalej danych zestawów klatek do wszystkich modułów w celu uzyskania predykcji (pkt 6.1.2.2) 4. Oczekiwanie na odpowiedź wszystkich modułów 5. Obliczenie końcowej klasyfikacji na podstawie wcześniej zdefiniowanego wzoru 		
Udziałowiec	UOB 01, UOB 03, UNB 02, UNP 01		
Wymagania powiązane	WF 03, PW 02, PW 03, PW 09		

KARTA WYMAGANIA			
Identyfikator:	WF 02	Priorytet:	M – must (musi być)
Nazwa	Odczyt logów		
Opis	System musi zapewnić wgląd do logów zawierających wyniki klasyfikacji z możliwością filtrowania		
Kryteria akceptacji	Odświeżane logi w czasie bliskim rzeczywistemu oraz poprawne filtrowanie		
Dane wejściowe	Brak		
Warunki początkowe	Zapisanie wyników do systemu logów		
Warunki końcowe	Poprawne odczytanie logów		
Sytuacje wyjątkowe	Zerwanie połączenia i brak dostępu do serwera logów		
Szczegóły implementacji	Wdrożenie Seq jako bazy oraz przeglądarki logów		
Udziałowiec	UOB 01, UOB 03, UNP 02		
Wymagania powiązane	WF 01, WI 02, NF 04		

KARTA WYMAGANIA			
Identyfikator:	WF 03	Priorytet:	M – must (musi być)
Nazwa	Wybór źródła		
Opis	System musi umożliwiać wybór źródła danych, tj. adresu kamery		
Kryteria akceptacji	Poprawne połączenie się do kamery pod podanym adresem		
Dane wejściowe	Plik konfiguracyjny		
Warunki początkowe	Adres kamery istnieje oraz posiadamy do niej dostęp		
Warunki końcowe	Podłączenie kamery do systemu		
Sytuacje wyjątkowe	Niepoprawny adres (brak dostępu/nie istniejące)		
Szczegóły implementacji	<ol style="list-style-type: none"> 1. Pozyskanie adresu z pliku konfiguracyjnego 2. Implementacja obsługi pliku konfiguracyjnego 		
Udziałowiec	UOB 01, UOB 03, UNP 01		
Wymagania powiązane	WF 04		

KARTA WYMAGANIA			
Identyfikator:	WF 04	Priorytet:	M – must (musi być)
Nazwa	Konfiguracja		
Opis	System musi udostępniać plik konfiguracyjny w celu dostosowania działania systemu pod klienta		
Kryteria akceptacji	Zmiany w pliku konfiguracyjnym wpływają adekwatnie na działanie systemu		
Dane wejściowe	Plik konfiguracyjny		
Warunki początkowe	Klient udostępnia informacje potrzebne do skonfigurowania systemu		
Warunki końcowe	Uruchomienie systemu według podanej konfiguracji		
Sytuacje wyjątkowe	Błędna konfiguracja lub jej brak		
Szczegóły implementacji	<ol style="list-style-type: none"> 1. Przygotowanie pliku konfiguracyjnego 2. Implementacja obsługi pliku konfiguracyjnego 		
Udziałowiec	UOB 01, UOB 03		
Wymagania powiązane			

5.2.1. Interfejs z otoczeniem

KARTA WYMAGANIA			
Identyfikator:	WI 01	Priorytet:	M – must (musi być)
Nazwa	Połączenie		
Opis	System po pozyskaniu od użytkownika adresu kamery, nawiązuje stałe połączenie (do momentu wyłączenia systemu)		
Kryteria akceptacji	Połączenie z kamerą jest ciągłe i nie zostaje zerwane w innym wypadku niż wyłączenie systemu		
Dane wejściowe	Adres kamery		
Warunki początkowe	Poprawnie podane źródło danych w pliku konfiguracyjnym		
Warunki końcowe	Stałe połączenie między kamerą a interfejsem		
Sytuacje wyjątkowe	Zerwanie połączenia pomiędzy kamerą a systemem		
Szczegóły implementacji	<ol style="list-style-type: none"> 1. Pozyskanie adresu kamery z pliku konfiguracyjnego 2. Nawiązanie i utrzymanie połączenia z kamerą do momentu zatrzymania systemu 		
Udziałowiec	UOB 01, UOB 03, UNP 01		
Wymagania powiązane	WF 03		

KARTA WYMAGANIA			
Identyfikator:	WI 02	Priorytet:	M – must (musi być)
Nazwa	Zapis logów		
Opis	System musi zapisywać zebrane wyniki klasyfikacji do zewnętrznego systemu logów		
Kryteria akceptacji	Zapisywanie wyników do logów w czasie bliskim rzeczywistemu		
Dane wejściowe	Źródło, ogólny wynik klasyfikacji, wyniki klasyfikacji poszczególnych modułów, data + czas		
Warunki początkowe	Zakończona klasyfikacja		
Warunki końcowe	Zapisanie rekordu do logów		
Sytuacje wyjątkowe	Brak połączenia z systemem logów		
Szczegóły implementacji	<ol style="list-style-type: none"> 1. Wdrożenie systemu logów Seq 2. Połączenie z systemem logów według konfiguracji wskazanej w pliku 3. Implementacja obsługi strukturalnego formatu logów Seq 4. Implementacja dodawania nowych wpisów do bazy Seq 		
Udziałowiec	UOB 01, UNP 01, UNP 02		
Wymagania powiązane	WF 01, WF 03, WI 01, NF 04		

5.3. Wymagania niefunkcjonalne

KARTA WYMAGANIA			
Identyfikator:	NF 01	Priorytet:	M – must (musi być)
Nazwa	Szybkość klasyfikacji		
Opis	Szybkość klasyfikacji z jaką system analizuje dane wideo		
Kryteria akceptacji	Zakładając, że źródło danych przesyła dane z minimalną prędkością 30 klatek na sekundę, system musi zwrócić wynik predykcji z danego zestawu klatek w czasie do 2 sekund		
Udziałowiec	UOB 01, UNB 02, UNP 01		
Wymagania powiązane	WF 01		

KARTA WYMAGANIA			
Identyfikator:	NF 02	Priorytet:	M – must (musi być)
Nazwa	Dostępność systemu		
Opis	System musi posiadać wysoką dostępność i bliski 100% up-time		
Kryteria akceptacji	System działa 24/7		
Udziałowiec	UOB 01, UNP 01		
Wymagania powiązane	WI 01		

KARTA WYMAGANIA			
Identyfikator:	NF 03	Priorytet:	M – must (musi być)
Nazwa	Czas wdrożenia		
Opis	System należy ukończyć do końca stycznia 2021 r.		
Kryteria akceptacji	System spełniający podstawowe wymagania zostanie ukończony do stycznia 2021 r.		
Udziałowiec	UOB 01, UOB 02		
Wymagania powiązane			

KARTA WYMAGANIA			
Identyfikator:	NF 04	Priorytet:	S – should (powinno być)
Nazwa	Seq		
Opis	Wykorzystanie Seq jako bazy danych oraz przeglądarki logów		
Kryteria akceptacji	Wersja ≥ 5.0		
Udziałowiec	UOB 01		
Wymagania powiązane			

Identyfikator:	NF 05	Priorytet:	M – must (musi być)
Nazwa	Uczenie sieci		
Opis	Wszystkie zakładane sieci muszą zostać w pełni wyuczone na wcześniej przygotowanym zbiorze danych		
Kryteria akceptacji	Dokładność klasyfikacji na poziomie minimum 80%		
Udziałowiec	UOB 01, UNB 01, UNB 02		
Wymagania powiązane	PW 01, PW 02, PW 03		

Identyfikator:	NF 06	Priorytet:	M – must (musi być)
Nazwa	Wzór klasyfikujący		
Opis	W celu uzyskania końcowego wyniku klasyfikacji, potrzebne jest utworzenie oraz implementacja wzoru liczącego wynik predykcji na podstawie wyników wszystkich modułów		
Kryteria akceptacji	Wagi wzoru dobrane w sposób poprawiający finalny wynik detekcji względem wyników pojedynczych modułów		
Udziałowiec	UOB 01		
Wymagania powiązane			

5.4. Wymania dotyczące procesu wytwarzania

KARTA WYMAGANIA			
Identyfikator:	PW 01	Priorytet:	M – must (musi być)
Nazwa	YOLOv3		
Opis	Użycie biblioteki YOLOv3 do wyuczenia modelu sieci, który zajmuje się wykrywaniem przedmiotów niebezpiecznych		
Kryteria akceptacji	Brak		
Udziałowiec	UOB 01		
Wymagania powiązane	WF 01		

KARTA WYMAGANIA			
Identyfikator:	PW 02	Priorytet:	M – must (musi być)
Nazwa	Python		
Opis	Użycie języka Python przy implementacji wszystkich modułów sieci neuronowej		
Kryteria akceptacji	Wersja >= 3.7		
Udziałowiec	UOB 01		
Wymagania powiązane	WF 01		

KARTA WYMAGANIA			
Identyfikator:	PW 03	Priorytet:	M – must (musi być)
Nazwa	Keras		
Opis	Użycie biblioteki Keras, która rozszerza bibliotekę TensorFlow do tworzenia sieci neuronowych		
Kryteria akceptacji	Wersja >= 2.4.3		
Udziałowiec	UOB 01		
Wymagania powiązane	WF 01, PW 02		

KARTA WYMAGANIA			
Identyfikator:	PW 04	Priorytet:	S – should (powinno być)
Nazwa	TensorFlow Serving		
Opis	Użycie kontenera TensorFlow Serving do obsługi wyuczonych modeli oraz wykonywania predykcji		
Kryteria akceptacji	Każdy z modeli jest udostępniany z wykorzystaniem TF Serving.		
Udziałowiec	UOB 01		
Wymagania powiązane	PW 05		

KARTA WYMAGANIA			
Identyfikator:	PW 05	Priorytet:	S – should (powinno być)
Nazwa	Docker		
Opis	Wykorzystanie do konteneryzowania serwisu oraz modułów		
Kryteria akceptacji	Każdy z modułów znajduje się w osobnym i niezależnym kontenerze		
Udziałowiec	UOB 01		
Wymagania powiązane			

KARTA WYMAGANIA			
Identyfikator:	PW 06	Priorytet:	S – should (powinno być)
Nazwa	Podział pracy		
Opis	Podział prac na zadania jak najbardziej od siebie niezależne		
Kryteria akceptacji	Podzielenie zadań pomiędzy członków zespołu projektowego względem umiejętności, po równo		
Udziałowiec	UOB 01		
Wymagania powiązane			

KARTA WYMAGANIA			
Identyfikator:	PW 07	Priorytet:	S – should (powinno być)
Nazwa	System wersjonowania		
Opis	Wersjonowanie i synchronizacja kodu za pośrednictwem zdalnego repozytorium		
Kryteria akceptacji	Po zakończeniu implementacji danej części systemu, kod powinien zostać umieszczony w serwisie Github przez osobę, która ten kod napisała		
Udziałowiec	UOB 01		
Wymagania powiązane			

KARTA WYMAGANIA			
Identyfikator:	PW 08	Priorytet:	S – should (powinno być)
Nazwa	Komunikacja zespołu		
Opis	Pozostawanie w ciągłym kontakcie za pomocą odpowiednich narzędzi		
Kryteria akceptacji	Trzymanie się ustalonych terminów spotkań w serwisie Discord		
Udziałowiec	UOB 01		
Wymagania powiązane			

KARTA WYMAGANIA			
Identyfikator:	PW 09	Priorytet:	S – should (powinno być)
Nazwa	Udostępnianie plików		
Opis	Utrzymywanie dokumentacji oraz zbioru danych na zasobie udostępnionym		
Kryteria akceptacji	Umieszczanie aktualnych wersji zbiorów danych w serwisie Synthing, a dokumentacji w chmurze Microsoftu		
Udziałowiec	UOB 01, UNB 01		
Wymagania powiązane			

5.5. Wymagania jakościowe i inne

Z racji wcześniej postawionych założeń, nasz system powinien spełniać poniższe wymagania:

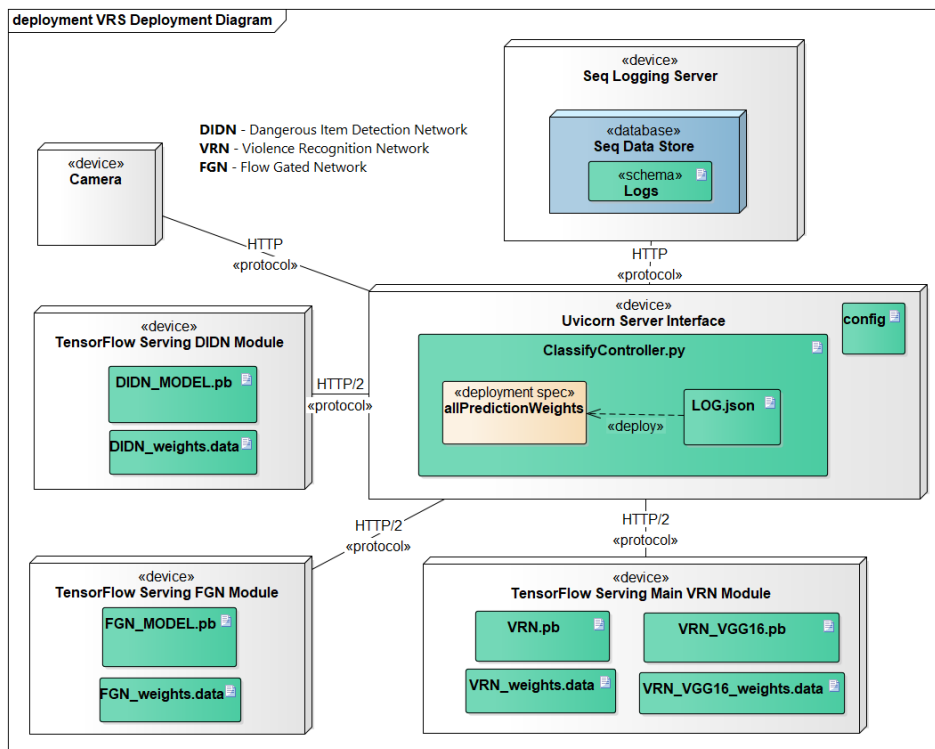
- system musi mieć minimum 80% poprawności w klasyfikacji przemocy tj. system poprawnie rozpoznała przemoc w 8 na 10 przypadków,
- system musi klasyfikować przemoc w czasie bliskim rzeczywistemu tzn. zakładając, że źródło danych przesyła dane z minimalną prędkością 30 klatek na sekundę, system musi zwrócić wynik predykcji z danego zestawu klatek w czasie do 2 sekund,
- system musi mieć możliwość postawienia na lokalnym serwerze jak i chmurze,
- system musi nawiązywać połączenie z kamerami,
- system musi umożliwiać wgląd do wyników klasyfikacji oraz możliwość ich filtrowania.

6. Projektowanie

6.1. Ogólny schemat systemu

6.1.1. Architektura systemu

Nasz system przyjmuje architekturę modułową. Składa się na nią główna aplikacja koordynująca pracę całego systemu, 3 kontenery TensorFlow Serving [9], serwer logów Seq [10] oraz kamera jako źródło sygnału wideo. Komunikacja między głównym serwerem a kontenerami TensorFlow Serving odbywa się przy pomocy gRPC wykorzystującego format danych Protobuf i protokół HTTP/2. Natomiast do porozumienia się z kamerą oraz serwerem Seq – wykorzystujemy protokołu HTTP. Do rozwiązania problemu wykrywania sytuacji niebezpiecznych zdecydowaliśmy się stworzyć 3 różne moduły analizujące problem na różne sposoby, składają się z jednego bądź dwóch wyuczonych modeli sieci neuronowych załadowanych do wyżej wspomnianych instancji TensorFlow Serving. Moduły DIDN (pkt 6.4.3) oraz FGN (pkt 6.4.1) zawierają po jednym wyuczonym modelu, natomiast moduł VRN (pkt 6.4.2) składa się z dwóch, które są osadzone w jednym kontenerze. Do przetrzymywania wyników z wcześniej wykonanych predykcji używany jest wbudowany w rozwiązanie Seq, Seq Data Store.

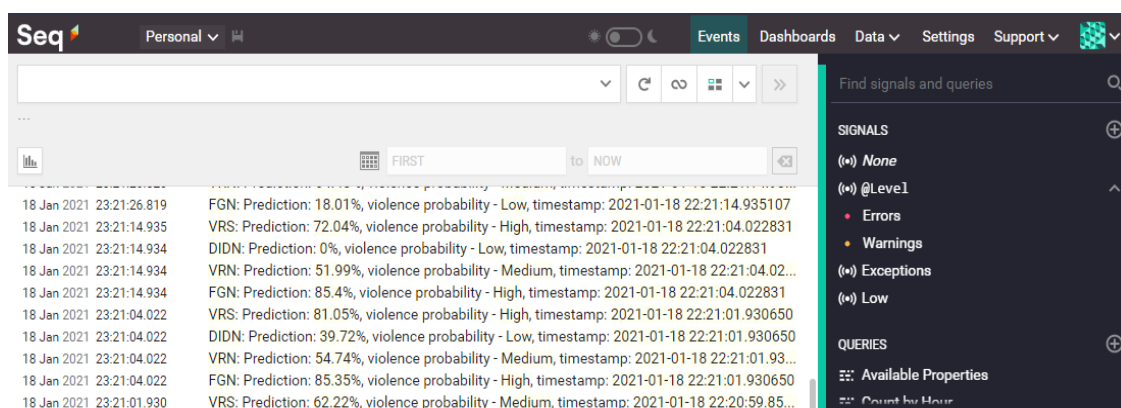


Rysunek 6.1 Diagram wdrożenia przedstawiający architekturę systemu

6.1.2. Działanie systemu

Sercem naszego systemu jest aplikacja, której zadaniem jest utrzymywanie połączenia z kamerą, zbieranie oraz dalsze rozsyłanie klatek do analizy dla poszczególnych modułów. Spełnia ona również zadanie źródła komunikacji ze wszystkimi pozostałymi komponentami, jak i również punktu startowego dla użytkownika.

Działanie rozpoczyna się od konfiguracji aplikacji, której wymagany parametrem jest adres kamery, na bazie której będzie pobierać dane do dalszej analizy. Po skonfigurowaniu systemu przechodzimy do jego uruchomienia. Jeżeli adres kamery był poprawny i na drodze połączenia nie wystąpił żaden problem, serwer zaczyna nasłuchiwać sygnał wideo. Następnie przygotowuje serie danych i rozsyła do wszystkich modułów. Z kolei aplikacja oczekuje na odpowiedź zawierającą prawdopodobieństwo wystąpienia sytuacji niebezpiecznej na analizowanym zestawie klatek od wszystkich modułów i używa naszej autorskiej funkcji (Równanie 1), która wydaje ostateczny werdykt dotyczący klasyfikacji. Ostatnim krokiem jest przesłanie zebranych wyników do serwera Seq, gdzie użytkownik może na bieżąco przeglądać oraz filtrować wpisy. Dodatkowo z poziomu panelu Seq użytkownik ma możliwość zarządzania przechowywanymi danymi, jak również dodatkowej opcji utworzenia alertów dostosowanych do jego potrzeb.



Timestamp	Log Entry
18 Jan 2021 23:21:26.819	FGN: Prediction: 18.01%, violence probability - Low, timestamp: 2021-01-18 22:21:14.935107
18 Jan 2021 23:21:14.935	VRS: Prediction: 72.04%, violence probability - High, timestamp: 2021-01-18 22:21:04.022831
18 Jan 2021 23:21:14.934	DIDN: Prediction: 0%, violence probability - Low, timestamp: 2021-01-18 22:21:04.022831
18 Jan 2021 23:21:14.934	VRN: Prediction: 51.99%, violence probability - Medium, timestamp: 2021-01-18 22:21:04.02...
18 Jan 2021 23:21:14.934	FGN: Prediction: 85.4%, violence probability - High, timestamp: 2021-01-18 22:21:04.022831
18 Jan 2021 23:21:04.022	VRS: Prediction: 81.05%, violence probability - High, timestamp: 2021-01-18 22:21:01.930650
18 Jan 2021 23:21:04.022	DIDN: Prediction: 39.72%, violence probability - Low, timestamp: 2021-01-18 22:21:01.930650
18 Jan 2021 23:21:04.022	VRN: Prediction: 54.74%, violence probability - Medium, timestamp: 2021-01-18 22:21:01.93...
18 Jan 2021 23:21:04.022	FGN: Prediction: 85.35%, violence probability - High, timestamp: 2021-01-18 22:21:01.930650
18 Jan 2021 23:21:01.930	VRS: Prediction: 62.22%, violence probability - Medium, timestamp: 2021-01-18 22:20:59.85...

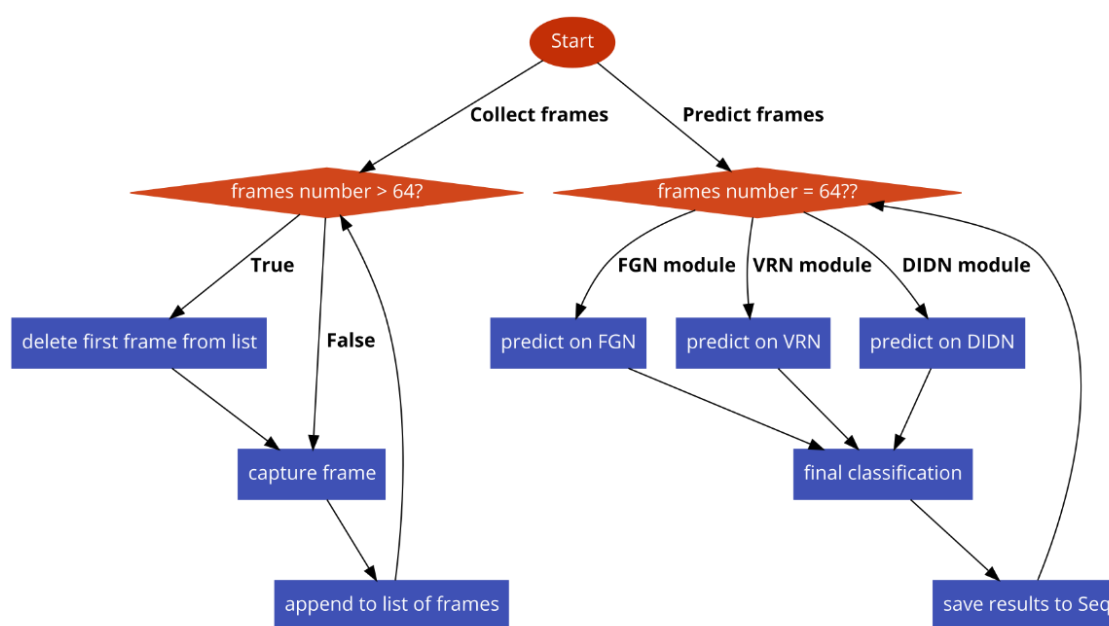
Rysunek 6.2 Zrzut ekranu przykładowych logów systemu Seq

6.1.2.1. Zbieranie klatek

Proces składa się z dwóch etapów: pierwotna inicjalizacja zestawu oraz ciągłe jego uzupełnianie. Analiza na modelu FGN wymaga zestawu liczącego 64 elementy.

Dedykowany temu zadaniu wątek nasłuchuje źródło sygnału i kolekcjonuje klatkę po klatce do momentu zebrania wszystkich 64. Następnie przed pobraniem następnej klatki system wyrzuca najstarszą z kolekcji i dodaje najnowszą na koniec listy. W międzyczasie aplikacja sprawdza liczbę elementów w zbiorze i jeżeli jest równa 64, to adekwatnie dla każdego modułu zleca proces transformacji na odpowiedniej liczbie elementów zbioru. Następnie przekazując go do analizy. Moduł VRN potrzebuje 2 klatek, natomiast DIDN tylko jednej. Ze względu na to, że do wydania ostatecznego werdyktu aplikacja potrzebuje wyniku z każdego modułu, system czeka na zebranie wystarczającej dużej zbioru na rozpoczęcie całego procesu

Drugi etap to kontynuacja monitorowania sygnału źródłowego oraz dalsze zbieranie klatek. W przypadku, gdy z jakiegokolwiek powodu, wątek monitorujący nie będzie w stanie pobrać nowej klatki, nie dojdzie do zlecenia transformacji oraz wykonania następnej predykcji. Natomiast system, cały czas nasłuchując źródło będzie próbował pobrać nową klatkę, do momentu jego zatrzymania. Opisany proces można przedstawić za pomocą schematu blokowego widocznego poniżej.

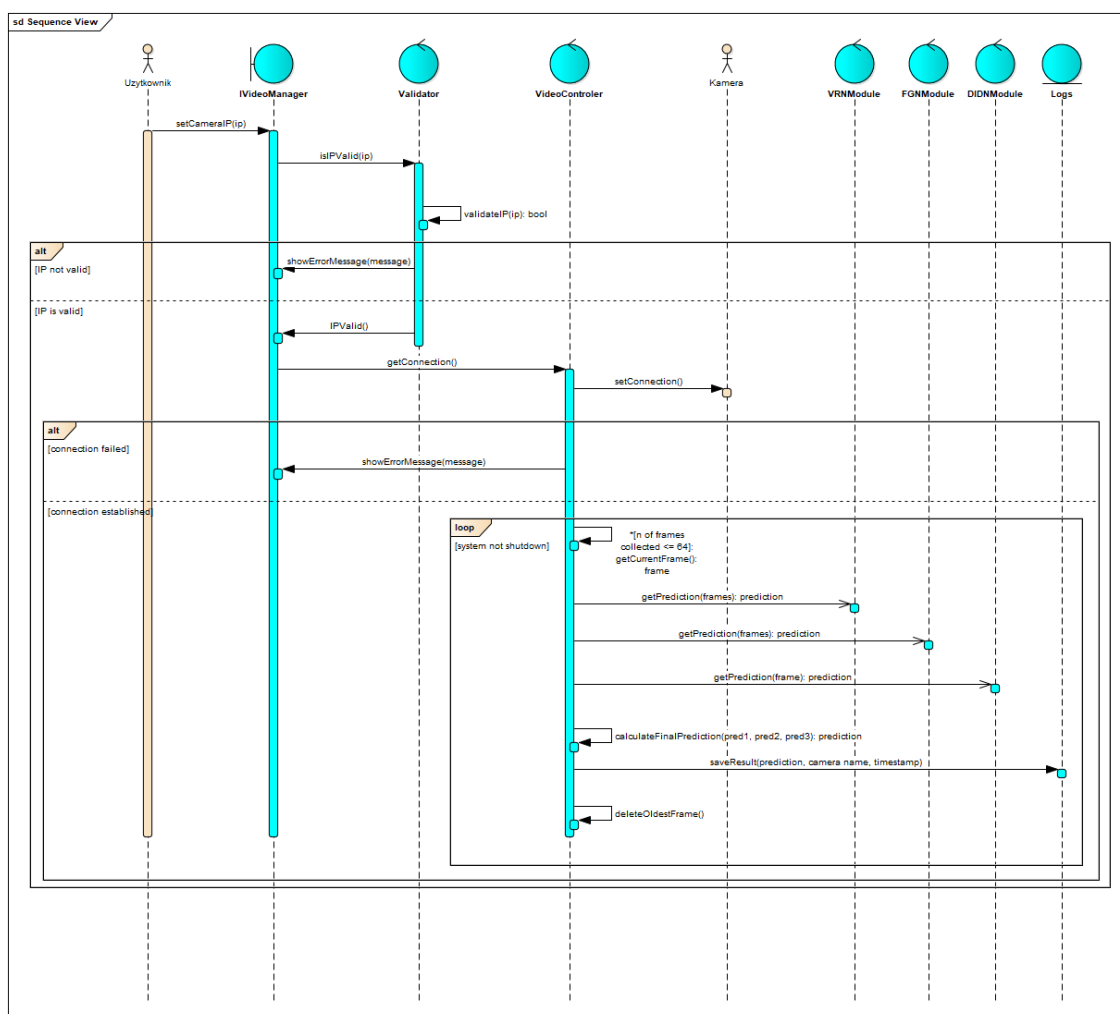


Rysunek 6.3 Schemat blokowy procesu głównej pętli systemu

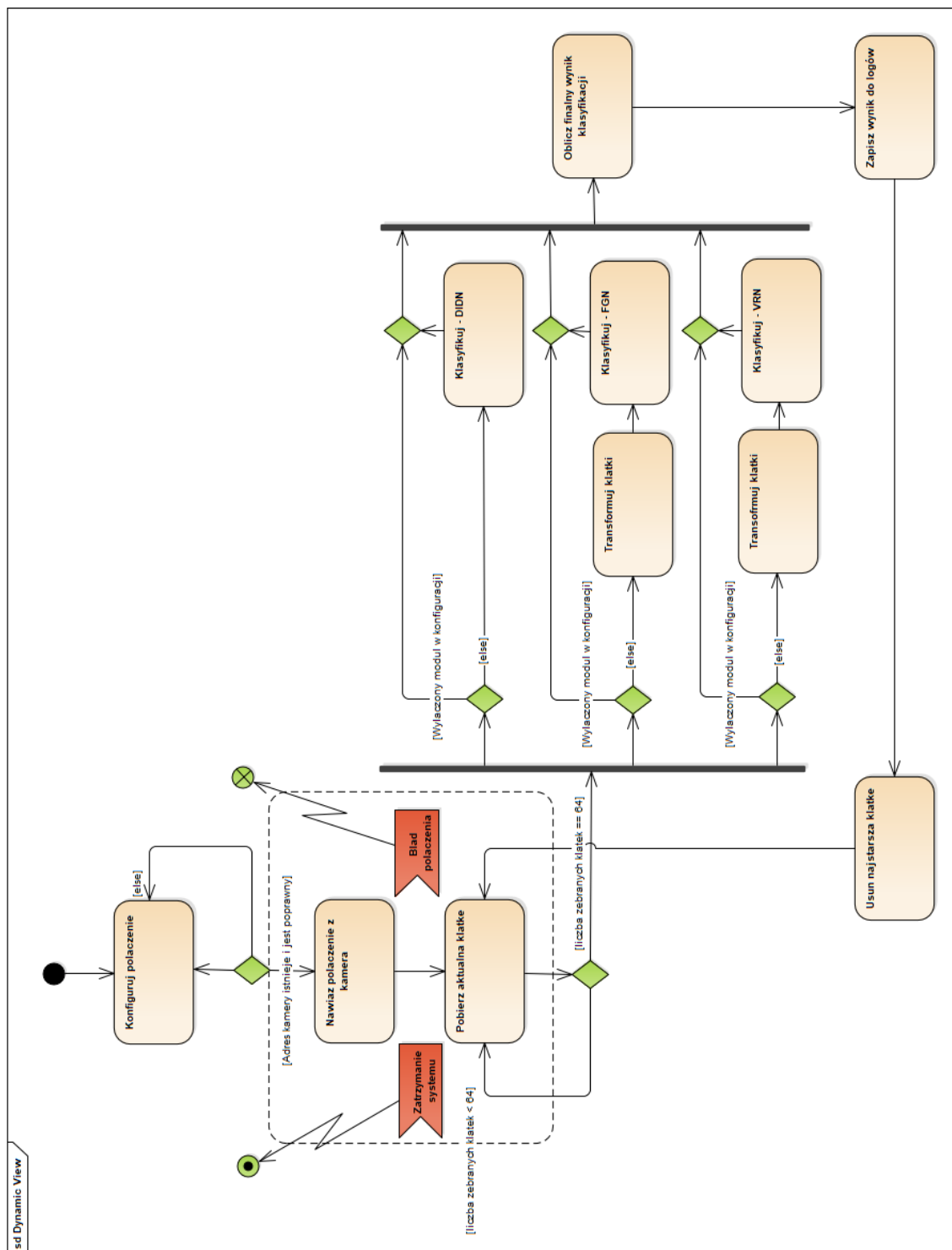
6.1.2.2. Rozesłanie klatek

Proces polegający na serializacji danych z wykorzystaniem protokołu Protobuf do formy binarnej oraz przesłaniu klatek do modułu. Jest to proces następujący po transformacji i przed wykonaniem samej predykcji w module.

