Raport z badania antyplagiatowego

szczegółowy

1 Próba | niski Poziom podobieństwa | Wynik zaakceptowany przez promotora

Spis treści

Metryka

Tytuł pracy	**Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego**
Zbadaj przetłumaczony tekst	Nie
Analiza SI	Nie

	Imię i Nazwisko	Numer albumu	Typ pracy	Jednostka lub instytucja	Kierunek studiów
Autorzy pracy	Damian Jamroży	113729	magisterska	Nauk Przyrodniczych	Informatyka

	Imię i Nazwisko	Instytucja		Imię i Nazwisko	Instytucja
Promotorzy	dr Bogusław Twaróg	Uniwersytet Rzeszowski	Recenzenci	dr hab. Jan Bazan	Uniwersytet Rzeszowski

Badane pliki	2024_IIstInfSR_113729_p.pdf
Grupa ustawień badania	Kolegium Nauk Przyrodniczych

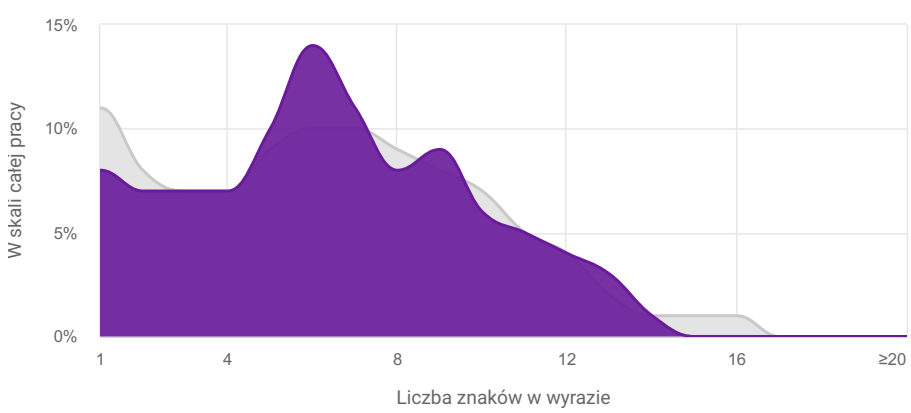
Numer badania	2870056	Numer próby	2958558	Przekazano próbę	13.09.2024, 11:29:08

Analiza tekstu

Statystyka

83 739	Znaki
5	Znaki specjalne lub spoza języka pracy
0.01%	względem wszystkich znaków
10 553	Wyrazy
706	Nierozpoznane wyrazy
6.69%	względem wszystkich wyrazów
46	Elementy graficzne
2	Fragmenty innego stylu

Rozkład długości wyrazów



Wyniki ogólne

≥ 40 wyrazów we frazie	≥ 20 wyrazów we frazie	≥ 10 wyrazów we frazie	≥ 5 wyrazów we frazie
2 fraz	2 fraz	2 fraz	2 fraz
2% PRP oryginalny	2% PRP oryginalny	2% PRP oryginalny	2% PRP oryginalny
		Wynik wiodący	

Wyniki szczegółowe

Nr	Referencyjna baza porównawcza	PRP dla fraz o zadanej długości ≥ 40	≥ 20	≥ 10	≥ 5
1	Akty prawne	0%	0%	0%	0%
2	Internet	0%	0%	0%	0%
3	ORPPD i BPA	2%	2%	2%	2%

Źródła wykrytych podobieństw

Nr	Tytuł lub adres dokumentu	Najdłuższa fraza	Liczba znalezionych fraz o zadanej długości ≥ 40	≥ 20	≥ 10	≥ 5

3.1	Metoda rozpoznawania języka migowego z obrazu	1813	1	1	**1**	1
3.2	Rozpoznawanie języka migowego z wykorzystaniem biblioteki M ediapipe i głębokiego uczenia maszynowego	503	1	1	**1**	1

4	**Baza instytucji**	0%	0%	**0%**	0%

Tekst pracy

Podobieństwa

1 Akty prawne	2 Internet	3 ORPPD i BPA	4 Baza instytucji
0 znalezionych	0 znalezionych	2 znalezionych	0 znalezionych

UNIWERSYTET RZESZOWSKI Kolegium Nauk Przyrodniczych Damian Jamroży Nr albumu: 113729 Kierunek: Informatyka Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego Praca magisterska Praca wykonana pod kierunkiem Dr. Inż. Bogusława Twaroga Rzeszów, 2024 Pragnę serdecznie podziękować Panu dr inż. Bogusławowi Twarogowi za nieocenione wsparcie oraz życzliwość okazywaną na każdym etapie mojej edukacji. Jego pomoc i zaangażowanie miały kluczowy wpływ nie tylko na proces powstawania niniejszej pracy magisterskiej, ale także na moją pracę inżynierską i cały tok studiów. Bez Jego merytorycznej wiedzy oraz wsparcia, ukończenie tych etapów byłoby znacznie trudniejsze. Jednocześnie chciałbym serdecznie podziękować moim przyjaciołom z UCI UR. To dzięki ich namowom i wsparciu udało mi się zrealizować ww. etap życia. Spis treści Wstęp 7 Cel i zakres pracy 8 1. Przegląd literatury i analiza istniejących rozwiązań 9 1.1. Przegląd technologii rozpoznawania gestów 9 1.2. Przegląd metod uczenia maszynowego 12 1.3. Wyzwania w rozpoznawaniu gestów języka migowego 13 1.4. Analiza problematyki oraz istniejących aplikacji dotyczących języka migowego 15 2. Techniczne aspekty funkcjonowania aplikacji 17 2.1. Wymagania systemowe 17 2.2. Narzędzia modelowania sztucznej inteligencji 19 3. Budowa modelu rozpoznawania gestów 21 3.1. Przygotowanie danych wejściowych 21 3.2. Preprocessing danych 24 3.3. Architektura modelu oraz algorytm klasyfikacji 26 3.4. Trening, walidacja oraz ocena skuteczności modelu 27 4. Implementacja platformy 32 4.1. Koncepcja wizualna platformy 32 4.2. Mechanizmy komunikacji między frontendem a backendem 33 4.2.1. Komunikacja serwer – użytkownik 33 4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) 34 4.3. Widoki aplikacji 34 5. Ocena działania platformy 48 5.1. Testy w rzeczywistych warunkach 48 5.1.1. Ekstrakcja punktów kluczowych 48 5.1.2. Uczenie modelu 50 .1.3. Testy w czasie rzeczywistym 52 5.2. Skuteczność i dokładność rozpoznawania gestów 53 5.3. Analiza błędów i propozycje ulepszeń 61 Podsumowanie 62 Bibliografia oraz Netografia 64 Spis ilustracji, tabel oraz wykresów 66 Wstęp Obecny rozwój technologii pozwala na automatyzację wielu procesów, w tym procesów finansowych, administracyjnych oraz sprzedażowych. Wymienione elementy są związane z obsługą biznesów, które umożliwiają komercyjny zarobek twórcom oprogramowania. Algorytmy, mają w tym przypadku ułatwić pracę lub całkowicie zastąpić osoby fizyczne w ich obowiązkach. Dzięki takim praktykom, pracodawcy, czy też różnego rodzaju organizacje, zwiększają swoje dochody poprzez przyspieszenie wykonywanej pracy lub w gorszym przypadku, oszczędzają fundusze poprzez zredukowanie etatów. Szerokie zastosowanie sztucznej inteligencji jest widoczne w każdym aspekcie naszego życia. Coraz większa ilość sklepów, banków czy też producentów wszelkiego rodzaju produktów, decyduje się na wdrażanie sztucznej inteligencji. Zgodnie z opinią specjalistów z Wyższej Szkoły Biznesu National-Louis University, AI (Artificial Intelligence) może powodować utratę miejsc pracy oraz restrukturyzację zawodów, jednakże tym samym może zwiększać zapotrzebowanie na specjalistów w branży kreatywnej oraz IT. Autor artykułu, zwraca uwagę na problematykę dotyczącą etyki związanej ze sztuczną inteligencją oraz potrzebę nieustannej nauki i rozwoju[12]. Istnieją również aspekty SI (Sztucznej Inteligencji), które są niezaprzeczalnie pozytywne, chociażby zastosowane w strefach pożytku publicznego, czy też w rozwiązaniach dla osób z dysfunkcjami. Wszelkiego rodzaju protezy, pojazdy, syntezatory mowy, algorytmy analizujące tekst, dźwięk czy też obraz, pozwalają na łatwiejsze funkcjonowanie osób niepełnosprawnych. Niestety obszary te są często pomijane, ze względu na stosunkowo niewielkie grono odbiorców, gotowych zapłacić za wprowadzenie takowych rozwiązań. Podejmując dalszą próbę analizy problemu, możemy zaobserwować wysokie zainteresowanie wadami wzroku oraz problemami ruchowymi, natomiast niskie zainteresowanie dysfunkcją głosową w odniesieniu do osób głuchoniemych. Istnieje wiele rozwiązań, które ułatwiają kontakt wzrokowy, chociażby takie jak regulacja wielkości czcionek we

wszelkiego rodzaju aplikacjach, asystenci głosowi, czy też operacyjne korekty wzroku. Funkcje ruchowe, wspierane są przez różnorodne protezy, pojazdy, a także specjalne miejsca dostosowane do ich potrzeb np. miejsca parkingowe. Niestety w odniesieniu do problematyki osób głuchoniemych, nie ma zbyt wielu rozwiązań technologicznych, które mogą ułatwić ich życie w sposób nieinwazyjny. Cel i zakres pracy Celem pracy dyplomowej jest wparcie ww. grupy docelowej poprzez utworzenie oraz analizę oprogramowania do rozpoznawania sekwencji ruchów w czasie rzeczywistym, w oparciu o metody uczenia maszynowego. Zastosowane rozwiązania, wykorzystane zostaną następnie w aplikacji służącej do nauki oraz tłumaczenia polskiego języka migowego. Oprogramowanie to może służyć jako wsparcie osób z dysfunkcjami w łatwiejszej adaptacji w środowisku osób pełnosprawnych. Aplikacja ta, również może działać w sposób edukacyjny, zwiększając świadomość użytkowników na temat dysfunkcji słuchowych oraz głosowych. Opracowanie autorskich skryptów, pozwalać ma na obsługę pełnego procesu pracy aplikacji, począwszy od rejestrowania próbek nagrań wideo, uczenia modelu SI, testowaniu jego poprawności, a kończąc na obsłudze aplikacji internetowej[4]. Niniejsza dysertacja, składa się z pięciu rozdziałów. Pierwszy z nich omawia zagadnienia teoretyczne związane z funkcjonowaniem aplikacji, takie jak przegląd dostępnych technologii związanych z rozpoznawaniem gestów oraz uczeniem maszynowym, wyzwaniami w rozpoznawaniu gestów oraz analizą istniejących platform dotyczących języka migowego. Następny rozdział omawia aspekty techniczne, które muszą zostać spełnione aby aplikacja działała w pełni poprawnie. W powyższym rozdziale znajdują się takie elementy jak: wymagania systemowe, zalecane oprogramowanie zewnętrzne oraz biblioteki, które należy zainstalować na urządzeniu docelowym. Rozdział trzeci porusza tematykę tworzenia modelu sztucznej inteligencji, przechodząc po każdym etapie jego powstawania, zaczynając od przygotowaniu danych wejściowych, a kończąc na ocenie skuteczności stworzonego modelu. Rozdział czwarty, zatytułowany „Implementacja platformy" skupia się na połączeniu aspektów sztucznej inteligencji wraz z aplikacją internetową. Zaprezentowane tam informacje dotyczą koncepcji wizualnej platformy, integracji modeli AI z aplikacją webową, mechanizmów komunikacji między klientem a serwerem oraz warstwy graficznej interfejsu użytkownika. Piąty, a zarazem ostatni rozdział niniejszej dysertacji omawia ocenę działania platformy, skupiając się na testach w warunkach rzeczywistych oraz symulowanych, ocenie skuteczności pracy aplikacji oraz propozycji ulepszeń oprogramowania. Praca zakończona została podsumowaniem, w którym zaprezentowano ogólny opis aplikacji wraz z jej analizą pod kątem przydatności oraz wydajności w rzeczywistych warunkach. 1. Przegląd literatury i analiza istniejących rozwiązań Rozpoznawanie wszelkiego rodzaju obiektów oraz ich właściwości, zyskały w ostatnich latach na popularności, głównie dzięki postępom w technikach uczenia maszynowego oraz rozwoju głębokich sieci neuronowych. W niniejszym rozdziale, omówione zostaną najważniejsze technologie oraz metody stosowane w tematyce rozpoznawania gestów, ze szczególnym uwzględnieniem języka migowego. 1.1. Przegląd technologii rozpoznawania gestów Technologie rozpoznawania gestów opierają się na stałej analizie ruchów ciała, ze szczególnym uwzględnieniem dłoni i palców, przy pomocy różnych sensorów oraz kamer. Systemy te wykorzystują zarówno dane wideo, jak i informacje trójwymiarowe, uzyskane za pomocą czujników głębi. Interdyscyplinarna dziedzina badań rozpoznawania gestów, łączy ze sobą elementy przetwarzania obrazów, komputerowego widzenia, sztucznej inteligencji oraz interakcji człowiek-komputer. Podstawą rozpoznawania gestów jest przetwarzanie obrazu, które obejmuje szereg technik służących do analizy danych wizualnych. Można wyłonić takie etapy jak: detekcja obrazów, śledzenie ruchu oraz ekstrakcja cech. Detekcja obrazów opiera się na rozpoznaniu dłoni oraz pozostałych istotnych części ciała na przetwarzanym obrazie. Techniki te bazują na algorytmach segmentacji obrazu, które identyfikują obiekty na podstawie ich koloru, kształtu lub ruchu. Algorytmy takie jak YOLO (You Only Look Once) oraz SSD (Single Shot Multibox Detector) są powszechnie stosowane do szybkiej i dokładnej detekcji dłoni w badanych obrazach. Śledzenie ruchu, opiera się na analizie każdej klatki nagrania w celu uchwycenia przesunięć, występujących w punktach wykrytych przez poprzedni proces. W tym celu stosuje się miedzy innymi algorytmy KLT (Kanade-Lucas-Tomasi) oraz Mean Shift, które pozwalają na dokładne monitorowanie trajektorii ruchu dłoni. Ostatnią techniką jest ekstrakcja cech, która umożliwia dokonanie wyodrębnienia z obrazów stawów oraz kości, niezbędnych do identyfikacji wykonanego gestu. W tym celu stosowane są takie narzędzia jak OpenPose oraz MediaPipe[3]. Biblioteka OpenPose została utworzona przez Perceptual Lab na Uniwersytecie Carnegie Mellon (CMU), pod nadzorem profesora Asuyuki Matsumoto oraz doktoranta Zhe Cao, który jest głównym autorem pracy badawczej, opisującej możliwości biblioteki OpenPose. Pierwsza wersja OpenPose została opublikowana w 2017 roku, a jej wyniki były oparte na pracy

naukowej pt. "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", której współautorem był wspomniany wcześniej Zhe Cao. Praca ta po raz pierwszy zaprezentowana została na konferencji CVPR (Computer Vision and Pattern Recognition) w 2017 roku. Początkowo OpenPose służył głównie do analizy oraz detekcji pozycji wielu osób w czasie rzeczywistym na obrazach 2D. Pierwsza wersja koncentrowała się na problemie, który dotąd stanowił duże wyzwanie, a mianowicie detekcji pozycji ciała wielu osób na jednym obrazie, w jak najszybszym czasie, z zachowaniem wysokiej precyzji. Istniejące wcześniej rozwiązania, były skoncentrowane głównie na wykryciu jednego człowieka lub działały wolniej, co uniemożliwiało zastosowanie ich w czasie rzeczywistym. Biblioteka ta zyskała spore zainteresowanie, co przyczyniło się do rozszerzenia jej o dodatkowe funkcje, takie jak detekcja szczegółowych ruchów dłoni i mimiki twarzy oraz analizy 3D. Głównym zadaniem obecnej wersji biblioteki jest wykrywanie oraz śledzenie pozycji ciała człowieka w obrazie 2D oraz 3D, uwzględniając różne części ciała takie jak kończyny, głowa oraz palce rąk i stóp. Podstawą działania OpenPose są techniki głębokiego uczenia, a dokładniej konwolucyjne sieci neuronowe (CNN), które za pomocą punktów kluczowych (tzw. keypoints), rozpoznają położenie ciała, dzieląc je na odpowiednie struktury. W początkowej fazie programu, biblioteka przeprowadza detekcję punktów kluczowych, następnie tworzy mapę ufności, która określa z jakim prawdopodobieństwem dany punkt (np. łokieć) znajduje się w określonej lokalizacji na obrazie. Utworzona mapa jest dwuwymiarową siatką, gdzie wartości w poszczególnych komórkach reprezentują stopień pewności. Po utworzeniu mapy ufności, biblioteka tworzy mapy części ciała, analizując położenie konkretnych elementów, co w dalszym etapie służy do utworzenia szkieletu osoby. W tym celu wykorzystywane są mapy PAF (Part Affinity Fields), czyli wektorowe pola, które wskazują, jak prawdopodobne jest, że dwa punkty kluczowe są ze sobą powiązane. Gdy mapy te zostaną utworzone, biblioteka przystępuje do połączenia punktów tworząc szkielet[2]. Druga spośród wymienionych bibliotek (MediaPipe) została udostępniona jako projekt open-source przez firmę Google w czerwcu 2019 roku. MediaPipe, jest to otwartoźródłowe środowisko, które umożliwia wykonywanie różnych zadań związanych z przetwarzaniem obrazów, wideo oraz danych w czasie rzeczywistym. Zbudowana została na podstawie struktury pipelinów przetwarzania multimodalnych danych, co oznacza, że pozwala na tworzenie i uruchamianie strumieniowych systemów analizy danych, w sposób modułowy oraz wydajny. Pipeline to zbiór skomponowanych kroków przetwarzania (tzw. modułów) w ustalonej kolejności. Każdy z modułów odpowiedzialny jest za określone zadanie np. detekcję obiektów, segmentację czy śledzenie ruchu. Dzięki temu, biblioteka ta jest bardzo elastyczna, gdyż użytkownik może z łatwością tworzyć własne pipeline'y, dostosowując je do specyficznych potrzeb. Pierwotnym założeniem MediaPipe było dostarczenie wydajnej biblioteki, która umożliwiałaby wieloplatformowe przetwarzanie strumieni wideo i obrazów w czasie rzeczywistym[6]. Przechodząc do kwestii hardware, najczęściej wykorzystywanymi urządzeniami do rejestracji gestów są kamery RGB. Ich popularność jest uargumentowana stosunkowo niską ceną oraz dostarczanym kolorowym obrazem w wysokiej rozdzielczości, który może być przetwarzany przez algorytmy komputerowego widzenia. Nie są to jednakże jedyne narzędzia, wykorzystywane w procesie rejestrowania ruchów. Elementy takie jak kamery głębi pozwalają na trójwymiarową rejestrację sceny, co umożliwia bardziej precyzyjną analizę ruchów dłoni i ciała. Dużą zaletą kamer głębi jest wysoka dokładność w trudnych warunkach oświetleniowych, gdzie w porównaniu z kamerami RGB, nie mają problemów z dokładnym uchwyceniem gestu. Dodatkowym asortymentem, mogą być również czujniki ruchu, które mierzą przyspieszenie i orientację dłoni. Zastosowanie ww. technologii, pozwala na rejestrowanie subtelnych ruchów i położenia dłoni w przestrzeni. Ostatnia z przestawionych opcji, daje największe możliwości rejestrowania ruchów. Niestety ze względu na koszt zaawansowanych technologii stosowanych w czujnikach ruchu, produkty te są nieprzystępne cenowo dla potencjalnego odbiorcy oprogramowania, omawianego w niniejszej dysertacji[9]. Rysunek 1 Porównanie technologii (Źródło: opracowanie własne) 1.2. Przegląd metod uczenia maszynowego Uczenie maszynowe (ang. Machine Learning (ML)) to dziedzina sztucznej inteligencji, która skupia się na rozwijaniu algorytmów oraz modeli zdolnych do uczenia i automatycznego podejmowania decyzji na podstawie dostarczonych danych. W odniesieniu do platformy tłumaczącej język migowy jest kluczowym rozwiązaniem, umożliwiając przekształcanie danych wideo w rozpoznawalne wzorce, które są możliwe do przetłumaczenia na słowa lub zdania. Pierwsza sieć neuronowa została opracowana przez Warrena McCullocha, wykładowcy z University of Illinois i Waltera Pittsa, niezależnego badacza. Ich artykuł pt. „A Logical Calculus of the Ideas Immanent in Nervous Activity", opublikowany w 1943 roku w

czasopiśmie „Bulletin of Mathematical Biophysics" został przyjęty z szerokim entuzjazmem, kładąc tym samym podwaliny pod rozwój sztucznych sieci neuronowych. Praca ta okazała się kamieniem milowym w późniejszych badaniach w dziedzinie informatyki, neurobiologii i kognitywistki. Autorzy, przedstawili w nim jak mogą działać neurony, prezentując swoje pomysły i tworząc prostą sieć neuronową za pomocą obwodów elektrycznych[15]. Obecnie sieci neuronowe są fundamentem współczesnych metod uczenia maszynowego. Składają się z wielu warstw neuronów, które przetwarzają dane wejściowe poprzez szereg operacji matematycznych, co umożliwia uczenie się złożonych wzorców. Konwolucyjne sieci neuronowe (CNN) są szczególnie efektywne w przetwarzaniu danych obrazowych oraz wideo, dzięki swojej zdolności do automatycznego wyodrębnienia cech z surowych danych. Sieci te, składają się z warstw konwolucyjnych, poolingowych oraz gęstych, które współpracują ze sobą, w celu przetwarzania informacji o strukturze przestrzennej obiektów. Autorzy publikacji Sign Language Recognition Using Convolutional Neural Networks wykazali, że CNN mogą być używane do skutecznego przetwarzania i rozpoznawania gestów na podstawie obrazu wideo, umożliwiając automatyczną ekstrakcję cech, co jest kluczowe dla poprawnej klasyfikacji gestów. Podkreślili również, że ich model nie tylko osiąga wysoką dokładność w rozpoznawaniu gestów, ale także jest skalowalny i może być rozszerzany o nowe gesty lub języki migowe z relatywnie niewielkim wysiłkiem. Badanie to otwiera drogę do tworzenia bardziej zaawansowanych systemów rozpoznawania języka migowego, które mogą być wykorzystywane w aplikacjach takich jak tłumacze języka migowego na mowę w czasie rzeczywistym, tworzenie interfejsów użytkownika dla osób niesłyszących, a także w edukacji[10]. Modele hybrydowe to połączenie różnych algorytmów klasyfikacyjnych, które mogą prowadzić do poprawy dokładności i stabilności rozpoznawania gestów. Przykładem takiego podejścia jest kombinacja CNN oraz HMM, co pozwala na skuteczniejsze modelowanie zależności czasowych w danych sekwencyjnych. Zastosowanie hybrydy modeli CNN oraz HMM opisuje publikacja pt. „SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition". Autorzy publikacji wykazują, że hybrydowe podejście może być skutecznie wykorzystywane do rozpoznawania ciągłych gestów w języku migowym. Ich badania przeprowadzone zostały na dużych zestawach danych, zawierających filmy z gestami języka migowego, przy czym wykorzystali takie techniki jak augmentacja danych, zwiększając tym samym różnorodność przykładów i poprawiając zdolność generalizacji modelu. Wyniki eksperymentu pokazują, że SubUNets osiąga znacznie lepsze wyniki w porównaniu z wcześniejszymi podejściami, zarówno w kontekście rozpoznawania kształtów dłoni, jak i ciągłego rozpoznawania gestów języka migowego. Autorzy podkreślają, że ich architektura nie tylko rozpoznaje gesty z większą dokładnością, ale także lepiej radzi sobie z różnorodnością kształtów dłoni i płynnością sekwencji[1]. 1.3. Wyzwania w rozpoznawaniu gestów języka migowego Rozpoznanie gestów języka migowego stanowi złożone wyzwanie, które wynika z unikalnych cech ww. sposobu komunikacji. W przeciwieństwie do języków mówionych, język migowy wykorzystuje mimikę oraz posturę ciała, co wymaga zaawansowanych algorytmów do skutecznej analizy i rozpoznania. Język ten charakteryzuje się ogromną różnorodnością gestów. Każdy z nich może mieć wiele wariantów, które różnią się w zależności od regionu, kultury, a nawet osoby wykonującej gest. Ponadto, gesty mogą obejmować stosunkowo nieistotne różnice w ruchach dłoni, ułożeniu palców, kierunku spojrzenia, a także w mimice twarzy, co sprawia, że poprawne rozpoznanie jest niezwykle trudne, zwłaszcza jeżeli model zostaje rozszerzany o nowe gesty oraz ich warianty. Badania prowadzone przez Koller i innych naukowców wskazują, że klasyczne podejścia oparte na modelach HMM mogą być niewystarczające do modelowania złożonych i różnorodnych gestów języka migowego. Aby sprostać temu wyzwaniu, autorzy sugerują zastosowanie hybrydowych modeli łączących CNN oraz HMM, co pozwala na skuteczniejsze modelowanie złożoności gestów[5]. Gesty języka migowego często wykonywane są w sposób płynny oraz nieprzerwany co stwarza jeden z największych problemów w przypadku zastosowania algorytmów SI. Kwestia ta jest niezwykle problematyczna dla modeli, które do poprawnego działania potrzebują wyraźnych przerw pomiędzy poszczególnymi znakami, aby określić gdzie badany gest się zaczyna, a gdzie kończy. Kluczowym problemem podczas tworzenia modelu sztucznej inteligencji jest również sama tematyka. W związku z niszowością tematu, utrudniony jest dostęp do dużych, zróżnicowanych zbiorów danych, które są niezbędne dla skutecznego trenowania modeli uczenia maszynowego. W przypadku języka migowego, zbiory danych są ograniczone pod względem wielkości i różnorodności. Brakuje danych pochodzących od różnych użytkowników, wykonanych w zróżnicowanych warunkach oświetleniowych i środowiskach, co może prowadzić do słabszej generalizacji modelu. Problematykę tą porusza w swoich badaniach Patel oraz inni naukowcy, zwracając

uwagę na problem ograniczonych zbiorów danych, proponując jednocześnie zastosowanie techniki transfer learningu, które pozwalają na wykorzystanie wiedzy zebranej na większych zbiorach danych do trenowania modeli na mniejszych, specyficznych zbiorach języka migowego[6]. Rozpoznawanie gestów w czasie rzeczywistym wymaga również znacznej mocy obliczeniowej, zwłaszcza w przypadku samego tworzenia modelu oraz ekstrakcji punktów na zbiorach treningowych. Programy wykorzystujące przetwarzanie obrazu w czasie rzeczywistym muszą przetwarzać duże ilości danych wideo o wysokiej rozdzielczości, minimalizując poziom opóźnienia do minimum. Optymalizacja rozwiązań w stosunku do urządzeń o gorszych parametrach również stwarza dodatkowe problemy, zwłaszcza w przypadku urządzeń mobilnych. Ostatnim powszechnie znanym wyzwaniem może być różnica w indywidualnym zachowaniu użytkownika. Każda osoba ma inną długość oraz szerokość kończyn. Inna szybkość, kąt nachylenia czy też indywidualny styl poruszania się każdej osoby sprawia, że opracowany model musi być odporny na takowe odchylenia. Na elastyczność modelu zwraca uwagę między innymi Zhang i Liu, wskazując konieczność opracowania metod, które mogą uwzględnić różnice indywidualne, takie jak ww. style gestykulacji, poprzez wykorzystanie cech całej dłoni, co pozwala na dokładniejsze rozpoznawanie gestów w różnych warunkach[11]. 1.4. Analiza problematyki oraz istniejących aplikacji dotyczących języka migowego W ciągu ostatnich lat rozwój technologii głębokiego uczenia oraz komputerowego widzenia przyczynił się do powstania wielu aplikacji, służących do rozpoznawania i tłumaczenia języka migowego. Aplikacja opracowana przez Patel i Sahah, pozwala użytkownikom uczyć się języka migowego za pomocą interaktywnych ćwiczeń. System ten, analizuje gesty użytkownika za pomocą technologii rozpoznawania wideo w czasie rzeczywistym, umożliwiając otrzymanie natychmiastowej informacji zwrotnej, co przyczynia się do skuteczniejszej nauki[8]. Aplikacja o nazwie DeepSign, również funkcjonuje w przestrzeni publicznej jako tłumacz języka migowego. Aplikacja ta wykorzystuje CNN do rozpoznawania gestów i przekształcania ich w tekst w czasie rzeczywistym. System ten jest bardzo zaawansowany, co pozwala tłumaczyć gesty na pełne zdania, co znacznie ułatwia komunikację[7]. Niestety każda z wyżej wymienionych aplikacji jest stworzona na rynek anglojęzyczny, a co za tym idzie, nie obsługuje polskiego języka migowego. Zapoznając się z aktualną na dzień 27.08.2024 ofertą na rynek polskojęzyczny, można odnaleźć jedynie rozwiązania, które nie są zautomatyzowane. Specjalne ośrodki czy też firmy prywatne obsługują klientów w ramach spotkań lub połączeń online z tłumaczem języka migowego. Analizując problematykę zastosowania sztucznej inteligencji, można dostrzec, iż nisza ta nie została jeszcze odpowiednio obsłużona w Polsce. Rozwiązania takie jak wideokonferencje, są skuteczną alternatywą, jednakże wymagają osoby fizycznej, która takową rozmowę przetłumaczy. Wiąże się to z kosztami zatrudnienia specjalisty oraz utrzymania infrastruktury, takiej jak stabilne połączenie z Internetem, odpowiedni sprzęt audio-wizualny oraz platformy obsługującej powyższe rozmowy. Zagłębiając się bardziej w poruszaną tematykę, można odnieść wrażenie, że osoby z dysfunkcjami instrumentów głosowych są w pewien sposób wykluczone społecznie, poprzez brak powszechnego narzędzia, które ułatwiałoby im życie codzienne w sferze komunikacyjnej. W przypadku każdego typu rozmów, należy skupić się na zagadnieniu prywatności, do której każdy obywatel ma prawo. Chociażby w taki sferach jak zdrowie czy też finanse. Dzięki istnieniu aplikacji, która byłaby w stanie w pełni przetłumaczyć język migowy, osoby z dysfunkcjami mowy, mogłyby korzystać z większości znanych usług, bez obecności osoby trzeciej, która będzie towarzyszyć w prowadzeniu rozmowy. Zaleca się aby aspekt ten został szerzej zbadany przez specjalistów w dziedzinie badań społecznych, analizując jednocześnie potrzeby osób niepełnosprawnych, a tym samym zwiększając świadomość społeczeństwa na temat poruszany w omawianej dysertacji. 2. Techniczne aspekty funkcjonowania aplikacji Każde oprogramowanie wymaga uprzedniego przygotowania środowiska pracy. Tak też jest w przypadku aplikacji dołączonej do niniejszej dysertacji. Wybrane etapy posiadają własne wymagania odnośnie sprzętu (ang. hardware), oprogramowania (ang. software) oraz środowiska programistycznego, dlatego w poniższych punktach przedstawione zostały kroki do poprawnej konfiguracji środowiska. 2.1. Wymagania systemowe Program podzielony został na dwa etapy główne. Pierwszy etap odpowiada za przygotowanie próbek oraz modelu sztucznej inteligencji, natomiast drugi etap jest stricte związany z platformą, która ma za zadanie odczytywać przygotowany w etapie pierwszym model SI. Każdy program posiada inne wymagania systemowe, dlatego też zostały one zawyżone w odniesieniu do najbardziej wymagającego programu z etapu pierwszego. Komponent Minimalne wymagania Zalecane wymagania Procesor (CPU) Sześciordzeniowy, np. Intel Core