Po transformacji danych, adekwatnych dla modułu, system przetwarza je używając gRPC i tworzy zapytanie, które następnie zostanie wysłane do odpowiedniego kontenera z wytrenowanym modelem. Po dokonaniu predykcji moduł zwraca odpowiedź z liczbą zmiennoprzecinkową reprezentującą szansę na występowanie przemocy w analizowanym zbiorze danych. Przed zapisaniem wyniku w logu, system oczekuje na zwrócenie predykcji z pozostałych wątków.



Rysunek 6.4 Diagram sekwencji



Rysunek 6.5 Diagram czynności

6.1.3. Funkcja finalnej klasyfikacji

Niżej opisany wzór służy nam do finalnej klasyfikacji przemocy na danej serii klatek:

$$f_0(x) = \sum_{i=0}^n x_i w_i$$
$$f(x) = \begin{cases} 1, & f_0(x) > 1 \\ f_0(x), & f_0(x) \leq 1 \end{cases}$$

gdzie:

f – funkcja finalnej klasyfikacji, $f \in \langle 0, 1 \rangle$

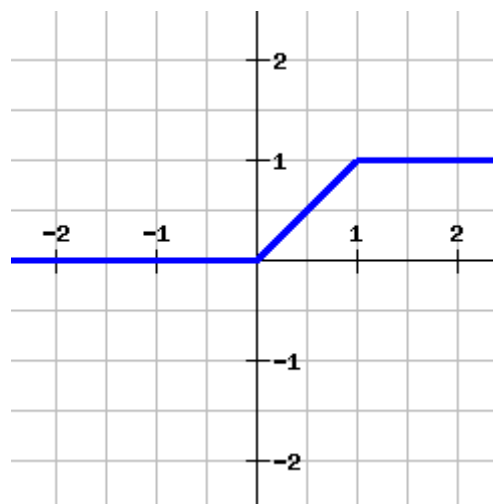
n – ilość modułów, $n > 0, n \in N$

x_i – wynik klasyfikacji i – tego modułu, $x_i \in \langle 0, 1 \rangle$

w_i – waga i – tego modułu, $w_i \in \langle 0, 1 \rangle$

Równanie 1 Funkcja finalnej klasyfikacji systemu VRS

Założeniem było stworzenie wzoru, który brałby pod uwagę dokładność modułów i uwzględniał to w swoim wyniku. Dlatego postanowiliśmy zastosować wzór, w którym mnożymy każdy z wyników przez przypisaną mu wagę (w zależności od modułu) a następnie ograniczamy od góry jego wartość do 1 (Rysunek 6.6). Wagi nie sumują się do jedynki, stąd konieczność ograniczenia.



Rysunek 6.6 Wykres funkcji finalnej klasyfikacji

6.2. Wybrane technologie

6.2.1. Język programowania

Python jest językiem wysokiego poziomu, o szerokim wachlarzu dostępnych bibliotek wspomagających proces wytwarzania rozwiązań. Zdecydowaliśmy się na niego ze względu na dużą ilość rozszerzeń umożliwiających transformację obrazu oraz narzędzi do uczenia sztucznej inteligencji.

6.2.2. Biblioteki i framework'i

6.2.2.1. Keras

Narzędzie do tworzenia sieci neuronowych, wykorzystujące TensorFlow jako backend. Wybraliśmy je ze względu na nasze doświadczenie z nim. W porównaniu z konkurencyjnymi rozwiązaniami jest bardzo dobrze udokumentowane oraz dzięki swojej popularności zyskało potężną grupę wsparcia online.

6.2.2.2. OpenCV

Biblioteka, którą umożliwiła nam ciągłe nasłuchiwanie sygnału wideo z kamer oraz późniejszego przetworzenia obrazów przed wysłaniem ich do analizy. Okazała się również przydatna w procesie walidacyjnym, gdzie zaznaczała wykryte akty przemocy na klatkach.

6.2.2.3. gRPC

Framework o wysokiej wydajności do transferu danych w formie binarnej. gRPC wykorzystaliśmy do przesyłania klatek w celu analizy do modeli osadzonych w kontenerach TensorFlow Serving [9]. Zrezygnowaliśmy na jego rzecz z tradycyjnego sposobu transferu danych protokołem HTTP, w którym dużo czasu poświęca się na konwersję danych do tekstu. W naszym przypadku, gdzie liczy się każda sekunda gRPC pozwolił się bardziej zbliżyć ku detekcji w czasie rzeczywistym.

6.2.2.4. Seq

System do agregowania logów w formacie strukturalnym (więcej informacji patrz [10]). Jego największą zaletą jest prostota we wdrożeniu oraz używaniu. Zarazem posiada wszystkie potrzebne funkcjonalności do spełnienia roli monitoringu oraz przeglądarki wydarzeń. Pozwana na wyszukiwane i ustawianie filtrów z wykorzystaniem języka

podobnego do SQL, co pozwoli osobnie nietechnicznej na bezproblemowe poruszanie się po systemie.

6.2.2.5. FastAPI

Nowoczesne narzędzie do budowy asynchronicznego REST API. Wyróżnia się prostotą w obsłudze oraz wydajnością w porównaniu do konkurencyjnych rozwiązań. W pierwotnym planie każdy model miał być udostępniany przez API zbudowane z wykorzystaniem FastAPI. Narzędzie zostało porzucone w momencie odkrycia TensorFlow Serving, które zostało stworzone do udostępniania modeli zapisanych w formacie TensorFlow.

6.2.2.6. TensorFlow Serving

Skonteneryzowany obraz do serwowania modeli zapisanych w formacie TensorFlow z wykorzystaniem Docker (patrz [9]). Zauważyliśmy jego przewagę nad autorskim API opartym o FastAPI głównie w sferze możliwości i optymalizacji sposobu obsługi wyuczonych modeli. Pozwala na umieszczenie wielu modeli oraz specyfikowanie indywidualnej konfiguracji dla każdego z nich. Jako kanał transferu danych udostępnia protokół HTTP oraz gRPC. Posiada wbudowaną obsługę wsadowego przetwarzania danych oraz rozgrzewania modelu co okazuje się przydatne w przypadku, gdzie pierwsze kilka predykcji zajmuje więcej czasu niż inne, np. przy analizie obrazu lub dzięki.

6.2.3. Inne narzędzia

6.2.3.1. Discord

Aplikacja, która ułatwiła nam komunikację oraz koordynowanie pracy zespołu. Discord pozwala na tworzenie kanałów z czatem tekstowym, głosowym oraz opcją udostępniania plików i ekranu.

6.2.3.2. YouTrack

Tablica Kanban do organizacji pracy w metodykach zwinnych (patrz [11]). Umożliwiła nam organizowanie pracy w przyroście oraz śledzenie progresu całego zespołu.

6.2.3.3. Syncthing

Syncthing [12] to aplikacja pozwalająca na proste współdzielenie plików w modelu peer-to-peer. Wykorzystaliśmy ją do zbierania i dzielenia się zbiorem materiałów wideo służącym do uczenia sieci.

6.2.3.4. MS Teams

Komunikator głosowy z czatem tekstowym oraz opcją współdzielenia plików i ekranu. MS Teams używamy do komunikacji z konsultantami projektowymi.

6.2.3.5. Google Hangouts

Komunikator głosowy z czatem tekstowym. Google Hangouts jest wykorzystany przez nas do organizowania spotkań z resztą konsultantów.

6.2.3.6. PyCharm

Środowisko programistyczne dla języka Python. Podczas optymalizacji czasu wykonywania analizy, wyjątkowo przydatna okazała się funkcja profilowania kodu. Pomogła zlokalizować funkcje, które zajmowały najwięcej czasu.

6.2.3.7. Docker

Narzędzie do konteneryzacji oprogramowania. Dzięki niemu przygotowaliśmy uniwersalne kontenery, które mogły być szybko wdrożone bez wglądu na konfigurację systemu klienta. Kontenery definiujemy wykorzystując pliki Dockerfile w formacie yaml.

6.2.3.8. Docker Compose

Pozwana na definiowanie i uruchamianie złożonych aplikacji składających się z wielu komponentów w środowisku Docker. Szczegóły dotyczące konfiguracji i odpalenia zestawu kontenerów ustalamy w plikach yaml.

6.2.3.9. Enterprise Architect

Zaawansowane środowisko do modelowania diagramów, głównie z wykorzystaniem UML. W przypadku diagramów klas pozwala na wygenerowanie kodu oraz synchronizację powstałego kodu z modelem.

6.2.3.10. Github

System kontrolowania wersji umożliwiający pracę nad współdzielonym projektem. Github umożliwił nam zespołową pracę nad implementacją.

6.3. Dataset

6.3.1. Dataset z filmami

Do wyuczenia sieci modułów opartych na serii klatek, potrzebny był ogromny zestaw danych wideo (w formacie mp4 bądź avi) w liczbie co najmniej 1,000 plików zawierających sceny przemocy, oraz 1,000 plików zawierających brak jakiejkolwiek sceny walki. W celu zebrania odpowiedniego zbioru danych podjęliśmy się próby znalezienia publicznych datasetów oraz na podstawie analizy i prób uczenia sieci, wybrać ten, który najbardziej spełnia nasze oczekiwania.

6.3.1.1. Wstępne wymagania

W celu pozyskania odpowiedniego zbioru danych przyjęliśmy ogólne wymagania jakich oczekujemy od datasetu. Niżej w tabeli opisano te wymagania oraz ich wagę w skali 1-5 (gdzie 1 oznacza bardzo mało istotne, a 5 bardzo ważne wymaganie):

Nazwa	Opis	Waga
Ilość	Posiada co najmniej 1,000 plików wideo z przemocą oraz 1,000 plików bez	5
Format	Posiada pliki wideo w formacie mp4 bądź avi	3
Zbiór walidacyjny	Zawiera podział na zbiór treningowy oraz walidacyjny	1
Czas	Długość plików wideo nie przekracza 10 sekund	2
Długość	Długość wszystkich plików wideo w zbiorze jest równa	5
Dostęp	Jest publicznie dostępny i nie posiada żadnych licencji uniemożliwiających wykorzystanie go w projekcie	4
Jakość	Dane wideo są w jakości co najmniej 480p	4
Obraz	Obraz w nagraniach wideo jest statyczny	5
FPS	Pliki wideo posiadają minimum 30 klatek na sekundę	5

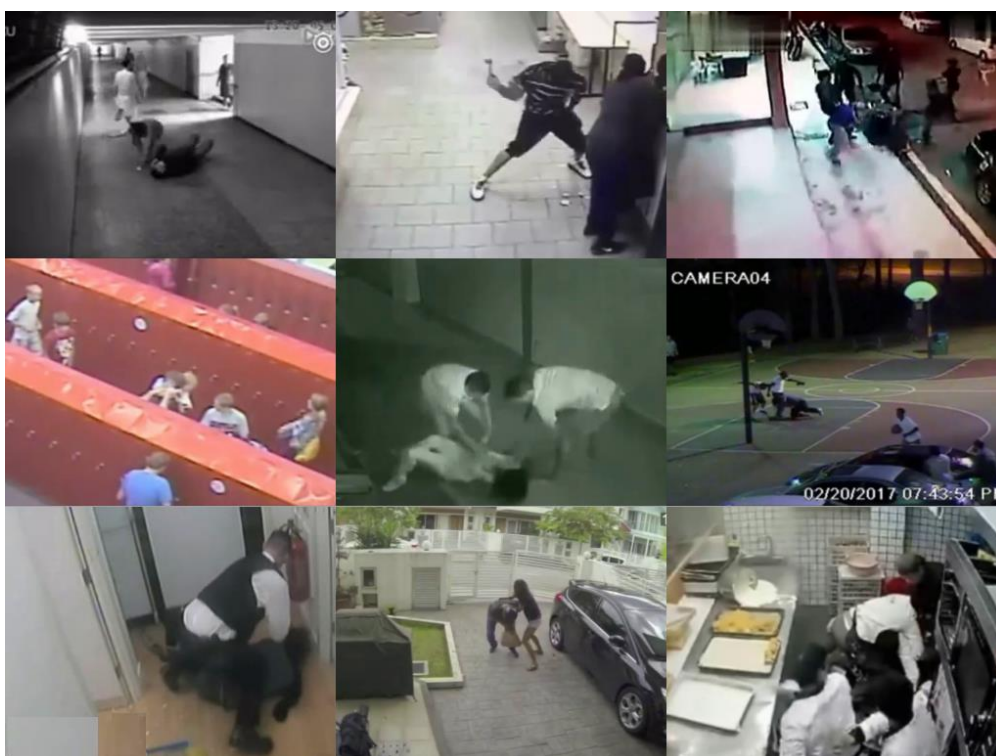
Tabela 6.1 Przedstawienie głównych wymagań dotyczących zbioru danych z filmami

6.3.1.2. RWF-2000 Dataset

Na etapie przeszukiwania sieci w celu odnalezienia odpowiedniego zbioru danych natrafiliśmy na RWF-2000 Dataset [13]. Biorąc pod uwagę nasze wstępne założenia opisane w punkcie wyżej, uznaliśmy ten zestaw danych jako najbardziej właściwy do rozwiązania naszego problemu.

Charakterystyka

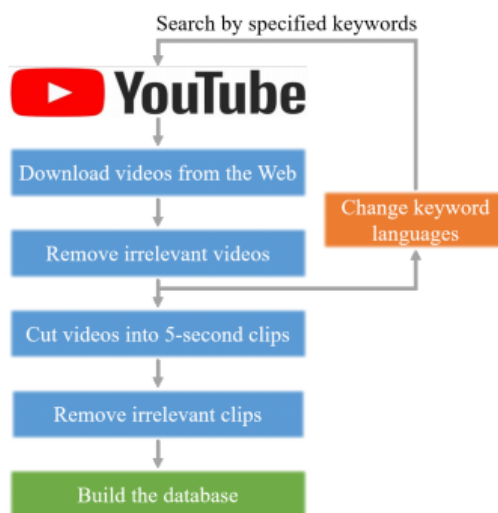
Łączny zbiór 2,000 plików wideo pobranych z platformy YouTube zawierających 1,000 nagrań z kamer monitorujących na których znajduje się przemoc, oraz 1,000 nagrań bez jakichkolwiek aktów przemocy. Długość wszystkich plików wideo jest jednakowa (5 sekund) oraz dataset zawiera uprzedni podział na zbiór treningowy (80%) oraz walidacyjny (20%).



Rysunek 6.7 Zestaw kilku wyciętych scen walki z plików wideo zawierających się w RWF-2000 Dataset

Jak opisują trzej profesorowie z Chin w swojej dokumentacji, pozyskanie zbioru RWF-2000 opierało się na wyszukaniu słów kluczowych w różnych językach, w tym angielski, chiński, niemiecki, francuski, japoński, rosyjski itp. w serwisie YouTube

podając słowa kluczowe związane z przemocą jak np. real fights, violence undersurveillance, violent events.



Rysunek 6.8 Przebieg pozyskiwania kolekcji danych RWF-2000 [5]

Zalety

- Zawiera łącznie 2,000 plików wideo
- Jednakowa długość filmów (5 sekund)
- Format danych avi
- Dobra jakość danych wideo
- Obraz na nagraniach wideo jest statyczny
- Minimum 30 klatek na sekundę w każdym wideo
- Podział na zbiór walidacyjny

Wady

- Dostęp po uprzedniej prośbie

Podsumowanie

Jeżeli prośba o uzyskanie dostępu zostanie zaakceptowana (jak w naszym przypadku), można za pomocą tego datasetu uzyskać bardzo dobre wyniki klasyfikacji przemocy (Rysunek 7.1 Korespodencja między właścicielem RWF-2000 Dataset a członkami zespołu). Zbiór danych ten jest idealnie przystosowany do rozwiązania tego

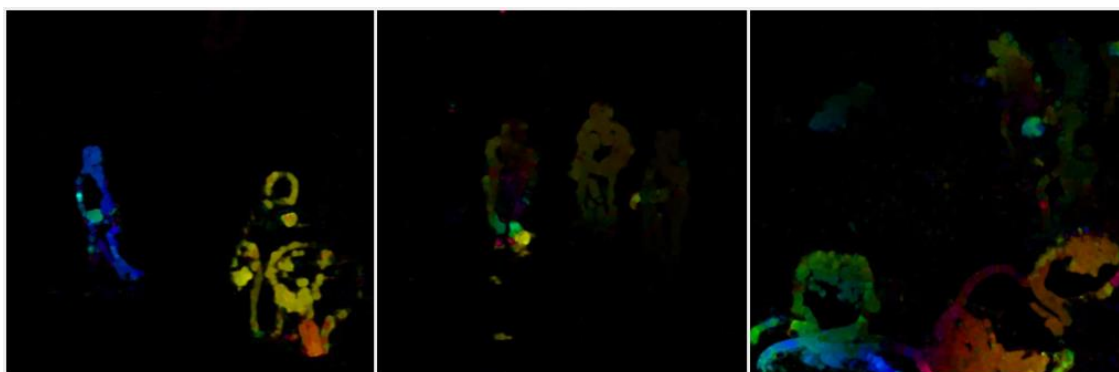
typu problemu poprzez np. równą długość plików wideo, podział na zbiór walidacyjny i treningowy, format danych jak nawet jakość danych i liczba klatek na sekundę.

6.3.1.2.1. RWF-2000 Image Subset

Zbiór danych RWF-2000 Image Subset to podzbiór RWF-2000 powstały w procesie ekstrakcji klatek z plików wideo przy użyciu skryptu napisanego w języku Python. Obrazy zostały pozyskane z około dwustu pięciosekundowych nagrań zawierających przemoc i odpowiednio taką samą ilość niezawierających przemocy. Proces ekstrakcji polegał na wyciągnięciu 30 klatek z każdej sekundy plików wideo. Rezultatem tego otrzymaliśmy zbiór danych posiadający łącznie 57 900 zdjęć.

6.3.1.2.2. RWF-2000 OpenCV Subset

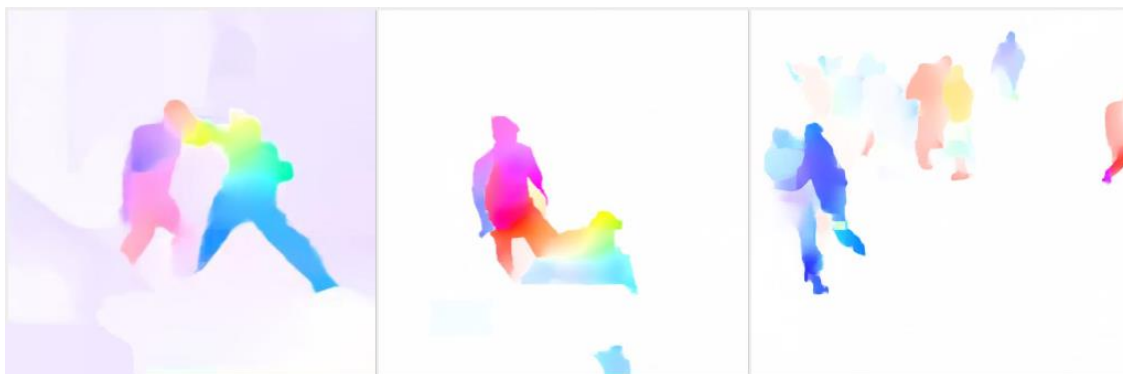
Zbiór danych RWF-2000 OpenCV Subset to podzbiór RWF-2000 powstały przy pomocy skryptu napisanego w języku Python. Zadaniem był proces ekstrakcji przepływu optycznego (ang. optical flow) pomiędzy klatkami za pomocą biblioteki OpenCV [14] i zapisania wyników w postaci odrębnego zbioru.



Rysunek 6.9 Przykłady klatek znajdujących się w RWF-2000 OpenCV Subset

6.3.1.2.3. RWF-2000 RAFT Subset

Zbiór danych RWF-2000 RAFT Subset to podzbiór RWF-2000 powstały przy pomocy skryptu napisanego w języku Python. Zadaniem było wykorzystanie aktualnie najlepszej metody typu Optical Flow [15] o nazwie RAFT [16] i zapisanie wyników w postaci odrębnego zbioru.



Rysunek 6.10 Przykład klatek znajdujących się w RWF-2000 RAFT Subset

6.3.1.3. Inne zbiory danych

Zanim udało nam się znaleźć odpowiedni zbiór danych jakim jest RWF-2000, przeprowadziliśmy krótką analizę wszystkich zebranych przez nas datasetów.

W poniższych podpunktach zostały opisane trzy zestawy danych, do których udało nam się uzyskać dostęp.

6.3.1.3.1. Real Life Violence Situations Dataset

Charakterystyka

Real Life Violence Situations Dataset [17] zawiera 1,000 plików wideo z przemocą oraz 1,000 plików wideo bez przemocy, zebranych z platformy YouTube. Głównie zawiera sceny walk ulicznych w różnych okolicznościach oraz wszelakich akcji takich jak: chodzenie, jedzenie, sport itp.

Zalety

- Jest publicznie dostępny
- Zawiera łącznie 2,000 plików wideo
- Format danych mp4
- Dobra jakość danych wideo

Wady

- Obraz na nagraniu wideo nie jest statyczny
- Różne długości plików wideo (od 3 sekund do 2 minut)

Podsumowanie

Ten zbiór danych może służyć do wyuczenia sieci na zadawalającym poziomie. Natomiast brak statycznego obrazu nie pozwala nam wybrać tego datasetu ze względu na planowane użycie metody Optical Flow.

6.3.1.3.2. UCF-ARG Dataset

Charakterystyka

UCF-ARG Dataset [18] zawiera między innymi 144 pliki wideo scen boksowania z różnych kątów, oraz 1,296 plików wideo z akcjami takimi jak: chód, bieg, machanie itd.

Zalety

- Jest publicznie dostępny
- Format danych mp4
- Dobra jakość danych wideo
- Obraz na nagraniu wideo jest statyczny

Wady

- Różne długości plików wideo (od 40 sekund do 2 minut)
- Zbyt mała ilość plików wideo
- 90% zbioru zawiera pliki wideo bez przemocy

Podsumowanie

Ten zbiór danych może służyć głównie jako uzupełnienie już istniejącego zbioru, natomiast nie nadaje się, aby oprzeć na nim całe uczenie sieci ze względu na małą ilość plików wideo oraz zbyt małą ich różnorodność.

6.3.1.3.3. NTU CCTV-Fights Dataset

Charakterystyka

NTU CCTV-Fights Dataset [19] zawiera 1,000 plików wideo zebranych z platformy YouTube, z podziałem na nagrania z kamer CCTV i kamer mobilnych.

Liczba plików wideo	
Wszystkie	1,000
CCTV	280
Kamera mobilna	720

Tabela 6.2 Podział plików wideo w zależności od rodzaju kamery w zbiorze danych NTU CCTV-Fights [19]

Zalety

- Format danych mp4
- Obraz na części nagrań wideo jest statyczny (CCTV)
- Każda sekunda plików wideo jest opisana

Wady

- Różne długości plików wideo (od 3 sekund do 12 minut)
- Zbyt mała ilość plików wideo statycznych
- Zbyt zróżnicowana jakość danych wideo
- Dostęp po uprzedniej prośbie

Podsumowanie

Ten zbiór danych może służyć głównie jako wzbogacenie już istniejącego zbioru, natomiast nie nadaje się, aby oprzeć na nim całe uczenie sieci ze względu na małą ilość statycznych plików wideo.

6.3.2. Dataset ze zdjęciami

W celu wyuczenia sieci wykrywającej obiekty, należało zdobyć dataset zdjęć zawierający zdjęcia broni wraz z plikami (adnotacjami), które posiadały informacje na temat tego, gdzie na danym zdjęciu znajduje się broń.

6.3.2.1. Wstępne wymagania

W celu pozyskania odpowiedniego zbioru danych przyjęliśmy ogólne wymagania jakich oczekujemy od tego datasetu. Niżej w tabeli opisano te wymagania oraz ich wagę w skali 1-5 (gdzie 1 oznacza bardzo mało istotne, a 5 bardzo ważne wymaganie):

Nazwa	Opis	Waga
Ilość	Posiada co najmniej 500 zdjęć	5
Format	Preferowany format zdjęć to jpg lub png	1
Posiadanie plików z lokalizacją	Zdjęcia powinny posiadać pliki zawierające dane opisujące położenie interesującego nas przedmiotu na zdjęciu	5
Dostęp	Jest publicznie dostępny i nie posiada żadnych licencji uniemożliwiających wykorzystanie go w projekcie	4

Tabela 6.3 Przedstawienie głównych wymagań dotyczących zbioru zdjęć

6.3.2.2. Open Images

Ręczne tworzenie takich zbiorów danych jest zadaniem wyjątkowo żmudnym oraz czasochłonnym. Na szczęście istnieje publiczny dataset - Open Images [20], zawierający przeszło 9 milionów zdjęć wraz z odpowiednimi adnotacjami. Dzięki czemu stworzenie odpowiedniego zbioru danych nie było większym wyzwaniem.

Charakterystyka

Zbiór plików pobrany z publicznego datasetu Open Images zawierał 600 zdjęć oraz taką samą ilość plików tekstowych opisujących położenia przedmiotów na obrazie.



Rysunek 6.11 Przykładowe zdjęcia zawarte w Open Images Dataset

Zalety

- Łatwy dostęp do zbioru
- Interfejs użytkownika umożliwiający podgląd zdjęć oraz filtrację
- Pliki w formacie jpg
- Zbiory dostępne publicznie
- Pliki z lokalizacją (adnotacjami)

Wady

- Mała różnorodność prezentowanej broni

Podsumowanie

Fakt, że zbiór jest publicznie dostępny oraz posiada łatwy dostęp, sprawiają, że właśnie ten dataset został wybrany do tej pracy. Jednym z niewielu problemów tego rozwiązania jest mała różnorodność prezentowanych broni co potencjalnie może wpłynąć na możliwość rozpoznawania pewnych typów oręża. Zdecydowaną większość przykładów stanowiły karabiny, co za tym idzie broń krótka może być gorzej rozpoznawana.

6.3.3. Dataset z audio

Zbiór plików audio był nam potrzebny do wyuczeniu sieci neuronowej opartej na spektrogramach jako dodatek do architektury systemu oraz w celu zwiększenia finalnego wyniku predykcji.

6.3.3.1. Wstępne wymagania

W celu pozyskania odpowiedniego zbioru danych przyjęliśmy ogólne wymagania jakich oczekujemy od tego datasetu. W następnej tabeli opisano te wymagania oraz ich wagę w skali 1-5 (gdzie 1 oznacza bardzo mało istotne, a 5 bardzo ważne wymaganie)

Nazwa	Opis	Waga
Ilość	Posiada co najmniej 1,000 plików audio z dźwiękami przemocy oraz 1,000 plików bez	5
Format	Posiada pliki audio w formacie wav	4
Zbiór walidacyjny	Zawiera podział na zbiór treningowy oraz walidacyjny	1
Czas	Długość plików audio nie przekracza 10 sekund	3
Długość	Długość wszystkich plików audio w zbiorze jest równa	5
Dostęp	Jest publicznie dostępny i nie posiada żadnych licencji uniemożliwiających wykorzystanie go w projekcie	4

Tabela 6.4 Przedstawienie głównych wymagań dotyczących zbioru plików audio

6.3.3.2. Zbiór plików audio z przemocą

Niestety nie udało nam się znaleźć żadnych gotowych zbiorów plików audio zawierających przemoc. Dlatego postanowiliśmy napisać skrypt, który wyciąga ścieżki dźwiękowe z wcześniej pobranych plików wideo (pkt 6.3.1.2) i na tej podstawie stworzyć dataset. Natomiast to rozwiązanie również nie dało zakładanych rezultatów i nie spełniało wyżej określonych wymagań (Tabela 6.4).

6.3.3.3. Zbiór plików audio z wystrzałem broni

Zbiór plików audio potrzebnych w celu nauki sieci klasyfikującej wystrzały broni ze ścieżki dźwiękowej. Dla przeciwwagi dźwięku wystrzału broni szukamy również zbiorów zawierających inne dźwięki jak np. krzyk, płacz, upadające przedmioty, klaskanie czy fajerwerki.

6.3.3.3.1. AudioSet

Szukając odpowiedniego zbioru danych audio na stronie AudioSet [21]. Znajduje się tam ogromna ilość przeróżnych zbiorów plików dźwiękowych, w tym również te zawierające wystrzały broni.

Charakterystyka

Zbiór plików dźwiękowych razem z adnotacjami pobrany z platformy YouTube, podzielony na 3 segmenty: zbalansowany treningowy, niezbalansowany treningowy oraz zbalansowany ewaluacyjny.

Zalety

- Duża ilość różnych klas, które będą przydane do walidacji rozwiązania (krzyk, klaskanie, fajerwerki)
- Łączna ilość plików nam potrzebnych wynosi ponad 4,000
- Zbiory dostępne publicznie
- Pliki w formacie wav
- Długość plików dźwiękowych takiej samej długości (10 sekund)

Wady

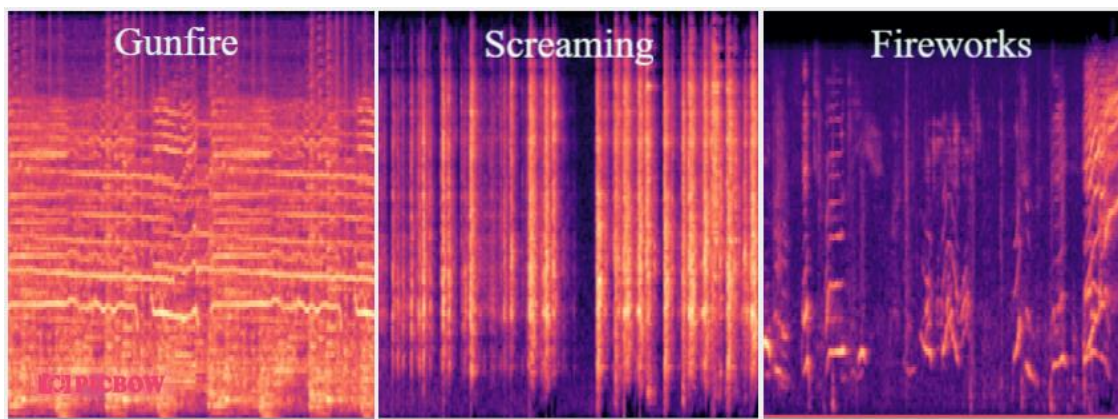
- Pliki audio zawierają dodatkowo plik csv z adnotacjami. Jest konieczność napisania skryptu, który ekstrahuje interesujące nas obszary.

Podsumowanie

Różnorodność oraz ilość klas i plików audio powoduje, że AudioSet jest idealnym rozwiązaniem. Jedyną przeszkodą to brak wyekstrahowanych fragmentów pliku i konieczność zrobienia tego samemu.

6.3.3.3.2. Spektrogramy

Duża część sieci do klasyfikacji dźwięku korzysta z wcześniej przygotowanych obrazów przedstawiających dźwięk tzw. spektrogramów. Dlatego w celu uzyskania finalnej wersji datasetu potrzebne było napisanie skryptu transformującego ścieżki audio na obraz przy pomocy Pythonowej biblioteki – librosa [22].



Rysunek 6.12 Przykład wygenerowanych spektrogramów

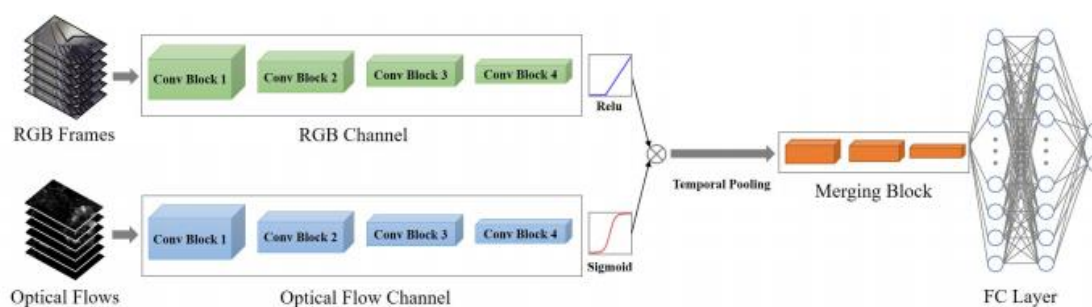
6.4. Moduły

6.4.1. Moduł FGN

Moduł FGN został oparty na sieci opracowanej przez trzech chińskich profesorów Ming Cheng, Kunjing Cai oraz Ming Li, o nazwie Flow Gated Network [13]. Wybraliśmy właśnie tę architekturę sieci ze względu na niesamowite wyniki jakie zostały osiągnięte i opisane w pracy wspomnianych profesorów.

6.4.1.1. Struktura sieci

Struktura sieci FGN składa się z: kanału RGB, kanału Optical Flow, bloku łączącego oraz warstwy Fully Connected.



Rysunek 6.13 Struktura sieci FGN [13]

Kanały RGB oraz Optical Flow składają się z kaskadowego 3D CNN i posiadają spójną strukturę w celu późniejszego ich złączenia. Dodatkowo na końcu kanału RGB jest zastosowana funkcja aktywacji Relu, natomiast na kanale Optical Flow zastosowano

Sigmoidalną funkcję aktywacji. Następnie wyjścia z kanału RGB oraz Optical Flow są mnożone i przetwarzane przez Temporal Pooling. Ponieważ funkcja Sigmoidalna zwraca wynik między 0 a 1, wynik kanału Optical Flow jest współczynnikiem skalującym w celu dostosowania wyniku z kanału RGB. Dlatego też wynik z kanału RGB pomnożony przez 1 będzie posiadać duże prawdopodobieństwo na zostanie zachowanym, a wartość pomnożona przez zero ma większe szanse na zostanie usuniętym. Podsumowując, mechanizm Temporal Pooling wykorzystuje kanał Optical Flow jako współczynnik do określenia, które informacje model powinien zachować a które porzucić. Po warstwie odpowiedzialnej za przetworzenie Temporal Pooling znajduje się blok łączący, który jest również złożony z kaskadowego 3D CNN. Na koniec, warstwy Fully Connected generują wynik wyjściowy.

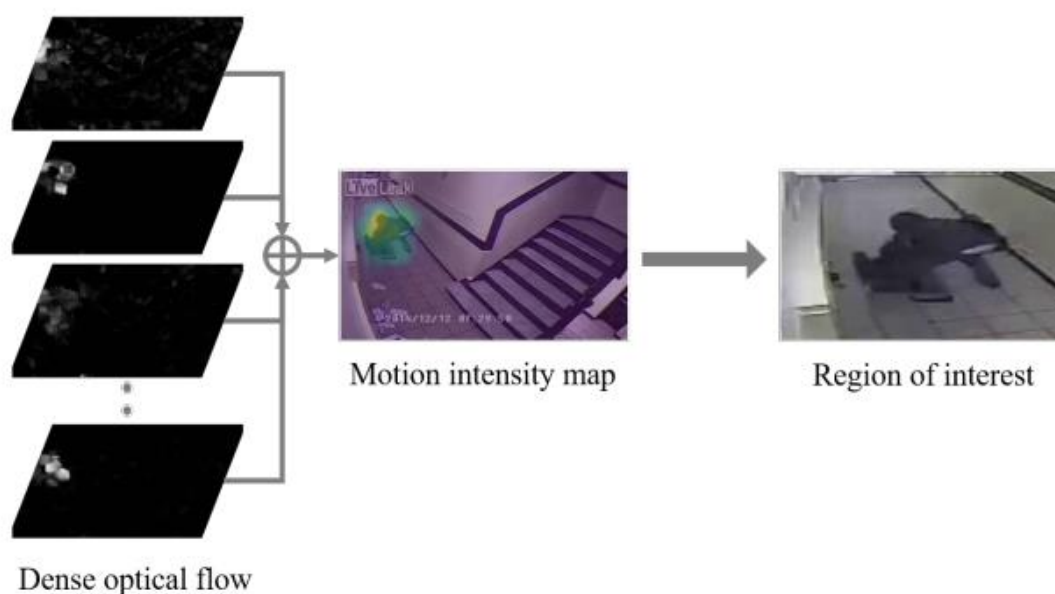
Nazwa bloku	Typ	Kształt filtru	Ilość powtórzeń
Kanał RGB/Optical Flow	Conv3d	$1 \times 3 \times 3 @ 16$	2
	Conv3d	$3 \times 1 \times 1 @ 16$	
	MaxPool3d	$1 \times 2 \times 2$	
	Conv3d	$1 \times 3 \times 3 @ 32$	2
	Conv3d	$3 \times 1 \times 1 @ 32$	
	MaxPool3d	$1 \times 2 \times 2$	
Fuzja i Pooling	Multiply	None	1
	MaxPool3d	$8 \times 1 \times 1$	1
Blok łączący	Conv3d	$1 \times 3 \times 3 @ 64$	2
	Conv3d	$3 \times 1 \times 1 @ 64$	
	MaxPool3d	$2 \times 2 \times 2$	
	Conv3d	$1 \times 3 \times 3 @ 128$	1
	Conv3d	$3 \times 1 \times 1 @ 128$	
	MaxPool3d	$2 \times 3 \times 3$	
	MaxPool3d	$2 \times 3 \times 3$	
Warstwy Fully Connected	FC layer	128	2
	Softmax	2	1

Tabela 6.5 Parametry architektury modelu FGN

6.4.1.2. Podstawowe ustawienia

Przeważnie interesujący nas obszar, na którym znajduję się aktywność jest bardzo mały względem wielkości klatki. Dlatego zaimplementowano strategię przycinania i próbkowania w celu redukcji wejściowych danych wideo.

Najpierw stosujemy metodę Gunner Farnebacka [23] do obliczenia gęstego Optical Flow pomiędzy sąsiadującymi klatkami. Obliczony gęsty Optical Flow jest wektorem 2D przemieszczenia. Tak więc obliczamy normę każdego wektora w celu uzyskania mapy cieplnej jako ostatecznej mapy natężenia ruchu. Następnie obszar zainteresowania jest wydobywany z miejsca o największym natężeniu ruchu.



Rysunek 6.14 Strategia przycinania przy użyciu gęstego Optical Flow [13]

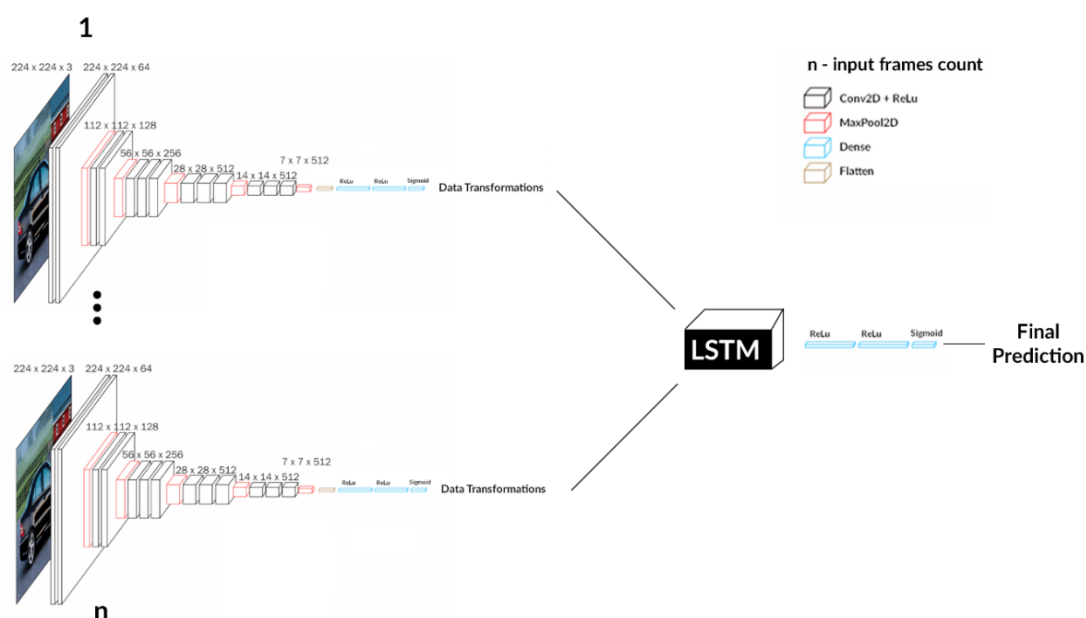
Rozmiar danych wejściowych to: liczba klatek \times wysokość \times szerokość \times 5. Gdzie wysokość \times szerokość to wcześniej ustalone hiperparametry (rozmiar zdjęcia wejściowego) a ostatni wymiar to 5 kanałów zawierających 3 kanały RGB oraz 2 kanały Optical Flow (poziomy i pionowy).

6.4.2. Moduł VRN

Moduł VRN (Violence Recognition Network) to nasz autorski pomysł na rozwiązanie problemu detekcji przemocy. Jego działanie opiera się na znajdowaniu cech charakterystycznych ruchów kojarzonych z aktami przemocy. Motywacją do zrobienia tego modułu była chęć wytworzenia przez nas autorskiej sieci i użycia nowego połączenia wcześniej wykorzystywanych technologii.

6.4.2.1. Struktura sieci

Podstawą sieci VRN jest sieć VGG16 [24] zaimplementowana od zera bez korzystania z ogólnodostępnego zestawu wag ImageNet [25], wyuczona zbiorze RWF-2000 Image Subset (pkt 6.3.1.2.1). Po wyuczeniu wyżej opisanej sieci VGG16, wykonywane są predykcje na określonym zbiorze klatek a wyniki predykcji są transformowane i dalej wysyłane do sieci LSTM [26], która analizuje dane w kontekście przestrzeni czasu. Na końcu po dodaniu warstw zmniejszających ilość neuronów wyjściowych generuje wynik klasyfikacji binarnej zawierającej prawdopodobieństwo wystąpienia aktu przemocy.



Rysunek 6.15 Architektura modułu VRN [27]

6.4.2.2. Podstawowe ustawienia

Rozmiar danych wejściowych to: $224 \times 224 \times 3$. Gdzie 224×224 to rozmiar wejściowy zdjęć z góry założony przez strukturę VGG16 (dlatego przed wejściem na sieć należy dokonać transformacji rozmiaru). Ostatni wymiar to 3 kanały RGB.

6.4.3. Moduł DIDN

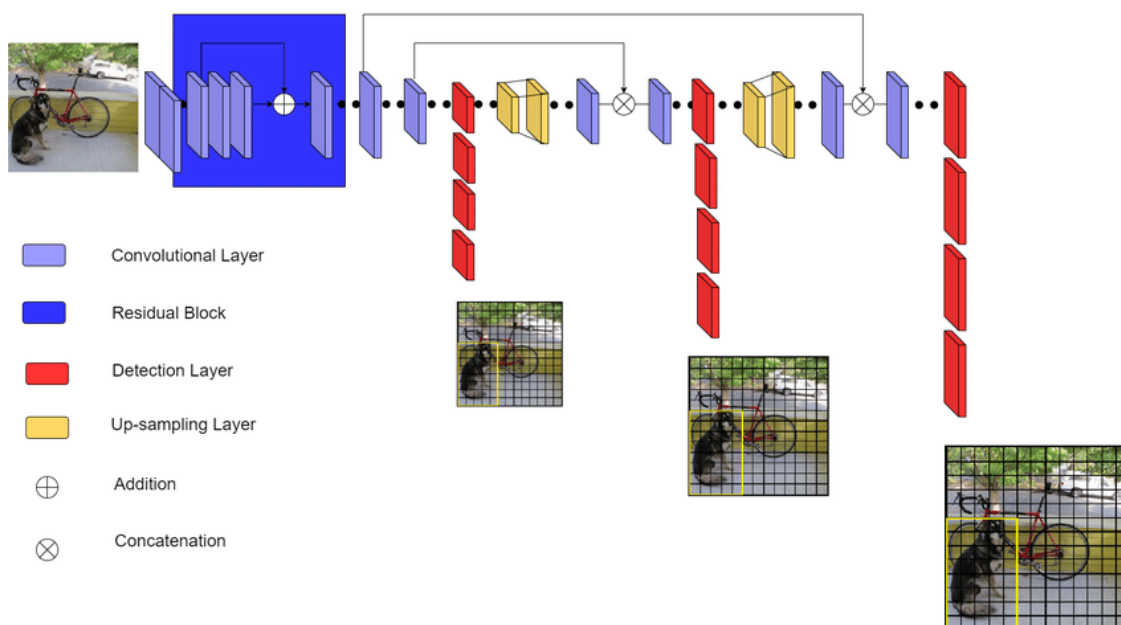
Moduł DIDN (Dangerous Item Detection Network) opiera się na detekcji przedmiotów niebezpiecznych takich jak broń palna na fragmentach wideo.

Zdecydowaliśmy się na utworzenie tego modułu w celu poprawy detekcji przemocy w sytuacjach, gdzie jest widoczna broń.

6.4.3.1. Struktura sieci

Cały moduł oparty został na algorytmie należącym do rodziny YOLO [28]. Ten typ algorytmów wykorzystuje konwolucyjne sieci neuronowe w celu wykrywania obiektów. Są to jedne z najszybszych dostępnych algorytmów. W tym module zdecydowaliśmy się wykorzystać YOLOv3 [29], czyli trzecią wersję tego algorytmu, gdyż oferuje ona dużo lepsze wyniki względem swoich poprzedników.

Algorytm ten wykorzystuje model sieci Darknet-53 (patrz [29]) w celu ekstrakowania cech charakterystycznych ze zdjęć, a następnie używa ich w celu wykrycia niebezpiecznych przedmiotów. Detekcja odbywa się trzykrotnie. Podczas pierwszej, obraz jest podzielony na 169 równych kwadratów (13 wierszy, 13 kolumn). Następnie wybierany jest środkowy kwadrat (7 wiersz, 7 kolumna) i dla tego kwadratu przypisywany jest wynik analizy. Operacja powtarza się dla kolejnych sąsiednich kwadratów. Wyniki tych operacji są grupowane, a znalezionym obszarom przypisywana jest trafność detekcji oraz lokalizacja na obrazie. Warto dodać, iż obszary posiadające niską trafność zostają odrzucone. Taki sam proces powtarzany jest dla obrazu podzielonego na 676 kwadratów (26 wierszy, 26 kolumn) oraz 2704 kwadratów (52 wiersze, 52 kolumny). Kolejny krok to porównanie wyników i wybranie tych najlepszych. Taki zabieg pozwala na trafniejsze wykrywanie przedmiotów różnej wielkości.



Rysunek 6.16 Architektura sieci algorytmu YOLOv3 [29]

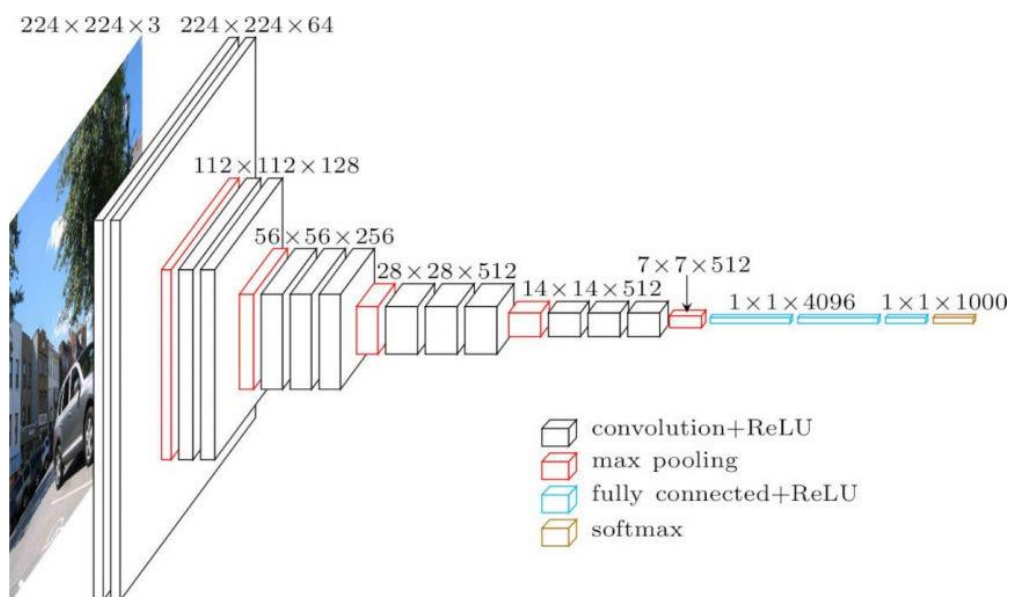
6.4.4. Moduł DSDN

Moduł DSDN (Dangerous Sound Detection Network) opiera się na detekcji sytuacji niebezpiecznej na podstawie ścieżki audio. Wybraliśmy ten moduł do naszego systemu w celu dopełnienia predykcji na wszystkich frontach.

6.4.4.1. Struktura sieci

Sieć VGG16 jest znana z wyciągania cech charakterystycznych ze zdjęć (patrz [24]), dlatego też to właśnie taką architekturę sieci przyjęliśmy w celu klasyfikacji ścieżki audio przekonwertowanej wcześniej na spektrogram (opis rozwiązania patrz [30]).

VGG16 to konwolucyjna architektura sieci neuronowej (CNN). Najbardziej unikalną cechą VGG16 jest to, że zamiast mieć dużą liczbę hiperparametrów, autorzy skupili się na warstwach splotu filtra 3x3 z pojedynczym krokiem i zawsze używali tego samego wypełnienia i warstwy maxpool z filtrem 2x2 z podwójnym krokiem. Ten schemat konwolucji i warstwy maxpool jest powtarzany spójnie w całej architekturze. Na końcu ma dwie warstwy Fully Connected, po których następuje aktywator softmax na wyjściu. Liczba 16 w nazwie „VGG16” odnosi się do szesnastu warstw, które mają wagi w tej sieci. Cała sieć posiada około 138 milionów parametrów.

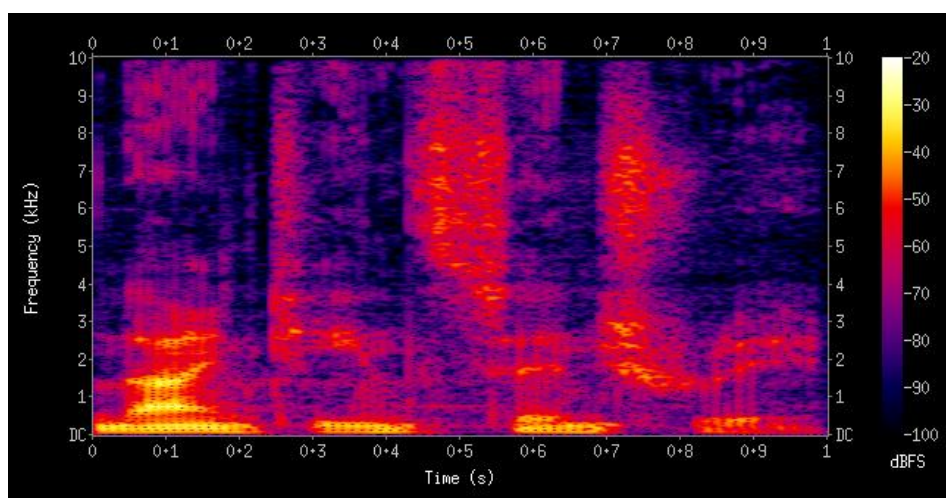


Rysunek 6.17 Architektura sieci VGG16 wykorzystana do modułu DSDN [27]

6.4.4.2. Podstawowe ustawienia

Sieć VGG16 przyjmuje obrazy na wejściu w formacie $224 \times 224 \times 3$, gdzie 224×224 to rozmiar zdjęcia a ostatni wymiar to kanał RGB w wymiarze 3. Najpierw jednak, aby uzyskać zdjęcie z nagrania audio musimy je zamienić na spektrogram.

Spektrogram jest to inaczej przedstawienie częstotliwości dźwięku w czasie (Rysunek 6.18). W naszym założeniu jest to wykres, na którym oś OX reprezentuje czas, OY częstotliwość oraz jako trzeci wymiar zakładamy intensywność koloru jako interpretacja amplitudy częstotliwości.



Rysunek 6.18 Przykład spektrogramu [31]

6.5. Proces nauczania

6.5.1. Maszyna

Niestety przeprowadzenie procesu nauczania na lokalnej maszynie było tylko możliwe przy odpowiednio ustawionych hiperparametrach, które nie zwracały zadowalających wymagań wyników. Brakowało nam mocy obliczeniowej potrzebnej do wyuczenia tak wymagających sieci. Na szczęście ten problem został rozwiązany dzięki uzyskanemu dostępowi do maszyny CloudLab2 posiadającej 6 kart graficznych typu Tesla K80. Możliwość przeprowadzenia procesu nauczania na owej maszynie pozwoliła na więcej sposobów dobierania hiperparametrów, które finalnie doprowadziły do lepszych wyników klasyfikacji. Jednak zanim było to możliwe maszyna musiała zostać skonfigurowana w odpowiedni sposób. W tym procesie zostały przygotowane odpowiednie środowiska przy użyciu narzędzia Conda dla użytku każdego modułu z osobna. Poza tym również została przygotowana architektura plików umożliwiająca archiwizację wszystkich procesów wykonywanych na ów maszynie. Dodatkowo korzystaliśmy również z Google Colaboratory, które oferowało darmowe maszyny wirtualne ze stosunkowo potężnymi kartami graficznymi – NVIDIA Tesla K80. Użycie tej maszyny wymagało pewnych zmian w kodzie, aczkolwiek pozwoliło pomyślnie zakończyć nauczanie.