i5- Ośmiordzeniowy, np. Intel Core 10600K lub AMD Ryzen 5 3600 i7-9700K lub AMD Ryzen 7 3700X Pamięć RAM 16 GB RAM 32 GB RAM Karta graficzna Zintegrowana Zintegrowana (GPU) Przestrzeń Minimum 20 GB wolnej przestrzeni na SSD NVMe z minimum 100 GB dyskowa dysku SSD wolnej przestrzeni System Windows 10 lub nowszy / Linux (Ubuntu Windows 10 Pro lub nowszy / operacyjny 20.04 lub nowszy) Linux (Ubuntu 20.04 LTS lub nowszy) Inne Python 3.8 lub nowszy Python 3.8 lub nowszy Informacje Procesor ośmiordzeniowy z obsługą wielowątkowości zapewni dobrą wydajność w przetwarzaniu wideo i trenowaniu modeli, choć na nieco niższym poziomie niż wcześniej wspomniane opcje. Zintegrowana karta graficzna wystarczy do obsługi podstawowych zadań związanych z wyświetlaniem i przetwarzaniem obrazu, ponieważ główne obciążenie będzie przeniesione na procesor, aby zmienić obciążenie z procesora na kartę graficzną należy dokonać modyfikacji programu „MachineLearning.py". 32 GB RAM pozwoli na komfortową pracę z większymi zbiorami danych i bardziej złożonymi modelami Tabela 1 Wymagania systemowe cz1 Druga część programu opiera się głównie na obsłudze aplikacji internetowej oraz przetwarzaniu obrazu z kamery na podstawie dostarczonego już modelu, co nie wymaga od użytkownika zwiększonej mocy obliczeniowej. Komponent Minimalne wymagania Zalecane wymagania Procesor Czterordzeniowy procesor, np. Intel Sześciordzeniowy procesor, np. Intel (CPU) Core i5-8265U lub AMD Ryzen 5 Core i7-10750H lub AMD Ryzen 5 2500U 4600H Pamięć RAM 8 GB RAM 16 GB RAM Karta Zintegrowana, np. Intel UHD Zintegrowana, np. Intel Iris Xe lub graficzna Graphics 620 lub AMD Radeon AMD Radeon Vega 11 (GPU) Vega 8 Przestrzeń Minimum 10 GB wolnej przestrzeni SSD NVMe z minimum 50 GB wolnej dyskowa na dysku SSD przestrzeni System Windows 10 lub nowszy / Linux Windows 10 Pro lub nowszy / Linux operacyjny (Ubuntu 20.04 lub nowszy) (Ubuntu 20.04 LTS lub nowszy) Inne Python 3.8 lub nowszy Python 3.8 lub nowszy Tabela 2 Wymagania systemowe cz2 2.2. Narzędzia modelowania sztucznej inteligencji Zgodnie z informacjami przedstawionymi w podpunkcie 2.1, każdy program posiada inne wymagania sprzętowe. W tym punkcie przedstawione zostaną indywidualne wymagania dotyczące poszczególnych programów, które nie zostały uwzględnione w poprzednich punktach. Pakiety wymagane do uruchomienia poszczególnych aplikacji, można zainstalować poprzez narzędzie do zarządzania pakietami w Pythonie (pip), wpisując wskazane komendy w terminalu. Program „NagrywanieVideo720p60FPS.py" do poprawnego działania potrzebuje podłączonej kamery internetowej obsługującej jakość 720p oraz przepustowość 60fps, a także importu pakietów takich jak os, time oraz opencv. Dwa pierwsze pakiety są dostępne w podstawowej wersji Pytthona, natomiast pakiet opencv musi zostać doinstalowany. Aby tego dokonać można skorzystać z poniższego polecenia: pip install opencv-python Do obsługi programu „MachineLearning.py" wymagane są pakiety: os, threading, warnings, random, time, webbrowser, multiprocessing, tkinter, cv2, numpy, mediapipe, tensorflow, keras, flatten, scikit-learn oraz pil , osiem pierwszych pakietów to standardowe moduły języka Python, brakujące pakiety można zainstalować inicjując polecenie: pip install opencv-python numpy mediapipe tensorflow keras scikit-learn pillow Program „Prediction Test.py" wymaga takich pakietów jak: cv2, numpy, mediapipe, tensorflow, matplotlib, keras, sklearn, os, tkinter. Brakujące pakiety można zainstalować za pomocą polecenia: pip install opencv-python numpy mediapipe tensorflow keras scikit-learn pillow Do poprawnej obsługi platformy, wymagane są również odpowiednie środki, takie jak pakiety języka python oraz zewnętrzne oprogramowanie inicjujące środowisko serwerowe. Omawiana platforma wymaga utworzenia lokalnego serwera w celu obsługi języka PHP. Przykładem oprogramowania pozwalającego na zainicjowanie ww. środowiska jest XAMPP. Do utworzenia niniejszej aplikacji wykorzystano najnowszą (na dzień 22.08.2024) wersję XAMPP w wersji 3.3.0 oraz PHP w wersji 8.2.12. Po zainstalowaniu aplikacji XAMPP, należy uruchomić opcję Apache, oznaczoną na Rysunku 2 numerem 1. Rysunek 2 Interfejs programu XAMPP Platforma, została projektowana oraz testowana za pomocą przeglądarki internetowej Chrome, która jest zalecana do obsługi ww. aplikacji. Program „app.py" potrzebuje podłączonej kamery internetowej o przepustowości minimum 30 fps oraz zainstalowanych pakietów takich jak: Flask, flask-cors, opencv, numpy, mediapipe, tensorflow, keras, pillow oraz scikit-learn. Elementy te, można zainstalować poprzez komendy w terminalu: pip install Flask flask-cors opencv-python numpy mediapipe tensorflow keras Pillow scikit-learn 3. Budowa modelu rozpoznawania gestów Aplikacja internetowa do właściwego działania potrzebuje wcześniej przygotowanego modelu sztucznej inteligencji, który będzie rozpoznawał wybrane sekwencje ruchów. W tym rozdziale zostanie przedstawiony opis procesu przygotowania ww. modelu. 3.1. Przygotowanie danych wejściowych Przygotowanie danych wejściowych jest kluczowym etapem każdego programu opartego na uczeniu maszynowym. Proces ten obejmuje takie elementy jak wybór odpowiednich źródeł danych, ich organizację, walidację

oraz przetworzenie do odpowiedniego formatu. W związku z niszowością tematyki polskiego języka migowego oraz utrudnionym dostępem do próbek badawczych, autor dysertacji utworzył własną bazę gestów, które poprzez algorytm napisany w języku python zostały zarejestrowane za pomocą kamery internetowej. Każde z nagrań musiało zostać ustandaryzowane w celu przystosowania ich do pracy z jednolitym modelem sztucznej inteligencji. Każda z zarejestrowanych próbek posiadała długość 4 sekund, rozdzielczość 720p oraz przepustowość 60 klatek na sekundę. Dodatkowo, autor przyjął metodę 25-50-25, która zaowocowała nagraniem 100 próbek dla każdego gestu. W późniejszym etapie projektu powtórzono ten proces, aby zyskać większą ilość próbek na innych modelach ciał. Metoda ta opierała się na odpowiednim ustawieniu kadru. Pierwszy etap tworzył 25 nagrań od czubka głowy do dolnej części żeber. Taka konfiguracja, pozwalała na najdokładniejsze zarejestrowanie pracy dłoni, wliczając w to układ palców. Drugi etap tworzył 50 nagrań od czubka głowy do górnej części uda. Nagrania te służyły jako baza do całości gestu. Ujęcia te, ujmowały każdą fazę ruchu, od fazy startowej do fazy końcowej, uwzględniając wszystkie niezbędne elementy ludzkiego działa. Ostatni etap tworzył 25 nagrań od czubka głowy do dolnej części kolan. Proces ten miał za zadanie zwiększyć obszar widzenia programu, uwzględniając dalsze położenie szukanych punktów, dzięki czemu utworzony w późniejszym etapie model, mógł z większą precyzją rozpoznawać pozycję statyczną oraz ruch odpowiednich fragmentów ciała człowieka. Ustawienie każdego z etapów zostało zobrazowane na ilustracji 3. Rysunek 3 Etapy zbierania próbek (Źródło: opracowanie własne) Dzięki zastosowaniu ww. rozwiązania, późniejsze algorytmy przetwarzania danych, mogły z większą łatwością zlokalizować wybrane punkty na ludzkim ciele, a następnie przetworzyć je w ramach sekwencji ruchu. Każdy z etapów dostarczał inną ilość punktów, dzięki czemu model uczył się rozpoznawania gestów ze zróżnicowanej perspektywy. W ramach części praktycznej pracy magisterskiej, został dołączony program „NagrywanieVideo720p60FPS.py", który realizuje wyżej wymienione etapy. Po zakończeniu procesu przygotowywania próbek badawczych autor dysertacji opracował program „MachineLearning.py", który realizuje wszystkie pozostałe kroki do uzyskania działającego modeli SI. Po uruchomieniu ww. programu, użytkownik ma możliwość skorzystania z graficznego interfejsu programu, utworzonego poprzez bibliotekę tkinter. Biblioteka ta jest standardowym pakietem w języku Python. Początkowo, program weryfikuje czy w obecnym katalogu istnieje plik z przetworzonymi punktami o nazwie „preprocessed_data.npz". Jeżeli takowy plik istnieje, zostaje wyświetlone okno z odpowiednim komunikatem (Rys.4). Rysunek 4 Uczenie maszynowe (GUI 1) (Źródło: opracowanie własne) Użytkownik może wybrać jedną z trzech dostępnych opcji. Pierwsza z nich prowadzi do pliku PDF, w którym opisane zostały wymagania sprzętowe dotyczące obsługi programu. Przycisk po prawej stronie, rozpoczyna proces uczenia maszynowego na obecnie wykrytych danych, natomiast przycisk „Wróć do wyboru danych" pozwala na otworzenie widoku okna startowego. Okno to pojawia się również, jeżeli program w początkowej fazie działania nie odnajdzie utworzonego pliku z wyekstraktowanymi punktami. Rysunek 5 Uczenie maszynowe (GUI 2) (Źródło: opracowanie własne) W ww. oknie, użytkownik może wybrać ścieżkę do katalogu, w którym znajdują się wcześniej przygotowane nagrania. Nagrania te, muszą być zgrupowane w odpowiedni sposób. Aby program działał poprawnie, struktura plików powinna wyglądać następująco: Katalog wybrany przez użytkownika / Nazwa sekwencji ruchu / pliki o rozszerzeniu *.avi. Jeżeli program wykryje błędną strukturę plików, zwróci komunikat w kolorze czerwonym. W przypadku poprawnie przetworzonych plików, wyświetli stosowne powiadomienie w kolorze zielonym, tworzy listę wszystkich ścieżek do plików wideo, a następnie odblokuje przycisk „Rozpocznij analizę". Każdy z powyższych przykładów został zobrazowany, za pomocą poniższej grafiki. Rysunek 6 Uczenie maszynowe (GUI 2 - Struktura) (Źródło: opracowanie własne) 3.2. Preprocessing danych Preprocessing polega na wstępnym przetwarzaniu danych, a dokładniej na przekształcaniu surowych danych w formę, która jest bardziej odpowiednia i optymalna dla modelu uczenia maszynowego. Celem tego fragmentu kodu, jest utrzymanie spójności oraz kompatybilności, a także poprawa jakości danych, potrzebnych do uczenia modelu. Usuwa on wszystkie zbędne informacje, czyli tzw. szumy, pozostawiając jedynie potrzebne dane, wymagane przez model SI. Po rozpoczęciu analizy nagrań, program wyświetla nowe okno z paskiem postępów oraz możliwością rozszerzenia widoku o szczegóły. Na rysunku 7, po lewej stronie przedstawiono podstawowe okno przetwarzania danych, po prawej stronie natomiast jest widoczne okno po rozszerzeniu szczegółów. Rysunek 7 Uczenie maszynowe (GUI 3) (Źródło: opracowanie własne) W trakcie przetwarzania danych, algorytm wykorzystuje bibliotekę MediaPipe, opracowaną przez Google. Za jej

pomocą, przeprowadza ekstrakcję punktów kluczowych z dostarczonych nagrań, analizując każdą z 240 klatek, a następnie przypisuje do nich odpowiednie etykiety. W ten sposób tworzy pełen spis punktowy, przedstawionego na nagraniu gestu. Zastosowanie powyższego rozwiązania, skutkuje zmniejszeniem ilości danych przetwarzanych przez etap uczenia maszynowego. W przypadku standardowego podejścia, program przetwarzałby każdy pikseli wideo, co mogłoby wprowadzić dodatkowy szum. Skutkowałoby to mniej precyzyjnym uczeniem modelu, a co za tym idzie gorszą precyzją rozpoznawania gestów. Program, posiada również możliwość podglądu wizualizacji procesu ekstrakcji punktów w czasie rzeczywistym. Gdy użytkownik po rozwinięciu szczegółów, uruchomi przycisk „Wyświetl wizualizację" otworzy się nowe okno, które zobrazuje przetwarzanie danych w sposób graficzny. Rysunek 8 Uczenie maszynowe (GUI 3 - Wizualizacja) (Źródło: opracowanie własne) Podczas przetwarzania informacji, program normalizuje dane poprzez pomijanie wideo o dłuższym czasie trwania lub mniejszej ilości klatek na sekundę, jeżeli natomiast nagranie jest krótsze, brakująca część czasu jest dodawana poprzez funkcję uzupełniania sekwencji pad_sequences. Funkcja ta, tworzy macierz wypełnioną zerami o wymiarach zdefiniowanych w kodzie programu, następnie dla każdej sekwencji kopiuje jej wartość do nowej macierzy, a jeżeli sekwencja jest krótsza niż maksymalna długość, resztę wypełnia zerami. Program, ze względu na konieczność przetwarzania dużej ilości danych oraz potencjalnie długi okres trwania ww. procesu, posiada obsługę wieloprocesowości. Wieloprocesowe podejście pozwala na jednoczesne przetwarzanie wielu nagrań na rdzeniach procesora, co znacząco redukuje czas potrzebny na preprocessowanie danych. Funkcje, takie jak blokada oraz kolejka zapewniają synchronizację między procesami, aby zapobiec ewentualnym kolizjom podczas dostępu do wspólnych zasobów, takich jak zapis do listy danych, czy aktualizacja wskaźnika postępu. Gdy dane zostaną poprawnie przetworzone, program tworzy plik „preprocessed_data.npz", w którym zapisuje przetworzone dane oraz „classes.npy" w którym przechowuje nazwy etykiet. Następnie pasek postępu dobiegnie końca, a na ekranie użytkownika wyświetla się komunikat, który umożliwia przejście do procesu uczenia modelu SI lub zamknięcie programu. Rysunek 9 Uczenie maszynowe (GUI 3 - Zakończenie przetwarzania danych) (Źródło: opracowanie własne) 3.3. Architektura modelu oraz algorytm klasyfikacji Architektura modelu odnosi się do struktury sieci neuronowej. To ona determinuje sposób, w jaki dane są przetwarzane, jakie operacje są wykonywane oraz jakie mechanizmy są stosowane do obsługi modelu. W omawianym oprogramowaniu, autor dysertacji zastosował model oparty na konwolucyjnych sieciach neuronowych (CNN), co miało na celu uchwycenie wzorców w danych sekwencyjnych, takich jak punkty kluczowe wyodrębnione z nagrań wideo. W kontekście przetwarzania sekwencji, warstwy konwolucyjne (Conv1D) przetwarzają dane jednokierunkowo. Filtrują sekwencje punktów kluczowych, tym samym wykrywając wzorce przestrzenno-czasowe. Filtry, inaczej zwane jądrami, są przesuwane wzdłuż sekwencji, stosując operację splotu. Czynność ta, pozwala na uchwycenie lokalnych wzorców danych. Po każdej warstwie konwolucyjnej zastosowano warstwę MaxPooling, która redukuje wymiarowość danych, zachowując najbardziej istotne cechy. MaxPooling wybiera maksymalną wartość z każdej lokalnej grupy wyników, co pomaga w uodpornieniu modelu na przesunięcia w danych. Równocześnie dokonuje redukcji liczby parametrów, co skutkuje zmniejszeniem ryzyka wystąpienia zjawiska przeuczenia modelu. Model został zaimplementowany przy pomocy biblioteki Keras, która działa na bazie TensorFlow. Zastosowane rozwiązanie oferuje szeroką gamę narzędzi do pracy z przetwarzaniem danych, budowaniem architektury sieci, a także optymalizacji modeli. Omawiany program, tworzy model w sposób sekwencyjny, co oznacza, że warstwy są dodawane jedna po drugiej, tworząc stos warstw. Każda z warstw przyjmuje jako argumenty liczbę filtrów o wymiarze 64, rozmiar jądra równy 3 oraz funkcję aktywacji „relu". Liczba filtrów określa, ile różnych cech model będzie próbował wyodrębnić. Rozmiar jądra to zakres, na którym filtr jest stosowany w danej chwili. Funkcja aktywacji relu, poprzez zamianę wszystkich wartości ujemnych na zero, zapobiega problemowi zanikania gradientu i przyspiesza proces nauki, umożliwiając sieci efektywne uczenie się złożonych wzorców. Warstwa Flatten w niniejszym programie, używana jest do przekształcenia danych z formatu wielowymiarowego do jednowymiarowego, który jest wymagany przez kolejne, w pełni połączone warstwy takie jak warstwa gęsta Dense. Warstwa ta, używana jest zarówno do przekształcania wyodrębnionych cech w postać o mniejszej wymiarowości, jak i do generowania ostatecznego wyniku klasyfikacji. Units, czyli liczba neuronów w warstwie została ustawiona na 256, co pozwala na modelowanie bardziej złożonych zależności, jednakże może prowadzić do przeuczenia. W ostatnim kroku, warstwa Dense generuje wyniki klasyfikacji za pomocą funkcji aktywacji softmax. Funkcja ta, przekształca wyniki w prawdopodobieństwa, które sumują się do 1, co pozwala na przypisanie prawdopodobieństwa do

klasy. 3.4. Trening, walidacja oraz ocena skuteczności modelu Trening modelu to proces optymalizacji jego parametrów tak, aby jak najlepiej uczył się rozpoznawania wzorców dostarczonych przy pomocy danych wejściowych, a następnie właściwie przypisywał je do odpowiednich etykiet. W kontekście sieci neuronowych, trening polega na minimalizacji funkcji straty, przy użyciu danych treningowych oraz odpowiedniego optymalizatora. W pierwszej fazie treningu modelu, zastosowany został tzw. forwadr propagation, czyli etap w którym dane przekazywane są do modelu, a każdy neuron przekształca wejścia, oblicza wyjścia i przekazuje je do następnej warstwy. Na końcu tego procesu, model generuje wynik, który jest porównywany z rzeczywistą etykietą danych przy użyciu funkcji straty. Następnie występuje backpropagation, czyli kluczowy mechanizm w treningu sieci neuronowych. Polega on na propagacji błędu wstecz przez sieć neuronową. Błąd ten jest obliczany jako różnica pomiędzy przewidywaniami modelu, a rzeczywistymi wynikami. Proces ten, aktualizuje wagi w sieci tak, aby minimalizować funkcję straty. W tym przypadku zastosowany został algorytm Adam, który jest odmianą algorytmu gradientu prostego (Stochastic Gradient Descent, SGD). Algorytm ten stosuje poniższy wzór matematyczny: Rysunek 10 Wzór algorytmu gradientu prostego gdzie: - w to wagi, które są optymalizowane, η (eta) to współczynnik uczenia się (ang. learning rate), który określa, jak duży krok robimy w kierunku minimalizacji funkcji kosztu (ang. Loss function), natomiast pozostała część to to pochodna funkcji kosztu Loss względem wag w, czyli gradient funkcji kosztu. Pokazuje kierunek i szybkość zmiany funkcji kosztu w zależności od zmian wag. Późniejszy etap obejmuje tworzenie tzw. epok. W każdej epoce, model przetwarza cały zbiór treningowy. Aby zwiększyć efektywność aktualizacji wag, zastosowane zostało porcjowanie (batch), które pozwala na przetwarzanie mniejszej ilości informacji jednocześnie. Podczas treningu, model uczy się na danych treningowych, jednakże istnieje pewne ryzyko, że nauczy się on wzorców specyficznych dla tych danych. Aby zapobiec ww. problematyce, zastosowano techniki regularyzacji takie jak Dropout oraz L2 Regularization (Ridge). Pierwsza z nich polega na losowym wyłączeniu części neuronów w warstwie podczas treningu. W omawianym programie wartość ta została ustawiona na 0.5, co oznacza, że połowa neuronów w warstwie jest wyłączana w każdej iteracji. Druga zaś, dodaje karę do funkcji straty za duże wartości wag, co ogranicza złożoność modelu i zapobiega jego przeuczeniu. Walidacja modelu jest niezbędna w procesie uczenia maszynowego. W jego trakcie program ocenia, jak dobrze model generalizuje, czyli jak radzi sobie z danymi, których nie uwzględnił podczas treningu. Autor pracy dyplomowej, przyjął złożenie 20% / 80%, gdzie 20% danych jest wykorzystywanych do walidacji, natomiast 80% do trenowania modelu. W trakcie uczenia, stosowane są callback'i. EarlyStopping, czyli wczesne zatrzymanie uczenia maszynowego, wykorzystane jest w celu zabezpieczenia modelu przed ewentualnym przeuczeniem. Proces ten stale monitoruje wydajność modelu na zbiorze walidacyjnym i przerywa trening w przypadku braku poprawy wyników. Istnieje również callback nazwany ModelCheckpoint, który zapisuje najlepsze wagi modelu na podstawie wybranej metryki. Cały proces uczenia maszynowego, został zobrazowany poprzez interfejs graficzny użytkownika, który pojawia się po zatwierdzeniu przycisku dostępnego po przetworzeniu nagrań. Rysunek 11 Uczenie maszynowe (GUI 4) (Źródło: opracowanie własne) Gdy uczenie maszynowe dobiegnie końca, program tworzy plik o nazwie best_model.h5, w którym zapisane są wszystkie informacje opracowywane przez powyższe algorytmy. Użytkownik, otrzymuje również powiadomienie dotyczące zakończenia pracy programu. Na tym etapie, może on zweryfikować poprawność uczenia maszynowego, a także sprawdzić ile epok przetworzył program w celu utworzenia modelu. Zatwierdzając przycisk „Zakończ", użytkownik kończy pracę programu, otrzymując tym samym dostęp do wszystkich niezbędnych plików do obsługi platformy, opisanej w rozdziale czwartym. Rysunek 12 Uczenie maszynowe (GUI 4 - Zakończenie pracy) (Źródło: opracowanie własne) Działanie powyższego programu obrazuje diagram aktywności zawarty na rysunku 13. Rysunek 13 Diagram aktywności (Uczenie maszynowe) (Źródło: opracowanie własne) 4. Implementacja platformy Implementacja platformy opiera się na utworzeniu aplikacji internetowej wraz z odpowiednim połączeniem jej z modelem sztucznej inteligencji. Implementację możemy podzielić w tym przypadku na dwie części. Część frontendową, która opiera się na wdrożeniu wersji graficznej, z której będzie korzystać użytkownik, a także część backendową, która będzie obsługiwać komunikację z serwerem oraz modelem SI. 4.1. Koncepcja wizualna platformy Każda aplikacja weebowa, która będzie posiadać interfejs graficzny (GUI), powinna być uprzednio zaprojektowana przez osobę, która dobrze rozumie zasady korzystania z interfejsów aplikacji internetowych. Doświadczenie w projektowaniu platform przyjaznym użytkownikom pozwala na zdobycie szerszego grona odbiorców poprzez pozytywne odczucia z

korzystania z aplikacji. Funkcjonalności wybranego oprogramowania stanowią dobrą podstawę, natomiast design może wzmocnić i uatrakcyjnić aplikację, czyniąc ją bardziej przyjazną i angażującą dla użytkowników[13]. W obecnej dobie Internetu, gdzie panuje przesyt informacji, a użytkownicy są często przebodźcowani, bardzo istotne jest utworzenie odpowiedniej warstwy graficznej. Użytkownicy zazwyczaj oceniają aplikację w ciągu kilku sekund od jej uruchomienia. Atrakcyjna i spójna oprawa graficzna może przyciągnąć uwagę odbiorcy i zachęcić go do dalszej eksploracji. Niezależnie od tego, jak zaawansowane oprogramowanie zostało zaoferowane użytkownikowi, pierwsze wrażenie bazuje na wyglądzie. Intuicyjny interfejs, który jest łatwy w nawigacji, ułatwia korzystanie z aplikacji, pozostawiając pozytywne wrażenia. Wygląd aplikacji nie tylko zwiększa jej zainteresowanie wśród odbiorców ale również pozytywnie wpływa na wzmocnienie marki, poprzez identyfikację marki z wybraną nazwą, kolorystyką czy też utworzonym logotypem[14]. W związku z powyższymi informacjami, omawiana aplikacja również posiada swój projekt wizualny, który został utworzony na początku procesu projektowego w oprogramowaniu figma, a plik projektowy został umieszczony pod nazwą „Projekt koncepcyjny.fig" oraz „Projekt koncepcyjny.pdf" w części praktycznej pracy dyplomowej. Rysunek 14 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 4.2. Mechanizmy komunikacji między frontendem a backendem Większość zaawansowanych aplikacji, korzystających z graficznego interfejsu użytkownika, a zwłaszcza aplikacje webowe, korzystają z warstwy frontendowej oraz backendowej w celu optymalizacji obsługi dużej ilości poleceń lub wykorzystania niedostępnych z poziomu wybranej warstwy narzędzi. 4.2.1. Komunikacja serwer – użytkownik Omawiana aplikacja napisana jest głównie w języku PHP, JavaScript oraz Python co wymaga stałej komunikacji pomiędzy frontendem, a backendem. Kod w języku PHP dzięki stałemu połączeniu z serwerem, może w sposób dynamiczny, kontrolować dostępem do witryny oraz zawartości, która jest wyświetlana na ekranie użytkownika. Poprzez zarządzanie witryną kod PHP umożliwia wczytanie odpowiedniego kodu CSS oraz JavaScript w wybranych zakładkach, dzięki funkcji rozpoznawania adresów URL. Oprócz aspektów wizualnych, PHP obsługuje również wczytywanie wizualizacji dla gestów, poprzez przeszukiwanie katalogów zamieszonych na serwerze w poszukiwaniu nazwy wprowadzonej w warstwie frontendowej oraz przekazanej odpowiednio przez skrypt napisany w języku JavaScript. 4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) Integracja modelu AI z aplikacją internetową jest realizowana poprzez połączenie części backendu, napisanego w języku programowania Python z częścią frontendową platformy napisaną w języku JavaScript. Wspomniana część backendowa jest zapisana w pliku o nazwie „app.py". Program ten uruchamia model sztucznej inteligencji o nazwie „best_model.h5" oraz listę etykiet „classes.npy", które są wykorzystywane do przetwarzania danych w czasie rzeczywistym. Dane z kamery użytkownika są przetwarzane, a wyodrębnione punkty kluczowe są przekazywane do modelu, który dokonuje predykcji gestów. Wyniki te są następnie udostępniane poprzez endpointy, które zwracają szczegóły dotyczące rozpoznanych gestów w formacie JSON. Frontend aplikacji komunikuje się z backendem za pomocą zapytań http, korzystając z fetch API do wysyłania zapytań do serwera Flask. Za komunikację z serwerową częścią oprogramowania, odpowiedzialne są trzy skrypty napisane w języku JavaScript, a są nimi: „edu,js", „translator.js" oraz „sign_video_loader.js". Każdy z trzech wymienionych skryptów, korzysta z wywołań endpointów poprzez serwer lokalny o porcie 5000 „http://localhost:5000/gesture_details", które służą do pobierania informacji o rozpoznanych gestach i aktualizowania interfejsu użytkownika. W pliku app.py, endpoint dostarcza strumień wideo z kamery internetowej, który następnie jest przetwarzany przez model SI. Frontend inicjuje kamerę oraz przesyła obraz do backendu gdzie obraz jest przetwarzany, a następnie odtwarza przesyłany strumień, umożliwiając tym samym interaktywną analizę gestów w czasie rzeczywistym. 4.3. Widoki aplikacji Omawiana platforma internetowa posiada tzw. widoki, czyli warstwę frontendową, która jest odpowiednio ostylowana poprzez arkusze stylów CSS, starając się jak najdokładniej odwzorować utworzony w początkowych fazach pracy projekt wizualizacji aplikacji. Rysunek 15 Interfejs graficzny aplikacji – zakładka Home (Źródło: opracowanie własne) Platforma składa się z siedmiu widoków. Trzy pierwsze widoki odnoszą się do informacji wstępnych dotyczących realizowanego projektu. Są to zakładki „Home", „O projekcie" oraz „O nas". Każda z zakładek posiada statyczne elementy informacyjne oraz dekoracyjne, które mają zachęcić użytkownika do dalszej interakcji z platformą. Poniżej opisane zostały bardziej interesujące elementy, każdej z zakładek. Rysunek 16 Zakładka Home (Źródło: opracowanie własne) Na stronie Index.php (Home) zostały przedstawione podstawowe informacje dotyczące platformy. Zastosowany został również slider, który wyświetla logotyp Uniwersytetu Rzeszowskiego, przechodząc od prawej strony do lewej, a następnie cofając się do początku. Rysunek 17 Slider - strona Index.php

(Źródło: opracowanie własne) Zakładka O projekcie (Project.php) zawiera informacje dotyczące funkcjonowania utworzonej aplikacji. Można znaleźć tam między innymi informacje o zastosowanych rozwiązaniach technologicznych, genezie powstania dysertacji, poruszanej problematyce, skrócony opis działania aplikacji, a także linię czasu, która w sposób skrótowy ukazuje etapy powstawania aplikacji. Rysunek 18 Widok zakładki O projekcie (Project.php) (Źródło: opracowanie własne) Zakładka O nas (About.php) skupia się na opisie osób zaangażowanych w powstanie projektu. W związku ze stałym rozwojem tworzonego oprogramowania oraz wysoką czasochłonnością prac, a także innymi losowymi aspektami, w projekt były zaangażowane 4 osoby. Oprócz autora oprogramowania oraz niniejszej dysertacji, w projekt zaangażował się również Dr. Inż. Wojciech Kozioł, który był pomysłodawcą tematu omawianej pracy, Dr. Inż. Bogusław Twaróg, który prowadził nadzór merytoryczny nad powstającą pracą oraz inż. Patryk Arendt, który służył jako model motion capture dla 5 gestów (tj. 500 próbek nagrań wideo). Rysunek 19 Widok zakładki O nas (About.php) (Źródło: opracowanie własne) W ramach uhonorowania wkładu, jaki włożyły ww. osoby w powstanie pracy dyplomowej, zostały one wymienione w wersji aplikacyjnej w zakładce O nas. Rysunek 20 Uhonorowanie wsparcia (About.php) (Źródło: opracowanie własne) Kliknięcie przycisku „Przejdź do aplikacji" powoduje wczytanie ostatniego statycznego widoku jakim jest „Dashboard.php". W widoku tym opisane są podstawowe informacje dotyczące ilości gestów, ilości wykorzystanych próbek, sumarycznego czasu nagrań, a także wiele innych. Rysunek 21 Widok zakładki Dashboard (Dashboard.php) (Źródło: opracowanie własne) W zakładce tej przedstawiona została również animacja, która w trzech etapach prezentuje w jaki sposób program rozpoznaje sekwencje ruchów ludzkiego ciała, na przykładzie gestu z etykietą „Dzięki". Rysunek 22 Działanie systemu rozpoznawania gestu (Źródło: opracowanie własne) Pozostałe trzy widoki, odpowiadają pośrednio lub bezpośrednio za komunikację z systemem rozpoznawania gestów. Działają one na podobnej zasadzie, lecz skupiają się na innym rodzaju wykonywanego zadania przez użytkownika. Rysunek 23 Widok zakładki Tłumacz (Translator.php) (Źródło: opracowanie własne) Zakładka Tłumacz, umożliwia użytkownikowi tłumaczenie słów w języku migowym w obie strony. Może on skorzystać z funkcji udostępnienia obrazu z kamery co pozwoli na tłumaczenie języka migowego na język naturalny lub wyszukać gest w języku migowym, dostępny w systemie poprzez polecenie pisemne lub komendę głosową. W celu wyszukanie gestu języka migowego, wystarczy, że użytkownik uzupełni pole tekstowe odpowiednią nazwą gestu i zatwierdzi zmianę klikając przycisk po prawej stronie lub wciskając klawisz „Enter". Rysunek 24 Wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) Istnieje również alternatywny sposób wyszukiwania gestu w systemie. Użytkownik może skorzystać z przycisku, znajdującego się po lewej stronie okna, aby rozpocząć nasłuchiwanie mikrofonu. Po wypowiedzeniu frazy np. „Cześć", możemy użyć polecenia głosowego „... stop szukaj", co automatycznie zakończy nasłuchiwanie mikrofonu i rozpocznie proces szukania gestu o etykiecie „Cześć". Nasłuchiwanie mikrofonu można także wyłączyć w sposób ręczny, klikając ponownie na ikonę mikrofonu. Wyszukiwanie głosowe, oparte zostało na polskiej wersji językowej Web Speech API dla języka JavaScript. Po aktywacji funkcji nasłuchiwania program sprawdza dostępność API w przeglądarce i konfiguruje obiekt rozpoznawania mowy. Podczas nasłuchiwania mikrofonu, program przetwarza rozpoznane fragmenty, a następnie wstawia je do pola tekstowego. Jeżeli program wykryje określoną komendę tj. „stop szukaj", przerywa nasłuchiwanie i uruchamia proces wyszukiwania nagrania. Program obsługuje najczęściej pojawiające się błędy, a także automatycznie restartuje nasłuchiwanie, jeżeli nie zostało ono przerwane ręcznie. Rysunek 25 Głosowe wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) Gdy gest nie zostanie odnaleziony, na ekranie pojawia się stosowny komunikat. Gdy gest zostanie wyszukany poprawnie, oczom użytkownika ukaże się wizualizacja gestu odtwarzana w postaci nagrania. Przykład takowej wizualizacji został ukazany na grafice poniżej. Rysunek 26 Wizualizacja gestu (Źródło: opracowanie własne) W drugim przypadku użycia, użytkownik może udostępnić obraz z kamery, a następnie wykonać wybrany przez siebie gest. Na wykrytą postać zostanie naniesiona siatka punktów, która będzie obrazować przetwarzane dane. Program w czasie rzeczywistym, w lewym górnym rogu ekranu wyświetlać będzie prawdopodobne gesty pokazane przez użytkownika. Gdy gest zostanie wykryty wystarczającą liczbę razy, oczom użytkownika ukaże się również przycisk „Wyświetl wizualizację pokazywanych gestów". Rysunek 27 Prezentacja gestu "Dzięki" (Źródło: opracowanie własne) Gdy użytkownik kliknie na komunikat „Wyświetl wizualizację pokazywanych gestów" po prawej stronie, automatycznie w sposób płynny wczyta się nagranie z wizualizacją gestu. Wizualizacja ta będzie się zmieniać

za każdym razem, gdy program wykryje pokazanie innego gestu z odpowiednią pewnością. Rysunek 28 Synchronizacja wizualizacji z obrazem w czasie rzeczywistym (Źródło: opracowanie własne) Po przejściu na inną zakładkę, w przeglądarce internetowej klienta, śledzenie kamery zostaje zatrzymane, zarówno przez warstwę forntendową jak i backendową, a algorytmy wstrzymują swoją pracę oczekując na wznowienie połączenia. Poniżej przedstawiono diagram przypadków użycia dla zakładki Tłumacz: Rysunek 29 Diagram przypadków użycia (Tłumacz) (Źródło: opracowanie własne) Dla powyższego widoku został również przygotowany diagram aktywności, który pozwala na jeszcze bardziej precyzyjne zapoznanie się z architekturą działania oprogramowania. Poniższa grafika obrazuje proces przetwarzania obrazu z kamery użytkownika, wraz z obsługą wizualizacji, która jest aktualizowana w czasie rzeczywistym. Rysunek 30 Diagram aktywności (Tłumacz - kamera / wizualizacja) (Źródło: opracowanie własne) Zobrazowany został również diagram aktywności, odpowiedzialny za funkcję wyszukiwania wizualizacji za pomocą poleceń tekstowych oraz głosowych. Diagram UML ww. procesu znajduje się poniżej. Rysunek 31 Diagram aktywności (Tłumacz – Wizualizacja) (Źródło: opracowanie własne) Ostatnie dwa widoki aplikacji omawiają działanie zakładki „Nauka języka migowego”. Oprogramowanie, dzięki zaawansowanym algorytmom wspiera proces uczenia języka migowego. W tym celu stworzone zostały kategorie oraz „kafelki”, które umożliwiają wybór gestu, którego chce się nauczyć użytkownik. Funkcja ta skierowana jest dla osób początkujących, które dopiero rozpoczynają swoją przygodę z PJM. Rysunek 32 Widok zakładki Nauka Języka Migowego (Edu.php) (Źródło: opracowanie własne) Na stronie podstronie edu.pl możemy wybrać kategorię, która nas interesuje, a następnie odpowiedni dla nas gest. Po wybraniu odpowiedniej kategorii użytkownik będzie mógł zaobserwować zmianę w wyświetlanych elementach. Zamiast kategorii, na ekranie pojawią się gesty, które są z nią powiązane. Rysunek 33 Gesty wybranej kategorii (Źródło: opracowanie własne) Po wybraniu jednego z dostępnych gestów pojawia się nowy widok (Edu-progress.php), który wizualnie jest zbliżony do widoku tłumacza. Tutaj również możemy uruchomić podgląd z kamery, natomiast po prawej stronie odtwarzane jest w zapętleniu nagranie z wybranym przez nas gestem. W widoku tym, niedostępny jest input do wyszukiwania gestów, czy też zmiany wizualizacji. Po uruchomieniu kamery i poprawnym wykonaniu gestu, program zatrzyma udostępnianie obrazu z kamery, a użytkownik otrzyma odpowiedni komunikat na ekranie. Rysunek 34 Wykonanie poprawnego gestu (Źródło: opracowanie własne) 5. Ocena działania platformy Utworzona platforma została przetestowana pod względem działania modelu sztucznej inteligencji oraz kilku powiązanych z nią czynników. Przebadana została precyzja modeli CNN, skonstruowanych z różnych ilości próbek, w konfiguracji 100 próbek na 1 gest, 200 próbek na 1 gest oraz 300 próbek na 1 gest. Wyszczególnione zostały etapy przetwarzania danych, uczenia maszynowego oraz ogólnej pracy modelu w czasie rzeczywistym, a także na bazie przygotowanych wcześniej nagrań. Następnie, wyniki te zostały zestawione z utworzonym hybrydowym modelem CNN + LSTM. 5.1. Testy w rzeczywistych warunkach Testy związane z wydajnością w rzeczywistych warunkach, a co za tym idzie w czasie rzeczywistym są stosunkowo ciężkie do realizacji. Wpływ ma na to rozbieżność pomiędzy wykonywanymi gestami, które ciężko odtworzyć w sposób identyczny, zróżnicowane oświetlenie, ułożenie ciała oraz wykorzystywany hardware. Istnieje wiele zmiennych, które mogą zakłócić obiektywną ocenę modelu, zwłaszcza jeżeli ten został przygotowany na niewielkiej ilości próbek. W omawianym punkcie, przedstawione zostaną zarówno obiektywne informacje, podparte dowodami, jak i subiektywne odczucia autora dysertacji. 5.1.1. Ekstrakcja punktów kluczowych Pierwszym elementem, który zostanie poddany analizie jest proces przetwarzania informacji, potrzebnych do utworzenia modelu. Cały proces uczenia maszynowego czyli ekstrakcja punktów kluczowych oraz nauka modelu odbywała się na maszynie obliczeniowej z zamontowanym procesorem AMD EPYC 7702 64-Core, który posiadał 128 procesorów wirtualnych o szybkości 2GHz, pamięcią ram 256GB oraz dysku HDD. Przetwarzanie danych w modelu CNN, zastosowane na 6 gestach po 100 próbek każdy, co daje wynik 600 próbek, trwało 19 minut w zaokrągleniu czasu do pełnych minut. Czas trwania tego samego procesu na 1200 próbkach trwało 35 minut, natomiast finalny model omawiany w dysertacji, potrzebował 54 minut na przetworzenie wszystkich danych (1800 próbek). Po zestawieniu ww. informacji w postaci wykresu, możemy zaobserwować jak wygląda złożoność czasowa przetwarzania danych. Przetwarzanie próbek w stosunku do czasu 60 54 50) n im (a 40 35 in a z r a 30 w t e 19 z r p 20 s a z C 10 0 600 próbek 1200 próbek 1800 próbek Czas w minutach 19 35 54 Ilość próbek / Czas Wykres 1 Przetwarzanie próbek CNN Uśredniając czas przetwarzania każdej próbki wynosi 1,81 sekundy. Rozbijając na poszczególne partie: przy 600 próbkach, program potrzebował 1,9 sekundy na przetworzenie jednego nagrania, 1,75 sekundy dla 1200 próbek oraz 1,8

sekundy w przypadku 1800 próbek. Zestawiając dane przetwarzane przez program wykorzystujący CNN oraz CNN w połączeniu z LSTM, nie widać jednoznacznych różnic. Program w podobnym czasie przetworzył dane zarówno dla modelu opartym o CNN jak i modelu hybrydowym, przy czym mniej złożony model zakończył proces ponad 2 minuty wcześniej. Rysunek 35 Zestawienie metody CNN oraz CNN + LSTM (Źródło: opracowanie własne) Czas przetwarzania danych (1800 próbek) CNN+LSTM 56 u le d o m ja z d o R CNN 54 53 54 54 55 55 56 56 57 Czas przetwarzania (min) Wykres 2 Czas przetwarzania danych – porównanie CNN – CNN + LSTM 5.1.2. Uczenie modelu Uczenie modelu również zostało porównane w identycznych kryteriach. Z otrzymanych informacji wynika, że czas uczenia maszynowego nie jest stricte związany z ilością próbek. Funkcja przerywania uczenia, która za zadanie ma chronić model przed przeuczeniem, skutecznie skraca czas potrzebny do stworzenia modelu. Program kończy pracę za każdym razem gdy wykryje, że precyzja modelu nie poprawia się co skutkuje utworzeniem modelu w krótszym czasie niżeli z potencjalnie mniejszą ilością próbek. Uczenie maszynowe w stosunku do czasu 140 124 120)s (u 100 le d 76 o m 80 a in 60 e z c u s 40 30 a z C 20 0 600 próbek 1200 próbek 1800 próbek Czas w sekundach 30 124 76 Ilość próbek / Czas Wykres 3 Uczenie modelu CNN O ile nie było widocznej różnicy pomiędzy przetwarzaniem danych pomiędzy modelem wykorzystującym CNN, a modelem hybrydowym CNN + LSTM, o tyle w przypadku uczenia modelu różnica jest bardzo zauważalna. Model z mniejszą ilością warstw uczył się przez 1 minutę i 16 sekund, natomiast model z przetwarzaniem danych wstecz, trenował się przez 6 godzin, 9 minut i 36 sekund. Różnica w czasie nauczania modelu jest ogromna, a co za tym idzie model hybrydowy z zastosowaniem LSTM wydaje się mało wydajnym rozwiązaniem, zważywszy na małą ilość przetwarzanych danych. Program oparty na mniejszej ilości warstw uczył się ze stosunkowo wysoką precyzją, która oscylowała w zakresie 91-95%, oraz 81-88% podczas walidacji, co obrazuje poniższa ilustracja. Rysunek 36 Proces uczenia maszynowego CNN (Źródło: opracowanie własne) Druga wersja programu zatytułowana roboczo 2.1, wykazywała wyższą precyzję uczenia podczas kilku prób uczenia modelu. Dokładność modelu oscylowała pomiędzy 97-99%, a podczas walidacji wynik ten wynosił 93-99%. Wyniki te sugerują, że model utworzony za pomocą CNN oraz LSTM radzi sobie znacznie lepiej niżeli poprzednia wersja. Należy zwrócić uwagę również na wielkość pliku docelowego. Model utworzony w wersji 1.0 zajmuje 25,8 MB miejsca na dysku, natomiast model w wersji 2.1 zajmuje 1,26 GB. Kwestia ta staje się problematyczna, gdyż model reprezentujący niewiele niższą precyzję podczas uczenia 5 maszynowego, zajmuje 50 razy mniej miejsca na dysku, co staje się kuszącym rozwiązaniem w przypadku chęci zaoszczędzenia miejsca w przestrzeni dyskowej. Rysunek 37 Proces uczenia maszynowego CNN + LSTM (Źródło: opracowanie własne) Czas przetwarzania danych (1800 próbek) CNN+LSTM 22176 u le d o m ja z d o R CNN 76 0 5000 10000 15000 20000 25000 Czas przetwarzania (s) Wykres 4 Czas uczenia modelu – porównanie CNN – CNN + LSTM 5.1.3. Testy w czasie rzeczywistym Model z założenia miał zostać przeznaczony do platformy obsługującej rozpoznawanie polskiego języka migowego w czasie rzeczywistym. W związku z powyższą informacją, również ten aspekt został przetestowany. Niestety w związku z wieloma czynnikami losowymi, nie da się zmierzyć precyzyjnie różnicy pomiędzy rozpoznawaniem gestów w czasie rzeczywistym pomiędzy modelem utworzonym w wersji podstawowej oraz hybrydowej. 5 Jednakże dokonano pewnych obserwacji, które sugerują wykorzystanie jednego spośród dwóch modeli. W przypadku modelu podstawowego, nie zaobserwowano problemów. Model działał stabilnie, precyzja wykrywanych gestów była zadowalająca. W przypadku modelu hybrydowego, uruchomionego w warunkach czasu rzeczywistego zaobserwowano problemy. Model działał niestabilnie, rejestrując ciągłe „przycięcia" obrazu. Zachodziło tam zjawisko klatkowania, a dokładna przyczyna została opisana w punkcie 5.3. 5.2. Skuteczność i dokładność rozpoznawania gestów Każdy z utworzonych modeli został również przetestowany w warunkach odpowiednio wcześniej przygotowanych. Specjalnie zaprojektowany program, odtwarzał trzy nagrania których nie było zamieszonych w próbkach modelu, dla każdego spośród 6 istniejących gestów. Następnie za pomocą dostarczonych modeli klasyfikował wykryte gesty. Ocenie została poddana precyzja rozpoznanych gestów z podziałem procentowym. Spośród testowanych gestów jeden z nich otrzymał najgorszy wynik, rozpoznając tym samym inny gest. Pozostałe tj. 5/6 gestów zostało rozpoznanych poprawnie. Zestawienie wykresów wraz z uśrednieniem wyników, zostało zaprezentowane na wykresie 5. 5 Wykres 5 Zestawienie wykresów Uśredniając wyniki spośród trzech testów, gest „Cześć" został rozpoznany z precyzją 60,70%, program wykrył również możliwość zaistnienia gestu „Dzięki" z wynikiem 38,69% oraz „Prosić" (0,58%). Program podczas analizy nagrań

wykrył 2 z 3 wskazanych gestów poprawnie. Podsumowanie analizy gestu „Cześć" zostało przedstawione na poniższym wykresie. 5 Wykres 6 Analiza gestu Cześć - CNN Porównując wyniki z modelem CNN + LSTM, lepiej prezentuje się model CNN. Model składający się z dodatkowej warstwy, błędnie wykrył gest dzięki z precyzją 60,33%, oraz cześć z precyzją 39,67%. 5 W przypadku gestu „Dzięki" model nieprecyzyjnie rozpoznał wskazywany gest. Jedynie na drugiej próbce gest dzięki miał wyższe prawdopodobieństwo wystąpienia niżeli inne opcje. Program błędnie rozpoznał, że wskazanym modelem jest gest Proszę z prawdopodobieństwem 66,13%, drugim możliwym gestem było dzięki z wynikiem 33,44%. Dodatkowo program dostrzegł podobieństwo z gestem prosić (0,24%), Cześć (0,17%) oraz Przepraszać (0,02%). Wykres 7 Analiza gestu Dzięki – CNN W powyższym przypadku lepszy okazał się model z dodatkową warstwą, prezentując następujące wyniki: dzięki (67,62%), proszę (32,11%), cześć (0,26%). 5 W przypadku gestu „Dziękuję" nie było minimalnych wątpliwości. Gest ten został wykryty z precyzją aż 99,98%. Pozostałe 0.02% zostało podzielone po równo pomiędzy gestem „dzięki" i „Cześć". Wykres 8 Analiza gestu Dziękuję - CNN W powyższym przypadku, znacznie lepsza okazała się sieć CNN. Druga sieć wykryła gest: dziękuję (65,22%), dzięki (32,16%), przepraszać (2,38%), cześć (0,18%), prosić (0,05%), proszę (0,01%). 5 Czwarty badany gest (Prosić) został rozpoznany poprawnie z precyzją 75,69%, pozostałe wartości zostały podzielone pomiędzy: cześć (22,34%), proszę (1,88%), dziękuję (0,06%) oraz dzięki (0,03%) Wykres 9 Analiza gestu Prosić – CNN Gest prosić został wykryty poprawnie w każdej konfiguracji sieci, lecz sieć LSTM wykryła gest z większą precyzją, zwracając wynik precyzji: prosić (99,74%), dziękuję (0,24%), dzięki (0,02%), przepraszać (0,01%). 5 Piąty gest, tak jak gest trzeci, został rozpoznany z bardzo wysoką precyzją (99,91%). Pozostałe prawdopodobieństwo zostało podzielone jedynie przez dzięki (0,07%) oraz prosić (0,02%). Wykres 10 Analiza gestu Proszę – CNN W powyższym przypadku sieć CNN okazała się lepsza niżeli LSTM. Druga z wymienionych sieci wykryła gesty: proszę (77,73%), dzięki (21,90%), cześć (0,35%), prosić oraz przepraszać (0,01%). 5 Ostatnim badanym gestem jest gest przepraszać. Gest ten został sumarycznie wykryty z precyzją 49%, możliwe było również wykrycie etykiety dzięki, która miała również wysoki wynik (43,71%). Analizując wyniki zauważyć można także proszę (7,07) oraz cześć (0,22%). Wykres 11 Analiza gestu Przepraszać – CNN Model LSTM wykrył odpowiednio gesty: przepraszać (44,11%), dzięki (38,40%), proszę (17,01%), cześć (0,23%), dziękuję (0,14%) oraz prosić (0,10%). Ogólny wynik precyzji przetwarzanych gestów wskazuje na to, iż model CNN jest bardziej precyzyjny (sumarycznie 417,72%) niżeli model hybrydowy CNN + LSTM (sumarycznie 394,09%). 60 5.3. Analiza błędów i propozycje ulepszeń Platforma, a co za tym idzie również model SI posiada nie tylko swoje zalety ale również wady. W pierwszej kolejności należy wymienić warstwę backendową. Na przykładzie omawianej dysertacji udowodniono, że model hybrydowy w postaci CNN + LSTM, niekoniecznie działa poprawnie z przetwarzaniem obrazów w czasie rzeczywistym. LSTM jest przeznaczony głównie do pracy z sekwencjami danych, które mają wyraźny związek czasowy, takimi jak teksty lub sygnały czasowe. W przypadku obrazu, dane są przestrzenne, a nie czasowe. Do przetwarzania danych obrazowych lepiej nadają się sieci konwolucyjne (CNN), które są zaprojektowane specjalnie do pracy z danymi w formie siatek (takich jak obrazy). Zwiększenie złożoności, w tym przypadku wpłynęło negatywnie na działanie programu. Model wykorzystujący LSTM potrzebował znacznie więcej czasu na przetworzenie sekwencji, co skutkowało zacinaniem się obrazu z kamery oraz powolną predykcją gestów. Precyzja w przetwarzaniu statycznym, również nie została podniesiona. Zaobserwować można było spadek precyzji modelu na danych testowych. W związku z dokonaną analizą, nie zaleca się wykorzystywania LSTM do przetwarzania obrazu w czasie rzeczywistym. Model CNN, można jednakże rozwinąć o podejście hybrydowe CNN + HMM, które polecają autorzy publikacji „Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition.". Zastosowanie powyższego rozwiązania potencjalnie mogłoby podnieść poziom precyzji modelu, bez negatywnego wpływu na stabilność programu. Możliwe jest także dodanie Transformatorów takich jak ST- Transformer, które są aktualnym rozwiązaniem na rok 2024. Dodatkowo, model mógłby być wzbogacony o większą ilość próbek, wliczając w to zróżnicowanych modeli motion capture, różne warunki oświetlenia, różnego typu kamery oraz zmienne tła otoczenia. Powyższe elementy z pewnością poprawiłyby precyzję uczenia modelu. Utworzona platforma, mogłaby również zostać ulepszona poprzez dodanie elementów frontendowych, takich jak responsywność, a także możliwości obsługi wielu języków. Posiada ona również przygotowany moduł, który można rozwinąć o logowanie oraz rejestrację, wprowadzając tym samym system abonamentów (tokenów), znany chociażby z konkurencyjnych aplikacji SI. 61 Podsumowanie Platformy edukacyjne stale cieszą się popularnością. W grupie ww. platform znajdują się aplikacje do

nauki języków obcych z których każdego dnia korzystają setki tysięcy osób, a ich stały rozwój pozwala na zwiększenie komunikatywności w środowisku międzynarodowym. Sytuacja ta jest analogiczna w przypadku zwiększenia świadomości ludzkiej na wszelkiego typu dysfunkcje organizmu. Strony internetowe coraz częściej dostosowują się do różnego rodzaju dyrektyw, chociażby takiej jak WCAG 2.0, które to w jasny sposób określają jak powinny wyglądać strony przyjazne dla osób z niepełnosprawnościami. Pomimo zwiększenia dostępności stron internetowych dla osób z dysfunkcjami, nadal istnieje problem wykluczenia społecznego w warunkach kontaktu bezpośredniego. Osoby, które są głuchonieme, potrzebują obecności tłumacza, który stale przekazuje informacje pomiędzy osobami pełnosprawnymi, a osobą dotknięta dysfunkcją. Niniejsza dysertacja łączy możliwości platform internetowych oraz tematyki niepełnosprawności, tworząc prototyp platformy do rozpoznawania polskiego języka migowego. Pomimo istnienia takowych aplikacji na rynku anglojęzycznym, nadal nie istnieje stabilne rozwiązanie na polskim rynku. W związku z powyższą informacją, praca ta jest innowacyjna, łącząc ze sobą prostotę obsługi, z zaawansowanymi technologiami takimi jak model sztucznej inteligencji rozpoznający gesty języka migowego w czasie rzeczywistym. W ramach niniejszej pracy dyplomowej, opracowany został projekt platformy internetowej, wraz z jej pełną implementacją przy zastosowaniu języków webowych. Dodatkowo na potrzeby dysertacji, autor utworzył kilka modeli sztucznej inteligencji, składających się z różnej ilości próbek opracowanych przez twórcę. Następnie każdy model został przetestowany pod względem wydajność oraz możliwości wykorzystania ich w aplikacji korzystającej z kamery internetowej w czasie rzeczywistym. Finalnie, utworzonych zostało 1800 nagrań o długości 4 sekund, przedstawiających osobę, wykonującą określony gest polskiego języka migowego. Najlepszym modelem okazał się program oparty na architekturze CNN, który wykazał się wysoką precyzją oraz szybkością działania. Należy również uwzględnić, że wszystkie elementy estetyczne w tym grafiki, wizualizacje oraz wykresy zostały opracowane przez autora pracy. Praca dyplomowa, może również zostać rozwinięta o dodatkowe elementy, takie jak responsywność oraz moduł logowania i rejestracji, pozwalający na wykorzystanie ww. dysertacji w celach materialnych. Sugerowane jest również wzbogacenie opracowanego 62 modelu o większą ilość próbek, opracowanych w różnych warunkach oświetleniowych, zmiennym otoczeniu, a także na innych osobach. Przebudowa modelu, dodając warstwę architektury HMM lub ST-Transformer również może zwiększyć precyzję wykrywanych gestów, a co za tym idzie poprawić działanie całej platformy edukacyjno-translatorycznej. 63 Bibliografia oraz Netografia 1. Camgoz, N. C., Koller, O., Hadfield, S., & Bowden, R. (2017). SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition.

[3.1.1 →] IEEE International Conference on Computer Vision (ICCV). 2. Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 7291-7299. 3. Huang, J., Zhou, W., Zhang, Q., Li, H., & Li, W. (2018). Attention-Based 3D-CNNs for Large- Vocabulary Sign Language Recognition. IEEE Transactions on Circuits and Systems for Video Technology, 29(9), 2822-2832. 4. Jamroży D.(2023): Aplikacja internetowa z graficznym interfejsem użytkownika opartym na technologii typu eye-tracking oraz speech recognition. [Praca inżynierska, Uniwersytet Rzeszowski] 5. Koller, O., Zargaran, S., Ney, H., & Bowden, R. (2015). Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition. In Proceedings of the British Machine Vision Conference (BMVC). 6. --- MediaPipe Documentation (n.d.). Google Developers. Retrieved from https://ai.google.dev/edge/mediapipe/solutions/guide?hl=pl 7. Pandey, A., Mishra, M., & Verma, A. K. (2022). Real-Time Sign Language Recognition Using Machine Learning and Neural Networks. IEEE Access. 8. Patel, K., Gil-González, A.-B., & Corchado, J. M. (2022). Deepsign: Sign Language Detection and Recognition Using Deep Learning. Electronics, 11(11), 1780. 9. Patel, M., & Shah, N. (2021). Real-Time Gesture-Based Sign Language Recognition System. IEEE International Conference on Information Technology and Engineering (ICITE). 10. Pigou, L., Dieleman, S., Kindermans, P. J., & Schrauwen, B. (2015). Sign Language Recognition Using Convolutional Neural Networks. European Conference on Computer Vision Workshops (ECCVW). 11. Zhang, Z., & Liu, C. (2020). Skeleton-Based Sign Language Recognition Using Whole-Hand Features. **[←][3.2.1 →]** IEEE Access, 8, 68827-68837. 12. https://www.wsb-nlu.edu.pl/pl/wpisy/wplyw-sztucznej-inteligencji-na-przyszlosc-pracy- nowe-perspektywy-i-wyzwania (22.08.2024) 64 13. https://www.tamoco.com/blog/blog-app-design-app-functionality-ux-ui/ (22.08.2024) 14. https://echoinnovateit.com/ux-or-ui-whats-more-important-in-app-development/ (22.08.2024) 15. McCulloch, W. S., & Pitts, W. (1943). A logical calculus

of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5(4), 115-133. [←]65 Spis ilustracji, tabel oraz wykresów Spis Ilustracji Rysunek 1 Porównanie technologii (Źródło: opracowanie własne) 11 Rysunek 2 Interfejs programu XAMPP 20 Rysunek 3 Etapy zbierania próbek (Źródło: opracowanie własne) 22 Rysunek 4 Uczenie maszynowe (GUI 1) (Źródło: opracowanie własne) 22 Rysunek 5 Uczenie maszynowe (GUI 2) (Źródło: opracowanie własne) 23 Rysunek 6 Uczenie maszynowe (GUI 2 - Struktura) (Źródło: opracowanie własne) 24 Rysunek 7 Uczenie maszynowe (GUI 3) (Źródło: opracowanie własne) 24 Rysunek 8 Uczenie maszynowe (GUI 3 - Wizualizacja) (Źródło: opracowanie własne) 25 Rysunek 9 Uczenie maszynowe (GUI 3 - Zakończenie przetwarzania danych) (Źródło: opracowanie własne) 26 Rysunek 10 Wzór algorytmu gradientu prostego 28 Rysunek 11 Uczenie maszynowe (GUI 4) (Źródło: opracowanie własne) 29 Rysunek 12 Uczenie maszynowe (GUI 4 - Zakończenie pracy) (Źródło: opracowanie własne) .. 30 Rysunek 13 Diagram aktywności (Uczenie maszynowe) (Źródło: opracowanie własne) 31 Rysunek 14 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 33 Rysunek 15 Interfejs graficzny aplikacji – zakładka Home (Źródło: opracowanie własne) 35 Rysunek 16 Zakładka Home (Źródło: opracowanie własne) 35 Rysunek 17 Slider - strona Index.php (Źródło: opracowanie własne) 36 Rysunek 18 Widok zakładki O projekcie (Project.php) (Źródło: opracowanie własne) 36 Rysunek 19 Widok zakładki O nas (About.php) (Źródło: opracowanie własne) 37 Rysunek 20 Uhonorowanie wsparcia (About.php) (Źródło: opracowanie własne) 38 Rysunek 21 Widok zakładki Dashboard (Dashboard.php) (Źródło: opracowanie własne) 38 Rysunek 22 Działanie systemu rozpoznawania gestu (Źródło: opracowanie własne) 39 Rysunek 23 Widok zakładki Tłumacz (Translator.php) (Źródło: opracowanie własne) 39 Rysunek 24 Wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) 40 Rysunek 25 Głosowe wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) 40 Rysunek 26 Wizualizacja gestu (Źródło: opracowanie własne) 41 Rysunek 27 Prezentacja gestu "Dzięki" (Źródło: opracowanie własne) 42 66 Rysunek 28 Synchronizacja wizualizacji z obrazem w czasie rzeczywistym (Źródło: opracowanie własne) 42 Rysunek 29 Diagram przypadków użycia (Tłumacz) (Źródło: opracowanie własne) 43 Rysunek 30 Diagram aktywności (Tłumacz - kamera / wizualizacja) (Źródło: opracowanie własne) 44 Rysunek 31 Diagram aktywności (Tłumacz – Wizualizacja) (Źródło: opracowanie własne) 45 Rysunek 32 Widok zakładki Nauka Języka Migowego (Edu.php) (Źródło: opracowanie własne) 46 Rysunek 33 Gesty wybranej kategorii (Źródło: opracowanie własne) 46 Rysunek 34 Wykonanie poprawnego gestu (Źródło: opracowanie własne) 47 Rysunek 35 Zestawienie metody CNN oraz CNN + LSTM (Źródło: opracowanie własne) 49 Rysunek 36 Proces uczenia maszynowego CNN (Źródło: opracowanie własne) 51 Rysunek 37 Proces uczenia maszynowego CNN + LSTM (Źródło: opracowanie własne) 52 Spis Tabel Tabela 1 Wymagania systemowe cz1 18 Tabela 2 Wymagania systemowe cz2 18 Spis Wykresów Wykres 1 Przetwarzanie próbek CNN 49 Wykres 2 Czas przetwarzania danych – porównanie CNN – CNN + LSTM 50 Wykres 3 Uczenie modelu CNN 50 Wykres 4 Czas uczenia modelu – porównanie CNN – CNN + LSTM 52 Wykres 5 Zestawienie wykresów 54 Wykres 6 Analiza gestu Cześć - CNN 55 Wykres 7 Analiza gestu Dzięki – CNN 56 Wykres 8 Analiza gestu Dziękuję - CNN 57 Wykres 9 Analiza gestu Prosić – CNN 58 Wykres 10 Analiza gestu Proszę – CNN 59 Wykres 11 Analiza gestu Przepraszać – CNN 60 67 Streszczenie: Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego. Niniejsza dysertacja dotyczy utworzenia prototypu platformy do rozpoznawania polskiego języka migowego (PJM) w czasie rzeczywistym poprzez zastosowanie metod uczenia maszynowego, których efektem jest model sztucznej inteligencji (SI). Głównym celem pracy było opracowanie platformy, która umożliwi osobom niesłyszącym łatwiejszą komunikację z otoczeniem bez potrzeby korzystania z dodatkowego sprzętu czy też usług tłumacza, a także zwiększenie ludzkiej świadomości na temat dysfunkcji organizmu. W pracy dokonano również przeglądu technologii i metod związanych z rozpoznawaniem gestów, takich jak konwolucyjne sieci neuronowe (CNN) oraz narzędzia przetwarzania obrazu, między innymi MediaPipe i OpenPose. Autor dysertacji, zaprojektował oraz przeanalizował modele SI, które rozpoznawały gesty PJM na podstawie danych wideo, przetwarzanych oraz trenowanych w modelu CNN. W dalszej części prac, opracowano aplikację webową, która korzysta z kamery internetowej w celu rozpoznania gestów użytkownika w czasie rzeczywistym. Przeprowadzone testy, zarówno w czasie rzeczywistym jak i na przygotowanych wcześniej danych wykazały, że aplikacja działa skutecznie, choć istnieją pewne wyzwania związane z precyzją rozpoznawania gestów, wliczając w to ograniczony dostęp do bazy gestów. Utworzona platforma oferuje potencjalne rozwiązania, które mogą wspierać

osoby z dysfunkcjami słuchu oraz mowy w codziennej komunikacji. 68 Abstract: Prototype and Analysis of a Real-time Polish Sign Language Gesture Recognition Platform Using Machine Learning Method. This dissertation concerns the creation of a prototype of a platform for real-time recognition of Polish Sign Language (PJM) through the use of machine learning methods, which result in an artificial intelligence (AI) model. The main objective of the work was to develop a platform that would enable deaf people to communicate more easily with their surroundings without the need for additional equipment or interpreter services, as well as to increase human awareness of body dysfunctions. The paper also reviews technologies and methods related to gesture recognition, such as convolutional neural networks (CNNs) and image processing tools, including MediaPipe and OpenPose. Author of the dissertation, he designed and analyzed AI models that recognized PJM gestures based on video data processed and trained in a CNN model. Further on, a web application was developed that uses a webcam to recognize user gestures in real time. Tests, both in real time and on pre-prepared data, have shown that the application works effectively, although there are some challenges related to the precision of gesture recognition, including limited access to the gesture database. The created platform offers potential solutions that can support people with hearing and speech impairments in everyday communication. 69 70