6.5.2. Wyuczanie modułu FGN

Zebrany wcześniej zbiór plików wideo (pkt 6.3.1) posłużył nam do wytrenowania wcześniej zaimplementowanej sieci FGN (pkt 6.4.1). Przebieg uczenia zaczęliśmy od ustawienia wstępnych hiperparametrów a następnie puszczenia serii „eksperymentów” po których dokonywaliśmy dokładnej analizy wyników.

Dla przyspieszenia oraz możliwości weryfikacji uczonego modelu i ułatwienia trenowania zastosowano autorskie callbacki [32]. Ich zadaniem jest: zapisywanie stanu modelu po każdej epoce, zapisywanie logów z informacjami o przebiegu nauczania oraz optymalizacja współczynnika uczenia.

6.5.2.1. Założenia modelu

W celu akceptacji modułu do części integracyjnej ustaliliśmy następujące wymagania:

- Dokładność modelu FGN na poziomie co najmniej 85% (tj. poprawne rozpoznanie przemocy w 85 przypadkach na 100)
- Predykcja zajmuje jak najmniej czasu (maximum 500 ms)
- Odejście od używania hiperparametrów zasugerowanych przez twórców w celu jak największej optymalizacji

6.5.2.2. Eksperymenty

6.5.2.2.1. Eksperyment 1

Zanim będziemy optymalizować hiperparametry musimy posiadać informację na temat podstawowych ustawień zasugerowanych przez twórców FGN.

Nazwa hiperparametru	Wartość
Liczba klatek wejściowych	64
Rozmiar klatek wejściowych	224×224
Dataset	RWF-2000
Liczba epoch	50
Batch size	8
Callbacks	Zapis do logów co epokę
	Zapis modelu co epokę
	Optymalizacja współczynnika uczenia co epokę
Optymalizator	Adam
Funkcja kosztu	Categorical crossentropy
Metryki	Accuracy

Tabela 6.6 Hiperparametry w pierwszym eksperymencie sieci FGN

Czas uczenia

Pierwsza próba uczenia nie powiodła się ze względu na zbyt małą moc obliczeniową na maszynie lokalnej. Dlatego po uzyskaniu dostępu do mocniejszego sprzętu, drugie uczenie zajęło niecałe 65 godzin na maszynie CloudLab02 (pkt 6.5.1).

Wyniki uczenia



Rysunek 6.19 Wykres przedstawiający przebieg uczenia sieci FGN w pierwszym eksperymencie

Jak można zauważyć na rysunku powyżej, sieć osiągnęła przewidywane wyniki około 87% parametru accuracy przy dwudziestej epoce. Później możemy zauważyć fluktuację wartości parametru val_loss, która informuje nas o przeuczeniu sieci. Czas predykcji wynosił około 3.5 sekundy.

6.5.2.2.2. Eksperyment 2

W związku z długim procesem uczenia sieci FGN na oryginalnych hiperparametrach oraz długim czasem predykcji, zdecydowaliśmy się przeprowadzić kolejny eksperyment przy innych hiperparametrach, w celu optymalizacji czasu predykcji modelu.

Nazwa hiperparametru	Wartość
Liczba klatek wejściowych	32
Rozmiar klatek wejściowych	96×96
Dataset	RWF-2000
Liczba epoch	75
Batch size	16
Callbacks	Zapis do logów co epokę
	Zapis modelu co epokę
	Optymalizacja współczynnika uczenia co epokę
Optymalizator	Adam
Funkcja kosztu	Categorical crossentropy
Metryki	Accuracy

Tabela 6.7 Hiperparametry w drugim eksperymencie sieci FGN

Czas uczenia

Uczenie zajęło niecałe 49 godzin na maszynie CloudLab02 (pkt 6.5.1), czyli mniej niż w eksperymencie pierwszym.

Wyniki uczenia



Rysunek 6.20 Wykres przedstawiający przebieg uczenia sieci FGN w drugim eksperymencie

Na rysunku powyżej można zauważyć spadek wartości parametru *accuracy* względem pierwszego eksperymentu. Można też zauważyć, że dochodzi do przeuczenia sieci po trzydziestej epoce, podobnie jak w eksperymencie pierwszym. Może być to spowodowane doбором hiperparameterów, dlatego zdecydowaliśmy się więc na kolejny eksperyment.

6.5.2.2.3. Eksperyment 3

W związku z niepowodzeniem poprzedniego eksperymentu postanowiliśmy pozmienić niektóre hiperparametry, w tym: zredukować rozmiar wejściowy klatek, zmienić optymalizator, zwiększyć liczbę klatek.

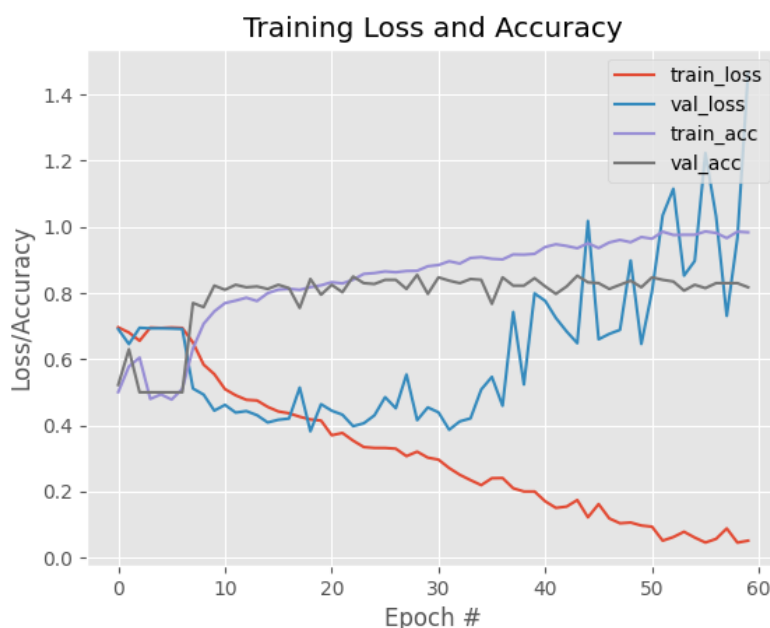
Nazwa hiperparametru	Wartość
Liczba klatek wejściowych	64
Rozmiar klatek wejściowych	64×64
Dataset	RWF-2000
Liczba epoch	60
Batch size	8
Callbacks	Zapis do logów co epokę
	Zapis modelu co epokę
	Optymalizacja współczynnika uczenia co epokę
Optymalizator	SGD
Funkcja kosztu	Categorical crossentropy
Metryki	Accuracy

Tabela 6.8 Hiperparametry w trzecim eksperymencie sieci FGN

Czas uczenia

Uczenie zajęło niecałe 34 godziny na maszynie CloudLab02 (pkt 6.5.1), czyli znacznie mniej niż w eksperymencie drugim.

Wyniki uczenia



Rysunek 6.21 Wykres przedstawiający przebieg uczenia sieci FGN w trzecim eksperymencie

Na rysunku powyżej można zauważyć przeuczenie po trzydziestej epoce, natomiast zdaje się, że zmiana hiperparametrów w tym eksperymencie poczyniła duże zmiany. Wartość parametru accuracy na stan trzydziestej pierwszej epoki wynosi 89%, czyli zbliżyliśmy się do wyników z pierwszego eksperymentu (pkt 6.5.2.2.1). Na tym etapie czas predykcji wynosił około 400 ms.

6.5.2.3. Podsumowanie

Biorąc pod uwagę wyżej opisane eksperymenty oraz ich wyniki, można zauważyć, że eksperyment 1 miał wartości współczynnika accuracy dokładnie takie same jak osiągnęli twórcy tej sieci. Natomiast był to też model najmniej zoptymalizowany pod względem czasowym. Bardzo dobre wyniki (tj. 89% parametru accuracy) posiadał eksperyment 3, który okazał się być dokładniejszy niż twórców sieci. To właśnie na nim postanowiliśmy zaprzestać kolejne eksperymenty, ze względu na: czas predykcji (400 ms) oraz dokładność (około 89 poprawnie rozpoznanych przypadków przemocy na 100). Eksperyment drugi zakończył się niepowodzeniem prawdopodobnie ze względu na zły dobór parametrów.

6.5.3. Wyuczanie modułu VRN

Zważywszy na wcześniej sformułowaną architekturę sieci (pkt 6.4.2.1), proces nauczania modelu VRN można podzielić na 2 etapy. Pierwszym etapem jest wyuczenie sieci w architekturze VGG16 używając zbioru danych RWF-2000 Image Subset (pkt 6.3.1.2.1) i dobraniu odpowiednich hiperparametrów. Drugim krokiem natomiast jest załadowanie wyuczonego już modelu VGG16 i przy użyciu oryginalnego zbioru RWF-2000 (pkt 6.3.1.2) generacji predykcji na seriach klatek wideo. Wyniki tych predykcji przechodzą przez proces kilku transformacji po czym trafiają do warstwy LSTM [26], gdzie sieć analizuje dane na przestrzeni czasu. Na końcu mamy kilka warstw zmniejszających ilość neuronów wyjściowych doprowadzając do formatu klasyfikacji binarnej. Oba opisane kroki zostały poddane serii różnych eksperymentów, które miały na celu uzyskanie jak największej dokładności finalnej predykcji.

6.5.3.1. Założenia modelu

W celu akceptacji powyższego modułu ustaliliśmy następujące wymagania:

- Wyuczony model sieci powinien na danym zbiorze klatek wideo poprawnie wykrywać przemoc lub jej brak w 8 na 10 przypadków
- Z wstępnych założeń wiemy, że system w całości powinien działać w czasie bliskim rzeczywistemu, dlatego też proces wykonywania predykcji na minimalnie dwóch następujących klatkach wideo powinien zajmować poniżej 1 sekundy

6.5.3.2. Eksperymenty

Poniżej opisane są eksperymenty, które na celu miały stworzyć gotowy moduł VRN spełniający wymagania opisane powyżej. Eksperymenty możemy podzielić na 2 rodzaje: wyuczanie bazowej sieci VGG16 oraz całej sieci już z warstwą LSTM.

6.5.3.2.1. Eksperymenty VGG16

Eksperymenty związane z wyuczaniem sieci na bazie architektury VGG16.

6.5.3.2.1.1. Eksperyment 1

Pierwszy eksperyment polegał na wyuczeniu sieci VGG16 od zera wykorzystując zbiór danych RWF-2000 Image Subset opisany w punkcie 6.3.1.2.1.

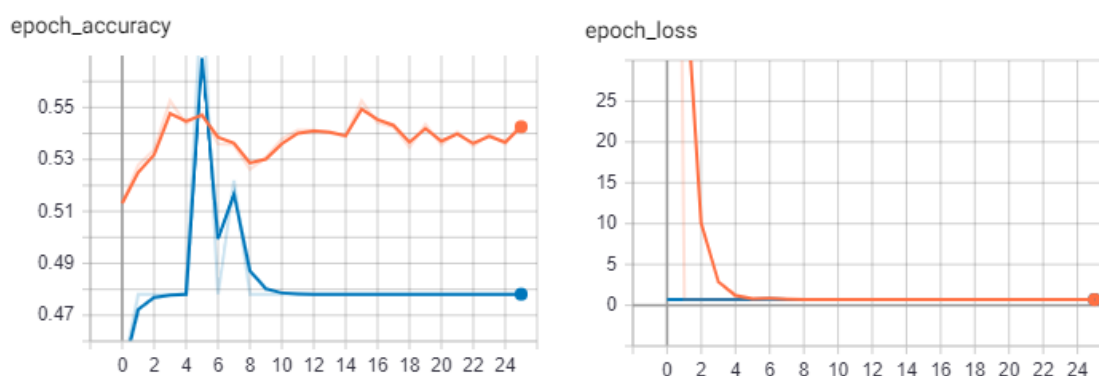
Nazwa hiperparametru	Wartość
Rozmiar klatek wejściowych	224×224
Dataset	RWF-2000 Image Subset
Liczba epoch	100
Liczba kroków na każdą epokę	100
Callbacks	Zapis do logów co epokę
	Zapis modelu co epokę
	Wcześniejsze zatrzymanie nauczania na bazie celności walidacyjnej z parametrem cierpliwości o wartości 20
Optymalizator	Adam
Funkcja kosztu	Categorical crossentropy
Metryki	Accuracy

Tabela 6.9 Tabela przedstawiająca dobór hiperparametrów do pierwszego eksperymentu VGG16

Czas uczenia

Uczenie na maszynie CloudLab2 (pkt 6.5.1) zajęło 4 godziny. Z racji wykorzystywania callbacku Early Stop proces zatrzymał się na 25 epoce.

Wyniki uczenia



Rysunek 6.22 Wykresy przedstawiające proces nauczania sieci VGG16 podczas pierwszego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji.

Jak można zauważyć na powyższym rysunku, względnie najlepszy wynik został uzyskany na 5 epoce, gdzie celność na zbiorze walidacyjnym osiągnęła wartość 57%. Na każdej kolejnej epoce widać znaczny spadek pod względem tego parametru. Ten sam eksperyment był kilkakrotnie powtarzany przy drobnych zmianach hiperparametrów, natomiast ten uzyskał najlepsze wyniki uzyskany przy użyciu RWF-2000 Image Subset.

6.5.3.2.1.2. Eksperyment 2

Następnym eksperymentem na sieci VGG16 było użycie innej wersji datasetu RWF-2000 – RWF-2000 OpenCV Subset opisanego w punkcie 6.3.1.2.2. Jest to pierwszy raz, kiedy w procesie nauczania użyliśmy technologii Optical Flow.

Nazwa hiperparametru	Wartość
Rozmiar klatek wejściowych	224×224
Dataset	RWF-2000 OpenCV Subset
Liczba epoch	100
Liczba kroków na każdą epokę	100
Callbacks	Zapis do logów co epokę
	Zapis modelu co epokę
	Wcześniejsze zatrzymanie nauczania na bazie celności walidacyjnej z parametrem cierpliwości o wartości 30
Optymalizator	Adam
Funkcja kosztu	Categorical crossentropy
Metryki	Accuracy

Tabela 6.10 Tabela przedstawiająca dobór hiperparametrów do drugiego eksperymentu VGG16

Czas uczenia

Proces nauczania na maszynie CloudLab2 (pkt 6.5.1) przed terminacją spowodowaną callbackiem typu Early Stop trwał 3 i pół godziny.

Wyniki uczenia



Rysunek 6.23 Wykresy przedstawiające proces nauczania sieci VGG16 podczas drugiego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji.

Spoglądając na powyższe wykresy możemy zauważyć, że sieć miała problem ze znalezieniem cech charakterystycznych podczas całego procesu nauczania. Parametry celności nie przekroczyły wartości 52%. Dany eksperyment został powtórzony kilkakrotnie zmieniając pojedyncze hiperparametry, natomiast nie przyniosło to lepszych rezultatów. Również pierwszy raz została podjęta próba wykorzystania metody finetunningu z użyciem wytrenowanej już wcześniej sieci VGG16 z wagami ImageNet [25], lecz to również nie przyniosło lepszych wyników.

6.5.3.2.1.3. Eksperyment 3

Ostatnim eksperymentem wykonanym na sieci VGG16 było użycie zbioru danych obrazów RWF-2000 RAFT Subset opisanego w punkcie 6.3.1.2.3.

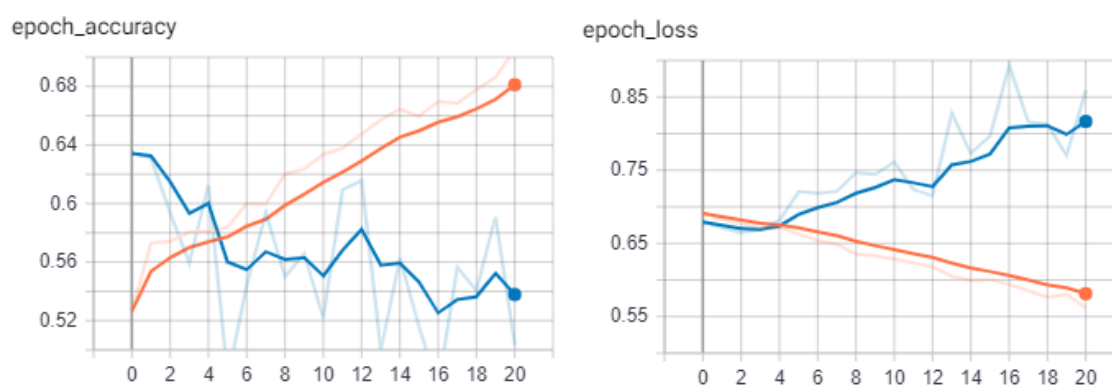
Nazwa hiperparametru	Wartość
Rozmiar klatek wejściowych	224×224
Dataset	RWF-2000 RAFT Subset
Liczba epoch	100
Liczba kroków na każdą epokę	100
Callbacks	Zapis do logów co epokę
	Zapis modelu co epokę
	Wcześniejsze zatrzymanie nauczania na bazie celności walidacyjnej z parametrem cierpliwości o wartości 50
Optymalizator	SGD
Funkcja kosztu	Binary crossentropy
Metryki	Accuracy

Tabela 6.11 Tabela przedstawiająca dobór hiperparametrów do trzeciego eksperymentu VGG16

Czas uczenia

Cały proces przy użyciu maszyny CloudLab2 (pkt 6.5.1) do czasu jego zatrzymania przez funkcję callback Early Stop zajął 3 godziny.

Wyniki uczenia



Rysunek 6.24 Wykresy przedstawiające proces nauczania sieci VGG16 podczas trzeciego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji.

Analizując powyższe wykresy możemy zauważyć wyniki nieco lepsze od tych, które zostały uzyskane w pierwszym eksperymencie. Najbardziej interesującymi

wynikami są te uzyskane podczas 2, 4 i 12 epoki, gdzie celność zarówno walidacyjna jak i treningowa łąpie się w przedziale 55%-60%. Spoglądając na wartości wykresów po 12 epoce możemy zauważyć zjawisko przeuczenia sieci. Ten eksperyment również został powtórzony kilkakrotnie natomiast zestawienie hiperparametrów w tabeli powyżej przyniosło najlepsze wyniki.

6.5.3.2.2. Eksperymenty VGG16 + LSTM

Eksperymenty związane z wyuczaniem sieci na bazie architektury opisanej w pkt 6.4.2.1.

6.5.3.2.2.1. Eksperyment 1

Pierwszy eksperyment wykonany w ramach wyuczenia całego modułu został wykonany przy użyciu oryginalnego zbioru RWF-2000 (pkt 6.3.1.2).

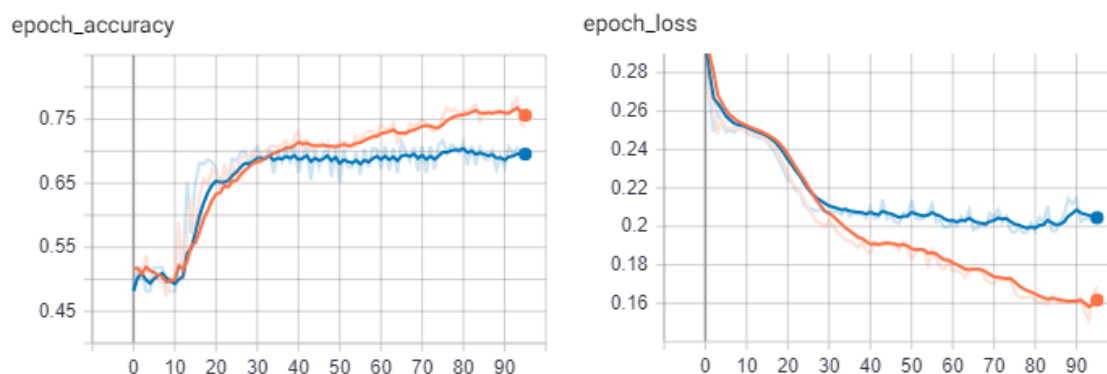
Nazwa hiperparametru	Wartość
Wagi VGG16	Wynik pierwszego eksperymentu z punktu 6.5.3.2.1.1
Wymiary wejściowe LSTM	20, 2
Rozmiar klatek wejściowych	224×224
Dataset	RWF-2000
Liczba epoch	1000
Batch size	500
Callbacks	Zapis do logów co epokę
	Zapis modelu co epokę
	Wcześniejsze zatrzymanie nauczania na bazie celności walidacyjnej z parametrem cierpliwości o wartości 30
Optymalizator	Adam
Funkcja kosztu	Binary crossentropy
Metryki	Accuracy

Tabela 6.12 Tabela przedstawiająca dobór hiperparametrów do pierwszego eksperymentu VGG16 + LSTM

Czas uczenia

Cały proces wyuczenia dla tego eksperymentu przy użyciu maszyny CloudLab2 (pkt 6.5.1) wyniósł około 9 godzin, po których funkcja Early Stop zakończyła proces.

Wyniki uczenia



Rysunek 6.25 Wykres przedstawiający proces nauczania sieci VGG16 + LSTM podczas pierwszego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji.

Jak możemy zauważyć na powyższych wykresach - wyniki wyglądają obiecująco. Po 10 epoce widać tendencje wzrostową w kontekście celności za równo na danych używanych do trenowania jak i walidacji oraz tendencje zniżkową dla funkcji kosztu też dla obu typów. Najlepszy wynik pod względem celności na zbiorze walidacyjnym został uzyskany na 76 epoce, gdzie uzyskał 71%. Czas potrzebny do wykonania predykcji na minimalnym zestawie klatek (2) w tym przypadku wynosił około 450ms. Ów eksperyment został także powtórzony wielokrotnie z drobnymi zmianami w hiperparametrach, lecz powyższe zestawienie przyniosło najlepsze wyniki.

6.5.3.2.2. Eksperyment 2

Następnym eksperymentem było użycie datasetu RWF-2000, gdzie dodatkowo klatki były przekonwertowywane na Optical Flow metodą z biblioteki OpenCV (tak samo jak przy transformacji datasetu RWF-2000 OpenCV Subset opisanego w punkcie 6.3.1.2.2) oraz wag VGG16 uzyskanych w drugim eksperymencie opisanym w punkcie 6.5.3.2.1.2.

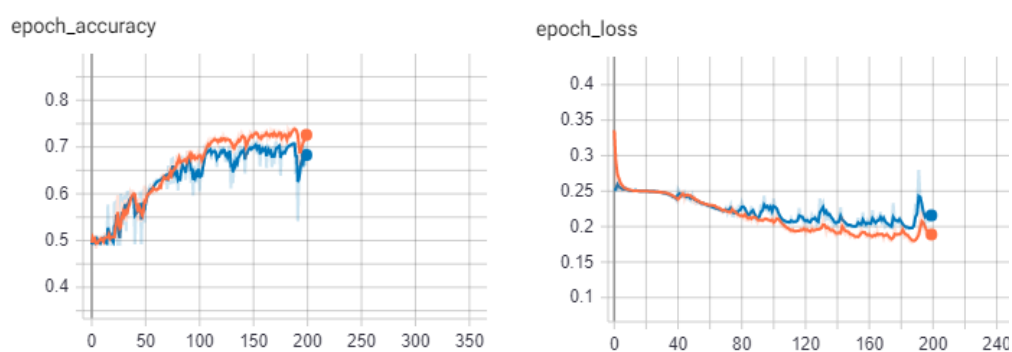
Nazwa hiperparametru	Wartość
Wagi VGG16	Wynik eksperymentu drugiego z punktu 6.5.3.2.1.2.
Wymiary wejściowe LSTM	20, 2
Rozmiar klatek wejściowych	224×224
Dataset	RWF-2000 z klatkami przekonwertowanymi na Optical Flow używając biblioteki OpenCV
Liczba epoch	1000
Batch size	500
Callbacks	Zapis do logów co epokę
	Zapis modelu co epokę
	Wcześniejsze zatrzymanie nauczania na bazie celności walidacyjnej z parametrem cierpliwości o wartości 50
Optymalizator	Adam
Funkcja kosztu	Binary crossentropy
Metryki	Accuracy

Tabela 6.13 Tabela przedstawiająca dobór hiperparametrów do drugiego eksperymentu VGG16 + LSTM

Czas uczenia

Biorąc pod uwagę dodatkowy proces przetwarzania klatek na Optical Flow cały proces zajął około 12 godzin. Funkcja callback Early Stop zatrzymała uczenie na 200 epoche.

Wyniki uczenia



Rysunek 6.26 Wykresy przedstawiające proces nauczania sieci VGG16 + LSTM podczas drugiego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji.

Na powyższych wykresach również możemy zauważyć całkiem dobre wyniki. Parametry celności predykcji wykazują trend zwyżkowy, a wartości zwracane z funkcji kosztu prezentują nurt zniżkowy. Niestety nawet najlepsze wyniki w tym eksperymencie nie przewyższyły tych uzyskanych w eksperymencie pierwszym. Czas potrzebny do wykonania predykcji na minimalnym zestawie klatek (2) w tym przypadku wynosił około 750ms. Ponowne próby wykonania tego eksperymentu z innymi kombinacjami hiperparametrów też nie pomogły w poprawie wyników.

6.5.3.2.2.3. Eksperyment 3

Ostatni eksperyment jaki został wykonany w kontekście całości modułu oparliśmy o zbiór danych RWF-2000 dodając proces konwersji klatek na Optical Flow przy użyciu metody RAFT (tej samej która została opisana w punkcie 6.3.1.2.3) oraz wag sieci VGG16 uzyskanych dzięki eksperymentowi opisanemu w punkcie 6.5.3.2.1.3.

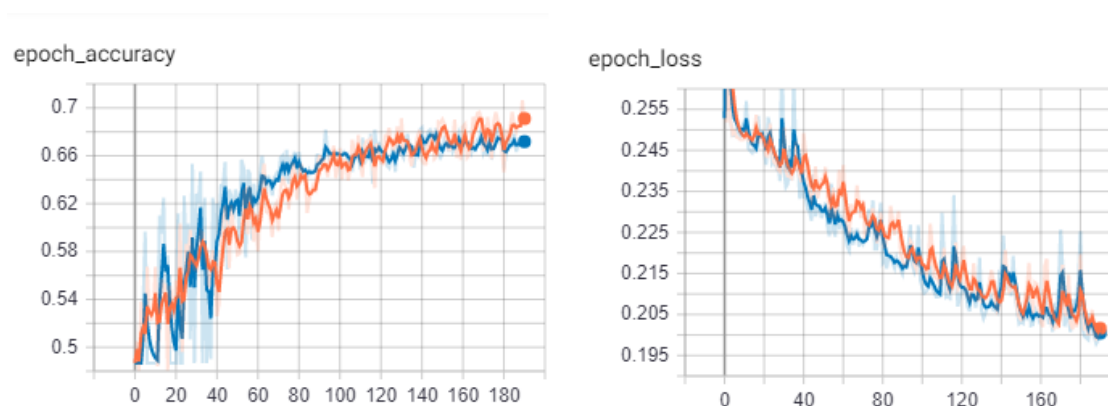
Nazwa hiperparametru	Wartość
Wagi VGG16	Wynik Eksperymentu 3 z punktu 6.5.3.2.1.3
Wymiary wejściowe LSTM	20, 2
Rozmiar klatek wejściowych	224×224
Dataset	RWF-2000 z klatkami przekonwertowanymi na Optical Flow używając metody RAFT
Liczba epoch	1000
Batch size	500
Callbacks	Zapis do logów co epokę
	Zapis modelu co epokę
	Wcześniejsze zatrzymanie nauczania na bazie celności walidacyjnej z parametrem cierpliwości o wartości 50
Optymalizator	Adam
Funkcja kosztu	Binary crossentropy
Metryki	accuracy

Tabela 6.14 Tabela przedstawiająca dobór hiperparametrów do trzeciego eksperymentu VGG16 + LSTM

Czas uczenia

Biorąc pod uwagę dodatkowy proces przetwarzania klatek na Optical Flow cały proces zajął około 16 godzin. Funkcja callback Early Stop zatrzymała uczenie na 190 epoce.