Wyrazy o ilości znaków z wybranego zakresu od 1 do 5 znaków

UNIWERSYTET RZESZOWSKI Kolegium Nauk Przyrodniczych Damian Jamroży Nr albumu: 113729 Kierunek: Informatyka Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego Praca magisterska Praca wykonana pod kierunkiem Dr. Inż. Bogusława Twaroga Rzeszów, 2024 Pragnę serdecznie podziękować Panu dr inż. Bogusławowi Twarogowi za nieocenione wsparcie oraz życzliwość okazywaną na każdym etapie mojej edukacji. Jego pomoc i zaangażowanie miały kluczowy wpływ nie tylko na proces powstawania niniejszej pracy magisterskiej, ale także na moją pracę inżynierską i cały tok studiów. Bez Jego merytorycznej wiedzy oraz wsparcia, ukończenie tych etapów byłoby znacznie trudniejsze. Jednocześnie chciałbym serdecznie podziękować moim przyjaciołom z UCI UR. To dzięki ich namowom i wsparciu udało mi się zrealizować ww. etap życia. Spis treści Wstęp 7 Cel i zakres pracy 8 1. Przegląd literatury i analiza istniejących rozwiązań 9 1.1. Przegląd technologii rozpoznawania gestów 9 1.2. Przegląd metod uczenia maszynowego 12 1.3. Wyzwania w rozpoznawaniu gestów języka migowego 13 1.4. Analiza problematyki oraz istniejących aplikacji dotyczących języka migowego 15 2. Techniczne aspekty funkcjonowania aplikacji 17 2.1. Wymagania systemowe 17 2.2. Narzędzia modelowania sztucznej inteligencji 19 3. Budowa modelu rozpoznawania gestów 21 3.1. Przygotowanie danych wejściowych 21 3.2. Preprocessing danych 24 3.3. Architektura modelu oraz algorytm klasyfikacji 26 3.4. Trening, walidacja oraz ocena skuteczności modelu 27 4. Implementacja platformy 32 4.1. Koncepcja wizualna platformy 32 4.2. Mechanizmy komunikacji między frontendem a backendem 33 4.2.1. Komunikacja serwer – użytkownik 33 4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) 34 4.3. Widoki aplikacji 34 5. Ocena działania platformy 48 5.1. Testy w rzeczywistych warunkach 48 5.1.1. Ekstrakcja punktów kluczowych 48 5.1.2. Uczenie modelu 50 .1.3. Testy w czasie rzeczywistym 52 5.2. Skuteczność i dokładność rozpoznawania gestów 53 5.3. Analiza błędów i propozycje ulepszeń 61 Podsumowanie 62 Bibliografia oraz Netografia 64 Spis ilustracji, tabel oraz wykresów 66 Wstęp Obecny rozwój technologii pozwala na automatyzację wielu procesów, w tym procesów finansowych, administracyjnych oraz sprzedażowych. Wymienione elementy są związane z obsługą biznesów, które umożliwiają komercyjny zarobek twórcom oprogramowania. Algorytmy, mają w tym przypadku ułatwić pracę lub całkowicie zastąpić osoby fizyczne w ich obowiązkach. Dzięki takim praktykom, pracodawcy, czy też różnego rodzaju organizacje, zwiększają swoje dochody poprzez przyspieszenie wykonywanej pracy lub w gorszym przypadku, oszczędzają fundusze poprzez zredukowanie etatów. Szerokie zastosowanie sztucznej inteligencji jest widoczne w każdym aspekcie naszego życia. Coraz większa ilość sklepów, banków czy też producentów wszelkiego rodzaju produktów, decyduje się na wdrażanie sztucznej inteligencji. Zgodnie z opinią specjalistów z Wyższej Szkoły Biznesu National-Louis University, AI (Artificial Intelligence) może powodować utratę miejsc pracy oraz restrukturyzację zawodów, jednakże tym samym może zwiększać zapotrzebowanie na specjalistów w branży kreatywnej oraz IT. Autor artykułu, zwraca uwagę na problematykę dotyczącą etyki związanej ze sztuczną inteligencją oraz potrzebę nieustannej nauki i rozwoju[12]. Istnieją również aspekty SI (Sztucznej Inteligencji), które są niezaprzeczalnie pozytywne, chociażby zastosowane w strefach pożytku publicznego, czy też w rozwiązaniach dla osób z dysfunkcjami. Wszelkiego rodzaju protezy, pojazdy, syntezatory mowy, algorytmy analizujące tekst, dźwięk czy też obraz, pozwalają na łatwiejsze funkcjonowanie osób niepełnosprawnych. Niestety obszary te są często pomijane, ze względu na stosunkowo niewielkie grono odbiorców, gotowych zapłacić za wprowadzenie takowych rozwiązań. Podejmując dalszą próbę analizy problemu, możemy zaobserwować wysokie zainteresowanie wadami wzroku oraz problemami ruchowymi, natomiast niskie zainteresowanie dysfunkcją głosową w odniesieniu do osób głuchoniemych. Istnieje wiele rozwiązań, które ułatwiają kontakt wzrokowy, chociażby takie jak regulacja wielkości czcionek we wszelkiego rodzaju aplikacjach, asystenci głosowi, czy też operacyjne korekty wzroku. Funkcje ruchowe, wspierane są przez różnorodne protezy, pojazdy, a także specjalne miejsca dostosowane do ich potrzeb np. miejsca parkingowe. Niestety w odniesieniu do problematyki osób głuchoniemych, nie ma zbyt wielu rozwiązań technologicznych, które mogą ułatwić ich życie w sposób nieinwazyjny. Cel i zakres pracy Celem pracy dyplomowej jest wparcie ww. grupy docelowej poprzez utworzenie oraz analizę oprogramowania do rozpoznawania sekwencji ruchów w czasie rzeczywistym, w oparciu o metody uczenia maszynowego. Zastosowane rozwiązania, wykorzystane zostaną

następnie w aplikacji służącej do nauki oraz tłumaczenia polskiego języka migowego. Oprogramowanie to może służyć jako wsparcie osób z dysfunkcjami w łatwiejszej adaptacji w środowisku osób pełnosprawnych. Aplikacja ta, również może działać w sposób edukacyjny, zwiększając świadomość użytkowników na temat dysfunkcji słuchowych oraz głosowych. Opracowanie autorskich skryptów, pozwalać ma na obsługę pełnego procesu pracy aplikacji, począwszy od rejestrowania próbek nagrań wideo, uczenia modelu SI, testowaniu jego poprawności, a kończąc na obsłudze aplikacji internetowej[4]. Niniejsza dysertacja, składa się z pięciu rozdziałów. Pierwszy z nich omawia zagadnienia teoretyczne związane z funkcjonowaniem aplikacji, takie jak przegląd dostępnych technologii związanych z rozpoznawaniem gestów oraz uczeniem maszynowym, wyzwaniami w rozpoznawaniu gestów oraz analizą istniejących platform dotyczących języka migowego. Następny rozdział omawia aspekty techniczne, które muszą zostać spełnione aby aplikacja działała w pełni poprawnie. W powyższym rozdziale znajdują się takie elementy jak: wymagania systemowe, zalecane oprogramowanie zewnętrzne oraz biblioteki, które należy zainstalować na urządzeniu docelowym. Rozdział trzeci porusza tematykę tworzenia modelu sztucznej inteligencji, przechodząc po każdym etapie jego powstawania, zaczynając od przygotowaniu danych wejściowych, a kończąc na ocenie skuteczności stworzonego modelu. Rozdział czwarty, zatytułowany „Implementacja platformy" skupia się na połączeniu aspektów sztucznej inteligencji wraz z aplikacją internetową. Zaprezentowane tam informacje dotyczą koncepcji wizualnej platformy, integracji modeli AI z aplikacją webową, mechanizmów komunikacji między klientem a serwerem oraz warstwy graficznej interfejsu użytkownika. Piąty, a zarazem ostatni rozdział niniejszej dysertacji omawia ocenę działania platformy, skupiając się na testach w warunkach rzeczywistych oraz symulowanych, ocenie skuteczności pracy aplikacji oraz propozycji ulepszeń oprogramowania. Praca zakończona została podsumowaniem, w którym zaprezentowano ogólny opis aplikacji wraz z jej analizą pod kątem przydatności oraz wydajności w rzeczywistych warunkach. 1 . Przegląd literatury i analiza istniejących rozwiązań Rozpoznawanie wszelkiego rodzaju obiektów oraz ich właściwości, zyskały w ostatnich latach na popularności, głównie dzięki postępom w technikach uczenia maszynowego oraz rozwoju głębokich sieci neuronowych. W niniejszym rozdziale, omówione zostaną najważniejsze technologie oraz metody stosowane w tematyce rozpoznawania gestów, ze szczególnym uwzględnieniem języka migowego. 1.1. Przegląd technologii rozpoznawania gestów Technologie rozpoznawania gestów opierają się na stałej analizie ruchów ciała, ze szczególnym uwzględnieniem dłoni i palców, przy pomocy różnych sensorów oraz kamer. Systemy te wykorzystują zarówno dane wideo, jak i informacje trójwymiarowe, uzyskane za pomocą czujników głębi. Interdyscyplinarna dziedzina badań rozpoznawania gestów, łączy ze sobą elementy przetwarzania obrazów, komputerowego widzenia, sztucznej inteligencji oraz interakcji człowiek-komputer. Podstawą rozpoznawania gestów jest przetwarzanie obrazu, które obejmuje szereg technik służących do analizy danych wizualnych. Można wyłonić takie etapy jak: detekcja obrazów, śledzenie ruchu oraz ekstrakcja cech. Detekcja obrazów opiera się na rozpoznaniu dłoni oraz pozostałych istotnych części ciała na przetwarzanym obrazie. Techniki te bazują na algorytmach segmentacji obrazu, które identyfikują obiekty na podstawie ich koloru, kształtu lub ruchu. Algorytmy takie jak YOLO (You Only Look Once) oraz SSD (Single Shot Multibox Detector) są powszechnie stosowane do szybkiej i dokładnej detekcji dłoni w badanych obrazach. Śledzenie ruchu, opiera się na analizie każdej klatki nagrania w celu uchwycenia przesunięć, występujących w punktach wykrytych przez poprzedni proces. W tym celu stosuje się miedzy innymi algorytmy KLT (Kanade-Lucas-Tomasi) oraz Mean Shift, które pozwalają na dokładne monitorowanie trajektorii ruchu dłoni. Ostatnią techniką jest ekstrakcja cech, która umożliwia dokonanie wyodrębnienia z obrazów stawów oraz kości, niezbędnych do identyfikacji wykonanego gestu. W tym celu stosowane są takie narzędzia jak OpenPose oraz MediaPipe[3]. Biblioteka OpenPose została utworzona przez Perceptual Lab na Uniwersytecie Carnegie Mellon (CMU) , pod nadzorem profesora Asuyuki Matsumoto oraz doktoranta Zhe Cao, który jest głównym autorem pracy badawczej, opisującej możliwości biblioteki OpenPose. Pierwsza wersja OpenPose została opublikowana w 2017 roku, a jej wyniki były oparte na pracy naukowej pt. "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", której współautorem był wspomniany wcześniej Zhe Cao. Praca ta po raz pierwszy zaprezentowana została na konferencji CVPR (Computer Vision and Pattern Recognition) w 2017 roku. Początkowo OpenPose służył głównie do analizy oraz detekcji pozycji wielu osób w czasie rzeczywistym na obrazach 2D. Pierwsza wersja koncentrowała się na problemie, który dotąd stanowił duże wyzwanie, a mianowicie detekcji pozycji ciała wielu osób na jednym obrazie, w jak najszybszym czasie, z zachowaniem wysokiej precyzji. Istniejące wcześniej rozwiązania, były skoncentrowane głównie na wykryciu jednego człowieka lub działały wolniej, co uniemożliwiało

zastosowanie ich w czasie rzeczywistym. Biblioteka ta zyskała spore zainteresowanie, co przyczyniło się do rozszerzenia jej o dodatkowe funkcje, takie jak detekcja szczegółowych ruchów dłoni i mimiki twarzy oraz analizy 3D. Głównym zadaniem obecnej wersji biblioteki jest wykrywanie oraz śledzenie pozycji ciała człowieka w obrazie 2D oraz 3D, uwzględniając różne części ciała takie jak kończyny, głowa oraz palce rąk i stóp. Podstawą działania OpenPose są techniki głębokiego uczenia, a dokładniej konwolucyjne sieci neuronowe (CNN), które za pomocą punktów kluczowych (tzw. keypoints), rozpoznają położenie ciała, dzieląc je na odpowiednie struktury. W początkowej fazie programu, biblioteka przeprowadza detekcję punktów kluczowych, następnie tworzy mapę ufności, która określa z jakim prawdopodobieństwem dany punkt (np. łokieć) znajduje się w określonej lokalizacji na obrazie. Utworzona mapa jest dwuwymiarową siatką, gdzie wartości w poszczególnych komórkach reprezentują stopień pewności. Po utworzeniu mapy ufności, biblioteka tworzy mapy części ciała, analizując położenie konkretnych elementów, co w dalszym etapie służy do utworzenia szkieletu osoby. W tym celu wykorzystywane są mapy PAF (Part Affinity Fields), czyli wektorowe pola, które wskazują, jak prawdopodobne jest, że dwa punkty kluczowe są ze sobą powiązane. Gdy mapy te zostaną utworzone, biblioteka przystępuje do połączenia punktów tworząc szkielet[2]. Druga spośród wymienionych bibliotek (MediaPipe) została udostępniona jako projekt open-source przez firmę Google w czerwcu 2019 roku. MediaPipe, jest to otwartoźródłowe środowisko, które umożliwia wykonywanie różnych zadań związanych z przetwarzaniem obrazów, wideo oraz danych w czasie rzeczywistym. Zbudowana została na podstawie struktury pipelinów przetwarzania multimodalnych danych, co oznacza, że pozwala na tworzenie i uruchamianie strumieniowych systemów analizy danych, w sposób modułowy oraz wydajny. Pipeline to zbiór skomponowanych kroków przetwarzania (tzw. modułów) w ustalonej kolejności. Każdy z modułów odpowiedzialny jest za określone zadanie np. detekcję obiektów, segmentację czy śledzenie ruchu. Dzięki temu, biblioteka ta jest bardzo elastyczna, gdyż użytkownik może z łatwością tworzyć własne pipeline'y, dostosowując je do specyficznych potrzeb. Pierwotnym założeniem MediaPipe było dostarczenie wydajnej biblioteki, która umożliwiałaby wieloplatformowe przetwarzanie strumieni wideo i obrazów w czasie rzeczywistym[6]. Przechodząc do kwestii hardware, najczęściej wykorzystywanymi urządzeniami do rejestracji gestów są kamery RGB. Ich popularność jest uargumentowana stosunkowo niską ceną oraz dostarczanym kolorowym obrazem w wysokiej rozdzielczości, który może być przetwarzany przez algorytmy komputerowego widzenia. Nie są to jednakże jedyne narzędzia, wykorzystywane w procesie rejestrowania ruchów. Elementy takie jak kamery głębi pozwalają na trójwymiarową rejestrację sceny, co umożliwia bardziej precyzyjną analizę ruchów dłoni i ciała. Dużą zaletą kamer głębi jest wysoka dokładność w trudnych warunkach oświetleniowych, gdzie w porównaniu z kamerami RGB, nie mają problemów z dokładnym uchwyceniem gestu. Dodatkowym asortymentem, mogą być również czujniki ruchu, które mierzą przyspieszenie i orientację dłoni. Zastosowanie ww. technologii, pozwala na rejestrowanie subtelnych ruchów i położenia dłoni w przestrzeni. Ostatnia z przestawionych opcji, daje największe możliwości rejestrowania ruchów. Niestety ze względu na koszt zaawansowanych technologii stosowanych w czujnikach ruchu, produkty te są nieprzystępne cenowo dla potencjalnego odbiorcy oprogramowania, omawianego w niniejszej dysertacji[9]. Rysunek 1 Porównanie technologii (Źródło: opracowanie własne) 1.2. Przegląd metod uczenia maszynowego Uczenie maszynowe (ang. Machine Learning (ML)) to dziedzina sztucznej inteligencji, która skupia się na rozwijaniu algorytmów oraz modeli zdolnych do uczenia i automatycznego podejmowania decyzji na podstawie dostarczonych danych. W odniesieniu do platformy tłumaczącej język migowy jest kluczowym rozwiązaniem, umożliwiając przekształcanie danych wideo w rozpoznawalne wzorce, które są możliwe do przetłumaczenia na słowa lub zdania. Pierwsza sieć neuronowa została opracowana przez Warrena McCullocha, wykładowcy z University of Illinois i Waltera Pittsa, niezależnego badacza. Ich artykuł pt. „A Logical Calculus of the Ideas Immanent in Nervous Activity", opublikowany w 1943 roku w czasopiśmie „Bulletin of Mathematical Biophysics" został przyjęty z szerokim entuzjazmem, kładąc tym samym podwaliny pod rozwój sztucznych sieci neuronowych. Praca ta okazała się kamieniem milowym w późniejszych badaniach w dziedzinie informatyki, neurobiologii i kognitywistki. Autorzy, przedstawili w nim jak mogą działać neurony, prezentując swoje pomysły i tworząc prostą sieć neuronową za pomocą obwodów elektrycznych[15]. Obecnie sieci neuronowe są fundamentem współczesnych metod uczenia maszynowego. Składają się z wielu warstw neuronów, które przetwarzają dane wejściowe poprzez szereg operacji matematycznych, co umożliwia uczenie się złożonych wzorców. Konwolucyjne sieci neuronowe (CNN) są szczególnie efektywne w przetwarzaniu danych obrazowych oraz wideo, dzięki swojej zdolności do automatycznego wyodrębnienia cech z surowych danych. Sieci te, składają się z warstw konwolucyjnych, poolingowych oraz

gęstych, które współpracują ze sobą, w celu przetwarzania informacji o strukturze przestrzennej obiektów. Autorzy publikacji Sign Language Recognition Using Convolutional Neural Networks wykazali, że CNN mogą być używane do skutecznego przetwarzania i rozpoznawania gestów na podstawie obrazu wideo, umożliwiając automatyczną ekstrakcję cech, co jest kluczowe dla poprawnej klasyfikacji gestów. Podkreślili również, że ich model nie tylko osiąga wysoką dokładność w rozpoznawaniu gestów, ale także jest skalowalny i może być rozszerzany o nowe gesty lub języki migowe z relatywnie niewielkim wysiłkiem. Badanie to otwiera drogę do tworzenia bardziej zaawansowanych systemów rozpoznawania języka migowego, które mogą być wykorzystywane w aplikacjach takich jak tłumacze języka migowego na mowę w czasie rzeczywistym, tworzenie interfejsów użytkownika dla osób niesłyszących, a także w edukacji[10]. Modele hybrydowe to połączenie różnych algorytmów klasyfikacyjnych, które mogą prowadzić do poprawy dokładności i stabilności rozpoznawania gestów. Przykładem takiego podejścia jest kombinacja CNN oraz HMM, co pozwala na skuteczniejsze modelowanie zależności czasowych w danych sekwencyjnych. Zastosowanie hybrydy modeli CNN oraz HMM opisuje publikacja pt. „SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition". Autorzy publikacji wykazują, że hybrydowe podejście może być skutecznie wykorzystywane do rozpoznawania ciągłych gestów w języku migowym. Ich badania przeprowadzone zostały na dużych zestawach danych, zawierających filmy z gestami języka migowego, przy czym wykorzystali takie techniki jak augmentacja danych, zwiększając tym samym różnorodność przykładów i poprawiając zdolność generalizacji modelu. Wyniki eksperymentu pokazują, że SubUNets osiąga znacznie lepsze wyniki w porównaniu z wcześniejszymi podejściami, zarówno w kontekście rozpoznawania kształtów dłoni, jak i ciągłego rozpoznawania gestów języka migowego. Autorzy podkreślają, że ich architektura nie tylko rozpoznaje gesty z większą dokładnością, ale także lepiej radzi sobie z różnorodnością kształtów dłoni i płynnością sekwencji[1]. 1.3. Wyzwania w rozpoznawaniu gestów języka migowego Rozpoznanie gestów języka migowego stanowi złożone wyzwanie, które wynika z unikalnych cech ww. sposobu komunikacji. W przeciwieństwie do języków mówionych, język migowy wykorzystuje mimikę oraz posturę ciała, co wymaga zaawansowanych algorytmów do skutecznej analizy i rozpoznania. Język ten charakteryzuje się ogromną różnorodnością gestów. Każdy z nich może mieć wiele wariantów, które różnią się w zależności od regionu, kultury, a nawet osoby wykonującej gest. Ponadto, gesty mogą obejmować stosunkowo nieistotne różnice w ruchach dłoni, ułożeniu palców, kierunku spojrzenia, a także w mimice twarzy, co sprawia, że poprawne rozpoznanie jest niezwykle trudne, zwłaszcza jeżeli model zostaje rozszerzany o nowe gesty oraz ich warianty. Badania prowadzone przez Koller i innych naukowców wskazują, że klasyczne podejścia oparte na modelach HMM mogą być niewystarczające do modelowania złożonych i różnorodnych gestów języka migowego. Aby sprostać temu wyzwaniu, autorzy sugerują zastosowanie hybrydowych modeli łączących CNN oraz HMM, co pozwala na skuteczniejsze modelowanie złożoności gestów[5]. Gesty języka migowego często wykonywane są w sposób płynny oraz nieprzerwany co stwarza jeden z największych problemów w przypadku zastosowania algorytmów SI. Kwestia ta jest niezwykle problematyczna dla modeli, które do poprawnego działania potrzebują wyraźnych przerw pomiędzy poszczególnymi znakami, aby określić gdzie badany gest się zaczyna, a gdzie kończy. Kluczowym problemem podczas tworzenia modelu sztucznej inteligencji jest również sama tematyka. W związku z niszowością tematu, utrudniony jest dostęp do dużych, zróżnicowanych zbiorów danych, które są niezbędne dla skutecznego trenowania modeli uczenia maszynowego. W przypadku języka migowego, zbiory danych są ograniczone pod względem wielkości i różnorodności. Brakuje danych pochodzących od różnych użytkowników, wykonanych w zróżnicowanych warunkach oświetleniowych i środowiskach, co może prowadzić do słabszej generalizacji modelu. Problematykę tą porusza w swoich badaniach Patel oraz inni naukowcy, zwracając uwagę na problem ograniczonych zbiorów danych, proponując jednocześnie zastosowanie techniki transfer learningu, które pozwalają na wykorzystanie wiedzy zebranej na większych zbiorach danych do trenowania modeli na mniejszych, specyficznych zbiorach języka migowego[6]. Rozpoznawanie gestów w czasie rzeczywistym wymaga również znacznej mocy obliczeniowej, zwłaszcza w przypadku samego tworzenia modelu oraz ekstrakcji punktów na zbiorach treningowych. Programy wykorzystujące przetwarzanie obrazu w czasie rzeczywistym muszą przetwarzać duże ilości danych wideo o wysokiej rozdzielczości, minimalizując poziom opóźnienia do minimum. Optymalizacja rozwiązań w stosunku do urządzeń o gorszych parametrach również stwarza dodatkowe problemy, zwłaszcza w przypadku urządzeń mobilnych. Ostatnim powszechnie znanym wyzwaniem może być różnica w indywidualnym zachowaniu użytkownika. Każda osoba ma inną długość oraz szerokość kończyn. Inna szybkość, kąt nachylenia czy też indywidualny styl poruszania się każdej osoby sprawia,

że opracowany model musi być odporny na takowe odchylenia. Na elastyczność modelu zwraca uwagę między innymi Zhang i Liu, wskazując konieczność opracowania metod, które mogą uwzględnić różnice indywidualne, takie jak ww. style gestykulacji, poprzez wykorzystanie cech całej dłoni, co pozwala na dokładniejsze rozpoznawanie gestów w różnych warunkach[11]. 1.4. Analiza problematyki oraz istniejących aplikacji dotyczących języka migowego W ciągu ostatnich lat rozwój technologii głębokiego uczenia oraz komputerowego widzenia przyczynił się do powstania wielu aplikacji, służących do rozpoznawania i tłumaczenia języka migowego. Aplikacja opracowana przez Patel i Sahah, pozwala użytkownikom uczyć się języka migowego za pomocą interaktywnych ćwiczeń. System ten, analizuje gesty użytkownika za pomocą technologii rozpoznawania wideo w czasie rzeczywistym, umożliwiając otrzymanie natychmiastowej informacji zwrotnej, co przyczynia się do skuteczniejszej nauki[8]. Aplikacja o nazwie DeepSign, również funkcjonuje w przestrzeni publicznej jako tłumacz języka migowego. Aplikacja ta wykorzystuje CNN do rozpoznawania gestów i przekształcania ich w tekst w czasie rzeczywistym. System ten jest bardzo zaawansowany, co pozwala tłumaczyć gesty na pełne zdania, co znacznie ułatwia komunikację[7]. Niestety każda z wyżej wymienionych aplikacji jest stworzona na rynek anglojęzyczny, a co za tym idzie, nie obsługuje polskiego języka migowego. Zapoznając się z aktualną na dzień 27.08.2024 ofertą na rynek polskojęzyczny, można odnaleźć jedynie rozwiązania, które nie są zautomatyzowane. Specjalne ośrodki czy też firmy prywatne obsługują klientów w ramach spotkań lub połączeń online z tłumaczem języka migowego. Analizując problematykę zastosowania sztucznej inteligencji, można dostrzec, iż nisza ta nie została jeszcze odpowiednio obsłużona w Polsce. Rozwiązania takie jak wideokonferencje, są skuteczną alternatywą, jednakże wymagają osoby fizycznej, która takową rozmowę przetłumaczy. Wiąże się to z kosztami zatrudnienia specjalisty oraz utrzymania infrastruktury, takiej jak stabilne połączenie z Internetem, odpowiedni sprzęt audio-wizualny oraz platformy obsługującej powyższe rozmowy. Zagłębiając się bardziej w poruszaną tematykę, można odnieść wrażenie, że osoby z dysfunkcjami instrumentów głosowych są w pewien sposób wykluczone społecznie, poprzez brak powszechnego narzędzia, które ułatwiałoby im życie codzienne w sferze komunikacyjnej. W przypadku każdego typu rozmów, należy skupić się na zagadnieniu prywatności, do której każdy obywatel ma prawo. Chociażby w taki sferach jak zdrowie czy też finanse. Dzięki istnieniu aplikacji, która byłaby w stanie w pełni przetłumaczyć język migowy, osoby z dysfunkcjami mowy, mogłyby korzystać z większości znanych usług, bez obecności osoby trzeciej, która będzie towarzyszyć w prowadzeniu rozmowy. Zaleca się aby aspekt ten został szerzej zbadany przez specjalistów w dziedzinie badań społecznych, analizując jednocześnie potrzeby osób niepełnosprawnych, a tym samym zwiększając świadomość społeczeństwa na temat poruszany w omawianej dysertacji. 2 . Techniczne aspekty funkcjonowania aplikacji Każde oprogramowanie wymaga uprzedniego przygotowania środowiska pracy. Tak też jest w przypadku aplikacji dołączonej do niniejszej dysertacji. Wybrane etapy posiadają własne wymagania odnośnie sprzętu (ang. hardware) , oprogramowania (ang. software) oraz środowiska programistycznego, dlatego w poniższych punktach przedstawione zostały kroki do poprawnej konfiguracji środowiska. 2.1. Wymagania systemowe Program podzielony został na dwa etapy główne. Pierwszy etap odpowiada za przygotowanie próbek oraz modelu sztucznej inteligencji, natomiast drugi etap jest stricte związany z platformą, która ma za zadanie odczytywać przygotowany w etapie pierwszym model SI. Każdy program posiada inne wymagania systemowe, dlatego też zostały one zawyżone w odniesieniu do najbardziej wymagającego programu z etapu pierwszego. Komponent Minimalne wymagania Zalecane wymagania Procesor (CPU) Sześciordzeniowy, np. Intel Core i5- Ośmiordzeniowy, np. Intel Core 10600K lub AMD Ryzen 5 3600 i7-9700K lub AMD Ryzen 7 3700X Pamięć RAM 16 GB RAM 32 GB RAM Karta graficzna Zintegrowana Zintegrowana (GPU) Przestrzeń Minimum 20 GB wolnej przestrzeni na SSD NVMe z minimum 100 GB dyskowa dysku SSD wolnej przestrzeni System Windows 10 lub nowszy / Linux (Ubuntu Windows 10 Pro lub nowszy / operacyjny 20.04 lub nowszy) Linux (Ubuntu 20.04 LTS lub nowszy) Inne Python 3.8 lub nowszy Python 3.8 lub nowszy Informacje Procesor ośmiordzeniowy z obsługą wielowątkowości zapewni dobrą wydajność w przetwarzaniu wideo i trenowaniu modeli, choć na nieco niższym poziomie niż wcześniej wspomniane opcje. Zintegrowana karta graficzna wystarczy do obsługi podstawowych zadań związanych z wyświetlaniem i przetwarzaniem obrazu, ponieważ główne obciążenie będzie przeniesione na procesor, aby zmienić obciążenie z procesora na kartę graficzną należy dokonać modyfikacji programu „MachineLearning.py”. 32 GB RAM pozwoli na komfortową pracę z większymi zbiorami danych i bardziej złożonymi modelami Tabela 1 Wymagania systemowe cz1 Druga część programu opiera się głównie na obsłudze aplikacji internetowej oraz przetwarzaniu obrazu z kamery na podstawie dostarczonego już modelu, co nie wymaga od użytkownika zwiększonej mocy