Wyniki nauczania



Rysunek 6.27 Wykresy przedstawiające proces nauczania sieci VGG16 + LSTM podczas trzeciego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji

Spoglądając na powyższe wykresy możemy również zauważyć obiecujące wyniki podobne do poprzednich eksperymentów. Niestety również ten eksperyment nie przyniósł oczekiwanych wyników, w najlepszym wypadku parametr celności na zbiorze walidacyjnym uzyskał 67%, a na zbiorze używanym do trenowania 69%. Czas potrzebny do wykonania predykcji na minimalnym zestawie klatek (2) w tym przypadku wynosił 900ms. Żadna inna próba uruchomienia tego eksperymentu z użyciem innych hiperparametrów nie uzyskała lepszych wyników.

6.5.3.3. Podsumowanie

Biorąc pod uwagę opisane powyżej eksperymenty, najbliższej wcześniej założonych wymagań był produkt eksperymentu opisanego w punkcie 6.5.3.2.2.1. Ze względu na jego najlepsze wyniki pod względem celności wydawanych predykcji oraz braku potrzeby wykonywania większej ilości transformacji pochłaniających dodatkowy czas w procesie generowania predykcji (używanie techniki Optical Flow), jest on najlepszym kandydatem do integracji z finalną wersją systemu. Niestety poza wymienionymi pozytywami nie spełnia on tak jak pozostałe wyniki eksperymentów, głównego wymagania celności na poziomie przynajmniej 80%.

6.5.4. Wyuczanie modułu DIDN

Zebrany wcześniej zbiór zdjęć posłuży do wyuczenia modelu za pomocą algorytmu YOLOv3 [29]. Nauczanie rozpoczęliśmy od przygotowania wymagań, które nasz model powinien spełniać.

6.5.4.1. Założenia modelu

Każdy wyuczony model w celu akceptacji powinien spełniać serię wcześniej ustalonych wymagań:

- Wykryje on poprawnie broń w przynajmniej 95% przypadkach tj. na 100 zdjęć testowych wykryje broń na przynajmniej 95 zdjęciach
- Będzie możliwa jednoczesna detekcja wielu przedmiotów
- Proces detekcji pojedynczego zdjęcia zajmie mniej niż 200ms

6.5.4.2. Eksperymenty

6.5.4.2.1. Eksperyment 1

Szczęśliwie wyniki już pierwszej iteracji spełniał wszystkie wcześniej stawiane założenia, z wyjątkiem ostatniego. Czas potrzebny na przetworzenie wynosił aż 300ms co było wartością znacząco przekraczającą postawione wcześniej wymaganie.

6.5.4.2.2. Eksperyment 2

Podczas kolejnej iteracji postanowiliśmy zmniejszyć wielkość przetwarzanego zdjęcia co pozwoli nam na przyspieszenie procesu wykrywania. Nasze założenie było słuszne i wynik tej iteracji pozwolił nam na uzyskanie gotowego modułu.

6.5.4.3. Translacja

Kolejnym podjętym przez nas krokiem było przekonwertowanie wyuczonego modelu, na model sieci wykorzystywany przez framework TensorFlow. Zdecydowaliśmy się na ujednolicenie wykorzystywanych technologii w celu łatwiejszej integracji. Konwersja odbyła się za pomocą skryptu napisanego przez Việt Hùng posiadającego licencję MIT [33].

6.5.4.4. Podsumowanie

Przekonwertowana wersja modelu powstałego podczas drugiej iteracji została wybrana do dalszego użycia w projekcie. Oferowała ona akceptowalny czas predykcji oraz bardzo wysoką trafność podczas detekcji. Dodatkowym plusem było użycie narzędzia TensorFlow co pozwoliło ujednolicić wykorzystywane technologie w projekcie.

6.5.5. Wyuczanie modułu DSDN

Zebrany wcześniej zbiór plików audio (pkt 6.3.3.3) posłuży nam do wytrenowania wcześniej zaimplementowanej sieci DSDN (pkt 6.4.4.1). Przebieg uczenia zaczęliśmy od ustawienia wstępnych hiperparametrów a następnie puszczania serii „eksperymentów” po których dokonywaliśmy dokładnej analizy wyników.

6.5.5.1. Założenia modelu

Założyliśmy, że satysfakcjonującym nas wynikiem będzie dokładność modelu na poziomie co najmniej 70% (tj. poprawne rozpoznanie przemocy w 70 przypadkach na 100). Relatywnie niskie założenie odnośnie do dokładności modelu wiąże się z małym znaczeniem tego modelu w ogólnej infrastrukturze systemu.

6.5.5.2. Eksperymenty

6.5.5.2.1. Eksperyment 1

Przy pierwszym eksperymencie hiperparametry dostosowaliśmy w sposób, który uważaliśmy za najbardziej trafny. Jest to jedynie wstępny eksperyment, na którym można oprzeć dalsze rozwiązania.

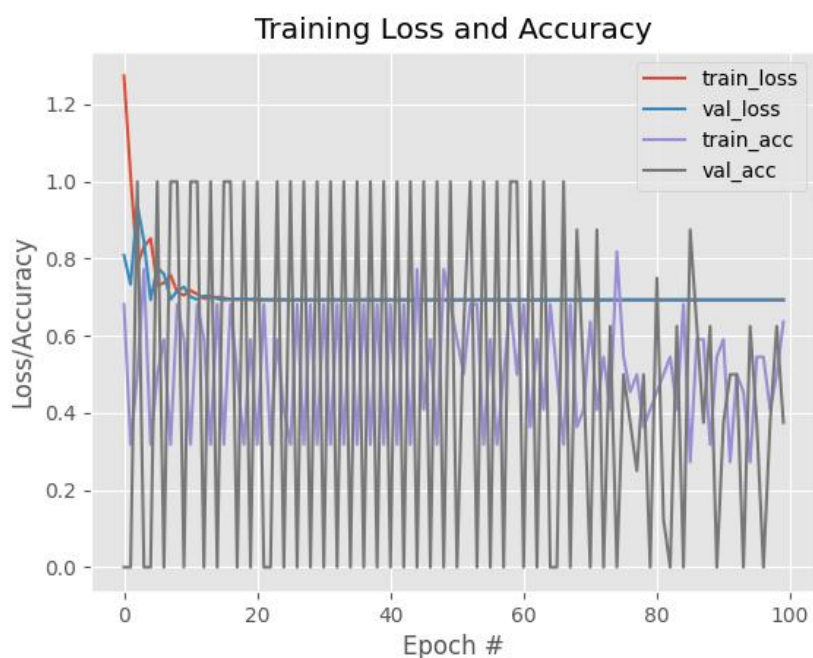
Nazwa hiperparametru	Wartość
Rozmiar klatek wejściowych	224×224
Dataset	AudioSet
Liczba epoch	10
Batch size	10
Optymalizator	Adam
Funkcja kosztu	Binary crossentropy
Metryki	accuracy

Tabela 6.15 Hiperparametry w pierwszym eksperymencie sieci DSDN

Czas uczenia

Uczenie zajęło niecałe 20 minut na lokalnej maszynie lokalnej z kartą graficzną NVIDIA GeForce GTX 970.

Wyniki uczenia



Rysunek 6.28 Wykres przedstawiający przebieg uczenia sieci DSDN w pierwszym eksperymencie

Jak można zauważyć na rysunku powyżej sieć nie była w stanie się nauczyć. Możemy również zauważyć ogromną fluktuację wartości parametru train_acc oraz

val_acc, co informuje nas niedouczeniu sieci. Pierwszy eksperyment można zaliczyć jako niepowodzenie.

6.5.5.3. Podsumowanie

Niestety pierwszy (i ostatni) eksperyment nie dostarczył zakładanych przez nas wyników. Dodatkowo nie udało nam się też wygenerować więcej eksperymentów ze względów czasowych (dokładniej opisane w punkcie 7.4.3.4).

6.6. API

Zgodnie z ideą modularności oraz dbałością o skalowalność i prostotę w obsłudze postanowiliśmy osadzić każdy model w osobnym kontenerze, gdzie udostępniany będzie przez protokół HTTP, dzięki autorskiemu API zbudowanym w oparciu o FastAPI. Autorskie rozwiązanie bardzo dobrze sprawdzało się w swojej roli do momentu obsługi wyuczonego modelu. Pojawił się szereg problemów, które musieliśmy rozwiązać.

6.6.1. Napotkane problemy

6.6.1.1. Inicjalizacja modelu

Pierwszym problemem okazał się długi czas ładowania modelu przed każdą predykcją. W przypadku modelu FGN średni czas inicjalizacji potrafił zajmować 2.5 sekundy, co wyraźnie spowalniało cały proces analizy. Za każdym razem przed analizą serii klatek była wymagana procedura ładowania modelu z dysku. Pojedyncze zapytanie o analizę trwało w sumie około 3.5 sekund. Praktycznie uniemożliwiało to detekcję przemocy w czasie bliskim rzeczywistego.

Rozwiązaniem problemu okazało się ładowanie modelu na samym początku inicjalizacji API. Dzięki temu uniknęliśmy potrzeby ładowania modelu za każdym razem z dysku co bardzo pozytywnie wpłynęło na czas trwania pojedynczej analizy. W ostatecznym rozrachunku udało się skrócić cały proces analizy pojedynczej serii klatek do jednej sekundy

6.6.1.2. Serializacja danych potrzebnych do analizy

Kolejną szansą na zoptymalizowanie czasu potrzebnego na analizę serii obrazu okazało się zrezygnowanie z protokołu HTTP i serializacji danych z wykorzystaniem tradycyjnych metod, tj. JSON i XML. Proces przetwarzania zestawu klatek potrzebnych

to analizy w sieci FGN trwał około 650 milisekund, gdzie analiza trwała tylko 350 milisekund. W przypadku protokołu HTTP i procesowanie danych do obsługiwanych przez API formatów nie udało nam się osiągnąć krótszego czasu.

6.6.2. Podsumowanie

W ostatecznym rozrachunku analiza pojedynczego zestawu klatek na modelu serwowanym przez API wykorzystujące protokół HTTP oraz serializację danych do formatu JSON trwała około sekundy, gdzie 350 milisekund trwała analiza, a 650 milisekund przetwarzanie danych. Ze względu na tę dysproporcję kontynuowaliśmy poszukiwania szybszego sposobu na przygotowywanie zapytania i transfer danych do analizy. Z pomocą przyszedł framework gRPC oraz wykorzystywany przez niego protokół Protobuf, serializujący dane do formy binarnej (opisane głębiej w punkcie 6.7). Dzięki tej zmianie proces tworzenia zapytania z serią klatek diametralnie został skrócony. Implementując to rozwiązanie musieliśmy zrezygnować z autorskiego rozwiązania opartego o API wykorzystujące protokół HTTP. W tym momencie zdecydowaliśmy się zamienić FastAPI na dedykowane rozwiązanie do udostępniania wytrenowanych modeli, TensorFlow Serving, którego implementację opisuje w dalszej części pracy.

6.7. TensorFlow Serving

Elastyczny i wysoce wydajny system do obsługi modeli uczenia maszynowego, zaprojektowany z myślą o środowiskach produkcyjnych. TensorFlow Serving umożliwia proste w obsłudze ładowanie wielu modeli, zachowując jednocześnie tę samą architekturę serwera oraz API. Pozwala na indywidualną konfigurację załadowanego modelu i umożliwia przetwarzanie wsadowe wraz z rozgrzewaniem modelu.

Zalety:

- Prostota w obsłudze i wdrożeniu
- Wsparcie dla transferu danych z wykorzystaniem gRPC i REST API
- Możliwość wdrożenia na chmurze Google w skalowanym środowisku produkcyjnym
- Wbudowane wsparcie dla przetwarzania wsadowego i rozgrzewania modelu
- Wersjonowanie modeli

TensorFlow Serving wspiera transfer danych z wykorzystaniem technologii gRPC, która pozwala na lekką i szybką serializację danych w formacie binarnym, według protokołu Protobuf. W przeciwieństwie do formatu JSON, dane w formacie Protobuf nie są w jasny sposób czytelne dla człowieka. Protobuf wymaga wcześniej zdefiniowanych kontraktów oraz formatu danych, od początku transferu do jego zakończenia jest sztywno ustalony i jasno określony.

Pierwotnie do komunikacji z modelem osadzonym w TensorFlow Serving wykorzystywaliśmy powszechnie opisywane REST API. W tym momencie jeszcze nie znaliśmy gRPC jako technologii oraz dobroci z niej płynących. Tradycyjne REST API oparte o przetwarzanie danych w formacie JSON, tworzenie zapytania z serią klatek do analizy na modelu FGN (pkt 6.4.1) trwało około 650 milisekund. Razem z przejściem na protokół gRPC czas serializacji identycznego zestawu klatek spadł do około 10 milisekund. Diametralna różnica przekonała nas do wyższości nowej technologii. Protobuf spisuje się idealnie w sytuacjach, gdzie przesyłane są duże ładunki danych oraz nie zależy nam na giętkości formatu danych oferowanej przez JSON.

6.8. Porzucone pomysły

W związku z eksperymentalnym charakterem naszej pracy w trakcie procesu twórczego zachodził u nas szereg analiz związanych z powodzeniem lub niepowodzeniem danej metody. W tym celu ciągle poszukiwaliśmy coraz to nowych metod mogących poprawić wyniki uczenia. Natomiast nie wszystkie techniki udało nam się przetestować ze względu na wąski czas oraz skalę projektu. Dlatego niżej w krótki sposób przedstawimy omawiane pomysły i ich możliwy wpływ na system.

6.8.1. Segmentacja

Przy tworzeniu podzbiorów datasetu RWF-2000 takich jak opisano w punktach 6.3.1.2.2 i 6.3.1.2.3, pojawił się pomysł segmentacji (patrz [34]) zbioru w celu polepszenia wyników uczenia sieci. Segmentacja tworzy łatwiejszy do przetworzenia dla sieci obraz, który zaznacza bardziej interesujące nas obszary w prosty sposób. Dalsze omówienie tego tematu znajduje się w pkt 7.2.2.8.

6.8.2. Szkieletyzacja

Jedną z rozważanych opcji była szkieletyzacja [35] obrazu, czyli znana technika wizji komputerowej. Natomiast biorąc pod uwagę jakość filmów w naszym zbiorze i skale zaawansowania projektu, musieliśmy porzucić ten pomysł.

6.8.3. Modele 3D

Ciekawym pomysłem było również wykorzystanie współczesnej metody, patrz [36], do generowania modeli 3D osób z nagrań wideo, w celu oceny ruchu wektorowego i na podstawie tego określenia występowania przemocy. Niestety metoda ta jest niezwykle kosztowna pod względem czasowym i mocy obliczeniowej. Dodatkowo technika ta jest stosunkowo młoda i podatna na wiele błędów.

6.8.4. Usuwanie tła

Dodatkowo rozważaliśmy również pomysł usuwania tła z klatek w celu wyostrenia obszaru najbardziej nas interesującego – czyli ludzi. Pomysł pojawił się podczas czytania jednej z dokumentacji [37]. Udało nam się napisać krótki skrypt w tym celu. Natomiast chwilę po tym zaniechaliśmy dalszych prac w tym kierunku, ponieważ jak w przypadku poprzednich metod, brakowało nam czasu na wszystkie możliwe eksperymenty.



Rysunek 6.29 Wizualizacja porzuconej techniki usuwania tła na zdjęciu ze zbioru RWF-2000 Image Subset

7. Realizacja projektu

7.1. Przyrost zero - planowanie

W tym przyroście skupiliśmy się na przygotowaniu paru sesji wykorzystujących technikę burzy mózgów w celu przedyskutowania naszych pomysłów odnośnie do projektu. Następnym krokiem było przygotowanie wstępnego planu prac oraz ustalenie wymagań.

7.1.1. Założenia przyrostu

- Przedyskutowanie pomysłów dotyczących projektu
- Ustalenie wymagań dla projektu
- Wybór narzędzi oraz preferowanych technologii
- Przygotowanie wstępnego planu prac
- Podzielenie prac

7.1.2. Zadania

7.1.2.1. Przedyskutowanie pomysłów dotyczących projektu

Pierwszym zadaniem, którego się podjęliśmy było przedyskutowanie naszych pomysłów.

Po przedstawieniu wszystkich idei dotyczących tematu pracy inżynierskiej, uzgodniliśmy spójną wizję projektu. Wybraliśmy ten pomysł spośród wielu, gdyż spełniał on nasze wymagania odnośnie złożoności i dostrzegliśmy realny problem, który możemy pomóc rozwiązać.

7.1.2.2. Ustalenie wymagań dla projektu

Kolejnym postawionym przez nas zadaniem było określenie wymagań jakie nasz projekt powinien spełniać. Niezwykle pomocne w tym procesie były konsultacje z udziałem person wyspecjalizowanych w dziedzinie sztucznej inteligencji. Założenia te były wstępne i w kolejnych etapach ulegały zmianom, aczkolwiek pozwoliły nam przejść do następnego zadania.

7.1.2.3. Wybór narzędzi oraz preferowanych technologii

Posiadając wstępną wersję wymagań dla systemu byliśmy w stanie zapoznać się z technologiami umożliwiającymi spełnienie tych wymagań. Kolejne spotkanie pozwoliły nam na wybranie interesujących nas technologii.

7.1.2.4. Przygotowanie wstępnego planu prac

Przygotowanie planu prac było przełomowym momentem projektu. Wymagał on od nas podjęcia poważnych decyzji architektonicznych popartych dogłębną analizą zalet i wad każdego z rozwiązań [38].

7.1.2.5. Podzielenie prac

Stworzenie planu pracy umożliwiło nam podzielenie zadań. Zadania zostały podzielone w drodze kompromisu oraz w celu dopasowania się do kręgu zainteresowań danej osoby.

7.1.3. Napotkane problemy

7.1.3.1. Komunikacja

W początkowej fazie projektu byliśmy zmuszeni do wypracowania metod komunikacji pomiędzy członkami zespołu. Proces ten wymagał wielu prób i dopiero po jakimś czasie udało nam się wypracować model komunikacji odpowiedni dla każdego uczestnika projektu.

7.1.3.2. Konflikt charakteru

Problem ten był szczególnie dotkliwy w sytuacjach wymagających podjęcia decyzji. Wymagało to wielu kompromisów oraz wypracowania wspólnego zdania.

7.1.3.3. Organizacja

Każdy z członków zespołu posiadał bardzo napięty grafik co często utrudniało spotkania. Problem został rozwiązany dzięki ustaleniu stałej daty spotkań, co pozwoliło na wcześniejszą rezerwację czasu.

7.2. Przyrost pierwszy – dataset

W tym przyroście skupiliśmy się głównie na pozyskaniu odpowiednich zbiorów danych potrzebnych do nauki sieci neuronowych. Następnym krokiem było odpowiednie przygotowanie datasetu oraz udostępnienie go na systemie wersjonowania w celu posiadania spójnej wersji przez wszystkich członków zespołu.


7.2.1. Założenia przyrostu

- Pozyskanie datasetu z nagraniami wideo zawierającymi przemoc oraz jej brak
- Pozyskanie datasetu zawierającego zdjęcia broni
- Pozyskanie datasetu z plikami audio zawierającymi dźwięki przemocy oraz jej brak
- Weryfikacja zbiorów (jakość danych wideo/audio/zdjęć, ilość plików itd.)
- Podział na zbiór walidacyjny i treningowy
- Ekstrakcja klatek z plików wideo
- Transformacja klatek na Optical Flow
- Segmentacja klatek
- Ekstrakcja spektrogramów z plików audio
- Udostępnienie pozyskanych zbiorów reszcie zespołu

7.2.2. Zadania

7.2.2.1. Pozyskanie zbioru nagrań wideo

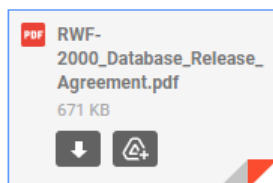
Jako zadanie przyjęliśmy pozyskanie zbioru danych wideo zawierających przemoc, jak i również te bez jakichkolwiek aktów przemocy jak opisano w punkcie 6.3.1. W tym celu sporządziliśmy wstępne wymagania, takie jak np. ilość plików wideo, ich jakość czy też dostępność (pkt 6.3.1.1). Następnie podjęliśmy się przeszukania sieci w celu znalezienia odpowiednich dla naszego zadania zbiorów oraz ich analizy, aby na podstawie wcześniej określonych wymagań pozyskać najlepszy dataset (analizę różnych zbiorów danych opisano w punkcie 6.3.1.3) Niektóre ze zbiorów wymagały od nas sformułowania pisemnej prośby o dostęp. W tym celu wydrukowaliśmy potrzebny dokument (miejsce dokumentu [39]), po uzupełnieniu odesłaliśmy skan do właściciela danych. Takim sposobem udało nam się uzyskać dostęp do RWF- 2000 Dataset.


 **Ola Piętka** <s17611@pjawst.edu.pl> 16 wrz 2020, 21:11
do ming.cheng ▾

Hello,
I am in a group of students that are working on Engineering Thesis and we are trying to create a Violence Recognition System. We would appreciate access to the RWF-2000 dataset very much.
The project is fully educational and will not be commercialized.

Required document is attached to the e-mail.

Best regards,
Students from Poland



 **Ming Cheng** <ming.cheng@dukekunshan.edu.cn> 17 wrz 2020, 03:04
do mnie ▾

angielski ▾ > polski ▾ Przetłumacz wiadomość Wyłącz dla następującego języka: angielski x

Hi,

Thanks for your application. Please download the RWF-2000 dataset from the links below.

Baidu Cloud:

link: [https://pan.baidu.com/s/1\[redacted\]](https://pan.baidu.com/s/1[redacted])

pin number: [redacted]

Google Drive:

[https://drive.google.com/drive/folders/\[redacted\]](https://drive.google.com/drive/folders/[redacted])

Box Drive:

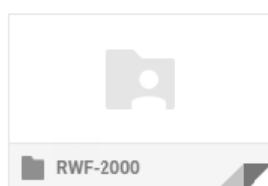
[https://duke.box.com/s/\[redacted\]](https://duke.box.com/s/[redacted])

If there is any problem, please feel free to contact me. Have a good day!

Best regards,

Ming

...



Rysunek 7.1 Korespondencja między właścicielem RWF-2000 Dataset a członkami zespołu

7.2.2.2. Pozyskanie zbioru zdjęć broni

Założeniem w tym przyroście było pozyskanie dużego zbioru zdjęć broni wraz z plikami zawierającymi wymagane dane opisane w podpunkcie 6.3.2.1.

Zebranie odpowiedniej ilości plików nie było wielkim problemem głównie dzięki ogromnym publicznym zbiorom takich zdjęć. Jeden z nich został opisany w podpunkcie 6.3.2.2.

7.2.2.3. Pozyskanie zbioru plików audio

W tym przyroście zadaniem było również pozyskanie zestawu plików audio zawierających dźwięki przemocy oraz ich brak. Dlatego wcześniej rozpisaliśmy wymagania dotyczące pozyskiwania datasetu (pkt 6.3.3.1) aby ułatwić i ustrukturyzować pracę nad poszukiwaniem zbioru.

Niestety pozyskaliśmy niewielką ilość plików dźwiękowych (160) zawierających zarówno dźwięki przemocy jak i ich brak (pkt 6.3.3.2). Dlatego w tym momencie podjęto decyzje o zmianie podejścia i zmienienie wcześniejszego założenia z dźwięków przemocy na dźwięki strzałów broni (pkt 6.3.3.3) jako dodatek do modułu rozpoznającego broń na obrazie (pkt 6.4.3). Udało nam się znaleźć świetne źródło, z którego można uzyskać wiele różnych zbiorów audio, jak opisano w punkcie 6.3.3.3.1. Dla przeciwwagi dźwięków broni pobrano również zbiór zawierający dźwięki krzyku (ok. 1,200), klaskania (790) oraz fajerwerk (ok. 3,000). Łączna ilość pobranych plików w tym zadaniu wynosiła ponad 4,000.

7.2.2.4. Weryfikacja zbiorów

Każdy pobrany zbiór danych musiał przejść weryfikację pod względem jakości oraz spójności z wcześniej postawionymi wymaganiami (wymagania opisane odpowiednio w punktach: 6.3.1.1, 6.3.2.1, 6.3.3.1).

7.2.2.5. Podział na zbiór walidacyjny i treningowy

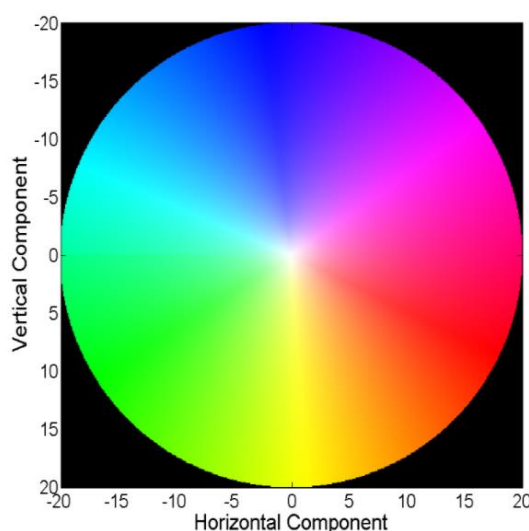
Przy datasetach które nie miały wcześniejszego podziału na zbiór walidacyjny i treningowy, przyjęliśmy ręczne jego podzielenie w stosunku 2-8 (tj. 20% zbiór walidacyjny i 80% zbiór treningowy).

7.2.2.6. Ekstrakcja klatek z plików wideo

RWF-2000 Dataset był nam potrzebny w formie zdjęć, dla późniejszego przyspieszenia uczenia oraz możliwej transformacji zbioru. W tym celu został sporządzony skrypt ułatwiający pracę nad wydobywaniem zdjęć i zapisaniem ich jako ostateczny zbiór o nazwie RWF-2000 Images Subset (pkt 6.3.1.2.1).

7.2.2.7. Transformacja klatek na Optical Flow

W celu wykorzystania technologii wizji komputerowej i potencjalnego zwiększenia wyników uczenia sieci, zdecydowaliśmy się na stworzenie zbioru opartego na metodzie Optical Flow. Metoda ta opiera się na detekcji ruszających się obiektów pomiędzy dwoma klatkami i oznaczeniu kolorem w zależności od kierunku ich poruszania się, oraz nasyceniem w zależności od ich wielkości.



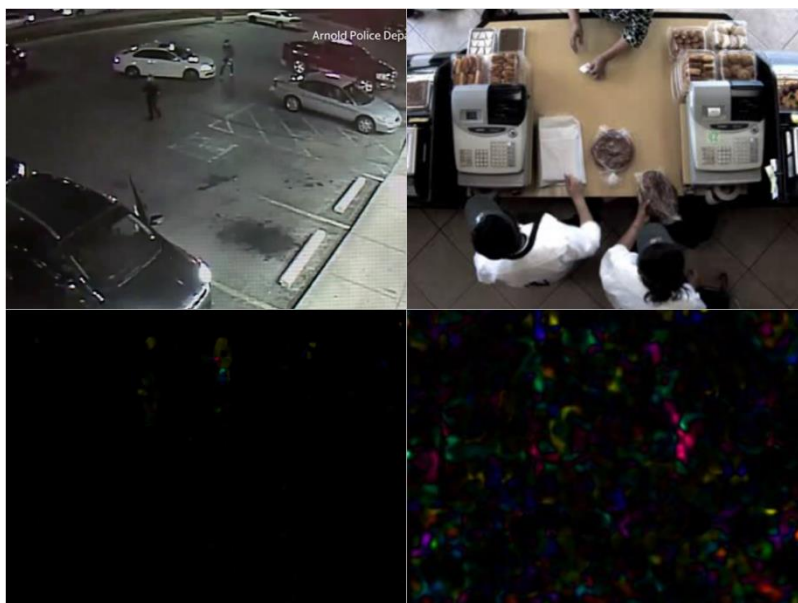
Rysunek 7.2 Legenda do wizualizacji optycznej pokazująca jak kierunek jest odwzorowywany na kolor, a wielkość na nasycenie [40]

Przyjęliśmy dwie metody umożliwiające dokonania takich transformacji i niżej opiszemy każdą z nich.