obliczeniowej. Komponent Minimalne wymagania Zalecane wymagania Procesor Czterordzeniowy procesor, np. Intel Sześciordzeniowy procesor, np. Intel (CPU) Core i5-8265U lub AMD Ryzen 5 Core i7-10750H lub AMD Ryzen 5 2500U 4600H Pamięć RAM 8 GB RAM 16 GB RAM Karta Zintegrowana, np. Intel UHD Zintegrowana, np. Intel Iris Xe lub graficzna Graphics 620 lub AMD Radeon AMD Radeon Vega 11 (GPU) Vega 8 Przestrzeń Minimum 10 GB wolnej przestrzeni SSD NVMe z minimum 50 GB wolnej dyskowa na dysku SSD przestrzeni System Windows 10 lub nowszy / Linux Windows 10 Pro lub nowszy / Linux operacyjny (Ubuntu 20.04 lub nowszy) (Ubuntu 20.04 LTS lub nowszy) Inne Python 3.8 lub nowszy Python 3.8 lub nowszy Tabela 2 Wymagania systemowe cz2 2.2. Narzędzia modelowania sztucznej inteligencji Zgodnie z informacjami przedstawionymi w podpunkcie 2.1, każdy program posiada inne wymagania sprzętowe. W tym punkcie przedstawione zostaną indywidualne wymagania dotyczące poszczególnych programów, które nie zostały uwzględnione w poprzednich punktach. Pakiety wymagane do uruchomienia poszczególnych aplikacji, można zainstalować poprzez narzędzie do zarządzania pakietami w Pythonie (pip), wpisując wskazane komendy w terminalu. Program „NagrywanieVideo720p60FPS.py" do poprawnego działania potrzebuje podłączonej kamery internetowej obsługującej jakość 720p oraz przepustowość 60fps, a także importu pakietów takich jak os, time oraz opencv. Dwa pierwsze pakiety są dostępne w podstawowej wersji Pytthona, natomiast pakiet opencv musi zostać doinstalowany. Aby tego dokonać można skorzystać z poniższego polecenia: pip install opencv-python Do obsługi programu „MachineLearning.py" wymagane są pakiety: os, threading, warnings, random, time, webbrowser, multiprocessing, tkinter, cv2, numpy, mediapipe, tensorflow, keras, flatten, scikit-learn oraz pil , osiem pierwszych pakietów to standardowe moduły języka Python, brakujące pakiety można zainstalować inicjując polecenie: pip install opencv-python numpy mediapipe tensorflow keras scikit-learn pillow Program „Prediction Test.py" wymaga takich pakietów jak: cv2 , numpy, mediapipe, tensorflow, matplotlib, keras, sklearn, os, tkinter. Brakujące pakiety można zainstalować za pomocą polecenia: pip install opencv-python numpy mediapipe tensorflow keras scikit-learn pillow Do poprawnej obsługi platformy, wymagane są również odpowiednie środki, takie jak pakiety języka python oraz zewnętrzne oprogramowanie inicjujące środowisko serwerowe. Omawiana platforma wymaga utworzenia lokalnego serwera w celu obsługi języka PHP. Przykładem oprogramowania pozwalającego na zainicjowanie ww. środowiska jest XAMPP. Do utworzenia niniejszej aplikacji wykorzystano najnowszą (na dzień 22.08.2024) wersję XAMPP w wersji 3.3.0 oraz PHP w wersji 8.2.12. Po zainstalowaniu aplikacji XAMPP, należy uruchomić opcję Apache, oznaczoną na Rysunku 2 numerem 1. Rysunek 2 Interfejs programu XAMPP Platforma, została projektowana oraz testowana za pomocą przeglądarki internetowej Chrome, która jest zalecana do obsługi ww. aplikacji. Program „app.py" potrzebuje podłączonej kamery internetowej o przepustowości minimum 30 fps oraz zainstalowanych pakietów takich jak: Flask, flask-cors, opencv, numpy, mediapipe, tensorflow, keras, pillow oraz scikit-learn. Elementy te, można zainstalować poprzez komendy w terminalu: pip install Flask flask-cors opencv-python numpy mediapipe tensorflow keras Pillow scikit-learn 3. Budowa modelu rozpoznawania gestów Aplikacja internetowa do właściwego działania potrzebuje wcześniej przygotowanego modelu sztucznej inteligencji, który będzie rozpoznawał wybrane sekwencje ruchów. W tym rozdziale zostanie przedstawiony opis procesu przygotowania ww. modelu. 3.1. Przygotowanie danych wejściowych Przygotowanie danych wejściowych jest kluczowym etapem każdego programu opartego na uczeniu maszynowym. Proces ten obejmuje takie elementy jak wybór odpowiednich źródeł danych, ich organizację, walidację oraz przetworzenie do odpowiedniego formatu. W związku z niszowością tematyki polskiego języka migowego oraz utrudnionym dostępem do próbek badawczych, autor dysertacji utworzył własną bazę gestów, które poprzez algorytm napisany w języku python zostały zarejestrowane za pomocą kamery internetowej. Każde z nagrań musiało zostać ustandaryzowane w celu przystosowania ich do pracy z jednolitym modelem sztucznej inteligencji. Każda z zarejestrowanych próbek posiadała długość 4 sekund, rozdzielczość 720p oraz przepustowość 60 klatek na sekundę. Dodatkowo, autor przyjął metodę 25-50-25, która zaowocowała nagraniem 100 próbek dla każdego gestu. W późniejszym etapie projektu powtórzono ten proces, aby zyskać większą ilość próbek na innych modelach ciał. Metoda ta opierała się na odpowiednim ustawieniu kadru. Pierwszy etap tworzył 25 nagrań od czubka głowy do dolnej części żeber. Taka konfiguracja, pozwalała na najdokładniejsze zarejestrowanie pracy dłoni, wliczając w to układ palców. Drugi etap tworzył 50 nagrań od czubka głowy do górnej części uda. Nagrania te służyły jako baza do całości gestu. Ujęcia te, ujmowały każdą fazę ruchu, od fazy startowej do fazy końcowej, uwzględniając wszystkie niezbędne elementy ludzkiego działa. Ostatni etap tworzył 25 nagrań od czubka głowy do dolnej części kolan. Proces ten miał za zadanie zwiększyć obszar widzenia

programu, uwzględniając dalsze położenie szukanych punktów, dzięki czemu utworzony w późniejszym etapie model, mógł z większą precyzją rozpoznawać pozycję statyczną oraz ruch odpowiednich fragmentów ciała człowieka. Ustawienie każdego z etapów zostało zobrazowane na ilustracji 3. Rysunek 3 Etapy zbierania próbek (Źródło: opracowanie własne) Dzięki zastosowaniu ww. rozwiązania, późniejsze algorytmy przetwarzania danych, mogły z większą łatwością zlokalizować wybrane punkty na ludzkim ciele, a następnie przetworzyć je w ramach sekwencji ruchu. Każdy z etapów dostarczał inną ilość punktów, dzięki czemu model uczył się rozpoznawania gestów ze zróżnicowanej perspektywy. W ramach części praktycznej pracy magisterskiej, został dołączony program „NagrywanieVideo720p60FPS.py", który realizuje wyżej wymienione etapy. Po zakończeniu procesu przygotowywania próbek badawczych autor dysertacji opracował program „MachineLearning.py", który realizuje wszystkie pozostałe kroki do uzyskania działającego modeli SI. Po uruchomieniu ww. programu, użytkownik ma możliwość skorzystania z graficznego interfejsu programu, utworzonego poprzez bibliotekę tkinter. Biblioteka ta jest standardowym pakietem w języku Python. Początkowo, program weryfikuje czy w obecnym katalogu istnieje plik z przetworzonymi punktami o nazwie „preprocessed_data.npz". Jeżeli takowy plik istnieje, zostaje wyświetlone okno z odpowiednim komunikatem (Rys.4). Rysunek 4 Uczenie maszynowe (GUI 1) (Źródło: opracowanie własne) Użytkownik może wybrać jedną z trzech dostępnych opcji. Pierwsza z nich prowadzi do pliku PDF, w którym opisane zostały wymagania sprzętowe dotyczące obsługi programu. Przycisk po prawej stronie, rozpoczyna proces uczenia maszynowego na obecnie wykrytych danych, natomiast przycisk „Wróć do wyboru danych" pozwala na otworzenie widoku okna startowego. Okno to pojawia się również, jeżeli program w początkowej fazie działania nie odnajdzie utworzonego pliku z wyekstraktowanymi punktami. Rysunek 5 Uczenie maszynowe (GUI 2) (Źródło: opracowanie własne) W ww. oknie, użytkownik może wybrać ścieżkę do katalogu, w którym znajdują się wcześniej przygotowane nagrania. Nagrania te, muszą być zgrupowane w odpowiedni sposób. Aby program działał poprawnie, struktura plików powinna wyglądać następująco: Katalog wybrany przez użytkownika / Nazwa sekwencji ruchu / pliki o rozszerzeniu *.avi. Jeżeli program wykryje błędną strukturę plików, zwróci komunikat w kolorze czerwonym. W przypadku poprawnie przetworzonych plików, wyświetli stosowne powiadomienie w kolorze zielonym, tworzy listę wszystkich ścieżek do plików wideo, a następnie odblokuje przycisk „Rozpocznij analizę" . Każdy z powyższych przykładów został zobrazowany, za pomocą poniższej grafiki. Rysunek 6 Uczenie maszynowe (GUI 2 - Struktura) (Źródło: opracowanie własne) 3.2. Preprocessing danych Preprocessing polega na wstępnym przetwarzaniu danych, a dokładniej na przekształcaniu surowych danych w formę, która jest bardziej odpowiednia i optymalna dla modelu uczenia maszynowego. Celem tego fragmentu kodu, jest utrzymanie spójności oraz kompatybilności, a także poprawa jakości danych, potrzebnych do uczenia modelu. Usuwa on wszystkie zbędne informacje, czyli tzw. szumy, pozostawiając jedynie potrzebne dane, wymagane przez model SI. Po rozpoczęciu analizy nagrań, program wyświetla nowe okno z paskiem postępów oraz możliwością rozszerzenia widoku o szczegóły. Na rysunku 7, po lewej stronie przedstawiono podstawowe okno przetwarzania danych, po prawej stronie natomiast jest widoczne okno po rozszerzeniu szczegółów. Rysunek 7 Uczenie maszynowe (GUI 3) (Źródło: opracowanie własne) W trakcie przetwarzania danych, algorytm wykorzystuje bibliotekę MediaPipe, opracowaną przez Google. Za jej pomocą, przeprowadza ekstrakcję punktów kluczowych z dostarczonych nagrań, analizując każdą z 240 klatek, a następnie przypisuje do nich odpowiednie etykiety. W ten sposób tworzy pełen spis punktowy, przedstawionego na nagraniu gestu. Zastosowanie powyższego rozwiązania, skutkuje zmniejszeniem ilości danych przetwarzanych przez etap uczenia maszynowego. W przypadku standardowego podejścia, program przetwarzałby każdy pikseli wideo, co mogłoby wprowadzić dodatkowy szum. Skutkowałoby to mniej precyzyjnym uczeniem modelu, a co za tym idzie gorszą precyzją rozpoznawania gestów. Program, posiada również możliwość podglądu wizualizacji procesu ekstrakcji punktów w czasie rzeczywistym. Gdy użytkownik po rozwinięciu szczegółów, uruchomi przycisk „Wyświetl wizualizację" otworzy się nowe okno, które zobrazuje przetwarzanie danych w sposób graficzny. Rysunek 8 Uczenie maszynowe (GUI 3 - Wizualizacja) (Źródło: opracowanie własne) Podczas przetwarzania informacji, program normalizuje dane poprzez pomijanie wideo o dłuższym czasie trwania lub mniejszej ilości klatek na sekundę, jeżeli natomiast nagranie jest krótsze, brakująca część czasu jest dodawana poprzez funkcję uzupełniania sekwencji pad_sequences. Funkcja ta, tworzy macierz wypełnioną zerami o wymiarach zdefiniowanych w kodzie programu, następnie dla każdej sekwencji kopiuje jej wartość do nowej macierzy, a jeżeli sekwencja jest krótsza niż maksymalna długość, resztę wypełnia zerami. Program, ze względu na konieczność przetwarzania dużej ilości danych oraz potencjalnie długi okres trwania ww. procesu, posiada obsługę wieloprocesowości. Wieloprocesowe podejście

pozwala na jednoczesne przetwarzanie wielu nagrań na rdzeniach procesora, co znacząco redukuje czas potrzebny na preprocessowanie danych. Funkcje, takie jak blokada oraz kolejka zapewniają synchronizację między procesami, aby zapobiec ewentualnym kolizjom podczas dostępu do wspólnych zasobów, takich jak zapis do listy danych, czy aktualizacja wskaźnika postępu. Gdy dane zostaną poprawnie przetworzone, program tworzy plik „preprocessed_data.npz", w którym zapisuje przetworzone dane oraz „classes.npy" w którym przechowuje nazwy etykiet. Następnie pasek postępu dobiegnie końca, a na ekranie użytkownika wyświetla się komunikat, który umożliwia przejście do procesu uczenia modelu SI lub zamknięcie programu. Rysunek 9 Uczenie maszynowe (GUI 3 - Zakończenie przetwarzania danych) (Źródło: opracowanie własne) 3.3. Architektura modelu oraz algorytm klasyfikacji Architektura modelu odnosi się do struktury sieci neuronowej. To ona determinuje sposób, w jaki dane są przetwarzane, jakie operacje są wykonywane oraz jakie mechanizmy są stosowane do obsługi modelu. W omawianym oprogramowaniu, autor dysertacji zastosował model oparty na konwolucyjnych sieciach neuronowych (CNN), co miało na celu uchwycenie wzorców w danych sekwencyjnych, takich jak punkty kluczowe wyodrębnione z nagrań wideo. W kontekście przetwarzania sekwencji, warstwy konwolucyjne (Conv1D) przetwarzają dane jednokierunkowo. Filtrują sekwencje punktów kluczowych, tym samym wykrywając wzorce przestrzenno-czasowe. Filtry, inaczej zwane jądrami, są przesuwane wzdłuż sekwencji, stosując operację splotu. Czynność ta, pozwala na uchwycenie lokalnych wzorców danych. Po każdej warstwie konwolucyjnej zastosowano warstwę MaxPooling, która redukuje wymiarowość danych, zachowując najbardziej istotne cechy. MaxPooling wybiera maksymalną wartość z każdej lokalnej grupy wyników, co pomaga w uodpornieniu modelu na przesunięcia w danych. Równocześnie dokonuje redukcji liczby parametrów, co skutkuje zmniejszeniem ryzyka wystąpienia zjawiska przeuczenia modelu. Model został zaimplementowany przy pomocy biblioteki Keras, która działa na bazie TensorFlow. Zastosowane rozwiązanie oferuje szeroką gamę narzędzi do pracy z przetwarzaniem danych, budowaniem architektury sieci, a także optymalizacji modeli. Omawiany program, tworzy model w sposób sekwencyjny, co oznacza, że warstwy są dodawane jedna po drugiej, tworząc stos warstw. Każda z warstw przyjmuje jako argumenty liczbę filtrów o wymiarze 64, rozmiar jądra równy 3 oraz funkcję aktywacji „relu". Liczba filtrów określa, ile różnych cech model będzie próbował wyodrębnić. Rozmiar jądra to zakres, na którym filtr jest stosowany w danej chwili. Funkcja aktywacji relu, poprzez zamianę wszystkich wartości ujemnych na zero, zapobiega problemowi zanikania gradientu i przyspiesza proces nauki, umożliwiając sieci efektywne uczenie się złożonych wzorców. Warstwa Flatten w niniejszym programie, używana jest do przekształcenia danych z formatu wielowymiarowego do jednowymiarowego, który jest wymagany przez kolejne, w pełni połączone warstwy takie jak warstwa gęsta Dense. Warstwa ta, używana jest zarówno do przekształcania wyodrębnionych cech w postać o mniejszej wymiarowości, jak i do generowania ostatecznego wyniku klasyfikacji. Units, czyli liczba neuronów w warstwie została ustawiona na 256, co pozwala na modelowanie bardziej złożonych zależności, jednakże może prowadzić do przeuczenia. W ostatnim kroku, warstwa Dense generuje wyniki klasyfikacji za pomocą funkcji aktywacji softmax. Funkcja ta, przekształca wyniki w prawdopodobieństwa, które sumują się do 1, co pozwala na przypisanie prawdopodobieństwa do klasy. 3.4. Trening, walidacja oraz ocena skuteczności modelu Trening modelu to proces optymalizacji jego parametrów tak, aby jak najlepiej uczył się rozpoznawania wzorców dostarczonych przy pomocy danych wejściowych, a następnie właściwie przypisywał je do odpowiednich etykiet. W kontekście sieci neuronowych, trening polega na minimalizacji funkcji straty, przy użyciu danych treningowych oraz odpowiedniego optymalizatora. W pierwszej fazie treningu modelu, zastosowany został tzw. forwadr propagation, czyli etap w którym dane przekazywane są do modelu, a każdy neuron przekształca wejścia, oblicza wyjścia i przekazuje je do następnej warstwy. Na końcu tego procesu, model generuje wynik, który jest porównywany z rzeczywistą etykietą danych przy użyciu funkcji straty. Następnie występuje backpropagation, czyli kluczowy mechanizm w treningu sieci neuronowych. Polega on na propagacji błędu wstecz przez sieć neuronową. Błąd ten jest obliczany jako różnica pomiędzy przewidywaniami modelu, a rzeczywistymi wynikami. Proces ten, aktualizuje wagi w sieci tak, aby minimalizować funkcję straty. W tym przypadku zastosowany został algorytm Adam, który jest odmianą algorytmu gradientu prostego (Stochastic Gradient Descent, SGD). Algorytm ten stosuje poniższy wzór matematyczny: Rysunek 10 Wzór algorytmu gradientu prostego gdzie: - w to wagi, które są optymalizowane, η (eta) to współczynnik uczenia się (ang. learning rate), który określa, jak duży krok robimy w kierunku minimalizacji funkcji kosztu (ang. Loss function), natomiast pozostała część to to pochodna funkcji kosztu Loss względem wag w, czyli gradient funkcji kosztu. Pokazuje kierunek i szybkość zmiany funkcji kosztu w zależności od zmian wag. Późniejszy etap obejmuje tworzenie tzw.

epok. W każdej epoce, model przetwarza cały zbiór treningowy. Aby zwiększyć efektywność aktualizacji wag, zastosowane zostało porcjowanie (batch), które pozwala na przetwarzanie mniejszej ilości informacji jednocześnie. Podczas treningu, model uczy się na danych treningowych, jednakże istnieje pewne ryzyko, że nauczy się on wzorców specyficznych dla tych danych. Aby zapobiec ww. problematyce, zastosowano techniki regularyzacji takie jak Dropout oraz L2 Regularization (Ridge). Pierwsza z nich polega na losowym wyłączeniu części neuronów w warstwie podczas treningu. W omawianym programie wartość ta została ustawiona na 0.5, co oznacza, że połowa neuronów w warstwie jest wyłączana w każdej iteracji. Druga zaś, dodaje karę do funkcji straty za duże wartości wag, co ogranicza złożoność modelu i zapobiega jego przeuczeniu. Walidacja modelu jest niezbędna w procesie uczenia maszynowego. W jego trakcie program ocenia, jak dobrze model generalizuje, czyli jak radzi sobie z danymi, których nie uwzględnił podczas treningu. Autor pracy dyplomowej, przyjął złożenie 20% / 80%, gdzie 20% danych jest wykorzystywanych do walidacji, natomiast 80% do trenowania modelu. W trakcie uczenia, stosowane są callback'i. EarlyStopping, czyli wczesne zatrzymanie uczenia maszynowego, wykorzystane jest w celu zabezpieczenia modelu przed ewentualnym przeuczeniem. Proces ten stale monitoruje wydajność modelu na zbiorze walidacyjnym i przerywa trening w przypadku braku poprawy wyników. Istnieje również callback nazwany ModelCheckpoint, który zapisuje najlepsze wagi modelu na podstawie wybranej metryki. Cały proces uczenia maszynowego, został zobrazowany poprzez interfejs graficzny użytkownika, który pojawia się po zatwierdzeniu przycisku dostępnego po przetworzeniu nagrań. Rysunek 11 Uczenie maszynowe (GUI 4) (Źródło: opracowanie własne) Gdy uczenie maszynowe dobiegnie końca, program tworzy plik o nazwie best_model.h5, w którym zapisane są wszystkie informacje opracowywane przez powyższe algorytmy. Użytkownik, otrzymuje również powiadomienie dotyczące zakończenia pracy programu. Na tym etapie, może on zweryfikować poprawność uczenia maszynowego, a także sprawdzić ile epok przetworzył program w celu utworzenia modelu. Zatwierdzając przycisk „Zakończ", użytkownik kończy pracę programu, otrzymując tym samym dostęp do wszystkich niezbędnych plików do obsługi platformy, opisanej w rozdziale czwartym. Rysunek 12 Uczenie maszynowe (GUI 4 - Zakończenie pracy) (Źródło: opracowanie własne) Działanie powyższego programu obrazuje diagram aktywności zawarty na rysunku 13. Rysunek 13 Diagram aktywności (Uczenie maszynowe) (Źródło: opracowanie własne) 4. Implementacja platformy Implementacja platformy opiera się na utworzeniu aplikacji internetowej wraz z odpowiednim połączeniem jej z modelem sztucznej inteligencji. Implementację możemy podzielić w tym przypadku na dwie części. Część frontendową, która opiera się na wdrożeniu wersji graficznej, z której będzie korzystać użytkownik, a także część backendową, która będzie obsługiwać komunikację z serwerem oraz modelem SI. 4.1. Koncepcja wizualna platformy Każda aplikacja weebowa, która będzie posiadać interfejs graficzny (GUI), powinna być uprzednio zaprojektowana przez osobę, która dobrze rozumie zasady korzystania z interfejsów aplikacji internetowych. Doświadczenie w projektowaniu platform przyjaznym użytkownikom pozwala na zdobycie szerszego grona odbiorców poprzez pozytywne odczucia z korzystania z aplikacji. Funkcjonalności wybranego oprogramowania stanowią dobrą podstawę, natomiast design może wzmocnić i uatrakcyjnić aplikację, czyniąc ją bardziej przyjazną i angażującą dla użytkowników[13]. W obecnej dobie Internetu, gdzie panuje przesyt informacji, a użytkownicy są często przebodźcowani, bardzo istotne jest utworzenie odpowiedniej warstwy graficznej. Użytkownicy zazwyczaj oceniają aplikację w ciągu kilku sekund od jej uruchomienia. Atrakcyjna i spójna oprawa graficzna może przyciągnąć uwagę odbiorcy i zachęcić go do dalszej eksploracji. Niezależnie od tego, jak zaawansowane oprogramowanie zostało zaoferowane użytkownikowi, pierwsze wrażenie bazuje na wyglądzie. Intuicyjny interfejs, który jest łatwy w nawigacji, ułatwia korzystanie z aplikacji, pozostawiając pozytywne wrażenia. Wygląd aplikacji nie tylko zwiększa jej zainteresowanie wśród odbiorców ale również pozytywnie wpływa na wzmocnienie marki, poprzez identyfikację marki z wybraną nazwą, kolorystyką czy też utworzonym logotypem[14]. W związku z powyższymi informacjami, omawiana aplikacja również posiada swój projekt wizualny, który został utworzony na początku procesu projektowego w oprogramowaniu figma, a plik projektowy został umieszczony pod nazwą „Projekt koncepcyjny.fig" oraz „Projekt koncepcyjny.pdf" w części praktycznej pracy dyplomowej. Rysunek 14 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 4.2. Mechanizmy komunikacji między frontendem a backendem Większość zaawansowanych aplikacji, korzystających z graficznego interfejsu użytkownika, a zwłaszcza aplikacje webowe, korzystają z warstwy frontendowej oraz backendowej w celu optymalizacji obsługi dużej ilości poleceń lub wykorzystania niedostępnych z poziomu wybranej warstwy narzędzi. 4.2.1. Komunikacja serwer – użytkownik Omawiana aplikacja napisana jest głównie w języku PHP, JavaScript oraz Python co wymaga stałej komunikacji pomiędzy frontendem, a

backendem. Kod w języku PHP dzięki stałemu połączeniu z serwerem, może w sposób dynamiczny, kontrolować dostępem do witryny oraz zawartości, która jest wyświetlana na ekranie użytkownika. Poprzez zarządzanie witryną kod PHP umożliwia wczytanie odpowiedniego kodu CSS oraz JavaScript w wybranych zakładkach, dzięki funkcji rozpoznawania adresów URL. Oprócz aspektów wizualnych, PHP obsługuje również wczytywanie wizualizacji dla gestów, poprzez przeszukiwanie katalogów zamieszonych na serwerze w poszukiwaniu nazwy wprowadzonej w warstwie frontendowej oraz przekazanej odpowiednio przez skrypt napisany w języku JavaScript. 4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) Integracja modelu AI z aplikacją internetową jest realizowana poprzez połączenie części backendu, napisanego w języku programowania Python z częścią frontendową platformy napisaną w języku JavaScript. Wspomniana część backendowa jest zapisana w pliku o nazwie „app.py". Program ten uruchamia model sztucznej inteligencji o nazwie „best_model.h5" oraz listę etykiet „classes.npy", które są wykorzystywane do przetwarzania danych w czasie rzeczywistym. Dane z kamery użytkownika są przetwarzane, a wyodrębnione punkty kluczowe są przekazywane do modelu, który dokonuje predykcji gestów. Wyniki te są następnie udostępniane poprzez endpointy, które zwracają szczegóły dotyczące rozpoznanych gestów w formacie JSON. Frontend aplikacji komunikuje się z backendem za pomocą zapytań http, korzystając z fetch API do wysyłania zapytań do serwera Flask. Za komunikację z serwerową częścią oprogramowania, odpowiedzialne są trzy skrypty napisane w języku JavaScript, a są nimi: „edu.js", „translator.js" oraz „sign_video_loader.js" . Każdy z trzech wymienionych skryptów, korzysta z wywołań endpointów poprzez serwer lokalny o porcie 5000 „http://localhost:5000/gesture_details", które służą do pobierania informacji o rozpoznanych gestach i aktualizowania interfejsu użytkownika. W pliku app.py, endpoint dostarcza strumień wideo z kamery internetowej, który następnie jest przetwarzany przez model SI. Frontend inicjuje kamerę oraz przesyła obraz do backendu gdzie obraz jest przetwarzany, a następnie odtwarza przesyłany strumień, umożliwiając tym samym interaktywną analizę gestów w czasie rzeczywistym. 4.3. Widoki aplikacji Omawiana platforma internetowa posiada tzw. widoki, czyli warstwę frontendową, która jest odpowiednio ostylowana poprzez arkusze stylów CSS, starając się jak najdokładniej odwzorować utworzony w początkowych fazach pracy projekt wizualizacji aplikacji. Rysunek 15 Interfejs graficzny aplikacji – zakładka Home (Źródło: opracowanie własne) Platforma składa się z siedmiu widoków. Trzy pierwsze widoki odnoszą się do informacji wstępnych dotyczących realizowanego projektu. Są to zakładki „Home", „O projekcie" oraz „O nas". Każda z zakładek posiada statyczne elementy informacyjne oraz dekoracyjne, które mają zachęcić użytkownika do dalszej interakcji z platformą. Poniżej opisane zostały bardziej interesujące elementy, każdej z zakładek. Rysunek 16 Zakładka Home (Źródło: opracowanie własne) Na stronie Index.php (Home) zostały przedstawione podstawowe informacje dotyczące platformy. Zastosowany został również slider, który wyświetla logotyp Uniwersytetu Rzeszowskiego, przechodząc od prawej strony do lewej, a następnie cofając się do początku. Rysunek 17 Slider - strona Index.php (Źródło: opracowanie własne) Zakładka O projekcie (Project.php) zawiera informacje dotyczące funkcjonowania utworzonej aplikacji. Można znaleźć tam między innymi informacje o zastosowanych rozwiązaniach technologicznych, genezie powstania dysertacji, poruszanej problematyce, skrócony opis działania aplikacji, a także linię czasu, która w sposób skrótowy ukazuje etapy powstawania aplikacji. Rysunek 18 Widok zakładki O projekcie (Project.php) (Źródło: opracowanie własne) Zakładka O nas (About.php) skupia się na opisie osób zaangażowanych w powstanie projektu. W związku ze stałym rozwojem tworzonego oprogramowania oraz wysoką czasochłonnością prac, a także innymi losowymi aspektami, w projekt były zaangażowane 4 osoby. Oprócz autora oprogramowania oraz niniejszej dysertacji, w projekt zaangażował się również Dr. Inż. Wojciech Kozioł, który był pomysłodawcą tematu omawianej pracy, Dr. Inż. Bogusław Twaróg, który prowadził nadzór merytoryczny nad powstającą pracą oraz inż. Patryk Arendt, który służył jako model motion capture dla 5 gestów (tj. 500 próbek nagrań wideo). Rysunek 19 Widok zakładki O nas (About.php) (Źródło: opracowanie własne) W ramach uhonorowania wkładu, jaki włożyły ww. osoby w powstanie pracy dyplomowej, zostały one wymienione w wersji aplikacyjnej w zakładce O nas. Rysunek 20 Uhonorowanie wsparcia (About.php) (Źródło: opracowanie własne) Kliknięcie przycisku „Przejdź do aplikacji" powoduje wczytanie ostatniego statycznego widoku jakim jest „Dashboard.php" . W widoku tym opisane są podstawowe informacje dotyczące ilości gestów, ilości wykorzystanych próbek, sumarycznego czasu nagrań, a także wiele innych. Rysunek 21 Widok zakładki Dashboard (Dashboard.php) (Źródło: opracowanie własne) W zakładce tej przedstawiona została również animacja, która w trzech etapach prezentuje w jaki sposób program rozpoznaje sekwencje ruchów ludzkiego ciała, na przykładzie gestu z etykietą „Dzięki". Rysunek 22 Działanie systemu rozpoznawania

gestu (Źródło: opracowanie własne) Pozostałe trzy widoki, odpowiadają pośrednio lub bezpośrednio za komunikację z systemem rozpoznawania gestów. Działają one na podobnej zasadzie, lecz skupiają się na innym rodzaju wykonywanego zadania przez użytkownika. Rysunek 23 Widok zakładki Tłumacz (Translator.php) (Źródło: opracowanie własne) Zakładka Tłumacz, umożliwia użytkownikowi tłumaczenie słów w języku migowym w obie strony. Może on skorzystać z funkcji udostępnienia obrazu z kamery co pozwoli na tłumaczenie języka migowego na język naturalny lub wyszukać gest w języku migowym, dostępny w systemie poprzez polecenie pisemne lub komendę głosową. W celu wyszukanie gestu języka migowego, wystarczy, że użytkownik uzupełni pole tekstowe odpowiednią nazwą gestu i zatwierdzi zmianę klikając przycisk po prawej stronie lub wciskając klawisz „Enter". Rysunek 24 Wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) Istnieje również alternatywny sposób wyszukiwania gestu w systemie. Użytkownik może skorzystać z przycisku, znajdującego się po lewej stronie okna, aby rozpocząć nasłuchiwanie mikrofonu. Po wypowiedzeniu frazy np. „Cześć", możemy użyć polecenia głosowego „... stop szukaj", co automatycznie zakończy nasłuchiwanie mikrofonu i rozpocznie proces szukania gestu o etykiecie „Cześć". Nasłuchiwanie mikrofonu można także wyłączyć w sposób ręczny, klikając ponownie na ikonę mikrofonu. Wyszukiwanie głosowe, oparte zostało na polskiej wersji językowej Web Speech API dla języka JavaScript. Po aktywacji funkcji nasłuchiwania program sprawdza dostępność API w przeglądarce i konfiguruje obiekt rozpoznawania mowy. Podczas nasłuchiwania mikrofonu, program przetwarza rozpoznane fragmenty, a następnie wstawia je do pola tekstowego. Jeżeli program wykryje określoną komendę tj. „stop szukaj", przerywa nasłuchiwanie i uruchamia proces wyszukiwania nagrania. Program obsługuje najczęściej pojawiające się błędy, a także automatycznie restartuje nasłuchiwanie, jeżeli nie zostało ono przerwane ręcznie. Rysunek 25 Głosowe wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) Gdy gest nie zostanie odnaleziony, na ekranie pojawia się stosowny komunikat. Gdy gest zostanie wyszukany poprawnie, oczom użytkownika ukaże się wizualizacja gestu odtwarzana w postaci nagrania. Przykład takowej wizualizacji został ukazany na grafice poniżej. Rysunek 26 Wizualizacja gestu (Źródło: opracowanie własne) W drugim przypadku użycia, użytkownik może udostępnić obraz z kamery, a następnie wykonać wybrany przez siebie gest. Na wykrytą postać zostanie naniesiona siatka punktów, która będzie obrazować przetwarzane dane. Program w czasie rzeczywistym, w lewym górnym rogu ekranu wyświetlać będzie prawdopodobne gesty pokazane przez użytkownika. Gdy gest zostanie wykryty wystarczającą liczbę razy, oczom użytkownika ukaże się również przycisk „Wyświetl wizualizację pokazywanych gestów". Rysunek 27 Prezentacja gestu "Dzięki" (Źródło: opracowanie własne) Gdy użytkownik kliknie na komunikat „Wyświetl wizualizację pokazywanych gestów" po prawej stronie, automatycznie w sposób płynny wczyta się nagranie z wizualizacją gestu. Wizualizacja ta będzie się zmieniać za każdym razem, gdy program wykryje pokazanie innego gestu z odpowiednią pewnością. Rysunek 28 Synchronizacja wizualizacji z obrazem w czasie rzeczywistym (Źródło: opracowanie własne) Po przejściu na inną zakładkę, w przeglądarce internetowej klienta, śledzenie kamery zostaje zatrzymane, zarówno przez warstwę forntendową jak i backendową, a algorytmy wstrzymują swoją pracę oczekując na wznowienie połączenia. Poniżej przedstawiono diagram przypadków użycia dla zakładki Tłumacz: Rysunek 29 Diagram przypadków użycia (Tłumacz) (Źródło: opracowanie własne) Dla powyższego widoku został również przygotowany diagram aktywności, który pozwala na jeszcze bardziej precyzyjne zapoznanie się z architekturą działania oprogramowania. Poniższa grafika obrazuje proces przetwarzania obrazu z kamery użytkownika, wraz z obsługą wizualizacji, która jest aktualizowana w czasie rzeczywistym. Rysunek 30 Diagram aktywności (Tłumacz - kamera / wizualizacja) (Źródło: opracowanie własne) Zobrazowany został również diagram aktywności, odpowiedzialny za funkcję wyszukiwania wizualizacji za pomocą poleceń tekstowych oraz głosowych. Diagram UML ww. procesu znajduje się poniżej. Rysunek 31 Diagram aktywności (Tłumacz – Wizualizacja) (Źródło: opracowanie własne) Ostatnie dwa widoki aplikacji omawiają działanie zakładki „Nauka języka migowego". Oprogramowanie, dzięki zaawansowanym algorytmom wspiera proces uczenia języka migowego. W tym celu stworzone zostały kategorie oraz „kafelki", które umożliwiają wybór gestu, którego chce się nauczyć użytkownik. Funkcja ta skierowana jest dla osób początkujących, które dopiero rozpoczynają swoją przygodę z PJM. Rysunek 32 Widok zakładki Nauka Języka Migowego (Edu.php) (Źródło: opracowanie własne) Na stronie podstronie edu.pl możemy wybrać kategorię, która nas interesuje, a następnie odpowiedni dla nas gest. Po wybraniu odpowiedniej kategorii użytkownik będzie mógł zaobserwować zmianę w wyświetlanych elementach. Zamiast kategorii, na ekranie pojawią się gesty, które są z nią powiązane. Rysunek 33 Gesty wybranej kategorii (Źródło: opracowanie własne) Po wybraniu jednego z dostępnych gestów pojawia się nowy widok

(Edu-progress.php), który wizualnie jest zbliżony do widoku tłumacza. Tutaj również możemy uruchomić podgląd z kamery, natomiast po prawej stronie odtwarzane jest w zapętleniu nagranie z wybranym przez nas gestem. W widoku tym, niedostępny jest input do wyszukiwania gestów, czy też zmiany wizualizacji. Po uruchomieniu kamery i poprawnym wykonaniu gestu, program zatrzyma udostępnianie obrazu z kamery, a użytkownik otrzyma odpowiedni komunikat na ekranie. Rysunek 34 Wykonanie poprawnego gestu (Źródło: opracowanie własne) 5. Ocena działania platformy Utworzona platforma została przetestowana pod względem działania modelu sztucznej inteligencji oraz kilku powiązanych z nią czynników. Przebadana została precyzja modeli CNN, skonstruowanych z różnych ilości próbek, w konfiguracji 100 próbek na 1 gest, 200 próbek na 1 gest oraz 300 próbek na 1 gest. Wyszczególnione zostały etapy przetwarzania danych, uczenia maszynowego oraz ogólnej pracy modelu w czasie rzeczywistym, a także na bazie przygotowanych wcześniej nagrań. Następnie, wyniki te zostały zestawione z utworzonym hybrydowym modelem CNN + LSTM. 5.1. Testy w rzeczywistych warunkach Testy związane z wydajnością w rzeczywistych warunkach, a co za tym idzie w czasie rzeczywistym są stosunkowo ciężkie do realizacji. Wpływ ma na to rozbieżność pomiędzy wykonywanymi gestami, które ciężko odtworzyć w sposób identyczny, zróżnicowane oświetlenie, ułożenie ciała oraz wykorzystywany hardware. Istnieje wiele zmiennych, które mogą zakłócić obiektywną ocenę modelu, zwłaszcza jeżeli ten został przygotowany na niewielkiej ilości próbek. W omawianym punkcie, przedstawione zostaną zarówno obiektywne informacje, podparte dowodami, jak i subiektywne odczucia autora dysertacji. 5.1.1. Ekstrakcja punktów kluczowych Pierwszym elementem, który zostanie poddany analizie jest proces przetwarzania informacji, potrzebnych do utworzenia modelu. Cały proces uczenia maszynowego czyli ekstrakcja punktów kluczowych oraz nauka modelu odbywała się na maszynie obliczeniowej z zamontowanym procesorem AMD EPYC 7702 64-Core, który posiadał 128 procesorów wirtualnych o szybkości 2GHz, pamięcią ram 256GB oraz dysku HDD. Przetwarzanie danych w modelu CNN, zastosowane na 6 gestach po 100 próbek każdy, co daje wynik 600 próbek, trwało 19 minut w zaokrągleniu czasu do pełnych minut. Czas trwania tego samego procesu na 1200 próbkach trwało 35 minut, natomiast finalny model omawiany w dysertacji, potrzebował 54 minut na przetworzenie wszystkich danych (1800 próbek). Po zestawieniu ww. informacji w postaci wykresu, możemy zaobserwować jak wygląda złożoność czasowa przetwarzania danych. Przetwarzanie próbek w stosunku do czasu 60 54 50) n im (a 40 35 in a z r a 30 w t e 19 z r p 20 s a z C 10 0 600 próbek 1200 próbek 1800 próbek Czas w minutach 19 35 54 Ilość próbek / Czas Wykres 1 Przetwarzanie próbek CNN Uśredniając czas przetwarzania każdej próbki wynosi 1,81 sekundy. Rozbijając na poszczególne partie: przy 600 próbkach, program potrzebował 1,9 sekundy na przetworzenie jednego nagrania, 1,75 sekundy dla 1200 próbek oraz 1,8 sekundy w przypadku 1800 próbek. Zestawiając dane przetwarzane przez program wykorzystujący CNN oraz CNN w połączeniu z LSTM, nie widać jednoznacznych różnic. Program w podobnym czasie przetworzył dane zarówno dla modelu opartym o CNN jak i modelu hybrydowym, przy czym mniej złożony model zakończył proces ponad 2 minuty wcześniej. Rysunek 35 Zestawienie metody CNN oraz CNN + LSTM (Źródło: opracowanie własne) Czas przetwarzania danych (1800 próbek) CNN+LSTM 56 u le d o m ja z d o R CNN 54 53 54 54 55 55 56 56 57 Czas przetwarzania (min) Wykres 2 Czas przetwarzania danych – porównanie CNN – CNN + LSTM 5.1.2. Uczenie modelu Uczenie modelu również zostało porównane w identycznych kryteriach. Z otrzymanych informacji wynika, że czas uczenia maszynowego nie jest stricte związany z ilością próbek. Funkcja przerywania uczenia, która za zadanie ma chronić model przed przeuczeniem, skutecznie skraca czas potrzebny do stworzenia modelu. Program kończy pracę za każdym razem gdy wykryje, że precyzja modelu nie poprawia się co skutkuje utworzeniem modelu w krótszym czasie niżeli z potencjalnie mniejszą ilością próbek. Uczenie maszynowe w stosunku do czasu 140 124 120)s (u 100 le d 76 o m 80 a in 60 e z c u s 40 30 a z C 20 0 600 próbek 1200 próbek 1800 próbek Czas w sekundach 30 124 76 Ilość próbek / Czas Wykres 3 Uczenie modelu CNN O ile nie było widocznej różnicy pomiędzy przetwarzaniem danych pomiędzy modelem wykorzystującym CNN, a modelem hybrydowym CNN + LSTM, o tyle w przypadku uczenia modelu różnica jest bardzo zauważalna. Model z mniejszą ilością warstw uczył się przez 1 minutę i 16 sekund, natomiast model z przetwarzaniem danych wstecz, trenował się przez 6 godzin, 9 minut i 36 sekund. Różnica w czasie nauczania modelu jest ogromna, a co za tym idzie model hybrydowy z zastosowaniem LSTM wydaje się mało wydajnym rozwiązaniem, zważywszy na małą ilość przetwarzanych danych. Program oparty na mniejszej ilości warstw uczył się ze stosunkowo wysoką precyzją, która oscylowała w zakresie 91-95%, oraz 81-88% podczas walidacji, co obrazuje poniższa ilustracja. Rysunek 36 Proces uczenia maszynowego CNN (Źródło: opracowanie własne) Druga wersja programu zatytułowana

roboczo 2.1, wykazywała wyższą precyzję uczenia podczas kilku prób uczenia modelu. Dokładność modelu oscylowała pomiędzy 97-99%, a podczas walidacji wynik ten wynosił 93-99%. Wyniki te sugerują, że model utworzony za pomocą CNN oraz LSTM radzi sobie znacznie lepiej niżeli poprzednia wersja. Należy zwrócić uwagę również na wielkość pliku docelowego. Model utworzony w wersji 1.0 zajmuje 25,8 MB miejsca na dysku, natomiast model w wersji 2.1 zajmuje 1,26 GB. Kwestia ta staje się problematyczna, gdyż model reprezentujący niewiele niższą precyzję podczas uczenia 5 maszynowego, zajmuje 50 razy mniej miejsca na dysku, co staje się kuszącym rozwiązaniem w przypadku chęci zaoszczędzenia miejsca w przestrzeni dyskowej. Rysunek 37 Proces uczenia maszynowego CNN + LSTM (Źródło: opracowanie własne) Czas przetwarzania danych (1800 próbek) CNN+LSTM 22176 u le d o m ja z d o R CNN 76 0 5000 10000 15000 20000 25000 Czas przetwarzania (s) Wykres 4 Czas uczenia modelu – porównanie CNN – CNN + LSTM 5.1.3. Testy w czasie rzeczywistym Model z założenia miał zostać przeznaczony do platformy obsługującej rozpoznawanie polskiego języka migowego w czasie rzeczywistym. W związku z powyższą informacją, również ten aspekt został przetestowany. Niestety w związku z wieloma czynnikami losowymi, nie da się zmierzyć precyzyjnie różnicy pomiędzy rozpoznawaniem gestów w czasie rzeczywistym pomiędzy modelem utworzonym w wersji podstawowej oraz hybrydowej. 5 Jednakże dokonano pewnych obserwacji, które sugerują wykorzystanie jednego spośród dwóch modeli. W przypadku modelu podstawowego, nie zaobserwowano problemów. Model działał stabilnie, precyzja wykrywanych gestów była zadowalająca. W przypadku modelu hybrydowego, uruchomionego w warunkach czasu rzeczywistego zaobserwowano problemy. Model działał niestabilnie, rejestrując ciągłe „przycięcia" obrazu. Zachodziło tam zjawisko klatkowania, a dokładna przyczyna została opisana w punkcie 5.3. 5.2. Skuteczność i dokładność rozpoznawania gestów Każdy z utworzonych modeli został również przetestowany w warunkach odpowiednio wcześniej przygotowanych. Specjalnie zaprojektowany program, odtwarzał trzy nagrania których nie było zamieszonych w próbkach modelu, dla każdego spośród 6 istniejących gestów. Następnie za pomocą dostarczonych modeli klasyfikował wykryte gesty. Ocenie została poddana precyzja rozpoznanych gestów z podziałem procentowym. Spośród testowanych gestów jeden z nich otrzymał najgorszy wynik, rozpoznając tym samym inny gest. Pozostałe tj. 5/6 gestów zostało rozpoznanych poprawnie. Zestawienie wykresów wraz z uśrednieniem wyników, zostało zaprezentowane na wykresie 5. 5 Wykres 5 Zestawienie wykresów Uśredniając wyniki spośród trzech testów, gest „Cześć" został rozpoznany z precyzją 60,70%, program wykrył również możliwość zaistnienia gestu „Dzięki" z wynikiem 38,69% oraz „Prosić" (0,58%). Program podczas analizy nagrań wykrył 2 z 3 wskazanych gestów poprawnie. Podsumowanie analizy gestu „Cześć" zostało przedstawione na poniższym wykresie. 5 Wykres 6 Analiza gestu Cześć - CNN Porównując wyniki z modelem CNN + LSTM, lepiej prezentuje się model CNN. Model składający się z dodatkowej warstwy, błędnie wykrył gest dzięki z precyzją 60,33%, oraz cześć z precyzją 39,67%. 5 W przypadku gestu „Dzięki" model nieprecyzyjnie rozpoznał wskazywany gest. Jedynie na drugiej próbce gest dzięki miał wyższe prawdopodobieństwo wystąpienia niżeli inne opcje. Program błędnie rozpoznał, że wskazanym modelem jest gest Proszę z prawdopodobieństwem 66,13%, drugim możliwym gestem było dzięki z wynikiem 33,44%. Dodatkowo program dostrzegł podobieństwo z gestem prosić (0,24%), Cześć (0,17%) oraz Przepraszać (0,02%). Wykres 7 Analiza gestu Dzięki – CNN W powyższym przypadku lepszy okazał się model z dodatkową warstwą, prezentując następujące wyniki: dzięki (67,62%), proszę (32,11%), cześć (0,26%). 5 W przypadku gestu „Dziękuję" nie było minimalnych wątpliwości. Gest ten został wykryty z precyzją aż 99,98%. Pozostałe 0.02% zostało podzielone po równo pomiędzy gestem „dzięki" i „Cześć". Wykres 8 Analiza gestu Dziękuję - CNN W powyższym przypadku, znacznie lepsza okazała się sieć CNN. Druga sieć wykryła gest: dziękuję (65,22%), dzięki (32,16%), przepraszać (2,38%), cześć (0,18%), prosić (0,05%), proszę (0,01%). 5 Czwarty badany gest (Prosić) został rozpoznany poprawnie z precyzją 75,69%, pozostałe wartości zostały podzielone pomiędzy: cześć (22,34%), proszę (1,88%), dziękuję (0,06%) oraz dzięki (0,03%) Wykres 9 Analiza gestu Prosić – CNN Gest prosić został wykryty poprawnie w każdej konfiguracji sieci, lecz sieć LSTM wykryła gest z większą precyzją, zwracając wynik precyzji: prosić (99,74%), dziękuję (0,24%), dzięki (0,02%), przepraszać (0,01%). 5 Piąty gest, tak jak gest trzeci, został rozpoznany z bardzo wysoką precyzją (99,91%). Pozostałe prawdopodobieństwo zostało podzielone jedynie przez dzięki (0,07%) oraz prosić (0,02%). Wykres 10 Analiza gestu Proszę – CNN W powyższym przypadku sieć CNN okazała się lepsza niżeli LSTM. Druga z wymienionych sieci wykryła gesty: proszę (77,73%), dzięki (21,90%), cześć (0,35%), prosić oraz przepraszać (0,01%). 5 Ostatnim badanym gestem jest gest przepraszać. Gest ten został sumarycznie wykryty z precyzją 49%, możliwe było również wykrycie etykiety dzięki, która miała również wysoki wynik (43,71%). Analizując wyniki zauważyć można także

proszę (7,07) oraz cześć (0,22%). Wykres 11 Analiza gestu Przepraszać – CNN Model LSTM wykrył odpowiednio gesty: przepraszać (44,11%), dzięki (38,40%), proszę (17,01%), cześć (0,23%), dziękuję (0,14%) oraz prosić (0,10%). Ogólny wynik precyzji przetwarzanych gestów wskazuje na to, iż model CNN jest bardziej precyzyjny (sumarycznie 417,72%) niżeli model hybrydowy CNN + LSTM (sumarycznie 394,09%). 60 5.3. Analiza błędów i propozycje ulepszeń Platforma, a co za tym idzie również model SI posiada nie tylko swoje zalety ale również wady. W pierwszej kolejności należy wymienić warstwę backendową. Na przykładzie omawianej dysertacji udowodniono, że model hybrydowy w postaci CNN + LSTM, niekoniecznie działa poprawnie z przetwarzaniem obrazów w czasie rzeczywistym. LSTM jest przeznaczony głównie do pracy z sekwencjami danych, które mają wyraźny związek czasowy, takimi jak teksty lub sygnały czasowe. W przypadku obrazu, dane są przestrzenne, a nie czasowe. Do przetwarzania danych obrazowych lepiej nadają się sieci konwolucyjne (CNN), które są zaprojektowane specjalnie do pracy z danymi w formie siatek (takich jak obrazy). Zwiększenie złożoności, w tym przypadku wpłynęło negatywnie na działanie programu. Model wykorzystujący LSTM potrzebował znacznie więcej czasu na przetworzenie sekwencji, co skutkowało zacinaniem się obrazu z kamery oraz powolną predykcją gestów. Precyzja w przetwarzaniu statycznym, również nie została podniesiona. Zaobserwować można było spadek precyzji modelu na danych testowych. W związku z dokonaną analizą, nie zaleca się wykorzystywania LSTM do przetwarzania obrazu w czasie rzeczywistym. Model CNN, można jednakże rozwinąć o podejście hybrydowe CNN + HMM, które polecają autorzy publikacji „Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition.”. Zastosowanie powyższego rozwiązania potencjalnie mogłoby podnieść poziom precyzji modelu, bez negatywnego wpływu na stabilność programu. Możliwe jest także dodanie Transformatorów takich jak ST- Transformer, które są aktualnym rozwiązaniem na rok 2024. Dodatkowo, model mógłby być wzbogacony o większą ilość próbek, wliczając w to zróżnicowanych modeli motion capture, różne warunki oświetlenia, różnego typu kamery oraz zmienne tła otoczenia. Powyższe elementy z pewnością poprawiłyby precyzję uczenia modelu. Utworzona platforma, mogłaby również zostać ulepszona poprzez dodanie elementów frontendowych, takich jak responsywność, a także możliwości obsługi wielu języków. Posiada ona również przygotowany moduł, który można rozwinąć o logowanie oraz rejestrację, wprowadzając tym samym system abonamentów (tokenów), znany chociażby z konkurencyjnych aplikacji SI. 61 Podsumowanie Platformy edukacyjne stale cieszą się popularnością. W grupie ww. platform znajdują się aplikacje do nauki języków obcych z których każdego dnia korzystają setki tysięcy osób, a ich stały rozwój pozwala na zwiększenie komunikatywności w środowisku międzynarodowym. Sytuacja ta jest analogiczna w przypadku zwiększenia świadomości ludzkiej na wszelkiego typu dysfunkcje organizmu. Strony internetowe coraz częściej dostosowują się do różnego rodzaju dyrektyw, chociażby takiej jak WCAG 2.0, które to w jasny sposób określają jak powinny wyglądać strony przyjazne dla osób z niepełnosprawnościami. Pomimo zwiększenia dostępności stron internetowych dla osób z dysfunkcjami, nadal istnieje problem wykluczenia społecznego w warunkach kontaktu bezpośredniego. Osoby, które są głuchonieme, potrzebują obecności tłumacza, który stale przekazuje informacje pomiędzy osobami pełnosprawnymi, a osobą dotknięta dysfunkcją. Niniejsza dysertacja łączy możliwości platform internetowych oraz tematyki niepełnosprawności, tworząc prototyp platformy do rozpoznawania polskiego języka migowego. Pomimo istnienia takowych aplikacji na rynku anglojęzycznym, nadal nie istnieje stabilne rozwiązanie na polskim rynku. W związku z powyższą informacją, praca ta jest innowacyjna, łącząc ze sobą prostotę obsługi, z zaawansowanymi technologiami takimi jak model sztucznej inteligencji rozpoznający gesty języka migowego w czasie rzeczywistym. W ramach niniejszej pracy dyplomowej, opracowany został projekt platformy internetowej, wraz z jej pełną implementacją przy zastosowaniu języków webowych. Dodatkowo na potrzeby dysertacji, autor utworzył kilka modeli sztucznej inteligencji, składających się z różnej ilości próbek opracowanych przez twórcę. Następnie każdy model został przetestowany pod względem wydajność oraz możliwości wykorzystania ich w aplikacji korzystającej z kamery internetowej w czasie rzeczywistym. Finalnie, utworzonych zostało 1800 nagrań o długości 4 sekund, przedstawiających osobę, wykonującą określony gest polskiego języka migowego. Najlepszym modelem okazał się program oparty na architekturze CNN, który wykazał się wysoką precyzją oraz szybkością działania. Należy również uwzględnić, że wszystkie elementy estetyczne w tym grafiki, wizualizacje oraz wykresy zostały opracowane przez autora pracy. Praca dyplomowa, może również zostać rozwinięta o dodatkowe elementy, takie jak responsywność oraz moduł logowania i rejestracji, pozwalający na wykorzystanie ww. dysertacji w celach materialnych. Sugerowane jest również wzbogacenie opracowanego 62 modelu o większą ilość próbek, opracowanych w różnych warunkach oświetleniowych,

zmiennym otoczeniu, a także na innych osobach. Przebudowa modelu, dodając warstwę architektury HMM lub ST-Transformer również może zwiększyć precyzję wykrywanych gestów, a co za tym idzie poprawić działanie całej platformy edukacyjno-translatorycznej. 63 Bibliografia oraz Netografia 1. Camgoz, N. C., Koller, O., Hadfield, S., & Bowden, R. (2017). SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition. IEEE International Conference on Computer Vision (ICCV). 2. Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 7291-7299. 3. Huang, J., Zhou, W., Zhang, Q., Li, H., & Li, W. (2018). Attention-Based 3D-CNNs for Large- Vocabulary Sign Language Recognition. IEEE Transactions on Circuits and Systems for Video Technology, 29(9), 2822-2832. 4. Jamroży D.(2023): Aplikacja internetowa z graficznym interfejsem użytkownika opartym na technologii typu eye-tracking oraz speech recognition. [Praca inżynierska, Uniwersytet Rzeszowski] 5. Koller, O., Zargaran, S., Ney, H., & Bowden, R. (2015). Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition. In Proceedings of the British Machine Vision Conference (BMVC). 6. --- MediaPipe Documentation (n.d.) . Google Developers. Retrieved from https://ai.google.dev/edge/mediapipe/solutions/guide?hl=pl 7. Pandey, A. , Mishra, M., & Verma, A. K. (2022). Real-Time Sign Language Recognition Using Machine Learning and Neural Networks. IEEE Access. 8. Patel, K., Gil-González, A.-B., & Corchado, J. M. (2022). Deepsign: Sign Language Detection and Recognition Using Deep Learning. Electronics, 11(11), 1780. 9. Patel, M., & Shah, N. (2021). Real-Time Gesture-Based Sign Language Recognition System. IEEE International Conference on Information Technology and Engineering (ICITE). 10. Pigou, L., Dieleman, S., Kindermans, P. J., & Schrauwen, B. (2015). Sign Language Recognition Using Convolutional Neural Networks. European Conference on Computer Vision Workshops (ECCVW). 11. Zhang, Z., & Liu, C. (2020). Skeleton-Based Sign Language Recognition Using Whole-Hand Features. IEEE Access, 8, 68827-68837. 12. https://www.wsb-nlu.edu.pl/pl/wpisy/wplyw-sztucznej-inteligencji-na-przyszlosc-pracy-nowe-perspektywy-i-wyzwania (22.08.2024) 64 13. https://www.tamoco.com/blog/blog-app-design-app-functionality-ux-ui/ (22.08.2024) 14. https://echoinnovateit.com/ux-or-ui-whats-more-important-in-app-development/ (22.08.2024) 15. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5(4), 115-133. 65 Spis ilustracji, tabel oraz wykresów Spis Ilustracji Rysunek 1 Porównanie technologii (Źródło: opracowanie własne) 11 Rysunek 2 Interfejs programu XAMPP 20 Rysunek 3 Etapy zbierania próbek (Źródło: opracowanie własne) 22 Rysunek 4 Uczenie maszynowe (GUI 1) (Źródło: opracowanie własne) 22 Rysunek 5 Uczenie maszynowe (GUI 2) (Źródło: opracowanie własne) 23 Rysunek 6 Uczenie maszynowe (GUI 2 - Struktura) (Źródło: opracowanie własne) 24 Rysunek 7 Uczenie maszynowe (GUI 3) (Źródło: opracowanie własne) 24 Rysunek 8 Uczenie maszynowe (GUI 3 - Wizualizacja) (Źródło: opracowanie własne) 25 Rysunek 9 Uczenie maszynowe (GUI 3 - Zakończenie przetwarzania danych) (Źródło: opracowanie własne) 26 Rysunek 10 Wzór algorytmu gradientu prostego 28 Rysunek 11 Uczenie maszynowe (GUI 4) (Źródło: opracowanie własne) 29 Rysunek 12 Uczenie maszynowe (GUI 4 - Zakończenie pracy) (Źródło: opracowanie własne) .. 30 Rysunek 13 Diagram aktywności (Uczenie maszynowe) (Źródło: opracowanie własne) 31 Rysunek 14 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 33 Rysunek 15 Interfejs graficzny aplikacji – zakładka Home (Źródło: opracowanie własne) 35 Rysunek 16 Zakładka Home (Źródło: opracowanie własne) 35 Rysunek 17 Slider - strona Index.php (Źródło: opracowanie własne) 36 Rysunek 18 Widok zakładki O projekcie (Project.php) (Źródło: opracowanie własne) 36 Rysunek 19 Widok zakładki O nas (About.php) (Źródło: opracowanie własne) 37 Rysunek 20 Uhonorowanie wsparcia (About.php) (Źródło: opracowanie własne) 38 Rysunek 21 Widok zakładki Dashboard (Dashboard.php) (Źródło: opracowanie własne) 38 Rysunek 22 Działanie systemu rozpoznawania gestu (Źródło: opracowanie własne) 39 Rysunek 23 Widok zakładki Tłumacz (Translator.php) (Źródło: opracowanie własne) 39 Rysunek 24 Wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) 40 Rysunek 25 Głosowe wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) 40 Rysunek 26 Wizualizacja gestu (Źródło: opracowanie własne) 41 Rysunek 27 Prezentacja gestu "Dzięki" (Źródło: opracowanie własne) 42 66 Rysunek 28 Synchronizacja wizualizacji z obrazem w czasie rzeczywistym (Źródło: opracowanie własne) 42 Rysunek 29 Diagram przypadków użycia (Tłumacz) (Źródło: opracowanie własne) 43 Rysunek 30 Diagram aktywności (Tłumacz - kamera / wizualizacja) (Źródło: opracowanie własne) 44 Rysunek 31 Diagram aktywności (Tłumacz – Wizualizacja) (Źródło: opracowanie własne) 45 Rysunek 32 Widok zakładki Nauka Języka Migowego (Edu.php) (Źródło: opracowanie własne) 46 Rysunek 33 Gesty wybranej kategorii (Źródło: opracowanie

Nierozpoznane wyrazy

UNIWERSYTET RZESZOWSKI Kolegium Nauk Przyrodniczych Damian Jamroży Nr albumu: 113729 Kierunek: Informatyka Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego Praca magisterska Praca wykonana pod kierunkiem Dr. Inż. Bogusława Twaroga Rzeszów, 2024 Pragnę serdecznie podziękować Panu dr inż. Bogusławowi Twarogowi za nieocenione wsparcie oraz życzliwość okazywaną na każdym etapie mojej edukacji. Jego pomoc i zaangażowanie miały kluczowy wpływ nie tylko na proces powstawania niniejszej pracy magisterskiej, ale także na moją pracę inżynierską i cały tok studiów. Bez Jego merytorycznej wiedzy oraz wsparcia, ukończenie tych etapów byłoby znacznie trudniejsze. Jednocześnie chciałbym serdecznie podziękować moim przyjaciołom z UCI UR. To dzięki ich namowom i wsparciu udało mi się zrealizować ww. etap życia. Spis treści Wstęp 7 Cel i zakres pracy 8 1. Przegląd literatury i analiza istniejących rozwiązań 9 1.1. Przegląd technologii rozpoznawania gestów 9 1.2. Przegląd metod uczenia maszynowego 12 1.3. Wyzwania w rozpoznawaniu gestów języka migowego 13 1.4. Analiza problematyki oraz istniejących aplikacji dotyczących języka migowego 15 2. Techniczne aspekty funkcjonowania aplikacji 17 2.1. Wymagania systemowe 17 2.2. Narzędzia modelowania sztucznej inteligencji 19 3. Budowa modelu rozpoznawania gestów 21 3.1. Przygotowanie danych wejściowych 21 3.2. Preprocessing danych 24 3.3. Architektura modelu oraz algorytm klasyfikacji 26 3.4. Trening, walidacja oraz ocena skuteczności modelu 27 4. Implementacja platformy 32 4.1. Koncepcja wizualna platformy 32 4.2. Mechanizmy komunikacji między frontendem a backendem 33 4.2.1. Komunikacja serwer – użytkownik 33 4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) 34 4.3. Widoki aplikacji 34 5. Ocena działania platformy 48 5.1. Testy w rzeczywistych warunkach 48 5.1.1. Ekstrakcja punktów kluczowych 48 5.1.2. Uczenie modelu 50 .1.3. Testy w czasie rzeczywistym 52 5.2. Skuteczność i dokładność rozpoznawania gestów 53 5.3. Analiza błędów i propozycje ulepszeń 61 Podsumowanie 62 Bibliografia oraz Netografia 64 Spis ilustracji, tabel oraz wykresów 66 Wstęp Obecny rozwój technologii pozwala na automatyzację wielu procesów, w tym procesów finansowych, administracyjnych oraz sprzedażowych. Wymienione elementy są związane z obsługą biznesów, które umożliwiają komercyjny zarobek twórcom oprogramowania. Algorytmy, mają w tym przypadku ułatwić pracę lub całkowicie zastąpić osoby fizyczne w ich obowiązkach. Dzięki takim praktykom, pracodawcy, czy też różnego rodzaju organizacje, zwiększają swoje dochody poprzez przyspieszenie wykonywanej pracy lub w gorszym przypadku, oszczędzają fundusze poprzez zredukowanie etatów. Szerokie zastosowanie sztucznej inteligencji jest widoczne w każdym aspekcie naszego życia. Coraz większa ilość sklepów, banków czy też producentów wszelkiego rodzaju produktów, decyduje się na wdrażanie sztucznej inteligencji. Zgodnie z opinią specjalistów z Wyższej Szkoły Biznesu National-Louis University, AI (Artificial Intelligence) może powodować utratę miejsc pracy oraz restrukturyzację zawodów, jednakże tym samym może zwiększać zapotrzebowanie na specjalistów w branży kreatywnej oraz IT. Autor artykułu, zwraca uwagę na problematykę dotyczącą etyki związanej ze sztuczną inteligencją oraz potrzebę nieustannej nauki i rozwoju[12]. Istnieją również aspekty SI (Sztucznej Inteligencji), które są niezaprzeczalnie pozytywne, chociażby zastosowane w strefach pożytku publicznego, czy też w rozwiązaniach dla osób z dysfunkcjami. Wszelkiego rodzaju protezy, pojazdy, syntezatory mowy, algorytmy analizujące tekst, dźwięk czy też obraz, pozwalają na łatwiejsze funkcjonowanie osób niepełnosprawnych. Niestety obszary te są często pomijane, ze względu na stosunkowo niewielkie grono odbiorców, gotowych zapłacić za wprowadzenie takowych rozwiązań. Podejmując dalszą próbę analizy problemu, możemy zaobserwować wysokie zainteresowanie wadami wzroku oraz problemami ruchowymi, natomiast niskie zainteresowanie dysfunkcją głosową w odniesieniu do osób głuchoniemych. Istnieje wiele rozwiązań, które ułatwiają kontakt wzrokowy, chociażby takie jak regulacja wielkości czcionek we wszelkiego rodzaju aplikacjach, asystenci głosowi, czy też operacyjne korekty wzroku. Funkcje ruchowe, wspierane są przez różnorodne protezy, pojazdy, a także specjalne miejsca dostosowane do ich potrzeb np. miejsca parkingowe. Niestety w odniesieniu do problematyki osób głuchoniemych, nie ma zbyt wielu rozwiązań technologicznych, które mogą ułatwić ich życie w sposób nieinwazyjny. Cel i zakres pracy Celem pracy dyplomowej jest wparcie ww. grupy docelowej poprzez utworzenie oraz analizę oprogramowania do rozpoznawania sekwencji ruchów w czasie rzeczywistym, w oparciu o metody uczenia maszynowego. Zastosowane rozwiązania, wykorzystane zostaną następnie w aplikacji