7.2.2.7.1. OpenCV

OpenCV jest to narzędzie umożliwiające między innymi transformacji dwóch klatek na Optical Flow. Jest ono bardzo szybkie (jedna transformacja na niecałe 100ms) ale niestety jest równocześnie niedokładne (Rysunek 7.3). Natomiast jest to metoda znana przy nauce wielu sieci neuronowych i potrafi znacznie poprawić wyniki. W tym celu

napisaliśmy skrypt, który transformuje RWF-2000 Dataset na zbiór o nazwie RWF-2000 OpenCV Subset (jak opisano w punkcie 6.3.1.2.2).



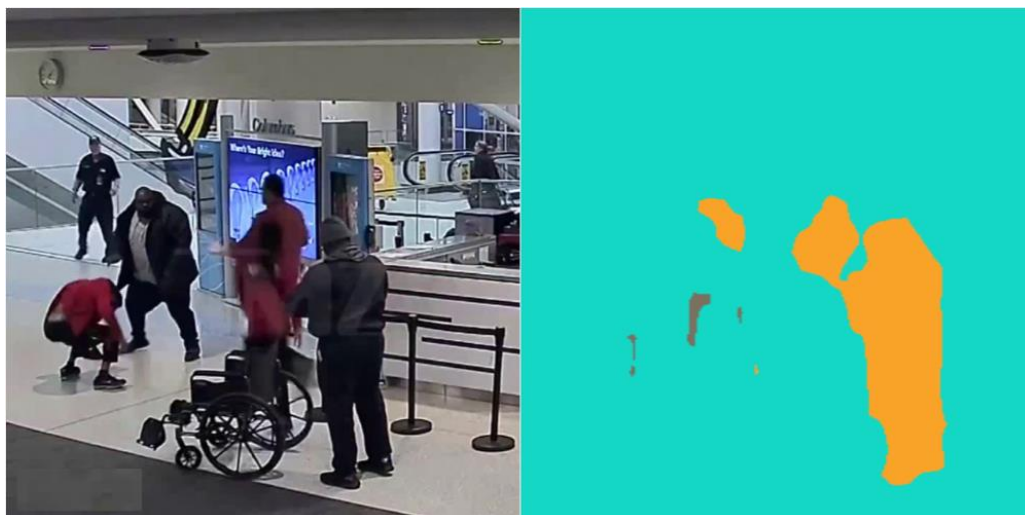
Rysunek 7.3 Przykłady niedokładnej transformacji klatek przy użyciu metody Optical Flow za pomocą technologii OpenCV

7.2.2.7.2. RAFT

RAFT jest aktualnie (na styczeń 2021r.) najbardziej dokładną metodą estymacji Optical Flow [15]. Model jest bardzo precyzyjny, ale niestety idzie to w parze z wydłużeniem czasu transformacji (około 2 sekundy na zestaw dwóch klatek). W celu wykorzystania tej metody, napisaliśmy skrypt dokonujący transformacji klatek z RWF- 2000 Dataset na jego podzbiór o nazwie RWF-2000 RAFT Subset (opisano w punkcie 6.3.1.2.3).

7.2.2.8. Segmentacja klatek

Jednym z pomysłów odnośnie poprawienia wyników klasyfikacji było stworzenie datasetu opartego o segmentację. Udało nam się napisać skrypt dokonujący segmentacji klatek ze zbioru RWF-2000 za pomocą modelu udostępnionego przez Keras [41], natomiast ze względów czasowych nie udało nam się przetestować tego rozwiązania i postanowiliśmy zaniechać pracy w tym kierunku.



Rysunek 7.4 Przykład segmentacji klatki

7.2.2.9. Ekstrakcja spektrogramów

Ponieważ moduł audio wymagał transformacji ścieżki dźwiękowej na obraz, wszystkie pobrane pliki wymagały od nas przepuszczenia przez skrypt ekstrahujący interesujące nas obszary, a następnie transformacje na spektrogramy (opisane w punkcie 6.3.3.3.2).

7.2.2.10. Udostępnienie pozyskanych zbiorów reszcie zespołu

W celu posiadania spójnych wersji datasetu przez wszystkich członków zespołu, został on finalnie udostępniony na systemie wersjonowania Syncthing (narzędzie opisane w punkcie 6.2.3.3).

7.2.3. Napotkane problemy

7.2.3.1. Czas

Pozyskanie datasetu z plikami wideo było bardzo czasochłonne i wymagało dużego przeglądu Internetu. Dodatkowo trzeba było kilkakrotnie wysyłać e-maile do właścicieli zbiorów w celu uzyskania dostępu, a to zajmowało dodatkowy cenny czas.

7.2.3.2. Pamięć

W przypadku datasetu zawierającego pliki wideo potrzeba było dużo wolnego miejsca na dysku. Niestety jedna z osób nie posiadała tyle by móc takowy zbiór posiadać lokalnie, dlatego konieczne było zakupienie nowego dysku pamięci.

7.2.3.3. Audio

Niestety znalezienie odpowiedniego zbioru audio przemocy okazało się nierealne ze względu na brak dostępności takowych danych. Wszelkie sprawdzone datasey nie zawierały odpowiedniej jakości/ilości plików potrzebnych do wyuczenia sieci. Dodatkowo okazało się również, że wcześniej pozyskane zbiory wideo nie zawierały ścieżek audio. Na tym etapie zdecydowano o zakończeniu pracy nad poszukiwaniem zbioru audio zawierającego przemoc i kontynuowano prace zmieniając założenia.

7.3. Przyrost drugi – implementacja modułów

W tym przyroście skupiliśmy się na implementacji modułów VRN, FGN, DIDN oraz DSDN.

7.3.1. Założenia przyrostu

- Implementacja modułu FGN
- Implementacja modułu VRN
- Implementacja modułu DIDN
- Implementacja modułu DSDN

7.3.2. Zadania

7.3.2.1. Implementacja modułu FGN

Opis zadania

W tym zadaniu, skupiono uwagę na implementacji architektury sieci stworzonej przez trzech Chińskich profesorów jak opisano w punkcie 6.4.1.1, do późniejszego jej wyuczenia.

Język i technologie

Do wykonania zadania wykorzystano język Python oraz następujące biblioteki:

- Keras
- TensorFlow
- NumPy [42]
- Matplotlib [43]
- OpenCV

7.3.2.2. Implementacja modułu VRN

Opis zadania

W tym zadaniu, skupiono uwagę na implementacji architektury sieci opisanej w punkcie 6.4.2.1.

Język i technologie

Do wykonania zadania wykorzystano język Python oraz następujące biblioteki:

- Keras
- TensorFlow
- NumPy
- Matplotlib
- OpenCV
- PyTorch [44]

7.3.2.3. Implementacja modułu DIDN

Opis zadania

W tym zadaniu, skupiono uwagę na zapoznaniu się z algorytmem YOLOv3 [29] oraz zdobyciu wszystkich informacji potrzebnych do późniejszego użycia tego algorytmu.

Język i technologie

Do wykonania zadania wykorzystano język Python, C++ oraz następujące biblioteki:

- TensorFlow
- NumPy
- OpenCV
- YOLOv3

7.3.2.4. Implementacja modułu DSDN

Opis zadania

W tym zadaniu skupiliśmy się na implementacji sieci konwolucyjnej opartej na wyciąganiu cech charakterystycznych z obrazów (spektrogramów), do późniejszego wyuczenia sieci rozpoznawania wystrzałów broni.

Język i technologie

Do wykonania zadania wykorzystano język Python oraz następujące biblioteki:

- Keras
- TensorFlow
- NumPy
- Matplotlib
- OpenCV
- Sklearn [45]
- Skimage [46]

7.3.3. Napotkane problemy

7.3.3.1. Wiedza

Nie każdy członek zespołu projektowego jest obeznany z językiem Python oraz potrzebnych bibliotekach. Dla niektórych z nas była to możliwość nauki nowych technologii i rozwiązań. Zdecydowanie jest to duży plus, natomiast takim sposobem straciliśmy część czasu, która mogłaby przydać się w późniejszych etapach projektu.

7.4. Przyrost trzeci – uczenie

W tym przyroście skupiliśmy uwagę na uczeniu zaimplementowanych wcześniej modułów (opis nauki modułów w punkcie 6.5).

7.4.1. Założenia przyrostu

- Wyuczenie modułu FGN z dokładnością co najmniej 85% (tj. poprawne rozpoznanie przemocy w 85 przypadkach na 100)
- Wyuczenie modułu VRN z dokładnością co najmniej 80% (tj. poprawne rozpoznanie przemocy w 80 przypadkach na 100)
- Wyuczenie modułu DIDN z dokładnością co najmniej 95% (tj. poprawne rozpoznanie broni w 95 przypadkach na 100)
- Wyuczenie modułu DSDN z dokładnością co najmniej 70% (tj. poprawne rozpoznanie wystrzału broni w 70 przypadkach na 100)

7.4.2. Zadania

7.4.2.1. Wyuczenie modułu FGN

W tym zadaniu skupiliśmy się na wyuczeniu sieci FGN, jak opisano w punkcie 6.5.2. Podeszliśmy do tego w sposób iteracyjny, nazywając każde nowe wyuczenie „eksperymentem”. W taki o to sposób wykonaliśmy trzy eksperymenty jak opisano w punkcie 6.5.2.2, z których ostatni został wybrany przez nas jako ostateczny model sieci FGN (z parametrem accuracy około 89%). Zakończyliśmy proces eksperymentów na tym etapie ze względu na osiągnięcie świetnych wyników w trzeciej iteracji uczenia. Takim sposobem udało nam się spełnić założone wymaganie co do dokładności modułu FGN.



Rysunek 7.5 Przykładowe predykcje końcowej wersji modułu FGN na filmach pozyskanych z serwisu YouTube

7.4.2.2. Wyuczenie modułu VRN

W tym zadaniu skupiliśmy się na wyuczeniu sieci VRN, jak opisano w punkcie 6.5.3. Zadanie te zostało rozwiązane w sposób iteracyjny i sformułowane w postaci eksperymentów opisanych w punkcie 6.5.3.2. Rezultatem tego zadania było 6

eksperymentów wykonanych na różne sposoby, które wyłoniły najlepsze rozwiązanie dla modułu VRN opisane w punkcie 6.5.3.3.



Rysunek 7.6 Obrazy z nagrań wideo przetworzone i sklasyfikowane przez finalną wersję modułu VRN. Nagrania pozyskane z serwisu LiveLeak i YouTube

7.4.2.3. Wyuczenie modułu DIDN

W tym zadaniu skupiliśmy się na wyuczeniu sieci DIDN, jak opisano w punkcie 6.5.4. Proces został podzielony na iteracje. W taki sposób została wybrana wersja będąca wynikiem drugiej iteracji. Następnym krokiem było przekonwertowanie modułu co zostało opisane w podpunkcie 6.5.4.3.

7.4.2.4. Wyuczenie modułu DSDN

W tym zadaniu chcieliśmy skupić naszą uwagę na wyuczeniu sieci DSDN opisanej w pkt 6.5.5. Niestety ze względów czasowych po wykonaniu jedynie jednego eksperymentu zdecydowaliśmy o porzuceniu tego modułu i skupieniu uwagi na bardziej znaczących sieciach.

7.4.3. Napotkane problemy

7.4.3.1. Czas

Uczenie poszczególnych modułów zajmowało sporo czasu, czasem nawet do 65 godzin (pkt 6.5.2.2.1). Tym sposobem traciliśmy cenny czas oczekując na wyniki.

7.4.3.2. Moc obliczeniowa

Do niektórych zadań moc obliczeniowa dostępna lokalnie przez zespół nie była wystarczająca. Dlatego potrzebowaliśmy zdobyć dostęp do zewnętrznej maszyny umożliwiającej szybsze i większe obliczenia (opisane w punkcie 6.5.1).

7.4.3.3. Hiperparametry

Ważnym aspektem każdej sieci było ustalenie odpowiednich hiperparametrów. Złe ich ustawienie skutkowało przeuczeniem bądź niedouczeniem sieci. W celu znalezienie odpowiednich wartości należało przeprowadzić serie „eksperymentów”.

7.4.3.4. Moduł DSDN

Moduł ten pochłaniał sporo czasu implementacyjnego, który mógł być wykorzystany na potrzebę innych ważniejszych modułów. Dlatego jak opisano w punkcie 7.4.3.4, zrezygnowaliśmy z używania tego modułu w finalnej wersji systemu.

7.4.3.5. Maszyna CloudLab2

W procesie przeprowadzania coraz to nowszych eksperymentów natrafiliśmy na problem związany z ograniczeniem pamięci fizycznej na maszynie CloudLab2 (pkt 6.5.1). Problem ten został rozwiązany dzięki korespondencji z osobami zarządzającymi maszyną, efektem tego otrzymaliśmy dostęp do nowej przestrzeni dyskowej, gdzie mogliśmy kontynuować pracę.

7.5. Przyrost czwarty - integracja

W tym przyroście skupiliśmy się na zintegrowaniu wyuczonych modułów, utworzeniu pliku konfiguracyjnego, utworzeniu połączeń pomiędzy źródłem danych a modułami oraz konfiguracji systemu logów.

7.5.1. Założenia

- Stworzenie uniwersalnego API dla każdego modelu
- Osadzenie modułów w API i późniejsza konteneryzacja
- Integracja modułów w jeden system
- Stworzenie konfigurowalnego pliku do uruchamiania systemu

7.5.2. Zadania

7.5.2.1. Stworzenie uniwersalnego API dla każdego modelu

Realizując to zadanie skupiliśmy się na stworzeniu uniwersalnego API dla każdego modelu. Wykorzystaliśmy do tego framework FastAPI używający języka programowania

Python. Jako przewodnią architekturę API wybraliśmy architekturę cebulową [47]. Składa się z trzech warstw: prezentacji, logiki biznesowej i danych. Warstwa prezentacji odpowiada za przyjmowanie i zwracanie odpowiedzi na zapytania do aplikacji oraz konfiguracją routerów zarządzających adresami zasobów. Logika biznesowa spełniała rolę koordynatora i zajmowała się transformacją danych a następnie przekazaniem ich do modelu. Warstwa danych w tym przypadku była bardzo skromna i zajmowała się ładowaniem konfiguracji oraz modelu z dysku. Szczegóły dotyczące FastAPI opisujemy w punkcie 6.6.

7.5.2.2. Osadzenie modeli w API i późniejsza konteneryzacja

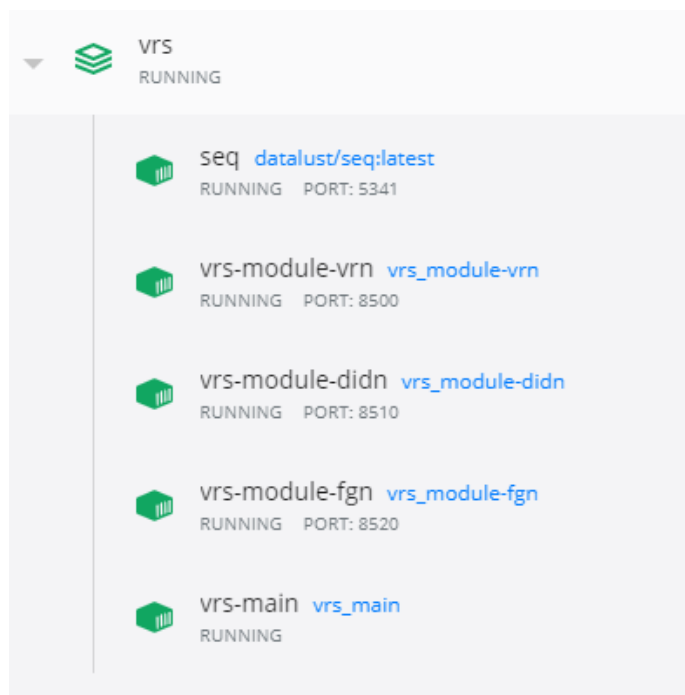
Każdy moduł został wyeksportowany do uniwersalnego formatu TensorFlow a następnie osadzony w strukturze programu. API posiadało endpoint, który przyjmował serię klatek do analizy i jako odpowiedź zwracał wynik predykcji. Ze względu na długi czas serializacji danych zrezygnowaliśmy z tradycyjnego REST API na rzecz gRPC z binarną serializacją do Protobuf. Modele ostatecznie zostały osadzone w kontenerach TensorFlow Serving, gdzie są serwowane przez gRPC. Obie technologie i motywy stojące za porzuceniem FastAPI opisujemy w punktach 6.6 oraz 6.7. Ze względu na specyfikację każdego z modeli, wymagają one indywidualnych transformacji na obrazie. Dlatego razem z modelami w każdym API został również umieszczony kod odpowiadający za wstępne przetwarzanie danych przed analizą.

7.5.2.3. Integracja modułów w jeden system

Do koordynowania procesem detekcji przemocy oraz zbierania materiału wideo z kamer stworzyliśmy wielowątkową aplikację napisaną w języku Python. Jeden z wątków aplikacji jest przeznaczony tylko do utrzymywania połączenia z kamerą oraz ciągłego zbierania klatek. Udało nam się to osiągnąć wykorzystując bibliotekę OpenCV oraz wielowątkowe podejście. Ze względu na indywidualne potrzeby każdego modelu w kwestii transformacji danych, ten proces również rozdzielamy na niezależne wątki per model. Dzięki temu podejściu udało nam się zaoszczędzić 12 sekund, kończąc z wynikiem około 1.5 sekundy dla całego procesu analizy. Od zbierania klatek po zwrócenie odpowiedzi z trzech modułów i zapisanie jej w logach Seq.

Każdy z modułów osadziliśmy w osobnym kontenerze. W celu uproszczenia procesu uruchamiania systemu oraz zwiększenia potencjału na skalowalności, zarazem

umożliwiając wdrożenie systemu w chmurze lub innym środowisku produkcyjnym stworzyliśmy plik konfiguracyjny (opisany w punkcie 7.5.2.4). Dzięki niemu, jednym kliknięciem uruchomimy cały system i rozpoczniemy analizę obrazu ze źródła wskazanego w konfiguracji. System pozwala na nasłuchiwanie sygnału wideo z każdej kamery, do której ma dostęp maszyna, na której został uruchomiony system. Ustawienie sieci pozostawiamy klientowi i nie narzucamy z góry jej konfiguracji.



Rysunek 7.7 Zrzut ekranu z Docker przedstawiający wynik uruchomienia systemu pliku Docker Compose.

7.5.2.4. Stworzenie konfigurowalnego pliku do uruchamiania systemu

W celu umożliwić prostą i zarazem skuteczną konfigurację systemu zdecydowaliśmy się na wykorzystanie plików yaml dla narzędzia Docker Compose. Umożliwiliśmy podanie adresu źródłowego kamery oraz wyłączenie i włączanie modułów. Jeżeli z jakiegoś powodu klient wymagałby udostępniania innego sposobu na konfigurację systemu to Docker Compose pozwala na wykorzystanie zmiennych środowiskowych oraz zwykłych plików tekstowych.

7.5.3. Napotkane problemy

7.5.3.1. Powtórne ładowanie modelu sieci przed każdą analizą

Przed rozpoczęciem analizy model wymaga załadowania. W pierwotnej wersji, system robił to przed każdą predykcją i było to wyjątkowo niewydajne. Za pierwszym razem ten problem rozwiązaliśmy przez wstępne załadowanie modelu na początku inicjalizacji aplikacji. Model był przechowywany w pamięci aplikacji. Po odrzuceniu autorskiego rozwiązania wykorzystującego FastAPI i przejściu na TensorFlow Serving problem ostatecznie rozwiązał się sam, dzięki nowemu rozwiązaniu. Kontener z TensorFlow Serving samemu optymalizuje proces ładowania modelu (pkt 6.7).

7.5.3.2. Długi czas potrzebny na serializację danych.

Wykorzystując nasze autorskie rozwiązanie oparte o API używające formatu JSON, serializacja zestawu klatek do analizy zajmował relatywnie dużo czasu względem samej predykcji. W trakcie poszukiwań alternatywy trafiliśmy na gRPC, który był jedną z dwóch metod transferu danych udostępnianych przez TensorFlow Serving. gRPC korzysta z protokołu Protobuf do przetwarzania danych i przesyła je w formie binarnej. Dzięki zaimplementowaniu rozwiązania opartego o gRPC udało nam się zaoszczędzić ponad 600 milisekund. Szczegóły opisujemy w punkcie 6.7.

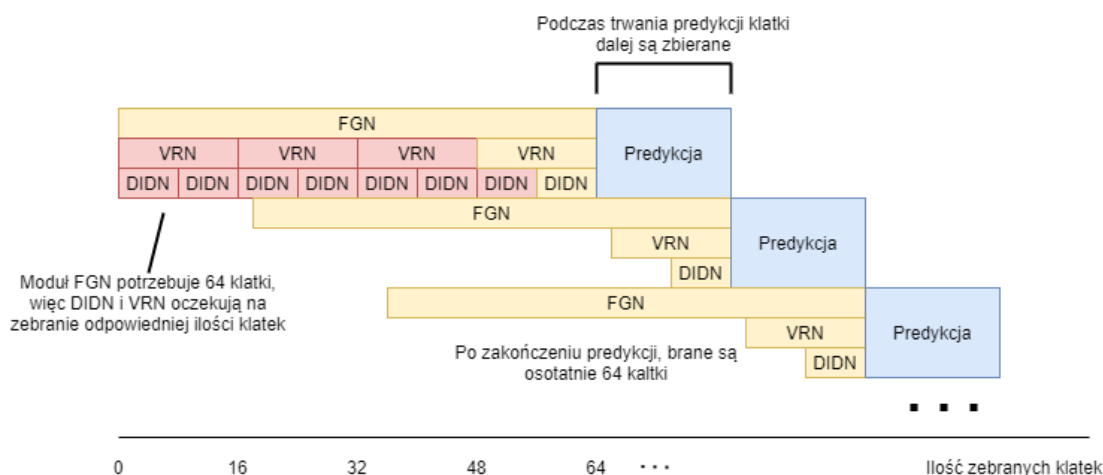
7.5.3.3. Synchroniczna transformacja i analiza materiału

Osadzenie i serwowanie modeli z niezależnych kontenerów pozostawia duże pole na optymalizację całego systemu. Do tego momentu transformacji, transfer danych i analiza był wykonywany synchronicznie. Każdy z trzech modułów przechodził ten sam proces po kolei. W takiej konfiguracji średni czas dla jednego pełnego cyklu z wykorzystaniem trzech modeli wynosił około 13 sekund. Wykorzystując podejście wielowątkowe, udało nam się zaoszczędzić blisko 7 sekund. Wyżej wymieniony proces aktualnie wykonuje się równolegle, z użyciem jednego wątku dla każdego z modeli.

7.5.3.4. Niewydajny sposób na zbieranie klatek przed każdą analizą

W pierwotnej wersji, zestaw klatek był zbierany za każdym razem na nowo, przed rozpoczęciem predykcji. W zależności od FPS sygnału źródłowego, kolekcjonowanie 64 klatek potrzebnych do analizy na modelu FGN trwało średnio 3 sekundy. Rozwiązaniem okazało przechowywanie ostatnich 64 klatek i dodawanie kolejnej, przy tym usuwając

najstarszą, w przypadku pojawienia się nowej. Ten proces osadziliśmy w osobnym wątku, który rozpoczyna swoją pracę jako pierwszy po rozpoczęciu programu. Wątek na bieżąco aktualizuje zestaw klatek, z którego czipią inne wątki odpowiedzialne za transformację i analizę danych.



Rysunek 7.8 Schemat predykcji rozłożony w czasie z podejściem wielowątkowym

7.6. Przyrost piąty – optymalizacja i walidacja

W tym przyroście skupiliśmy się na walidacji, optymalizacji zintegrowanego systemu oraz dostosowaniu parametrów w celu otrzymania jak najlepszych wyników.

7.6.1. Założenia

- Kalibracja wartości wag
- Walidacja systemu

7.6.2. Zadania

7.6.2.1. Kalibracja wartości wag

Proces ten polegał na dostosowaniu wartości wag do wzoru końcowej klasyfikacji opisanego w podpunkcie 6.1.3. Jest to ważny proces, ponieważ na podstawie tych wag będziemy klasyfikować prawdopodobieństwo przemocy na serii klatek. Wagi modułów FGN i VRN zbiegają się do 1, natomiast DIDN wykracza poza. Dzieje się tak, ponieważ nie chcemy by moduł DIDN wpływał na ocenę przemocy, a jedynie zwiększał prawdopodobieństwo wystąpienia sytuacji niebezpiecznej w przypadku pojawienia się broni. Przykładowo: gdyby moduły miały równe wagi o wartości 1/3, w razie wykrycia

przemocy rzędu 100% przez FGN i VRN i braku wykrycia broni przez DIDN (0%), ostateczny wynik byłby 66%. Co jak widać nie jest odzwierciedleniem rzeczywistej sytuacji. Niestety nie mamy żadnej idealnej metody dostosowującej najlepsze wagi do poszczególnych modułów. Przyjęliśmy wagi startowe i od nich w procesie „eksperymentów” odchodziliśmy w różne kierunki sprawdzając zmianę wyników systemu na jednym zestawie danych. Niżej w formie tabel opisaliśmy wyniki naszych eksperymentów i przyjmowanych wag:

7.6.2.1.1. Eksperyment 1

	FGN	VRN	DIDN
Waga	0.5	0.5	0.5

Tabela 7.1 Tabela przedstawiająca zestaw wartości dla pierwszego eksperymentu kalibracji wartości wag

7.6.2.1.1.1. Rezultaty i wnioski

Przy wyżej podanym zestawie wag, po wykonanych testach większość wyników finalnych nie zgrywała się z wcześniej założonymi oczekiwanymi wynikami. Tak wysoka waga dla modułu DIDN powodowała zbyt wysokie wyniki, kiedy tylko na obrazie pojawiała się broń.

7.6.2.1.2. Eksperyment 2

	FGN	VRN	DIDN
Waga	0.5	0.5	0.3

Tabela 7.2 Tabela przedstawiająca zestaw wartości dla drugiego eksperymentu kalibracji wartości wag

7.6.2.1.2.1. Rezultaty i wnioski

Zważając na efekty pierwszego eksperymentu postanowiliśmy obniżyć wagę modułu DIDN do wartości 0.3. Rozwiązało to problem zbyt wysokich wyników w sytuacjach pojawienia się broni na obrazie, jednak wygenerowało to kolejny problem, w którym to moduł VRN zaniżał wyniki.

7.6.2.1.3. Eksperyment 3

	FGN	VRN	DIDN
Waga	0.8	0.2	0.3

Tabela 7.3 Tabela przedstawiająca zestaw wartości dla trzeciego eksperymentu kalibracji wartości wag

7.6.2.1.3.1. Rezultaty i wnioski

Biorąc pod uwagę rezultaty poprzedniego eksperymentu postanowiliśmy podbić wartość wagi dla modułu FGN. Niestety poprzedni problem przekształcił się w inny, a mianowicie w niektórych przypadkach moduł VRN wykrywał lepiej przemoc i był bliżej wartości oczekiwanych, ale wysoka waga FGN zaniżała jego wynik.

7.6.2.1.4. Eksperyment 4

	FGN	VRN	DIDN
Waga	0.6	0.4	0.3

Tabela 7.4 Tabela przedstawiająca zestaw wartości dla czwartego eksperymentu kalibracji wartości wag

7.6.2.1.4.1. Rezultaty i wnioski

Biorąc pod lupę wyniki eksperymentu 3 postanowiliśmy trochę uśrednić i przybliżyć wartości wag modułów FGN i VRN do siebie. Rozwiązało to poprzedni problem, lecz niestety powrócił do nas problem zbyt wysokich wartości generowanych przez moduł DIDN, kiedy na obrazie pojawiała się broń.

7.6.2.1.5. Eksperyment 5

	FGN	VRN	DIDN
Waga	0.6	0.4	0.2

Tabela 7.5 Tabela przedstawiająca zestaw wartości dla piątego eksperymentu kalibracji wartości wag

7.6.2.1.5.1. Rezultaty i wnioski

Delikatnie obniżając wagę modułu DIDN udało nam się uzyskać wyniki najbardziej zbliżone do zdefiniowanych wcześniej wartości oczekiwanych. Na tym etapie zdecydowaliśmy, że wyniki są wystarczająco zadowalające.

7.6.2.2. Walidacja systemu

Cały proces walidacji oraz analizy systemu został szczegółowo opisany w punkcie 8.

7.6.3. Napotkane problemy

7.6.3.1. Czas

Walidacja przy użyciu dużego zbioru testowego jest procesem czasochłonnym z racji czasu wymaganego na wykonanie tego procesu jak i późniejszą analizę.

7.7. Przyrost szósty – poprawki

W tym przyroście skupiliśmy się na finalnych poprawkach dokumentacji oraz kodu źródłowego.