służącej do nauki oraz tłumaczenia polskiego języka migowego. Oprogramowanie to może służyć jako wsparcie osób z dysfunkcjami w łatwiejszej adaptacji w środowisku osób pełnosprawnych. Aplikacja ta, również może działać w sposób edukacyjny, zwiększając świadomość użytkowników na temat dysfunkcji słuchowych oraz głosowych. Opracowanie autorskich skryptów, pozwalać ma na obsługę pełnego procesu pracy aplikacji, począwszy od rejestrowania próbek nagrań wideo, uczenia modelu SI, testowaniu jego poprawności, a kończąc na obsłudze aplikacji internetowej[4]. Niniejsza dysertacja, składa się z pięciu rozdziałów. Pierwszy z nich omawia zagadnienia teoretyczne związane z funkcjonowaniem aplikacji, takie jak przegląd dostępnych technologii związanych z rozpoznawaniem gestów oraz uczeniem maszynowym, wyzwaniami w rozpoznawaniu gestów oraz analizą istniejących platform dotyczących języka migowego. Następny rozdział omawia aspekty techniczne, które muszą zostać spełnione aby aplikacja działała w pełni poprawnie. W powyższym rozdziale znajdują się takie elementy jak: wymagania systemowe, zalecane oprogramowanie zewnętrzne oraz biblioteki, które należy zainstalować na urządzeniu docelowym. Rozdział trzeci porusza tematykę tworzenia modelu sztucznej inteligencji, przechodząc po każdym etapie jego powstawania, zaczynając od przygotowaniu danych wejściowych, a kończąc na ocenie skuteczności stworzonego modelu. Rozdział czwarty, zatytułowany „Implementacja platformy" skupia się na połączeniu aspektów sztucznej inteligencji wraz z aplikacją internetową. Zaprezentowane tam informacje dotyczą koncepcji wizualnej platformy, integracji modeli AI z aplikacją webową, mechanizmów komunikacji między klientem a serwerem oraz warstwy graficznej interfejsu użytkownika. Piąty, a zarazem ostatni rozdział niniejszej dysertacji omawia ocenę działania platformy, skupiając się na testach w warunkach rzeczywistych oraz symulowanych, ocenie skuteczności pracy aplikacji oraz propozycji ulepszeń oprogramowania. Praca zakończona została podsumowaniem, w którym zaprezentowano ogólny opis aplikacji wraz z jej analizą pod kątem przydatności oraz wydajności w rzeczywistych warunkach. 1. Przegląd literatury i analiza istniejących rozwiązań Rozpoznawanie wszelkiego rodzaju obiektów oraz ich właściwości, zyskały w ostatnich latach na popularności, głównie dzięki postępom w technikach uczenia maszynowego oraz rozwoju głębokich sieci neuronowych. W niniejszym rozdziale, omówione zostaną najważniejsze technologie oraz metody stosowane w tematyce rozpoznawania gestów, ze szczególnym uwzględnieniem języka migowego. 1.1. Przegląd technologii rozpoznawania gestów Technologie rozpoznawania gestów opierają się na stałej analizie ruchów ciała, ze szczególnym uwzględnieniem dłoni i palców, przy pomocy różnych sensorów oraz kamer. Systemy te wykorzystują zarówno dane wideo, jak i informacje trójwymiarowe, uzyskane za pomocą czujników głębi. Interdyscyplinarna dziedzina badań rozpoznawania gestów, łączy ze sobą elementy przetwarzania obrazów, komputerowego widzenia, sztucznej inteligencji oraz interakcji człowiek-komputer. Podstawą rozpoznawania gestów jest przetwarzanie obrazu, które obejmuje szereg technik służących do analizy danych wizualnych. Można wyłonić takie etapy jak: detekcja obrazów, śledzenie ruchu oraz ekstrakcja cech. Detekcja obrazów opiera się na rozpoznaniu dłoni oraz pozostałych istotnych części ciała na przetwarzanym obrazie. Techniki te bazują na algorytmach segmentacji obrazu, które identyfikują obiekty na podstawie ich koloru, kształtu lub ruchu. Algorytmy takie jak YOLO (You Only Look Once) oraz SSD (Single Shot Multibox Detector) są powszechnie stosowane do szybkiej i dokładnej detekcji dłoni w badanych obrazach. Śledzenie ruchu, opiera się na analizie każdej klatki nagrania w celu uchwycenia przesunięć, występujących w punktach wykrytych przez poprzedni proces. W tym celu stosuje się miedzy innymi algorytmy KLT (Kanade-Lucas-Tomasi) oraz Mean Shift, które pozwalają na dokładne monitorowanie trajektorii ruchu dłoni. Ostatnią techniką jest ekstrakcja cech, która umożliwia dokonanie wyodrębnienia z obrazów stawów oraz kości, niezbędnych do identyfikacji wykonanego gestu. W tym celu stosowane są takie narzędzia jak OpenPose oraz MediaPipe[3]. Biblioteka OpenPose została utworzona przez Perceptual Lab na Uniwersytecie Carnegie Mellon (CMU), pod nadzorem profesora Asuyuki Matsumoto oraz doktoranta Zhe Cao, który jest głównym autorem pracy badawczej, opisującej możliwości biblioteki OpenPose. Pierwsza wersja OpenPose została opublikowana w 2017 roku, a jej wyniki były oparte na pracy naukowej pt. "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", której współautorem był wspomniany wcześniej Zhe Cao. Praca ta po raz pierwszy zaprezentowana została na konferencji CVPR (Computer Vision and Pattern Recognition) w 2017 roku. Początkowo OpenPose służył głównie do analizy oraz detekcji pozycji wielu osób w czasie rzeczywistym na obrazach 2D. Pierwsza wersja koncentrowała się na problemie, który dotąd stanowił duże wyzwanie, a mianowicie detekcji pozycji ciała wielu osób na jednym obrazie, w jak najszybszym czasie, z zachowaniem wysokiej precyzji. Istniejące wcześniej rozwiązania, były skoncentrowane głównie na wykryciu jednego człowieka lub działały wolniej, co uniemożliwiało zastosowanie ich w czasie rzeczywistym. Biblioteka ta zyskała spore zainteresowanie, co przyczyniło się do rozszerzenia jej o dodatkowe funkcje, takie jak detekcja szczegółowych ruchów dłoni i mimiki

twarzy oraz analizy 3D. Głównym zadaniem obecnej wersji biblioteki jest wykrywanie oraz śledzenie pozycji ciała człowieka w obrazie 2D oraz 3D, uwzględniając różne części ciała takie jak kończyny, głowa oraz palce rąk i stóp. Podstawą działania OpenPose są techniki głębokiego uczenia, a dokładniej konwolucyjne sieci neuronowe (CNN), które za pomocą punktów kluczowych (tzw. keypoints), rozpoznają położenie ciała, dzieląc je na odpowiednie struktury. W początkowej fazie programu, biblioteka przeprowadza detekcję punktów kluczowych, następnie tworzy mapę ufności, która określa z jakim prawdopodobieństwem dany punkt (np. łokieć) znajduje się w określonej lokalizacji na obrazie. Utworzona mapa jest dwuwymiarową siatką, gdzie wartości w poszczególnych komórkach reprezentują stopień pewności. Po utworzeniu mapy ufności, biblioteka tworzy mapy części ciała, analizując położenie konkretnych elementów, co w dalszym etapie służy do utworzenia szkieletu osoby. W tym celu wykorzystywane są mapy PAF (Part Affinity Fields), czyli wektorowe pola, które wskazują, jak prawdopodobne jest, że dwa punkty kluczowe są ze sobą powiązane. Gdy mapy te zostaną utworzone, biblioteka przystępuje do połączenia punktów tworząc szkielet[2]. Druga spośród wymienionych bibliotek (MediaPipe) została udostępniona jako projekt open-source przez firmę Google w czerwcu 2019 roku. MediaPipe, jest to otwartoźródłowe środowisko, które umożliwia wykonywanie różnych zadań związanych z przetwarzaniem obrazów, wideo oraz danych w czasie rzeczywistym. Zbudowana została na podstawie struktury pipelinów przetwarzania multimodalnych danych, co oznacza, że pozwala na tworzenie i uruchamianie strumieniowych systemów analizy danych, w sposób modułowy oraz wydajny. Pipeline to zbiór skomponowanych kroków przetwarzania (tzw. modułów) w ustalonej kolejności. Każdy z modułów odpowiedzialny jest za określone zadanie np. detekcję obiektów, segmentację czy śledzenie ruchu. Dzięki temu, biblioteka ta jest bardzo elastyczna, gdyż użytkownik może z łatwością tworzyć własne pipeline'y, dostosowując je do specyficznych potrzeb. Pierwotnym założeniem MediaPipe było dostarczenie wydajnej biblioteki, która umożliwiałaby wieloplatformowe przetwarzanie strumieni wideo i obrazów w czasie rzeczywistym[6]. Przechodząc do kwestii hardware, najczęściej wykorzystywanymi urządzeniami do rejestracji gestów są kamery RGB. Ich popularność jest uargumentowana stosunkowo niską ceną oraz dostarczanym kolorowym obrazem w wysokiej rozdzielczości, który może być przetwarzany przez algorytmy komputerowego widzenia. Nie są to jednakże jedyne narzędzia, wykorzystywane w procesie rejestrowania ruchów. Elementy takie jak kamery głębi pozwalają na trójwymiarową rejestrację sceny, co umożliwia bardziej precyzyjną analizę ruchów dłoni i ciała. Dużą zaletą kamer głębi jest wysoka dokładność w trudnych warunkach oświetleniowych, gdzie w porównaniu z kamerami RGB, nie mają problemów z dokładnym uchwyceniem gestu. Dodatkowym asortymentem, mogą być również czujniki ruchu, które mierzą przyspieszenie i orientację dłoni. Zastosowanie ww. technologii, pozwala na rejestrowanie subtelnych ruchów i położenia dłoni w przestrzeni. Ostatnia z przestawionych opcji, daje największe możliwości rejestrowania ruchów. Niestety ze względu na koszt zaawansowanych technologii stosowanych w czujnikach ruchu, produkty te są nieprzystępne cenowo dla potencjalnego odbiorcy oprogramowania, omawianego w niniejszej dysertacji[9]. Rysunek 1 Porównanie technologii (Źródło: opracowanie własne) 1.2. Przegląd metod uczenia maszynowego Uczenie maszynowe (ang. Machine Learning (ML)) to dziedzina sztucznej inteligencji, która skupia się na rozwijaniu algorytmów oraz modeli zdolnych do uczenia i automatycznego podejmowania decyzji na podstawie dostarczonych danych. W odniesieniu do platformy tłumaczącej język migowy jest kluczowym rozwiązaniem, umożliwiając przekształcanie danych wideo w rozpoznawalne wzorce, które są możliwe do przetłumaczenia na słowa lub zdania. Pierwsza sieć neuronowa została opracowana przez Warrena McCullocha, wykładowcy z University of Illinois i Waltera Pittsa, niezależnego badacza. Ich artykuł pt. „A Logical Calculus of the Ideas Immanent in Nervous Activity", opublikowany w 1943 roku w czasopiśmie „Bulletin of Mathematical Biophysics" został przyjęty z szerokim entuzjazmem, kładąc tym samym podwaliny pod rozwój sztucznych sieci neuronowych. Praca ta okazała się kamieniem milowym w późniejszych badaniach w dziedzinie informatyki, neurobiologii i kognitywistki. Autorzy, przedstawili w nim jak mogą działać neurony, prezentując swoje pomysły i tworząc prostą sieć neuronową za pomocą obwodów elektrycznych[15]. Obecnie sieci neuronowe są fundamentem współczesnych metod uczenia maszynowego. Składają się z wielu warstw neuronów, które przetwarzają dane wejściowe poprzez szereg operacji matematycznych, co umożliwia uczenie się złożonych wzorców. Konwolucyjne sieci neuronowe (CNN) są szczególnie efektywne w przetwarzaniu danych obrazowych oraz wideo, dzięki swojej zdolności do automatycznego wyodrębnienia cech z surowych danych. Sieci te, składają się z warstw konwolucyjnych, poolingowych oraz gęstych, które współpracują ze sobą, w celu przetwarzania informacji o strukturze przestrzennej obiektów. Autorzy publikacji Sign Language Recognition Using Convolutional Neural Networks wykazali, że CNN mogą być

używane do skutecznego przetwarzania i rozpoznawania gestów na podstawie obrazu wideo, umożliwiając automatyczną ekstrakcję cech, co jest kluczowe dla poprawnej klasyfikacji gestów. Podkreślili również, że ich model nie tylko osiąga wysoką dokładność w rozpoznawaniu gestów, ale także jest skalowalny i może być rozszerzany o nowe gesty lub języki migowe z relatywnie niewielkim wysiłkiem. Badanie to otwiera drogę do tworzenia bardziej zaawansowanych systemów rozpoznawania języka migowego, które mogą być wykorzystywane w aplikacjach takich jak tłumacze języka migowego na mowę w czasie rzeczywistym, tworzenie interfejsów użytkownika dla osób niesłyszących, a także w edukacji[10]. Modele hybrydowe to połączenie różnych algorytmów klasyfikacyjnych, które mogą prowadzić do poprawy dokładności i stabilności rozpoznawania gestów. Przykładem takiego podejścia jest kombinacja CNN oraz HMM, co pozwala na skuteczniejsze modelowanie zależności czasowych w danych sekwencyjnych. Zastosowanie hybrydy modeli CNN oraz HMM opisuje publikacja pt. „SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition". Autorzy publikacji wykazują, że hybrydowe podejście może być skutecznie wykorzystywane do rozpoznawania ciągłych gestów w języku migowym. Ich badania przeprowadzone zostały na dużych zestawach danych, zawierających filmy z gestami języka migowego, przy czym wykorzystali takie techniki jak augmentacja danych, zwiększając tym samym różnorodność przykładów i poprawiając zdolność generalizacji modelu. Wyniki eksperymentu pokazują, że SubUNets osiąga znacznie lepsze wyniki w porównaniu z wcześniejszymi podejściami, zarówno w kontekście rozpoznawania kształtów dłoni, jak i ciągłego rozpoznawania gestów języka migowego. Autorzy podkreślają, że ich architektura nie tylko rozpoznaje gesty z większą dokładnością, ale także lepiej radzi sobie z różnorodnością kształtów dłoni i płynnością sekwencji[1]. 1.3. Wyzwania w rozpoznawaniu gestów języka migowego Rozpoznanie gestów języka migowego stanowi złożone wyzwanie, które wynika z unikalnych cech ww. sposobu komunikacji. W przeciwieństwie do języków mówionych, język migowy wykorzystuje mimikę oraz posturę ciała, co wymaga zaawansowanych algorytmów do skutecznej analizy i rozpoznania. Język ten charakteryzuje się ogromną różnorodnością gestów. Każdy z nich może mieć wiele wariantów, które różnią się w zależności od regionu, kultury, a nawet osoby wykonującej gest. Ponadto, gesty mogą obejmować stosunkowo nieistotne różnice w ruchach dłoni, ułożeniu palców, kierunku spojrzenia, a także w mimice twarzy, co sprawia, że poprawne rozpoznanie jest niezwykle trudne, zwłaszcza jeżeli model zostaje rozszerzany o nowe gesty oraz ich warianty. Badania prowadzone przez Koller i innych naukowców wskazują, że klasyczne podejścia oparte na modelach HMM mogą być niewystarczające do modelowania złożonych i różnorodnych gestów języka migowego. Aby sprostać temu wyzwaniu, autorzy sugerują zastosowanie hybrydowych modeli łączących CNN oraz HMM, co pozwala na skuteczniejsze modelowanie złożoności gestów[5]. Gesty języka migowego często wykonywane są w sposób płynny oraz nieprzerwany co stwarza jeden z największych problemów w przypadku zastosowania algorytmów SI. Kwestia ta jest niezwykle problematyczna dla modeli, które do poprawnego działania potrzebują wyraźnych przerw pomiędzy poszczególnymi znakami, aby określić gdzie badany gest się zaczyna, a gdzie kończy. Kluczowym problemem podczas tworzenia modelu sztucznej inteligencji jest również sama tematyka. W związku z niszowością tematu, utrudniony jest dostęp do dużych, zróżnicowanych zbiorów danych, które są niezbędne dla skutecznego trenowania modeli uczenia maszynowego. W przypadku języka migowego, zbiory danych są ograniczone pod względem wielkości i różnorodności. Brakuje danych pochodzących od różnych użytkowników, wykonanych w zróżnicowanych warunkach oświetleniowych i środowiskach, co może prowadzić do słabszej generalizacji modelu. Problematykę tą porusza w swoich badaniach Patel oraz inni naukowcy, zwracając uwagę na problem ograniczonych zbiorów danych, proponując jednocześnie zastosowanie techniki transfer learningu, które pozwalają na wykorzystanie wiedzy zebranej na większych zbiorach danych do trenowania modeli na mniejszych, specyficznych zbiorach języka migowego[6]. Rozpoznawanie gestów w czasie rzeczywistym wymaga również znacznej mocy obliczeniowej, zwłaszcza w przypadku samego tworzenia modelu oraz ekstrakcji punktów na zbiorach treningowych. Programy wykorzystujące przetwarzanie obrazu w czasie rzeczywistym muszą przetwarzać duże ilości danych wideo o wysokiej rozdzielczości, minimalizując poziom opóźnienia do minimum. Optymalizacja rozwiązań w stosunku do urządzeń o gorszych parametrach również stwarza dodatkowe problemy, zwłaszcza w przypadku urządzeń mobilnych. Ostatnim powszechnie znanym wyzwaniem może być różnica w indywidualnym zachowaniu użytkownika. Każda osoba ma inną długość oraz szerokość kończyn. Inna szybkość, kąt nachylenia czy też indywidualny styl poruszania się każdej osoby sprawia, że opracowany model musi być odporny na takowe odchylenia. Na elastyczność modelu zwraca uwagę między innymi Zhang i Liu, wskazując konieczność opracowania metod, które mogą uwzględnić różnice indywidualne, takie jak ww. style gestykulacji, poprzez wykorzystanie cech całej dłoni, co pozwala na dokładniejsze rozpoznawanie gestów w różnych warunkach[11]. 1.4. Analiza

problematyki oraz istniejących aplikacji dotyczących języka migowego W ciągu ostatnich lat rozwój technologii głębokiego uczenia oraz komputerowego widzenia przyczynił się do powstania wielu aplikacji, służących do rozpoznawania i tłumaczenia języka migowego. Aplikacja opracowana przez Patel i Sahah, pozwala użytkownikom uczyć się języka migowego za pomocą interaktywnych ćwiczeń. System ten, analizuje gesty użytkownika za pomocą technologii rozpoznawania wideo w czasie rzeczywistym, umożliwiając otrzymanie natychmiastowej informacji zwrotnej, co przyczynia się do skuteczniejszej nauki[8]. Aplikacja o nazwie DeepSign, również funkcjonuje w przestrzeni publicznej jako tłumacz języka migowego. Aplikacja ta wykorzystuje CNN do rozpoznawania gestów i przekształcania ich w tekst w czasie rzeczywistym. System ten jest bardzo zaawansowany, co pozwala tłumaczyć gesty na pełne zdania, co znacznie ułatwia komunikację[7]. Niestety każda z wyżej wymienionych aplikacji jest stworzona na rynek anglojęzyczny, a co za tym idzie, nie obsługuje polskiego języka migowego. Zapoznając się z aktualną na dzień 27.08.2024 ofertą na rynek polskojęzyczny, można odnaleźć jedynie rozwiązania, które nie są zautomatyzowane. Specjalne ośrodki czy też firmy prywatne obsługują klientów w ramach spotkań lub połączeń online z tłumaczem języka migowego. Analizując problematykę zastosowania sztucznej inteligencji, można dostrzec, iż nisza ta nie została jeszcze odpowiednio obsłużona w Polsce. Rozwiązania takie jak wideokonferencje, są skuteczną alternatywą, jednakże wymagają osoby fizycznej, która takową rozmowę przetłumaczy. Wiąże się to z kosztami zatrudnienia specjalisty oraz utrzymania infrastruktury, takiej jak stabilne połączenie z Internetem, odpowiedni sprzęt audio-wizualny oraz platformy obsługującej powyższe rozmowy. Zagłębiając się bardziej w poruszaną tematykę, można odnieść wrażenie, że osoby z dysfunkcjami instrumentów głosowych są w pewien sposób wykluczone społecznie, poprzez brak powszechnego narzędzia, które ułatwiałoby im życie codzienne w sferze komunikacyjnej. W przypadku każdego typu rozmów, należy skupić się na zagadnieniu prywatności, do której każdy obywatel ma prawo. Chociażby w taki sferach jak zdrowie czy też finanse. Dzięki istnieniu aplikacji, która byłaby w stanie w pełni przetłumaczyć język migowy, osoby z dysfunkcjami mowy, mogłyby korzystać z większości znanych usług, bez obecności osoby trzeciej, która będzie towarzyszyć w prowadzeniu rozmowy. Zaleca się aby aspekt ten został szerzej zbadany przez specjalistów w dziedzinie badań społecznych, analizując jednocześnie potrzeby osób niepełnosprawnych, a tym samym zwiększając świadomość społeczeństwa na temat poruszany w omawianej dysertacji. 2. Techniczne aspekty funkcjonowania aplikacji Każde oprogramowanie wymaga uprzedniego przygotowania środowiska pracy. Tak też jest w przypadku aplikacji dołączonej do niniejszej dysertacji. Wybrane etapy posiadają własne wymagania odnośnie sprzętu (ang. hardware), oprogramowania (ang. software) oraz środowiska programistycznego, dlatego w poniższych punktach przedstawione zostały kroki do poprawnej konfiguracji środowiska. 2.1. Wymagania systemowe Program podzielony został na dwa etapy główne. Pierwszy etap odpowiada za przygotowanie próbek oraz modelu sztucznej inteligencji, natomiast drugi etap jest stricte związany z platformą, która ma za zadanie odczytywać przygotowany w etapie pierwszym model SI. Każdy program posiada inne wymagania systemowe, dlatego też zostały one zawyżone w odniesieniu do najbardziej wymagającego programu z etapu pierwszego. Komponent Minimalne wymagania Zalecane wymagania Procesor (CPU) Sześciordzeniowy, np. Intel Core i5- Ośmiordzeniowy, np. Intel Core 10600K lub AMD Ryzen 5 3600 i7-9700K lub AMD Ryzen 7 3700X Pamięć RAM 16 GB RAM 32 GB RAM Karta graficzna Zintegrowana Zintegrowana (GPU) Przestrzeń Minimum 20 GB wolnej przestrzeni na SSD NVMe z minimum 100 GB dyskowa dysku SSD wolnej przestrzeni System Windows 10 lub nowszy / Linux (Ubuntu Windows 10 Pro lub nowszy / operacyjny 20.04 lub nowszy) Linux (Ubuntu 20.04 LTS lub nowszy) Inne Python 3.8 lub nowszy Python 3.8 lub nowszy Informacje Procesor ośmiordzeniowy z obsługą wielowątkowości zapewni dobrą wydajność w przetwarzaniu wideo i trenowaniu modeli, choć na nieco niższym poziomie niż wcześniej wspomniane opcje. Zintegrowana karta graficzna wystarczy do obsługi podstawowych zadań związanych z wyświetlaniem i przetwarzaniem obrazu, ponieważ główne obciążenie będzie przeniesione na procesor, aby zmienić obciążenie z procesora na kartę graficzną należy dokonać modyfikacji programu „MachineLearning.py”. 32 GB RAM pozwoli na komfortową pracę z większymi zbiorami danych i bardziej złożonymi modelami Tabela 1 Wymagania systemowe cz1 Druga część programu opiera się głównie na obsłudze aplikacji internetowej oraz przetwarzaniu obrazu z kamery na podstawie dostarczonego już modelu, co nie wymaga od użytkownika zwiększonej mocy obliczeniowej. Komponent Minimalne wymagania Zalecane wymagania Procesor Czterordzeniowy procesor, np. Intel Sześciordzeniowy procesor, np. Intel (CPU) Core i5-8265U lub AMD Ryzen 5 Core i7-10750H lub AMD Ryzen 5 2500U 4600H Pamięć RAM 8 GB RAM 16 GB RAM Karta Zintegrowana, np. Intel UHD Zintegrowana, np. Intel Iris Xe lub graficzna Graphics 620 lub AMD Radeon AMD Radeon Vega 11 (GPU)

Vega 8 Przestrzeń Minimum 10 GB wolnej przestrzeni SSD NVMe z minimum 50 GB wolnej dyskowa na dysku SSD przestrzeni System Windows 10 lub nowszy / Linux Windows 10 Pro lub nowszy / Linux operacyjny (Ubuntu 20.04 lub nowszy) (Ubuntu 20.04 LTS lub nowszy) Inne Python 3.8 lub nowszy Python 3.8 lub nowszy Tabela 2 Wymagania systemowe cz2 2.2. Narzędzia modelowania sztucznej inteligencji Zgodnie z informacjami przedstawionymi w podpunkcie 2.1, każdy program posiada inne wymagania sprzętowe. W tym punkcie przedstawione zostaną indywidualne wymagania dotyczące poszczególnych programów, które nie zostały uwzględnione w poprzednich punktach. Pakiety wymagane do uruchomienia poszczególnych aplikacji, można zainstalować poprzez narzędzie do zarządzania pakietami w Pythonie (pip), wpisując wskazane komendy w terminalu. Program „NagrywanieVideo720p60FPS.py" do poprawnego działania potrzebuje podłączonej kamery internetowej obsługującej jakość 720p oraz przepustowość 60fps, a także importu pakietów takich jak os, time oraz opencv. Dwa pierwsze pakiety są dostępne w podstawowej wersji Pytthona, natomiast pakiet opencv musi zostać doinstalowany. Aby tego dokonać można skorzystać z poniższego polecenia: pip install opencv-python Do obsługi programu „MachineLearning.py" wymagane są pakiety: os, threading, warnings, random, time, webbrowser, multiprocessing, tkinter, cv2, numpy, mediapipe, tensorflow, keras, flatten, scikit-learn oraz pil , osiem pierwszych pakietów to standardowe moduły języka Python, brakujące pakiety można zainstalować inicjując polecenie: pip install opencv-python numpy mediapipe tensorflow keras scikit-learn pillow Program „Prediction Test.py" wymaga takich pakietów jak: cv2, numpy, mediapipe, tensorflow, matplotlib, keras, sklearn, os, tkinter. Brakujące pakiety można zainstalować za pomocą polecenia: pip install opencv-python numpy mediapipe tensorflow keras scikit-learn pillow Do poprawnej obsługi platformy, wymagane są również odpowiednie środki, takie jak pakiety języka python oraz zewnętrzne oprogramowanie inicjujące środowisko serwerowe. Omawiana platforma wymaga utworzenia lokalnego serwera w celu obsługi języka PHP. Przykładem oprogramowania pozwalającego na zainicjowanie ww. środowiska jest XAMPP. Do utworzenia niniejszej aplikacji wykorzystano najnowszą (na dzień 22.08.2024) wersję XAMPP w wersji 3.3.0 oraz PHP w wersji 8.2.12. Po zainstalowaniu aplikacji XAMPP, należy uruchomić opcję Apache, oznaczoną na Rysunku 2 numerem 1. Rysunek 2 Interfejs programu XAMPP Platforma, została projektowana oraz testowana za pomocą przeglądarki internetowej Chrome, która jest zalecana do obsługi ww. aplikacji. Program „app.py" potrzebuje podłączonej kamery internetowej o przepustowości minimum 30 fps oraz zainstalowanych pakietów takich jak: Flask, flask-cors, opencv, numpy, mediapipe, tensorflow, keras, pillow oraz scikit-learn. Elementy te, można zainstalować poprzez komendy w terminalu: pip install Flask flask-cors opencv-python numpy mediapipe tensorflow keras Pillow scikit-learn 3. Budowa modelu rozpoznawania gestów Aplikacja internetowa do właściwego działania potrzebuje wcześniej przygotowanego modelu sztucznej inteligencji, który będzie rozpoznawał wybrane sekwencje ruchów. W tym rozdziale zostanie przedstawiony opis procesu przygotowania ww. modelu. 3.1. Przygotowanie danych wejściowych Przygotowanie danych wejściowych jest kluczowym etapem każdego programu opartego na uczeniu maszynowym. Proces ten obejmuje takie elementy jak wybór odpowiednich źródeł danych, ich organizację, walidację oraz przetworzenie do odpowiedniego formatu. W związku z niszowością tematyki polskiego języka migowego oraz utrudnionym dostępem do próbek badawczych, autor dysertacji utworzył własną bazę gestów, które poprzez algorytm napisany w języku python zostały zarejestrowane za pomocą kamery internetowej. Każde z nagrań musiało zostać ustandaryzowane w celu przystosowania ich do pracy z jednolitym modelem sztucznej inteligencji. Każda z zarejestrowanych próbek posiadała długość 4 sekund, rozdzielczość 720p oraz przepustowość 60 klatek na sekundę. Dodatkowo, autor przyjął metodę 25-50-25, która zaowocowała nagraniem 100 próbek dla każdego gestu. W późniejszym etapie projektu powtórzono ten proces, aby zyskać większą ilość próbek na innych modelach ciał. Metoda ta opierała się na odpowiednim ustawieniu kadru. Pierwszy etap tworzył 25 nagrań od czubka głowy do dolnej części żeber. Taka konfiguracja, pozwalała na najdokładniejsze zarejestrowanie pracy dłoni, wliczając w to układ palców. Drugi etap tworzył 50 nagrań od czubka głowy do górnej części uda. Nagrania te służyły jako baza do całości gestu. Ujęcia te, ujmowały każdą fazę ruchu, od fazy startowej do fazy końcowej, uwzględniając wszystkie niezbędne elementy ludzkiego działa. Ostatni etap tworzył 25 nagrań od czubka głowy do dolnej części kolan. Proces ten miał za zadanie zwiększyć obszar widzenia programu, uwzględniając dalsze położenie szukanych punktów, dzięki czemu utworzony w późniejszym etapie model, mógł z większą precyzją rozpoznawać pozycję statyczną oraz ruch odpowiednich fragmentów ciała człowieka. Ustawienie każdego z etapów zostało zobrazowane na ilustracji 3. Rysunek 3 Etapy zbierania próbek (Źródło: opracowanie własne) Dzięki zastosowaniu ww. rozwiązania, późniejsze algorytmy przetwarzania danych, mogły z większą łatwością zlokalizować wybrane punkty na ludzkim ciele, a następnie przetworzyć je w ramach sekwencji ruchu. Każdy z etapów