7.7.1. Założenia

- Poprawa dokumentacji
- Oczyszczenie kodu
- Wzbogacenie kodu o komentarze

7.7.2. Zadania

7.7.2.1. Poprawa dokumentacji

Po otrzymaniu recenzji naszej pracy przez promotorów nanieśliśmy finalne poprawki w celu zwiększenia jakości dokumentacji.

7.7.2.2. Oczyszczenie kodu

Zdecydowaliśmy oczyścić kod źródłowy powstałego systemu, aby był on łatwiejszy w zrozumieniu oraz by ewentualna jego rozbudowa była łatwiejsza.

7.7.2.3. Wzbogacenie kodu o komentarze

Postanowiliśmy dodać komentarze w celu wyjaśnienia pewnych części kodu. Ułatwi to ewentualną dalszą pracę w przyszłości.

7.7.3. Napotkane problemy

7.7.3.1. Presja czasu

Wraz ze zbliżającym się terminem prac odczuliśmy sporą presję, która dodatkowo motywowała nas do jak najszybszego ukończenia wszelakich prac przewidywanych w tym projekcie.

7.8. Podsumowanie

Prace nad systemem zostały zakończone pomyślnie. Finalny produkt spełnia wszystkie wymagania przedstawione w rozdziale 5. Na jego podstawie przygotowana została prezentacja opisująca działanie systemu w praktyce oraz poruszająca kwestie implementacyjne. Jesteśmy bardzo zadowoleni z rezultatu jaki udało nam się osiągnąć i w przyszłości chcielibyśmy dalej rozwijać ten produkt.

8. Walidacja rozwiązania

8.1. Sposób walidacji

Aby dokładnie przetestować nasz system zdecydowaliśmy się na dwa sposoby walidacji. Jednym z nich jest określenie błędu predykcji naszego systemu, a drugim sposobem jest obliczenie dokładności predykcji (testując przy tym podstawowe założenie opisane w punkcie 4.1).

Proces walidacji działania systemu rozpoczęliśmy od przygotowania pięćdziesięciu streamów wideo, z których część zawiera przemoc, część przemoc + broń, część brak przemocy + broń i część bez przemocy i broni. Dodatkowo każdy stream został dopasowany w sposób by posiadał spójną liczbę klatek na sekundę (30 FPS).

8.1.1. Błąd predykcji systemu

Z każdego ze streamów na przestrzeni 10 sekund wyciągamy średnią dokładność predykcji naszego systemu. Ten sam proces powtarzamy dla oczekiwanych przez nas wyników. Następnie na podstawie tych danych obliczamy pierwiastek błędu średniego kwadratowego (RMSE) uzyskując w ten sposób błąd predykcji naszego systemu w punktach procentowych. Wyżej opisany schemat może zostać przedstawiony za pomocą następujących wzorów:

$$\bar{x} = \frac{1}{m} \sum_{i=0}^m x_i$$

$$\bar{y} = \frac{1}{m} \sum_{i=0}^m y_i$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=0}^n (\bar{x}_i - \bar{y}_i)^2} \text{ gdzie:}$$

x_i – średni wynik i – tej predykcji

y_i – wynik oczekiwany i – tej predykcji

\bar{x}_i – średni wynik i – tego testu

\bar{y}_i – średni wynik oczekiwany i – tego testu

m – liczba predykcji na dany test

n – liczba testów

Równanie 2 Średni błąd predykcji systemu VRS

8.1.2. Dokładność predykcji systemu

W celu sprawdzenia dokładności predykcji systemu, będziemy się przyglądać wynikom predykcji po normalizacji (Równanie 3). W tym celu wyznaczaliśmy punkt 70% jako granica wykrywania przemocy. Po transformacji wyników liczymy procent poprawnych wyników, uzyskując tym samym przybliżoną dokładność naszego systemu (Równanie 4)

$$f(x) = \begin{cases} 0, & x < 0.70 \\ 1, & x \geq 0.70 \end{cases}$$

gdzie:

x – wynik predykcji

Równanie 3 Normalizacja wyników predykcji

$$VRS\ accuracy = \frac{\text{number of accurate predictions}}{\text{number of all predictions}}$$

Równanie 4 Przybliżona dokładność predykcji systemu VRS

8.2. Przeprowadzone testy

Na zestawie wybranych wideo przeprowadziliśmy testy w postaci zaciągnięcia wyników klasyfikacji, normalizacji i zapisania ich w postaci tabeli. W poniższych punktach można znaleźć opis przykładowych dwunastu testów na różnych sytuacjach z wideo. Każdy film różnił się swoim charakterem, dlatego też jest to opisane na wstępie każdego testu. Reszta testów (38) znajduje się w podsumowaniu wyników w punkcie 8.3.

8.2.1. Test 1

Cechy charakterystyczne filmu: typowy przykład braku przemocy - na nagraniu jest mężczyzna idący korytarzem. Nagranie jest dobrej jakości w orientacji pionowej. Obraz jest statyczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	29.07	0	0	0
2	38.99	0	0	0
3	31.03	0	0	0
4	49.97	0	0	0
5	41.47	0	0	0
Średnia		Ilość poprawnych		
40.11		0		5

Tabela 8.1 Wyniki predykcji w teście nr 1

8.2.2. Test 2

Cechy charakterystyczne filmu: grupa osób wykonującą aktywności fizyczne. Nagranie jest dobrej jakości w orientacji pionowej. Obraz jest statyczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	27.64	0	0	0
2	19.20	0	0	0
3	45.41	0	0	0
4	47.4	0	0	0
5	45.1	0	0	0
Średnia		Ilość poprawnych		
36.95		0		5

Tabela 8.2 Wyniki predykcji w teście nr 2

8.2.3. Test 3

Cechy charakterystyczne filmu: brak ludzi – szybko poruszające się pojazdy i wypadek. Nagranie jest średniej jakości w orientacji pionowej. Obraz jest statyczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	26.34	0	0	0
2	26.37	0	0	0
3	27.38	0	0	0
4	58.87	0	0	0
5	44.20	0	0	0
Średnia		Ilość poprawnych		
36.63		0		
		5		

Tabela 8.3 Wyniki predykcji w teście nr 3

8.2.4. Test 4

Cechy charakterystyczne filmu: marsz uzbrojonych żołnierzy w broń długą. Nagranie jest średniej jakości w orientacji pionowej. Obraz jest dynamiczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	27.08	20	0	0
2	46.22	20	0	0
3	66.47	20	0	0
4	65.44	20	0	0
5	59.09	20	0	0
Średnia		Ilość poprawnych		
52.86		20		
		5		

Tabela 8.4 Wyniki predykcji w teście nr 4

8.2.5. Test 5

Cechy charakterystyczne filmu: napad na sklep z bronią w ręku (brak aktów przemocy). Nagranie jest dobrej jakości w orientacji poziomej. Obraz jest statyczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	17.09	0	0	0
2	47.06	20	0	0
3	39.20	20	0	0
4	41.78	20	0	0
5	40.39	20	0	0
Średnia		Ilość poprawnych		
37.10		16		5

Tabela 8.5 Wyniki predykcji w teście nr 5

8.2.6. Test 6

Cechy charakterystyczne filmu: uzbrojeni żołnierze przygotowujący się do strzału (brak przemocy). Nagranie słabej jakości w orientacji poziomej. Obraz statyczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	30.12	20	0	0
2	36.96	20	0	0
3	30.58	20	0	0
4	48.05	20	0	0
5	36.00	20	0	0
Średnia		Ilość poprawnych		
36.34		20		5

Tabela 8.6 Wyniki predykcji w teście nr 6

8.2.7. Test 7

Cechy charakterystyczne filmu: bójka przed domem. Walka następuje od pierwszych chwil ujęcia. Nagranie jest średniej jakości w orientacji pionowej. Obraz jest statyczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	33.91	100	0	1
2	65.80	100	0	1
3	64.62	100	0	1
4	83.71	100	1	1
5	87.12	100	1	1
Średnia		Ilość poprawnych		
67.03		100	2	

Tabela 8.7 Wyniki predykcji w teście nr 7

8.2.8. Test 8

Cechy charakterystyczne filmu: bójka na parkingu. Po chwili następuje atak. Nagranie jest dobrej jakości w orientacji pionowej. Obraz jest statyczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	56.14	0	0	0
2	63.40	100	0	1
3	64.54	100	0	1
4	70.50	100	1	1
5	58.46	100	0	1
Średnia		Ilość poprawnych		
62.60		80	2	

Tabela 8.8 Wyniki predykcji w teście nr 7

8.2.9. Test 9

Cechy charakterystyczne filmu: bójka w barze. Po chwili następuje atak. Nagranie jest dobrej jakości w orientacji poziomej. Obraz jest statyczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	30.66	0	0	0
2	27.27	0	0	0
3	46.41	100	0	1
4	61.52	100	0	1
5	75.81	100	1	1
Średnia		Ilość poprawnych		
48.33		60	3	

Tabela 8.9 Wyniki predykcji w teście nr 9

8.2.10. Test 10

Cechy charakterystyczne filmu: napad na policjanta. Broń użyta jest w połowie nagrania. Nagranie jest dobrej jakości, w orientacji poziomej. Obraz jest dynamiczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	36.47	0	0	0
2	36.68	0	0	0
3	82.76	100	1	1
4	69.18	100	0	1
5	58.49	100	0	1
Średnia		Ilość poprawnych		
56.72		60	3	

Tabela 8.10 Wyniki predykcji w teście nr 10

8.2.11. Test 11

Cechy charakterystyczne filmu: napad w sklepie. Po chwili następuje atak. Broń jest ledwo widoczna. Nagranie jest słabej jakości w orientacji poziomej. Obraz jest statyczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	25.40	0	0	0
2	32.64	0	0	0
3	52.55	100	0	1
4	66.19	100	0	1
5	29.83	100	0	1
Średnia		Ilość poprawnych		
41.32		60	2	

Tabela 8.11 Wyniki predykcji w teście nr 11

8.2.12. Test 12

Cechy charakterystyczne filmu: duża szarpanina w metrze. Nagranie jest stosunkowo dobrej jakości w orientacji poziomej. Obraz jest statyczny.

Numer predykcji	Wynik predykcji [%]	Oczekiwany wynik predykcji [%]	Normalizacja wyniku predykcji	Normalizacja oczekiwanego wyniku predykcji
1	29.57	0	0	0
2	56.00	100	0	1
3	90.38	100	1	1
4	71.52	100	1	1
5	68.66	100	0	1
Średnia		Ilość poprawnych		
63.23		80	3	

Tabela 8.12 Wyniki predykcji w teście nr 12

8.3. Analiza przeprowadzonych testów

Numer testu	Średni wynik predykcji [%]	Średni oczekiwany wynik predykcji [%]	Ilość poprawnych predykcji
1	40.11	0	5
2	36.95	0	5
3	36.63	0	5
4	52.86	20	5
5	37.10	16	5
6	36.34	20	5
7	67.03	100	2
8	62.60	80	2
9	48.33	60	3
10	56.72	60	3
11	41.32	60	2
12	63.23	80	3
13	76.38	100	4
14	56.11	60	5
15	34.81	20	5
16	84.16	100	5
17	76.22	100	4
18	50.25	20	3
19	27.02	0	5
20	37.38	0	5
21	38.86	0	5
22	61.91	20	3
23	60.35	20	3
24	81.34	100	5
25	86.16	100	5

Numer testu	Średni wynik predykcji [%]	Średni oczekiwany wynik predykcji [%]	Ilość poprawnych predykcji
26	81.19	100	5
27	87.13	100	5
28	82.11	80	5
29	85.57	80	5
30	71.11	80	5
31	84.75	100	5
32	84.62	100	5
33	78.52	100	4
34	10.33	0	5
35	7.93	0	5
36	10.12	0	5
37	77.78	60	5
38	81.34	100	5
39	66.92	60	3
40	73.09	100	4
41	82.97	80	5
42	78.60	60	4
43	21.58	0	5
44	23.87	0	5
45	13.72	0	5
46	56.92	20	4
47	9.44	0	5
48	33.49	20	5
49	83.24	80	5
50	16.01	0	5

Tabela 8.13 Wyniki średnich predykcji w testach błędu predykcji

8.3.1. Błąd predykcji systemu

Korzystając z wyżej omawianego wzoru (Równanie 2) obliczamy w przybliżeniu błąd predykcji naszego systemu w punktach procentowych:

$$\begin{aligned} RMSE &= \frac{\sqrt{(40.11 - 0)^2 + (36.95 - 0)^2 + \dots + (83.24 - 80)^2 + (16.01 - 0)^2}}{50} \\ &= \frac{\sqrt{25112.66}}{50} \approx 3.17 \% \end{aligned}$$

Na podstawie danych uzyskanych w testach obliczyliśmy, że błąd predykcji wyrażony w punktach procentowych z jaką nasz system rozpoznaje przemoc wynosi w przybliżeniu 3.17%.

8.3.2. Dokładność predykcji systemu

Korzystając z wyżej omawianego wzoru (Równanie 4) obliczamy w przybliżeniu dokładność predykcji naszego systemu:

$$VRS \text{ accuracy} = \frac{\text{number of accurate predictions}}{\text{number of all predictions}} = \frac{221}{n * m} = \frac{221}{250} = 88.40\%$$

Na podstawie danych uzyskanych w testach obliczyliśmy, że dokładność predykcji wyrażona w punktach procentowych z jaką nasz system rozpoznaje przemoc wynosi w przybliżeniu 88.40 %.

8.4. Podsumowanie

Wykonane przez nas procesy walidacyjne pomogły nam zweryfikować wcześniej postawione przez nas główne wymaganie dotyczące dokładności systemu (WF 01), patrz 5.2. W trakcie walidacji system został poddany serii testów, polegających na analizie różnorodnych materiałów wideo symulujących dane przychodzące z kamery. Dzięki tym testom udało nam się obliczyć średni błąd predykcji systemu, którego wartość wyniosła około 3,17% (pkt 8.3.1). Jednocześnie udało nam się obliczyć dokładność predykcji systemu, której wartość wyniosła w przybliżeniu 88,40% (pkt 8.3.2). Wynik ten uważamy za ogromny sukces. Argumentem za tym jest fakt, iż spełnia on jeden z głównych celów dla tego projektu, jakim była poprawna klasyfikacja przemocy lub jej braku na poziomie przynajmniej 80% (pkt 4.1).

9. Nakład sumaryczny

9.1. Sposoby obliczeń nakładu sumarycznego

9.1.1. Nakład pracy w postaci tabeli

Pierwszym sposobem do opisanego nakładu sumarycznego jaki sformułowaliśmy było sporządzenie go w formie podzielonej na przyrosty tabeli. W każdym przyroście jest zawarta wartość sumaryczna spędzonych dni danego członka zespołu projektowego oraz procent udziału względem trwania całego przyrostu. Ostatnią sekcją tabeli jest podsumowanie, gdzie możemy zaobserwować wyniki względem przebiegu wszystkich przyrostów.

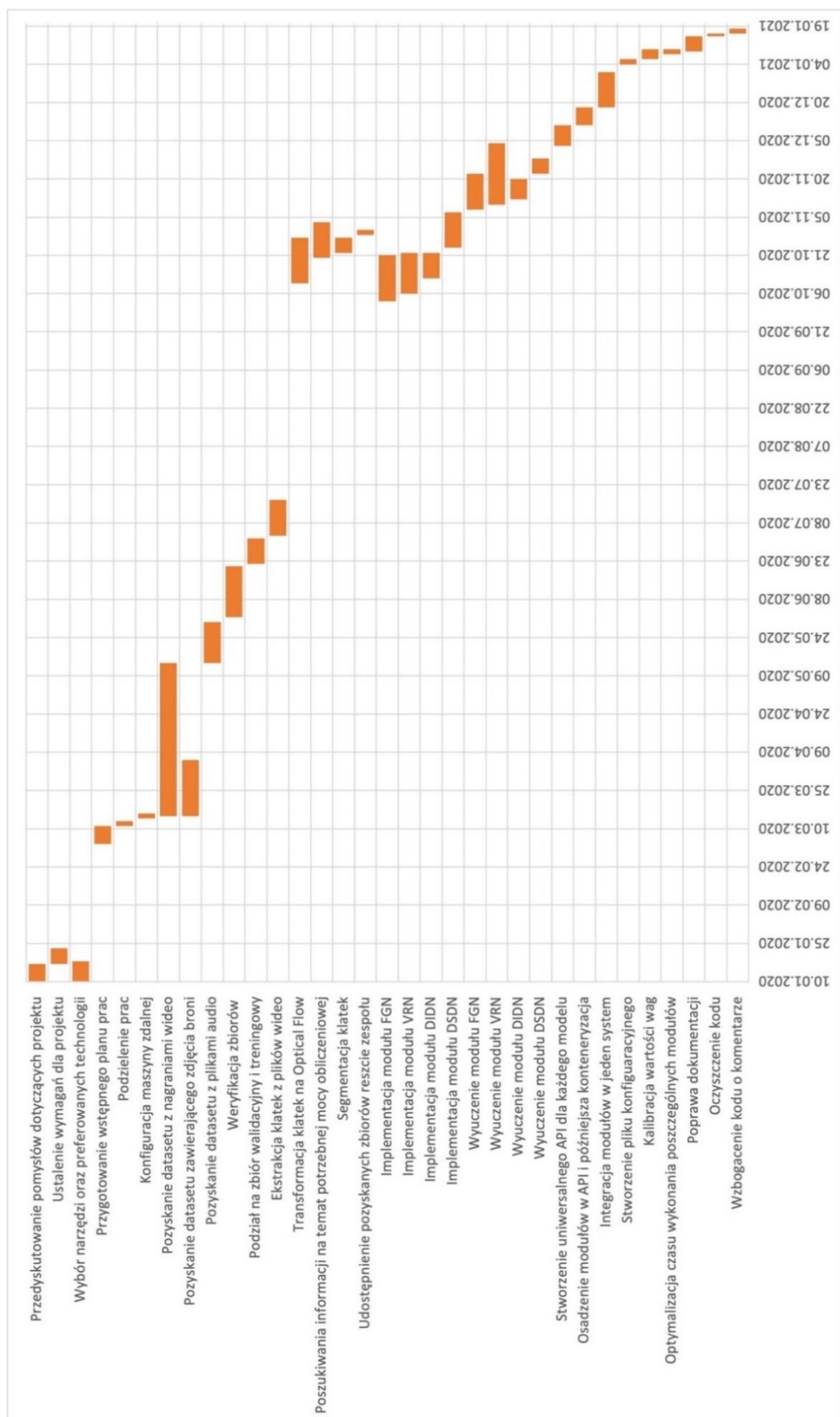
	Ola	Jakub	Mateusz	Benedykt
Przyrost 0 – 30 dni				
Ilość dni	30	30	30	30
Procent	100	100	100	100
Przyrost 1 – 170 dni				
Ilość dni	124	98	54	45
Procent	73	58	32	41
Przyrost 2 – 48 dni				
Ilość dni	22	16	10	0
Procent	46	34	21	0
Przyrost 3 – 70 dni				
Ilość dni	22	31	8	2
Procent	32	43	11	3
Przyrost 4 – 59 dni				
Ilość dni	0	0	0	59
Procent	0	0	0	100

	Ola	Jakub	Mateusz	Benedykt
Przyrost 5 – 8 dni				
Ilość dni	8	6	6	8
Procent	100	75	75	100
Przyrost 6 – 10 dni				
Ilość dni	10	10	10	10
Procent	100	100	100	100
PODSUMOWANIE – 395 dni				
Suma dni	216	191	118	155
Procent	55	48	30	40

Tabela 9.1 Nakład sumaryczny przedstawiony w formie tabeli

9.1.2. Nakład pracy w postaci diagramu Gantta

Drugim sposobem na zwizualizowanie nakładu sumarycznego naszego projektu było wpięrow przygotowanie odpowiednich danych dotyczących czasu spędzonego na poszczególnych zadaniach w postaci przedziałów czasowych. Następnie na bazie tych danych przy użyciu programu Excel wygenerowaliśmy diagram Gantta (Rysunek 9.1)



Rysunek 9.1 Diagram Gantta

10. Wkład własny

10.1. Ola Piętka

10.1.1. Prace wykonane na poczet systemu

10.1.1.1. Przyrost pierwszy – dataset

W tym przyroście byłam współodpowiedzialna za pozyskanie i weryfikację zbioru danych z filmami (pkt 7.2.2.1 i 7.2.2.4) oraz w pełni odpowiedzialna za zbiór plików audio (pkt 7.2.2.3).

Oprócz samego poszukiwania i pozyskania wyżej wymienionych datasetów, była potrzebna ich transformacja na wszelakie podzbiory. Dlatego tym kierunkiem został przeze mnie napisany szereg skryptów umożliwiających:

- ekstrakcje Optical Flow z klatek metodą OpenCV, w celu uzyskania podzbioru danych o nazwie RWF-2000 Optical Subset (pkt 7.2.2.7.1),
- ekstrakcje Optical Flow z klatek metodą RAFT, w celu uzyskania podzbioru danych o nazwie RWF-2000 RAFT Subset (pkt 7.2.2.7.2),
- segmentacje klatek przy pomocy modelu udostępnionego przez Keras, w celu uzyskania podzbioru danych (pkt 7.2.2.8),
- transformacje ścieżek audio na spektrogramy (pkt 7.2.2.9).

10.1.1.2. Przyrost drugi – implementacja

W tej części projektu zajęłam się implementacją wcześniej zdefiniowanej architektury modułów FGN oraz DSDN. Opis modułów oraz wykorzystane w implementacji technologie znajdują się odpowiednio w punktach: 7.3.2.1, 7.3.2.4.

10.1.1.3. Przyrost trzeci – uczenie

Podczas przyrostu trzeciego zajęłam się wyuczeniem wcześniej zaimplementowanych modułów FGN oraz DSDN (opisane dokładniej w punkcie 6.5.2 i 6.5.5). Dodatkowo uczenie wiązało się z przeprowadzeniem serii eksperymentów w celu uzyskania najlepszych wyników modeli. Bardziej dokładny opis przebiegu uczenia został opisany odpowiednio w punktach 7.4.2.1 oraz 7.4.2.4.

10.1.1.4. Przyrost czwarty – integracja

Zajęłam się przygotowaniem gotowego modułu FGN do stanu umożliwiającego jego integrację z całym systemem, nadzorowałam pracę przy zadaniach związanych z integracją wyuczonych modeli oraz wytworzyłam wzór do obliczenia finalnej klasyfikacji VRS (Równanie 1).

10.1.1.5. Przyrost piąty – optymalizacja i walidacja

Udało mi się czynnie uczestniczyć przy optymalizacji parametrów i doboru wag (pkt 7.6.2.1) do równania końcowej klasyfikacji (Równanie 1), dokonać poboru danych i obliczeń w procesie walidacji (rozdział 8) oraz stworzyłam wzór liczący błąd predykcji systemu (Równanie 2) oraz finalnej jego dokładności (Równanie 4).

10.1.1.6. Inne

Dodatkowo w trakcie realizacji projektu przeprowadzałam analizę i eksperymenty dotyczące różnych metod możliwych do wykorzystania w takiego typu systemie (opisane w punkcie 6.8). Finalnie metody te nie wzięły udziału w ostatecznej wersji systemu, ale praca nad nimi poszerzyła moje umiejętności i zakres wiedzy.

10.1.2. Prace wykonane na poczet dokumentacji

10.1.2.1. Dział 1 – wstęp

Napisałam wstęp do pracy w rozdziale 0.

10.1.2.2. Dział 2 – słownik pojęć

Współtworzyłam słownik pojęć podczas pracy nad dokumentacją opisaną w rozdziale 2.

10.1.2.3. Dział 3 – analiza problemu

W tym dziale byłam odpowiedzialna za przedstawienie problemu jaki chcemy rozwiązać (pkt 3.1) oraz ręczne narysowanie obrazów rich picture w punkcie 3.2 (Rysunek 3.2, Rysunek 3.3, Rysunek 3.4)

10.1.2.4. Dział 4 – planowanie

W trakcie realizacji czwartego rozdziału w pierwszej kolejności określiłam udziałowców projektu w formie tabel (pkt 4.3). Następnie stworzyłam przypadki użycia przy pomocy języka UML (Rysunek 4.1) i opisałam naszych aktorów (pkt 4.5). Sporządziłam również trzy przykładowe scenariusze korzystania z systemu (pkt 4.5.1).

Dodatkowo przyczyniłam się do wytworzenia charakterystyki klientów wewnętrznych i zewnętrznych (pkt 4.4) oraz narysowałam ręcznie rysunek przedstawiający naszą strategię wytwarzania systemu (Rysunek 4.2).

10.1.2.5. Dział 5 – analiza wymagań

Sporządziłam tabele do wyspecyfikowanych przez nas wcześniej wymagań oraz byłam współodpowiedzialna za wymagania ogólne (pkt 5.1), funkcjonalne (pkt 5.2) oraz niefunkcjonalne (pkt 5.3) naszego systemu. Dodatkowo pod koniec powiązałam udziałowców i wymagania ze sobą.

10.1.2.6. Dział 6 – projektowanie

W trakcie pracy nad rozdziałem szóstym, opisałam wymyślony wcześniej wzór końcowej klasyfikacji (Równanie 1). Następnie stworzyłam schemat głównej pętli naszego systemu (Rysunek 6.3). Zabrałam się również za opisanie wstępnych wymagań odnośnie datasetu z filmami (pkt 6.3.2.1), opisanie naszego ostatecznego zbioru RWF-2000 (pkt 6.3.1.2) oraz dwóch wygenerowanych podzbiorów Optical Flow (punkty 6.3.1.2.2 i 6.3.1.2.3). Ponieważ byłam odpowiedzialna za moduł wykorzystujący audio, w punkcie 6.3.3 określiłam wymagania, opisałam pozyskanie zbioru plików dźwiękowych oraz transformację dźwięku na spektrogramy.

Następnie opisałam moduły przeze mnie wytworzone oraz ich architekturę odpowiednio w punktach 6.4.1 i 6.4.4. Wyuczenie, iteracje i analiza tych modułów znajduje się odpowiednio w punktach 6.5.2 i 6.5.4.

Na koniec opisałam w punkcie 6.8 pokrótce o naszych porzuconych pomysłach i powodach, dla których z nich zrezygnowaliśmy.

10.1.2.7. Dział 7 – realizacja projektu

Zaczęłam od określenia struktury całego działu siódmego. Następnie wyjaśniłam co działo się w przyroście pierwszym (wykluczając punkt 7.1.2.2), opisałam implementacje modułów FGN i DSDN (punkty 7.3.2.1 i 7.3.2.4), proces wyuczenia modułów FGN i DSDN (punkty 7.4.2.1 i 7.4.2.4), wykonałam schemat dotyczący predykcji rozłożonej w czasie (Rysunek 7.8) oraz współtworzyłam punkty 7.6 i 7.7.

10.1.2.8. Dział 8 – walidacja rozwiązania

Wyjaśniłam wygenerowane wcześniej wzory (Równanie 2 i Równanie 4) i podejście do walidacji oraz analizy naszego systemu w punkcie 8.1. Dodatkowo zajęłam się strukturą tabel, ich opisem i uzupełnieniem w testach w podpunkcie 8.2. Następnie obliczyłam finalny wynik dokładności systemu oraz jego błędu predykcji w punkcie 8.3.

10.1.2.9. Dział 9 – nakład sumaryczny

Byłam współodpowiedzialna za przygotowanie danych potrzebnych do uzyskania wyników nakładu sumarycznego (Tabela 9.1).

10.1.2.10. Inne

- Pełniłam dużą rolę w redakcji punktów opisanych przez innych członków zespołu
- Dbałam o spójność punktów np. opisywanych modułów i zbiorów danych
- Dbałam o wygląd pracy (własnoręczne rysunki, diagramy i przykłady, formatowanie obrazków)

10.2. Jakub Kulaszewicz

10.2.1. Prace wykonane na poczet systemu

10.2.1.1. Przyrost pierwszy – dataset

Podczas pierwszego przyrostu byłem współodpowiedzialny za pozyskanie oraz weryfikację zbioru danych z filmami (pkt 7.2.2.1 i 7.2.2.4). Następnie przygotowałem skrypt w języku Python którego zadaniem była ekstrakcja klatek z plików wideo ze zbioru RWF-2000 (pkt 6.3.1.2) i zapis ich w danym formacie na lokalnej maszynie. Kolejno wyekstrahowany już zbiór klatek podałem procesowi manualnej segregacji na klatki, na

których występuje wyłącznie przemoc i na te na których jej kompletnie nie ma. Produktem tego procesu był RWF- 2000 Image Subset (pkt 6.3.1.2.1). Ostatnim krokiem wykonanym przez mnie w tym przyroście było przygotowanie aplikacji do synchronizacji Syncthing (pkt 6.2.3.3) i udostępnienie zbioru danych pozostałym członkom zespołu (pkt 7.2.2.10).

10.2.1.2. Przyrost drugi – implementacja

W tym przyroście zająłem się implementacją i przygotowaniem architektury modułu VRN opisanego w punkcie 6.4.2. Opis tego zadania oraz użyte technologie do jego wykonania zostały opisane w punkcie 7.3.2.2.

10.2.1.3. Przyrost trzeci – uczenie

W trakcie tego przyrostu zająłem się przygotowaniem środowiska na maszynie CloudLab2 do stanu, w którym były możliwość przeprowadzenia procesu nauczania dla różnych modułów, opisane zostało to w punkcie 6.5.1, a związane z tym problemy w punkcie 7.4.3.5. Po zakończeniu konfiguracji maszyny rozpocząłem proces wyuczania zaimplementowanego wcześniej modułu VRN. Do realizacji tego zadania została wykonana seria eksperymentów, która była walidowana wcześniej wyznaczonymi założeniami, opisane zostało to w punkcie 6.5.3. Natomiast sam proces i jego rezultaty został zredagowany w punkcie 7.4.2.2

10.2.1.4. Przyrost czwarty – integracja

W okresie trwania tego przyrostu brałem udział w dyskusjach dotyczących doboru technologicznego związanego z integracją. Również zająłem się przygotowaniem gotowego modułu VRN do stanu umożliwiającego jego integrację z całym systemem.

10.2.1.5. Przyrost piąty – optymalizacja i walidacja

Podczas trwania tego przyrostu zająłem się przetestowaniem modułu VRN. Testy te opierały się na podaniu wcześniej skompletowanych plików wideo procesowi predykcji w tempie 30 klatek na sekundę i zapisaniu ich kopii z nałożonymi wynikami predykcji w schemacie:

- Powyżej 0.5 wyniku predykcji przemocy -> Violence
- Równe i poniżej 0.5 wyniku predykcji przemocy -> NonViolence

Przykładowe wyniki tego testu można zobaczyć na Rysunek 7.6. Poza tym pomagałem przygotować zbiór filmików testowych potrzebnych do weryfikacji działania całego systemu opisanych w formie testów w punkcie 8.2.

10.2.1.6. Inne

Poza wyżej wspomnianą pracą pełniłem obowiązki organizacji pracy i zadań w narzędziu YouTrack (pkt 6.2.3.2).

10.2.2. Prace wykonane na poczet dokumentacji

10.2.2.1. Dział 1 – wstęp

Brałem udział w pisaniu wstępu do pracy.

10.2.2.2. Dział 2 – słownik pojęć

Współtworzyłem słownik pojęć podczas pracy nad dokumentacją.

10.2.2.3. Dział 3 – analiza problemu

W tym dziale byłem współodpowiedzialny za sformułowanie rozwiązań konkurencyjnych (pkt 3.3).

10.2.2.4. Dział 4 – planowanie

W ramach realizacji tego działu byłem współodpowiedzialny za sformułowanie charakterystyki klientów (pkt 4.4) oraz komercjalizacji projektu (pkt 4.6)

10.2.2.5. Dział 5 – analiza wymagań

Mój udział w tym dziale był w trakcie formułowania wymagań ogólnych i dziedzinowych (pkt 5.1) oraz wymagań funkcjonalnych (pkt 5.2)

10.2.2.6. Dział 6 – projektowanie

W procesie opisywania działu 6 zająłem się definiowaniem punktów 6.1.1 i 6.1.2 oraz stworzeniem schematu głównej pętli naszego systemu (Rysunek 6.3). Również w ramach punktu 6.1.1 wytworzyłem diagram wdrożenia widoczny na Rysunek 6.1. Następnie zdefiniowałem RWF-2000 Image Subset w punkcie 6.3.1.2.1, opisałem punkt definiujący moduł VRN (pkt 6.4.2) i jego proces nauczania w punkcie 6.5.3.

10.2.2.7. Dział 7 – realizacja projektu

W tym dziale opisałem zadania związane z modulem VRN (pkt 7.3.2.2 oraz 7.4.2.2). Poza tym opisałem napotkany problem związany z maszyną CloudLab2 (pkt 7.4.3.5). Ostatnią rzeczą sformułowaną przez mnie w tym dziale było opisanie procesu kalibracji wartości wag (pkt 7.6.2.1).

10.2.2.8. Dział 8 – walidacja rozwiązania

Współuczestniczyłem w zbieraniu danych testowych a uzyskane wyniki poddałem analizie w punkcie 8.4.

10.2.2.9. Dział 9 – nakład sumaryczny

Byłem współodpowiedzialny za przygotowanie danych potrzebnych do uzyskania wyników w Tabeli 9.1. Opisałem metody sformułowania nakładu sumarycznego (pkt 9.1.1 i 9.1.2)

10.2.2.10. Inne

Brałem czynny udział w procesie recenzji napływających zmian w całym dokumencie wygenerowanych przez innych członków zespołu.

10.3. Benedykt Kościński

10.3.1. Prace wykonane na poczet systemu

10.3.1.1. Przyrost pierwszy – dataset

Byłem odpowiedzialny za poszukiwanie i pozyskanie zbioru 150 nagrań z aktami przemocy, który został ostatecznie wykorzystany do walidacji systemu (pkt 8.2) oraz współuczestniczyłem w pozyskaniu zbioru RWF-2000 (pkt 7.2.2.1).

10.3.1.2. Przyrost drugi – implementacja

Podczas implementacji skupiałem się na poznaniu sposobu działania modeli oraz metod transformacji danych. Ta wiedza okazała się wyjątkowo przydatna na etapie tworzenia integracji i tworzenia API (pkt 6.6).

10.3.1.3. Przyrost trzeci – uczenie

W trakcie przyrostu trzeciego wspierałem zespół merytorycznie. Pomogłem członkom zespołu, odpowiedzialnym za uczenie sieci skonfigurować i zainstalować odpowiednie narzędzia do uczenia z wykorzystaniem karty graficznej.

Prowadziłem rozpoznanie wśród optymalnych rozwiązań sprzętowych do nauczania sieci neuronowych. Wynikiem, którego były dwie kwarty graficzne Nvidia GTX 2080 IT oraz RTX 3070. Szczególnie ta druga wypadła bardzo dobrze w kategorii efektywności/cena. Po tym dostaliśmy dostęp do maszyny na platformie CloudLab2 (pkt 6.5.1).

10.3.1.4. Przyrost czwarty – integracja

Integracja wszystkich modułów w jeden system było moim głównym zadaniem.

- Rozpoznanie wśród narzędzi do tworzenia API i wybranie najbardziej adekwatnego do naszej sytuacji – ze względu na stos technologiczny zespół zdecydowaliśmy się na FastAPI, bardzo lekki i zarazem elastyczny framework (pkt 6.6).
- Przygotowanie uniwersalnego API do serwowania modeli.
- Konteneryzacja modułów serwujących modele przez API.
- Ponowne rozpoznanie wśród produkcyjnych metod serwowania modeli sztucznej inteligencji – TensorFlow Serving (pkt 6.7)
- Wdrożenie TensorFlow Serving oraz gRPC jako ostatecznego rozwiązania problemu udostępniania modeli (pkt 6.7 i 7.5.3.2).
- Optymalizacja procesu transformacji danych oraz przekazywani ich do analizy (pkt 7.5.3.3 i 7.5.3.4).
- Ostateczna integracja całego systemu z wykorzystaniem Docker Compose i stworzenie pliku konfiguracyjnego (pkt 7.5.2.3).

10.3.1.5. Przyrost piąty – optymalizacja i walidacja

Przygotowałem środowisko testowe do nadawania sygnału wideo, w sposób naśladujący oryginalny sygnał z kamery. Uczestniczyłem w zbieraniu filmów oraz przeprowadzałem testy opisane w punkcie 8.2. Do nadawania filmów w celach testowych wykorzystałem program OBS Studio [48].

10.3.2. Prace wykonane na poczet dokumentacji

10.3.2.1. Dział 2 – słownik pojęć

Współtworzyłem słownik pojęć podczas pracy nad dokumentacją.

10.3.2.2. Dział 5 – analiza wymagań

Zajmowałem się współtworzeniem wymagań funkcjonalnych i нефункциональных, które dotyczyły API, systemu logów i integralności całego systemu.

10.3.2.3. Dział 6 – projektowanie

Współtworzyłem i redagowałem tekst opisujący architekturę oraz działanie systemu (pkt 6.1.1 i 6.1.2).

Stworzyłem opis:

- procedury zbierania klatek oraz ich późniejszego rozsyłania (pkt 6.1.2.1 i 6.1.2.2)
- stosu technologicznego oraz narzędzi wykorzystanych podczas procesu wytwarzania systemu (pkt 6.2),
- niefortunnego epizodu powstawiania API w oparciu o FastAPI. Wady i problemy jakie napotkaliśmy oraz sposoby na ich rozwiązanie (pkt 6.6),
- implementacji oraz wszystkich zalet TensorFlow Serving (pkt 6.7).

10.3.2.4. Dział 7 – realizacja projektu

Opisałem etap integracji wszystkich modułów w jeden system, napotkane problemy, sposoby na ich rozwiązanie i etap konteneryzacji razem z przygotowaniem systemu do wdrożenia na środowisku produkcyjnym (pkt 7.5).

10.3.2.5. Dział 8 – walidacja rozwiązania

Współuczestniczyłem w zbieraniu danych testowych (pkt 8.2)

10.3.2.6. Inne

Byłem odpowiedzialny za rozpoznanie wśród standardów formatowania prac dyplomowych i zaprojektowanie pierwotnego formatu dokumentacji. Dodałem

predefiniowane style, których używamy do szybkiego formatowania tekstu. Miałem duży udział w redagowaniu tekstu oraz pootrzymywania standardów pracy.

10.4. Mateusz Chodyna

10.4.1. Prace wykonane na poczet systemu

10.4.1.1. Przyrost pierwszy – dataset

Byłem odpowiedzialny za pozyskanie i weryfikację zbioru danych zdjęć zawierających niebezpieczne przedmioty co zostało opisanych w punkcie 6.4.3.

10.4.1.2. Przyrost drugi – implementacja

W tej części projektu zająłem się zapoznaniem z algorytmem YOLOv3, w celu jego późniejszego użycia. Dodatkowo pozyskiwałem informacje potrzebne w procesie detekcji. Zostało to opisane w punkcie 7.4.2.3.

10.4.1.3. Przyrost trzeci – uczenie

Podczas przyrostu trzeciego skupiłem się na wyuczeniu modułu DIDN i jego translacji. Proces ten został opisany w punkcie

10.4.1.4. Przyrost czwarty – integracja

W trakcie trwania tego przyrostu asystowałem oraz pomagałem w integracji wykonanego przeze mnie modułu.

10.4.1.5. Przyrost piąty – optymalizacja i walidacja

W tym przyroście skupiłem się głównie na pomaganiu w procesie walidacji oraz analizie wyników.

10.4.2. Prace wykonane na poczet dokumentacji

10.4.2.1. Dział 2 – słownik pojęć

Współtworzyłem słownik pojęć podczas pracy nad dokumentacją opisany w rozdziale 2.

10.4.2.2. Dział 3 – analiza problemu

W tym punkcie byłem odpowiedzialny za przedstawienie propozycji rozwiązania opisane w punkcie 3.4.

10.4.2.3. Dział 4 – planowanie

W czasie realizacji tego rozdziału określiłem cele projektu oraz jego spodziewane produkty, punkty 4.1 i 4.2. Odpowiedzialny byłem również za opisanie sposobu komercjalizacji (pkt 4.6), ograniczeń (pkt 4.7) jak i strategii wytwarzania wykorzystywanej w tym projekcie (pkt 4.8).

10.4.2.4. Dział 5 – analiza wymagań

Podczas pracy nad tym rozdziałem odpowiedzialny byłem za uzupełnienie punktów 5.4 i 5.5 opisujących wymagania dotyczące procesu wytwarzania jak i wymagania jakościowe.

10.4.2.5. Dział 6 – projektowanie

W trakcie pracy nad działem szóstym, opisałem pozyskiwanie datasetu ze zdjęciami w punkcie 6.3.2, strukturę sieci DIDN w punkcie 6.4.3 oraz jej proces nauczania w punkcie 6.5.4

10.4.2.6. Dział 7 – realizacja projektu

Podczas uzupełniania tego rozdziału opisałem przyrost pierwszy dotyczący datasetu 7.2. Podjąłem się również opisanie swoich zadań w punktach 7.3 oraz 7.4.

10.4.2.7. Dział 9 – nakład sumaryczny

Dane informujące o czasie spędzonym całego zespołu nad danym zadaniem przedstawiłem na diagramie Gantta (Rysunek 9.1).

10.4.2.8. Inne

- Pomagałem w procesie redakcji innych części dokumentacji

11. Bibliografia

- [1] "Artificial Intelligence's Growth," [Online]. Available: forbes.com/sites/louiscolumnbus/2018/01/12/10-charts-that-will-change-your-perspective-on-artificial-intelligences-growth/?sh=4edacfd94758.
- [2] "Bureau of Justice Statistics," [Online]. Available: <https://www.bjs.gov/>.
- [3] "Abto Software," [Online]. Available: <https://www.abtosoftware.com/>.
- [4] "Violence Detection for Smart Surveillance Systems," [Online]. Available: <https://www.abtosoftware.com/blog/violence-detection>.
- [5] O. Deniz, I. Serrano Gracia, G. Bueno and T.-T. Kim, "Fast violence detection in video," in *The 9th International Conference on Computer Vision Theory and Applications*, 2014.
- [6] P. Zhou, Q. Ding and X. Hou, "Violent Interaction Detection in Video Based on Deep Learning," 2017.
- [7] "Google Ads," [Online]. Available: <https://ads.google.com/home/>.
- [8] T. D. Nils Brede Moe, "Scrum and Team Effectiveness: Theory and Practice," in *Agile Processes in Software Engineering and Extreme Programming*, Limerick, 2008.
- [9] C. Olston, N. Fiedel, K. Gorovoy, J. Harmsen, L. Lao, F. Li, V. Rajashekhar, S. Ramesh and J. Soyke, "TensorFlow-Serving: Flexible, High-Performance ML Serving," 2017.
- [10] "Seq documentation," [Online]. Available: <https://docs.datalust.co/docs>.
- [11] "YouTrack documentation," [Online]. Available: <https://www.jetbrains.com/youtrack/documentation/>.
- [12] "Syncthing documentation," [Online]. Available: <https://docs.syncthing.net/pdf/FAQ.pdf>.

- [13] M. Cheng, K. Cai and M. Li, "RWF-2000: An Open Large Scale Video Database for Violence Detection," 2019.
- [14] "Optical Flow with OpenCV," [Online]. Available: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html.
- [15] "Optical Flow Estimation - methods comparison," [Online]. Available: <https://paperswithcode.com/task/optical-flow-estimation>.
- [16] Z. Teed and J. Deng, "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow," 2020.
- [17] "Real Life Violence Situations Dataset," [Online]. Available: <https://www.kaggle.com/mohamedmustafa/real-life-violence-situations-dataset>.
- [18] "UCF-ARG Dataset," [Online]. Available: <https://www.crcv.ucf.edu/research/datasets/ucf-arg/>.
- [19] "NTU CCTV-Fights Dataset," [Online]. Available: <http://rose1.ntu.edu.sg/datasets/cctvFights.asp>.
- [20] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, K. Ivan, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallocci, A. Kolesnikov, T. Duerig and V. Ferrari, "The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale," 2020.
- [21] "AudioSet - A large-scale dataset of manually annotated audio events," [Online]. Available: <https://research.google.com/audioset/>.
- [22] "Librosa documentation," [Online]. Available: <https://librosa.org/doc/latest/index.html>.
- [23] G. Farneback, "Two-Frame Motion Estimation Based on Polynomial Expansion," in *SCIA*, 2003.
- [24] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *ICLR*, 2015.

- [25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," 2015.
- [26] S. Hochreiter and u. Schmidhuber, "Long Short-Term Memory," 1997.
- [27] "VGG16 – Convolutional Network for Classification and Detection," [Online]. Available: <https://neurohive.io/en/popular-networks/vgg16/>.
- [28] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [29] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018.
- [30] L. Wyse, "Audio spectrogram representations for processing with Convolutional Neural Networks," 2017.
- [31] "Spectrogram - Wikipedia," [Online]. Available: <https://en.wikipedia.org/wiki/Spectrogram>.
- [32] "Keras Callbacks," [Online]. Available: <https://keras.io/api/callbacks/>.
- [33] "YOLO to Tensorflow model," [Online]. Available: <https://github.com/hunglc007/tensorflow-yolov4-tflite>.
- [34] R. Aslanzadeh, K. Qazanfari and M. Rahmati, "An Efficient Evolutionary Based Method For Image Segmentation," 2017.
- [35] Q. Li, X. Bai and W. Liu, "Skeletonization of gray-scale image from incomplete boundaries," in *ICIP*, 2008.
- [36] S. Saito, T. Simon, J. Saragih and H. Joo, "Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization," in *CVPR*, 2020.
- [37] A. Datta, M. Shah and N. Da Vitoria Lobo, "Person-on-Person Violence Detection in Video Data," 2003.

- [38] M. Ramzan, A. Abid, H. Ullah Khan, S. Mahmood Awan, A. Ismail, M. Ahmed, M. Ilyas and A. Mahmood, "A Review on State-of-the-Art Violence Detection Techniques," 2019.
- [39] "Github RWF-2000 project," [Online]. Available: <https://github.com/mchengny/RWF2000-Video-Database-for-Violence-Detection>.
- [40] D. Kondermann, "A Toolbox to Visualize Dense Image Correspondences," [Online]. Available: https://hci.iwr.uni-heidelberg.de/Correspondence_Visualization.
- [41] "Keras Image Segmentation," [Online]. Available: <https://divamgupta.com/image-segmentation/2019/06/06/deep-learning-semantic-segmentation-keras.html>.
- [42] "Numpy documentation," [Online]. Available: <https://numpy.org/doc/>.
- [43] "Matplotlib documentation," [Online]. Available: <https://devdocs.io/matplotlib~3.1/>.
- [44] "PyTorch documentation," [Online]. Available: <https://pytorch.org/docs/stable/index.html>.
- [45] "Sklearn documentation," [Online]. Available: <https://scikit-learn.org/0.21/documentation.html>.
- [46] "Skimage documentation," [Online]. Available: <https://scikit-image.org/docs/0.14.x/>.
- [47] J. Palermo, "The Onion Architecture : part 1 | Programming with Palermo," 29 7 2008. [Online]. Available: <https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/>.
- [48] "Open Broadcaster Software," [Online]. Available: <https://obsproject.com>.

12. Spis rysunków

Rysunek 3.1 Tabela udostępniona przez Bureau of Justice Statistics [2], przedstawiająca wykaz liczbowy oraz procentowy aktów przemocy z podziałem na lokalizacje w Stanach Zjednoczonych w latach 2004-2008	13
Rysunek 3.2 Rich picture – ogólne przedstawienie problemu.....	14
Rysunek 3.3 Rich picture – perspektywa ochroniarza.....	15
Rysunek 3.4 Rich picture – efekty korzystania z systemu	16
Rysunek 4.1 Diagram przypadków użycia przedstawiający schemat wykorzystania systemu	24
Rysunek 4.2 Diagram przyjętego przez nas modelu wytwarzania systemu	29
Rysunek 6.1 Diagram wdrożenia przedstawiający architekturę systemu	40
Rysunek 6.2 Zrzut ekranu przykładowych logów systemu Seq	41
Rysunek 6.3 Schemat blokowy procesu głównej pętli systemu	42
Rysunek 6.4 Diagram sekwencji.....	43
Rysunek 6.5 Diagram czynności.....	44
Rysunek 6.6 Wykres funkcji finalnej klasyfikacji	45
Rysunek 6.7 Zestaw kilku wyciętych scen walki z plików wideo zawierających się w RWF-2000 Dataset.....	50
Rysunek 6.8 Przebieg pozyskiwania kolekcji danych RWF-2000 [5]	51
Rysunek 6.9 Przykłady klatek znajdujących się w RWF-2000 OpenCV Subset	52
Rysunek 6.10 Przykład klatek znajdujących się w RWF-2000 RAFT Subset	53
Rysunek 6.11 Przykładowe zdjęcia zawarte w Open Images Dataset.....	56
Rysunek 6.12 Przykład wygenerowanych spektrogramów	60
Rysunek 6.13 Struktura sieci FGN [13].....	60
Rysunek 6.14 Strategia przycinania przy użyciu gęstego Optical Flow [13]	62
Rysunek 6.15 Architektura modułu VRN [27]	63
Rysunek 6.16 Architektura sieci algorytmu YOLOv3 [29].....	65
Rysunek 6.17 Architektura sieci VGG16 wykorzystana do modułu DSDN [27]	66
Rysunek 6.18 Przykład spektrogramu [31].....	66
Rysunek 6.19 Wykres przedstawiający przebieg uczenia sieci FGN w pierwszym eksperymencie.....	69

Rysunek 6.20 Wykres przedstawiający przebieg uczenia sieci FGN w drugim eksperymencie	70
Rysunek 6.21 Wykres przedstawiający przebieg uczenia sieci FGN w trzecim eksperymencie	72
Rysunek 6.22 Wykresy przedstawiające proces nauczania sieci VGG16 podczas pierwszego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji.	74
Rysunek 6.23 Wykresy przedstawiające proces nauczania sieci VGG16 podczas drugiego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji.....	76
Rysunek 6.24 Wykresy przedstawiające proces nauczania sieci VGG16 podczas trzeciego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji.....	77
Rysunek 6.25 Wykres przedstawiający proces nauczania sieci VGG16 + LSTM podczas pierwszego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji.	79
Rysunek 6.26 Wykresy przedstawiające proces nauczania sieci VGG16 + LSTM podczas drugiego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji. ..	80
Rysunek 6.27 Wykresy przedstawiające proces nauczania sieci VGG16 + LSTM podczas trzeciego eksperymentu. Pomarańcz oznacza proces trenowania, niebieski walidacji ...	82
Rysunek 6.28 Wykres przedstawiający przebieg uczenia sieci DSDN w pierwszym eksperymencie	85
Rysunek 6.29 Wizualizacja porzuconej techniki usuwania tła na zdjęciu ze zbioru RWF-2000 Image Subset	89
Rysunek 7.1 Korespondencja między właścicielem RWF-2000 Dataset a członkami zespołu.....	93
Rysunek 7.2 Legenda do wizualizacji optycznej pokazująca jak kierunek jest odwzorowywany na kolor, a wielkość na nasycenie [40]	95
Rysunek 7.3 Przykłady niedokładnej transformacji klatek przy użyciu metody Optical Flow za pomocą technologii OpenCV	96
Rysunek 7.4 Przykład segmentacji klatki.....	97
Rysunek 7.5 Przykładowe predykcje końcowej wersji modułu FGN na filmach pozyskanych z serwisu YouTube	101
Rysunek 7.6 Obrazy z nagrań wideo przetworzone i sklasyfikowane przez finalną wersję modułu VRN. Nagrania pozyskane z serwisu LiveLeak i YouTube	102

Rysunek 7.7 Zrzut ekranu z Docker przedstawiający wynik uruchomienia systemu pliku Docker Compose.....	105
Rysunek 7.8 Schemat predykcji rozłożony w czasie z podejściem wielowątkowym ..	107
Rysunek 9.1 Diagram Gantta.....	125

13. Spis tabel

Tabela 4.1 Charakterystyka klientów wewnętrznych oraz ich potrzeb	22
Tabela 4.2 Charakterystyka klientów zewnętrznych oraz ich potrzeb	23
Tabela 6.1 Przedstawienie głównych wymagań dotyczących zbioru danych z filmami	49
Tabela 6.2 Podział plików wideo w zależności od rodzaju kamery w zbiorze danych NTU CCTV-Fights [19]	55
Tabela 6.3 Przedstawienie głównych wymagań dotyczących zbioru zdjęć	56
Tabela 6.4 Przedstawienie głównych wymagań dotyczących zbioru plików audio	58
Tabela 6.5 Parametry architektury modelu FGN	61
Tabela 6.6 Hiperparametry w pierwszym eksperymencie sieci FGN	68
Tabela 6.7 Hiperparametry w drugim eksperymencie sieci FGN	70
Tabela 6.8 Hiperparametry w trzecim eksperymencie sieci FGN.....	71
Tabela 6.9 Tabela przedstawiająca dobór hiperparametrów do pierwszego eksperymentu VGG16	74
Tabela 6.10 Tabela przedstawiająca dobór hiperparametrów do drugiego eksperymentu VGG16	75
Tabela 6.11 Tabela przedstawiająca dobór hiperparametrów do trzeciego eksperymentu VGG16	77
Tabela 6.12 Tabela przedstawiająca dobór hiperparametrów do pierwszego eksperymentu VGG16 + LSTM	78
Tabela 6.13 Tabela przedstawiająca dobór hiperparametrów do drugiego eksperymentu VGG16 + LSTM	80
Tabela 6.14 Tabela przedstawiająca dobór hiperparametrów do trzeciego eksperymentu VGG16 + LSTM	81
Tabela 6.15 Hiperparametry w pierwszym eksperymencie sieci DSDN	85
Tabela 7.1 Tabela przedstawiająca zestaw wartości dla pierwszego eksperymentu kalibracji wartości wag.....	108
Tabela 7.2 Tabela przedstawiająca zestaw wartości dla drugiego eksperymentu kalibracji wartości wag.....	108
Tabela 7.3 Tabela przedstawiająca zestaw wartości dla trzeciego eksperymentu kalibracji wartości wag.....	109

Tabela 7.4 Tabela przedstawiająca zestaw wartości dla czwartego eksperymentu kalibracji wartości wag	109
Tabela 7.5 Tabela przedstawiająca zestaw wartości dla piątego eksperymentu kalibracji wartości wag	109
Tabela 8.1 Wyniki predykcji w teście nr 1	114
Tabela 8.2 Wyniki predykcji w teście nr 2	114
Tabela 8.3 Wyniki predykcji w teście nr 3	115
Tabela 8.4 Wyniki predykcji w teście nr 4	115
Tabela 8.5 Wyniki predykcji w teście nr 5	116
Tabela 8.6 Wyniki predykcji w teście nr 6	116
Tabela 8.7 Wyniki predykcji w teście nr 7	117
Tabela 8.8 Wyniki predykcji w teście nr 7	117
Tabela 8.9 Wyniki predykcji w teście nr 9	118
Tabela 8.10 Wyniki predykcji w teście nr 10	118
Tabela 8.11 Wyniki predykcji w teście nr 11	119
Tabela 8.12 Wyniki predykcji w teście nr 12	119
Tabela 8.13 Wyniki średnich predykcji w testach błędu predykcji	121
Tabela 9.1 Nakład sumaryczny przedstawiony w formie tabeli	124

14. Spis równań

Równanie 1 Funkcja finalnej klasyfikacji systemu VRS	45
Równanie 2 Średni błąd predykcji systemu VRS	112
Równanie 3 Normalizacja wyników predykcji	113
Równanie 4 Przybliżona dokładność predykcji systemu VRS	113

15. Załączniki

Dołączona do dokumentacji płyta CD zawiera następującą strukturę folderów:

- „doc” – dokumentacja
 - „Prezentacja.pdf” – prezentacja systemu
 - „Praca_dyplomowa_VRS.pdf” – dokumentacja systemu
- „src” – kod źródłowy systemu VRN
 - „Module.DIDN” – sieć wykrywająca broń na obrazach, YOLOv3
 - „Module.DSDN” – sieć klasyfikująca audio
 - „Module.FGN” – sieć wykrywająca przemoc, Flow Gate Network
 - „Module.Main” – główny moduł koordynujący pracę systemu
 - „Module.VRN” – sieć wykrywająca przemoc, Violence Recognition Network + sieć VGG16
- „tools” – zewnętrzne narzędzi wykorzystane podczas powstawania projektu
- „docker-compose.yml” – plik konfiguracyjny, który pozwala na zbudowanie i uruchomienie kontenerów