dostarczał inną ilość punktów, dzięki czemu model uczył się rozpoznawania gestów ze zróżnicowanej perspektywy. W ramach części praktycznej pracy magisterskiej, został dołączony program „NagrywanieVideo720p60FPS.py", który realizuje wyżej wymienione etapy. Po zakończeniu procesu przygotowywania próbek badawczych autor dysertacji opracował program „MachineLearning.py", który realizuje wszystkie pozostałe kroki do uzyskania działającego modeli SI. Po uruchomieniu ww. programu, użytkownik ma możliwość skorzystania z graficznego interfejsu programu, utworzonego poprzez bibliotekę tkinter. Biblioteka ta jest standardowym pakietem w języku Python. Początkowo, program weryfikuje czy w obecnym katalogu istnieje plik z przetworzonymi punktami o nazwie „preprocessed_data.npz". Jeżeli takowy plik istnieje, zostaje wyświetlone okno z odpowiednim komunikatem (Rys.4). Rysunek 4 Uczenie maszynowe (GUI 1) (Źródło: opracowanie własne) Użytkownik może wybrać jedną z trzech dostępnych opcji. Pierwsza z nich prowadzi do pliku PDF, w którym opisane zostały wymagania sprzętowe dotyczące obsługi programu. Przycisk po prawej stronie, rozpoczyna proces uczenia maszynowego na obecnie wykrytych danych, natomiast przycisk „Wróć do wyboru danych" pozwala na otworzenie widoku okna startowego. Okno to pojawia się również, jeżeli program w początkowej fazie działania nie odnajdzie utworzonego pliku z wyekstraktowanymi punktami. Rysunek 5 Uczenie maszynowe (GUI 2) (Źródło: opracowanie własne) W ww. oknie, użytkownik może wybrać ścieżkę do katalogu, w którym znajdują się wcześniej przygotowane nagrania. Nagrania te, muszą być zgrupowane w odpowiedni sposób. Aby program działał poprawnie, struktura plików powinna wyglądać następująco: Katalog wybrany przez użytkownika / Nazwa sekwencji ruchu / pliki o rozszerzeniu *.avi. Jeżeli program wykryje błędną strukturę plików, zwróci komunikat w kolorze czerwonym. W przypadku poprawnie przetworzonych plików, wyświetli stosowne powiadomienie w kolorze zielonym, tworzy listę wszystkich ścieżek do plików wideo, a następnie odblokuje przycisk „Rozpocznij analizę". Każdy z powyższych przykładów został zobrazowany, za pomocą poniższej grafiki. Rysunek 6 Uczenie maszynowe (GUI 2 - Struktura) (Źródło: opracowanie własne) 3.2. Preprocessing danych Preprocessing polega na wstępnym przetwarzaniu danych, a dokładniej na przekształcaniu surowych danych w formę, która jest bardziej odpowiednia i optymalna dla modelu uczenia maszynowego. Celem tego fragmentu kodu, jest utrzymanie spójności oraz kompatybilności, a także poprawa jakości danych, potrzebnych do uczenia modelu. Usuwa on wszystkie zbędne informacje, czyli tzw. szumy, pozostawiając jedynie potrzebne dane, wymagane przez model SI. Po rozpoczęciu analizy nagrań, program wyświetla nowe okno z paskiem postępów oraz możliwością rozszerzenia widoku o szczegóły. Na rysunku 7, po lewej stronie przedstawiono podstawowe okno przetwarzania danych, po prawej stronie natomiast jest widoczne okno po rozszerzeniu szczegółów. Rysunek 7 Uczenie maszynowe (GUI 3) (Źródło: opracowanie własne) W trakcie przetwarzania danych, algorytm wykorzystuje bibliotekę MediaPipe, opracowaną przez Google. Za jej pomocą, przeprowadza ekstrakcję punktów kluczowych z dostarczonych nagrań, analizując każdą z 240 klatek, a następnie przypisuje do nich odpowiednie etykiety. W ten sposób tworzy pełen spis punktowy, przedstawionego na nagraniu gestu. Zastosowanie powyższego rozwiązania, skutkuje zmniejszeniem ilości danych przetwarzanych przez etap uczenia maszynowego. W przypadku standardowego podejścia, program przetwarzałby każdy pikseli wideo, co mogłoby wprowadzić dodatkowy szum. Skutkowałoby to mniej precyzyjnym uczeniem modelu, a co za tym idzie gorszą precyzją rozpoznawania gestów. Program, posiada również możliwość podglądu wizualizacji procesu ekstrakcji punktów w czasie rzeczywistym. Gdy użytkownik po rozwinięciu szczegółów, uruchomi przycisk „Wyświetl wizualizację" otworzy się nowe okno, które zobrazuje przetwarzanie danych w sposób graficzny. Rysunek 8 Uczenie maszynowe (GUI 3 - Wizualizacja) (Źródło: opracowanie własne) Podczas przetwarzania informacji, program normalizuje dane poprzez pomijanie wideo o dłuższym czasie trwania lub mniejszej ilości klatek na sekundę, jeżeli natomiast nagranie jest krótsze, brakująca część czasu jest dodawana poprzez funkcję uzupełniania sekwencji pad_sequences. Funkcja ta, tworzy macierz wypełnioną zerami o wymiarach zdefiniowanych w kodzie programu, następnie dla każdej sekwencji kopiuje jej wartość do nowej macierzy, a jeżeli sekwencja jest krótsza niż maksymalna długość, resztę wypełnia zerami. Program, ze względu na konieczność przetwarzania dużej ilości danych oraz potencjalnie długi okres trwania ww. procesu, posiada obsługę wieloprocesowości. Wieloprocesowe podejście pozwala na jednoczesne przetwarzanie wielu nagrań na rdzeniach procesora, co znacząco redukuje czas potrzebny na preprocessowanie danych. Funkcje, takie jak blokada oraz kolejka zapewniają synchronizację między procesami, aby zapobiec ewentualnym kolizjom podczas dostępu do wspólnych zasobów, takich jak zapis do listy danych, czy aktualizacja wskaźnika postępu. Gdy dane zostaną poprawnie przetworzone, program tworzy plik „preprocessed_data.npz", w którym zapisuje przetworzone dane oraz „classes.npy" w którym przechowuje nazwy etykiet. Następnie pasek postępu dobiegnie końca, a na ekranie

użytkownika wyświetla się komunikat, który umożliwia przejście do procesu uczenia modelu SI lub zamknięcie programu. Rysunek 9 Uczenie maszynowe (GUI 3 - Zakończenie przetwarzania danych) (Źródło: opracowanie własne) 3.3. Architektura modelu oraz algorytm klasyfikacji Architektura modelu odnosi się do struktury sieci neuronowej. To ona determinuje sposób, w jaki dane są przetwarzane, jakie operacje są wykonywane oraz jakie mechanizmy są stosowane do obsługi modelu. W omawianym oprogramowaniu, autor dysertacji zastosował model oparty na konwolucyjnych sieciach neuronowych (CNN), co miało na celu uchwycenie wzorców w danych sekwencyjnych, takich jak punkty kluczowe wyodrębnione z nagrań wideo. W kontekście przetwarzania sekwencji, warstwy konwolucyjne (Conv1D) przetwarzają dane jednokierunkowo. Filtrują sekwencje punktów kluczowych, tym samym wykrywając wzorce przestrzenno-czasowe. Filtry, inaczej zwane jądrami, są przesuwane wzdłuż sekwencji, stosując operację splotu. Czynność ta, pozwala na uchwycenie lokalnych wzorców danych. Po każdej warstwie konwolucyjnej zastosowano warstwę MaxPooling, która redukuje wymiarowość danych, zachowując najbardziej istotne cechy. MaxPooling wybiera maksymalną wartość z każdej lokalnej grupy wyników, co pomaga w uodpornieniu modelu na przesunięcia w danych. Równocześnie dokonuje redukcji liczby parametrów, co skutkuje zmniejszeniem ryzyka wystąpienia zjawiska przeuczenia modelu. Model został zaimplementowany przy pomocy biblioteki Keras, która działa na bazie TensorFlow. Zastosowane rozwiązanie oferuje szeroką gamę narzędzi do pracy z przetwarzaniem danych, budowaniem architektury sieci, a także optymalizacji modeli. Omawiany program, tworzy model w sposób sekwencyjny, co oznacza, że warstwy są dodawane jedna po drugiej, tworząc stos warstw. Każda z warstw przyjmuje jako argumenty liczbę filtrów o wymiarze 64, rozmiar jądra równy 3 oraz funkcję aktywacji „relu". Liczba filtrów określa, ile różnych cech model będzie próbował wyodrębnić. Rozmiar jądra to zakres, na którym filtr jest stosowany w danej chwili. Funkcja aktywacji relu, poprzez zamianę wszystkich wartości ujemnych na zero, zapobiega problemowi zanikania gradientu i przyspiesza proces nauki, umożliwiając sieci efektywne uczenie się złożonych wzorców. Warstwa Flatten w niniejszym programie, używana jest do przekształcenia danych z formatu wielowymiarowego do jednowymiarowego, który jest wymagany przez kolejne, w pełni połączone warstwy takie jak warstwa gęsta Dense. Warstwa ta, używana jest zarówno do przekształcania wyodrębnionych cech w postać o mniejszej wymiarowości, jak i do generowania ostatecznego wyniku klasyfikacji. Units, czyli liczba neuronów w warstwie została ustawiona na 256, co pozwala na modelowanie bardziej złożonych zależności, jednakże może prowadzić do przeuczenia. W ostatnim kroku, warstwa Dense generuje wyniki klasyfikacji za pomocą funkcji aktywacji softmax. Funkcja ta, przekształca wyniki w prawdopodobieństwa, które sumują się do 1, co pozwala na przypisanie prawdopodobieństwa do klasy. 3.4. Trening, walidacja oraz ocena skuteczności modelu Trening modelu to proces optymalizacji jego parametrów tak, aby jak najlepiej uczył się rozpoznawania wzorców dostarczonych przy pomocy danych wejściowych, a następnie właściwie przypisywał je do odpowiednich etykiet. W kontekście sieci neuronowych, trening polega na minimalizacji funkcji straty, przy użyciu danych treningowych oraz odpowiedniego optymalizatora. W pierwszej fazie treningu modelu, zastosowany został tzw. forwadr propagation, czyli etap w którym dane przekazywane są do modelu, a każdy neuron przekształca wejścia, oblicza wyjścia i przekazuje je do następnej warstwy. Na końcu tego procesu, model generuje wynik, który jest porównywany z rzeczywistą etykietą danych przy użyciu funkcji straty. Następnie występuje backpropagation, czyli kluczowy mechanizm w treningu sieci neuronowych. Polega on na propagacji błędu wstecz przez sieć neuronową. Błąd ten jest obliczany jako różnica pomiędzy przewidywaniami modelu, a rzeczywistymi wynikami. Proces ten, aktualizuje wagi w sieci tak, aby minimalizować funkcję straty. W tym przypadku zastosowany został algorytm Adam, który jest odmianą algorytmu gradientu prostego (Stochastic Gradient Descent, SGD). Algorytm ten stosuje poniższy wzór matematyczny: Rysunek 10 Wzór algorytmu gradientu prostego gdzie: - w to wagi, które są optymalizowane, η (eta) to współczynnik uczenia się (ang. learning rate), który określa, jak duży krok robimy w kierunku minimalizacji funkcji kosztu (ang. Loss function), natomiast pozostała część to to pochodna funkcji kosztu Loss względem wag w, czyli gradient funkcji kosztu. Pokazuje kierunek i szybkość zmiany funkcji kosztu w zależności od zmian wag. Późniejszy etap obejmuje tworzenie tzw. epok. W każdej epoce, model przetwarza cały zbiór treningowy. Aby zwiększyć efektywność aktualizacji wag, zastosowane zostało porcjowanie (batch), które pozwala na przetwarzanie mniejszej ilości informacji jednocześnie. Podczas treningu, model uczy się na danych treningowych, jednakże istnieje pewne ryzyko, że nauczy się on wzorców specyficznych dla tych danych. Aby zapobiec ww. problematyce, zastosowano techniki regularyzacji takie jak Dropout oraz L2 Regularization (Ridge). Pierwsza z nich polega na losowym wyłączeniu części neuronów w warstwie podczas treningu. W omawianym programie wartość ta została ustawiona na 0.5, co oznacza, że połowa neuronów w warstwie jest wyłączana w każdej iteracji. Druga zaś, dodaje karę do funkcji straty za duże wartości wag, co ogranicza złożoność

modelu i zapobiega jego przeuczeniu. Walidacja modelu jest niezbędna w procesie uczenia maszynowego. W jego trakcie program ocenia, jak dobrze model generalizuje, czyli jak radzi sobie z danymi, których nie uwzględnił podczas treningu. Autor pracy dyplomowej, przyjął złożenie 20% / 80%, gdzie 20% danych jest wykorzystywanych do walidacji, natomiast 80% do trenowania modelu. W trakcie uczenia, stosowane są callback'i. EarlyStopping, czyli wczesne zatrzymanie uczenia maszynowego, wykorzystane jest w celu zabezpieczenia modelu przed ewentualnym przeuczeniem. Proces ten stale monitoruje wydajność modelu na zbiorze walidacyjnym i przerywa trening w przypadku braku poprawy wyników. Istnieje również callback nazwany ModelCheckpoint, który zapisuje najlepsze wagi modelu na podstawie wybranej metryki. Cały proces uczenia maszynowego, został zobrazowany poprzez interfejs graficzny użytkownika, który pojawia się po zatwierdzeniu przycisku dostępnego po przetworzeniu nagrań. Rysunek 11 Uczenie maszynowe (GUI 4) (Źródło: opracowanie własne) Gdy uczenie maszynowe dobiegnie końca, program tworzy plik o nazwie best_model.h5, w którym zapisane są wszystkie informacje opracowywane przez powyższe algorytmy. Użytkownik, otrzymuje również powiadomienie dotyczące zakończenia pracy programu. Na tym etapie, może on zweryfikować poprawność uczenia maszynowego, a także sprawdzić ile epok przetworzył program w celu utworzenia modelu. Zatwierdzając przycisk „Zakończ", użytkownik kończy pracę programu, otrzymując tym samym dostęp do wszystkich niezbędnych plików do obsługi platformy, opisanej w rozdziale czwartym. Rysunek 12 Uczenie maszynowe (GUI 4 - Zakończenie pracy) (Źródło: opracowanie własne) Działanie powyższego programu obrazuje diagram aktywności zawarty na rysunku 13. Rysunek 13 Diagram aktywności (Uczenie maszynowe) (Źródło: opracowanie własne) 4. Implementacja platformy Implementacja platformy opiera się na utworzeniu aplikacji internetowej wraz z odpowiednim połączeniem jej z modelem sztucznej inteligencji. Implementację możemy podzielić w tym przypadku na dwie części. Część frontendową, która opiera się na wdrożeniu wersji graficznej, z której będzie korzystać użytkownik, a także część backendową, która będzie obsługiwać komunikację z serwerem oraz modelem SI. 4.1. Koncepcja wizualna platformy Każda aplikacja weebowa, która będzie posiadać interfejs graficzny (GUI), powinna być uprzednio zaprojektowana przez osobę, która dobrze rozumie zasady korzystania z interfejsów aplikacji internetowych. Doświadczenie w projektowaniu platform przyjaznym użytkownikom pozwala na zdobycie szerszego grona odbiorców poprzez pozytywne odczucia z korzystania z aplikacji. Funkcjonalności wybranego oprogramowania stanowią dobrą podstawę, natomiast design może wzmocnić i uatrakcyjnić aplikację, czyniąc ją bardziej przyjazną i angażującą dla użytkowników[13]. W obecnej dobie Internetu, gdzie panuje przesyt informacji, a użytkownicy są często przebodźcowani, bardzo istotne jest utworzenie odpowiedniej warstwy graficznej. Użytkownicy zazwyczaj oceniają aplikację w ciągu kilku sekund od jej uruchomienia. Atrakcyjna i spójna oprawa graficzna może przyciągnąć uwagę odbiorcy i zachęcić go do dalszej eksploracji. Niezależnie od tego, jak zaawansowane oprogramowanie zostało zaoferowane użytkownikowi, pierwsze wrażenie bazuje na wyglądzie. Intuicyjny interfejs, który jest łatwy w nawigacji, ułatwia korzystanie z aplikacji, pozostawiając pozytywne wrażenia. Wygląd aplikacji nie tylko zwiększa jej zainteresowanie wśród odbiorców ale również pozytywnie wpływa na wzmocnienie marki, poprzez identyfikację marki z wybraną nazwą, kolorystyką czy też utworzonym logotypem[14]. W związku z powyższymi informacjami, omawiana aplikacja również posiada swój projekt wizualny, który został utworzony na początku procesu projektowego w oprogramowaniu figma, a plik projektowy został umieszczony pod nazwą „Projekt koncepcyjny.fig" oraz „Projekt koncepcyjny.pdf" w części praktycznej pracy dyplomowej. Rysunek 14 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 4.2. Mechanizmy komunikacji między frontendem a backendem Większość zaawansowanych aplikacji, korzystających z graficznego interfejsu użytkownika, a zwłaszcza aplikacje webowe, korzystają z warstwy frontendowej oraz backendowej w celu optymalizacji obsługi dużej ilości poleceń lub wykorzystania niedostępnych z poziomu wybranej warstwy narzędzi. 4.2.1. Komunikacja serwer – użytkownik Omawiana aplikacja napisana jest głównie w języku PHP, JavaScript oraz Python co wymaga stałej komunikacji pomiędzy frontendem, a backendem. Kod w języku PHP dzięki stałemu połączeniu z serwerem, może w sposób dynamiczny, kontrolować dostępem do witryny oraz zawartości, która jest wyświetlana na ekranie użytkownika. Poprzez zarządzanie witryną kod PHP umożliwia wczytanie odpowiedniego kodu CSS oraz JavaScript w wybranych zakładkach, dzięki funkcji rozpoznawania adresów URL. Oprócz aspektów wizualnych, PHP obsługuje również wczytywanie wizualizacji dla gestów, poprzez przeszukiwanie katalogów zamieszonych na serwerze w poszukiwaniu nazwy wprowadzonej w warstwie frontendowej oraz przekazanej odpowiednio przez skrypt napisany w języku JavaScript. 4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) Integracja modelu AI z aplikacją internetową jest realizowana poprzez połączenie części backendu,

napisanego w języku programowania Python z częścią frontendową platformy napisaną w języku JavaScript. Wspomniana część backendowa jest zapisana w pliku o nazwie „app.py". Program ten uruchamia model sztucznej inteligencji o nazwie „best_model.h5" oraz listę etykiet „classes.npy", które są wykorzystywane do przetwarzania danych w czasie rzeczywistym. Dane z kamery użytkownika są przetwarzane, a wyodrębnione punkty kluczowe są przekazywane do modelu, który dokonuje predykcji gestów. Wyniki te są następnie udostępniane poprzez endpointy, które zwracają szczegóły dotyczące rozpoznanych gestów w formacie JSON. Frontend aplikacji komunikuje się z backendem za pomocą zapytań http, korzystając z fetch API do wysyłania zapytań do serwera Flask. Za komunikację z serwerową częścią oprogramowania, odpowiedzialne są trzy skrypty napisane w języku JavaScript, a są nimi: „edu.js", „translator.js" oraz „sign_video_loader.js". Każdy z trzech wymienionych skryptów, korzysta z wywołań endpointów poprzez serwer lokalny o porcie 5000 „http://localhost:5000/gesture_details", które służą do pobierania informacji o rozpoznanych gestach i aktualizowania interfejsu użytkownika. W pliku app.py, endpoint dostarcza strumień wideo z kamery internetowej, który następnie jest przetwarzany przez model SI. Frontend inicjuje kamerę oraz przesyła obraz do backendu gdzie obraz jest przetwarzany, a następnie odtwarza przesyłany strumień, umożliwiając tym samym interaktywną analizę gestów w czasie rzeczywistym. 4.3. Widoki aplikacji Omawiana platforma internetowa posiada tzw. widoki, czyli warstwę frontendową, która jest odpowiednio ostylowana poprzez arkusze stylów CSS, starając się jak najdokładniej odwzorować utworzony w początkowych fazach pracy projekt wizualizacji aplikacji. Rysunek 15 Interfejs graficzny aplikacji – zakładka Home (Źródło: opracowanie własne) Platforma składa się z siedmiu widoków. Trzy pierwsze widoki odnoszą się do informacji wstępnych dotyczących realizowanego projektu. Są to zakładki „Home", „O projekcie" oraz „O nas". Każda z zakładek posiada statyczne elementy informacyjne oraz dekoracyjne, które mają zachęcić użytkownika do dalszej interakcji z platformą. Poniżej opisane zostały bardziej interesujące elementy, każdej z zakładek. Rysunek 16 Zakładka Home (Źródło: opracowanie własne) Na stronie Index.php (Home) zostały przedstawione podstawowe informacje dotyczące platformy. Zastosowany został również slider, który wyświetla logotyp Uniwersytetu Rzeszowskiego, przechodząc od prawej strony do lewej, a następnie cofając się do początku. Rysunek 17 Slider - strona Index.php (Źródło: opracowanie własne) Zakładka O projekcie (Project.php) zawiera informacje dotyczące funkcjonowania utworzonej aplikacji. Można znaleźć tam między innymi informacje o zastosowanych rozwiązaniach technologicznych, genezie powstania dysertacji, poruszanej problematyce, skrócony opis działania aplikacji, a także linię czasu, która w sposób skrótowy ukazuje etapy powstawania aplikacji. Rysunek 18 Widok zakładki O projekcie (Project.php) (Źródło: opracowanie własne) Zakładka O nas (About.php) skupia się na opisie osób zaangażowanych w powstanie projektu. W związku ze stałym rozwojem tworzonego oprogramowania oraz wysoką czasochłonnością prac, a także innymi losowymi aspektami, w projekt były zaangażowane 4 osoby. Oprócz autora oprogramowania oraz niniejszej dysertacji, w projekt zaangażował się również Dr. Inż. Wojciech Kozioł, który był pomysłodawcą tematu omawianej pracy, Dr. Inż. Bogusław Twaróg, który prowadził nadzór merytoryczny nad powstającą pracą oraz inż. Patryk Arendt, który służył jako model motion capture dla 5 gestów (tj. 500 próbek nagrań wideo). Rysunek 19 Widok zakładki O nas (About.php) (Źródło: opracowanie własne) W ramach uhonorowania wkładu, jaki włożyły ww. osoby w powstanie pracy dyplomowej, zostały one wymienione w wersji aplikacyjnej w zakładce O nas. Rysunek 20 Uhonorowanie wsparcia (About.php) (Źródło: opracowanie własne) Kliknięcie przycisku „Przejdź do aplikacji" powoduje wczytanie ostatniego statycznego widoku jakim jest „Dashboard.php". W widoku tym opisane są podstawowe informacje dotyczące ilości gestów, ilości wykorzystanych próbek, sumarycznego czasu nagrań, a także wiele innych. Rysunek 21 Widok zakładki Dashboard (Dashboard.php) (Źródło: opracowanie własne) W zakładce tej przedstawiona została również animacja, która w trzech etapach prezentuje w jaki sposób program rozpoznaje sekwencje ruchów ludzkiego ciała, na przykładzie gestu z etykietą „Dzięki". Rysunek 22 Działanie systemu rozpoznawania gestu (Źródło: opracowanie własne) Pozostałe trzy widoki, odpowiadają pośrednio lub bezpośrednio za komunikację z systemem rozpoznawania gestów. Działają one na podobnej zasadzie, lecz skupiają się na innym rodzaju wykonywanego zadania przez użytkownika. Rysunek 23 Widok zakładki Tłumacz (Translator.php) (Źródło: opracowanie własne) Zakładka Tłumacz, umożliwia użytkownikowi tłumaczenie słów w języku migowym w obie strony. Może on skorzystać z funkcji udostępnienia obrazu z kamery co pozwoli na tłumaczenie języka migowego na język naturalny lub wyszukać gest w języku migowym, dostępny w systemie poprzez polecenie pisemne lub komendę głosową. W celu wyszukanie gestu języka migowego, wystarczy, że użytkownik uzupełni pole tekstowe odpowiednią nazwą gestu i zatwierdzi zmianę klikając przycisk po prawej stronie lub wciskając klawisz „Enter". Rysunek 24 Wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) Istnieje również alternatywny sposób

wyszukiwania gestu w systemie. Użytkownik może skorzystać z przycisku, znajdującego się po lewej stronie okna, aby rozpocząć nasłuchiwanie mikrofonu. Po wypowiedzeniu frazy np. „Cześć", możemy użyć polecenia głosowego „... stop szukaj", co automatycznie zakończy nasłuchiwanie mikrofonu i rozpocznie proces szukania gestu o etykiecie „Cześć". Nasłuchiwanie mikrofonu można także wyłączyć w sposób ręczny, klikając ponownie na ikonę mikrofonu. Wyszukiwanie głosowe, oparte zostało na polskiej wersji językowej Web Speech API dla języka JavaScript. Po aktywacji funkcji nasłuchiwania program sprawdza dostępność API w przeglądarce i konfiguruje obiekt rozpoznawania mowy. Podczas nasłuchiwania mikrofonu, program przetwarza rozpoznane fragmenty, a następnie wstawia je do pola tekstowego. Jeżeli program wykryje określoną komendę tj. „stop szukaj", przerywa nasłuchiwanie i uruchamia proces wyszukiwania nagrania. Program obsługuje najczęściej pojawiające się błędy, a także automatycznie restartuje nasłuchiwanie, jeżeli nie zostało ono przerwane ręcznie. Rysunek 25 Głosowe wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) Gdy gest nie zostanie odnaleziony, na ekranie pojawia się stosowny komunikat. Gdy gest zostanie wyszukany poprawnie, oczom użytkownika ukaże się wizualizacja gestu odtwarzana w postaci nagrania. Przykład takowej wizualizacji został ukazany na grafice poniżej. Rysunek 26 Wizualizacja gestu (Źródło: opracowanie własne) W drugim przypadku użycia, użytkownik może udostępnić obraz z kamery, a następnie wykonać wybrany przez siebie gest. Na wykrytą postać zostanie naniesiona siatka punktów, która będzie obrazować przetwarzane dane. Program w czasie rzeczywistym, w lewym górnym rogu ekranu wyświetlać będzie prawdopodobne gesty pokazane przez użytkownika. Gdy gest zostanie wykryty wystarczającą liczbę razy, oczom użytkownika ukaże się również przycisk „Wyświetl wizualizację pokazywanych gestów". Rysunek 27 Prezentacja gestu "Dzięki" (Źródło: opracowanie własne) Gdy użytkownik kliknie na komunikat „Wyświetl wizualizację pokazywanych gestów" po prawej stronie, automatycznie w sposób płynny wczyta się nagranie z wizualizacją gestu. Wizualizacja ta będzie się zmieniać za każdym razem, gdy program wykryje pokazanie innego gestu z odpowiednią pewnością. Rysunek 28 Synchronizacja wizualizacji z obrazem w czasie rzeczywistym (Źródło: opracowanie własne) Po przejściu na inną zakładkę, w przeglądarce internetowej klienta, śledzenie kamery zostaje zatrzymane, zarówno przez warstwę forntendową jak i backendową, a algorytmy wstrzymują swoją pracę oczekując na wznowienie połączenia. Poniżej przedstawiono diagram przypadków użycia dla zakładki Tłumacz: Rysunek 29 Diagram przypadków użycia (Tłumacz) (Źródło: opracowanie własne) Dla powyższego widoku został również przygotowany diagram aktywności, który pozwala na jeszcze bardziej precyzyjne zapoznanie się z architekturą działania oprogramowania. Poniższa grafika obrazuje proces przetwarzania obrazu z kamery użytkownika, wraz z obsługą wizualizacji, która jest aktualizowana w czasie rzeczywistym. Rysunek 30 Diagram aktywności (Tłumacz - kamera / wizualizacja) (Źródło: opracowanie własne) Zobrazowany został również diagram aktywności, odpowiedzialny za funkcję wyszukiwania wizualizacji za pomocą poleceń tekstowych oraz głosowych. Diagram UML ww. procesu znajduje się poniżej. Rysunek 31 Diagram aktywności (Tłumacz – Wizualizacja) (Źródło: opracowanie własne) Ostatnie dwa widoki aplikacji omawiają działanie zakładki „Nauka języka migowego". Oprogramowanie, dzięki zaawansowanym algorytmom wspiera proces uczenia języka migowego. W tym celu stworzone zostały kategorie oraz „kafelki", które umożliwiają wybór gestu, którego chce się nauczyć użytkownik. Funkcja ta skierowana jest dla osób początkujących, które dopiero rozpoczynają swoją przygodę z PJM. Rysunek 32 Widok zakładki Nauka Języka Migowego (Edu.php) (Źródło: opracowanie własne) Na stronie podstronie edu.pl możemy wybrać kategorię, która nas interesuje, a następnie odpowiedni dla nas gest. Po wybraniu odpowiedniej kategorii użytkownik będzie mógł zaobserwować zmianę w wyświetlanych elementach. Zamiast kategorii, na ekranie pojawią się gesty, które są z nią powiązane. Rysunek 33 Gesty wybranej kategorii (Źródło: opracowanie własne) Po wybraniu jednego z dostępnych gestów pojawia się nowy widok (Edu-progress.php), który wizualnie jest zbliżony do widoku tłumacza. Tutaj również możemy uruchomić podgląd z kamery, natomiast po prawej stronie odtwarzane jest w zapętleniu nagranie z wybranym przez nas gestem. W widoku tym, niedostępny jest input do wyszukiwania gestów, czy też zmiany wizualizacji. Po uruchomieniu kamery i poprawnym wykonaniu gestu, program zatrzyma udostępnianie obrazu z kamery, a użytkownik otrzyma odpowiedni komunikat na ekranie. Rysunek 34 Wykonanie poprawnego gestu (Źródło: opracowanie własne) 5. Ocena działania platformy Utworzona platforma została przetestowana pod względem działania modelu sztucznej inteligencji oraz kilku powiązanych z nią czynników. Przebadana została precyzja modeli CNN, skonstruowanych z różnych ilości próbek, w konfiguracji 100 próbek na 1 gest, 200 próbek na 1 gest oraz 300 próbek na 1 gest. Wyszczególnione zostały etapy przetwarzania danych, uczenia maszynowego oraz ogólnej pracy modelu w czasie rzeczywistym, a także na bazie przygotowanych

wcześniej nagrań. Następnie, wyniki te zostały zestawione z utworzonym hybrydowym modelem CNN + LSTM. 5.1. Testy w rzeczywistych warunkach Testy związane z wydajnością w rzeczywistych warunkach, a co za tym idzie w czasie rzeczywistym są stosunkowo ciężkie do realizacji. Wpływ ma na to rozbieżność pomiędzy wykonywanymi gestami, które ciężko odtworzyć w sposób identyczny, zróżnicowane oświetlenie, ułożenie ciała oraz wykorzystywany hardware. Istnieje wiele zmiennych, które mogą zakłócić obiektywną ocenę modelu, zwłaszcza jeżeli ten został przygotowany na niewielkiej ilości próbek. W omawianym punkcie, przedstawione zostaną zarówno obiektywne informacje, podparte dowodami, jak i subiektywne odczucia autora dysertacji. 5.1.1. Ekstrakcja punktów kluczowych Pierwszym elementem, który zostanie poddany analizie jest proces przetwarzania informacji, potrzebnych do utworzenia modelu. Cały proces uczenia maszynowego czyli ekstrakcja punktów kluczowych oraz nauka modelu odbywała się na maszynie obliczeniowej z zamontowanym procesorem AMD EPYC 7702 64-Core, który posiadał 128 procesorów wirtualnych o szybkości 2GHz, pamięcią ram 256GB oraz dysku HDD. Przetwarzanie danych w modelu CNN, zastosowane na 6 gestach po 100 próbek każdy, co daje wynik 600 próbek, trwało 19 minut w zaokrągleniu czasu do pełnych minut. Czas trwania tego samego procesu na 1200 próbkach trwało 35 minut, natomiast finalny model omawiany w dysertacji, potrzebował 54 minut na przetworzenie wszystkich danych (1800 próbek). Po zestawieniu ww. informacji w postaci wykresu, możemy zaobserwować jak wygląda złożoność czasowa przetwarzania danych. Przetwarzanie próbek w stosunku do czasu 60 54 50) n im (a 40 35 in a z r a 30 w t e 19 z r p 20 s a z C 10 0 600 próbek 1200 próbek 1800 próbek Czas w minutach 19 35 54 Ilość próbek / Czas Wykres 1 Przetwarzanie próbek CNN Uśredniając czas przetwarzania każdej próbki wynosi 1,81 sekundy. Rozbijając na poszczególne partie: przy 600 próbkach, program potrzebował 1,9 sekundy na przetworzenie jednego nagrania, 1,75 sekundy dla 1200 próbek oraz 1,8 sekundy w przypadku 1800 próbek. Zestawiając dane przetwarzane przez program wykorzystujący CNN oraz CNN w połączeniu z LSTM, nie widać jednoznacznych różnic. Program w podobnym czasie przetworzył dane zarówno dla modelu opartym o CNN jak i modelu hybrydowym, przy czym mniej złożony model zakończył proces ponad 2 minuty wcześniej. Rysunek 35 Zestawienie metody CNN oraz CNN + LSTM (Źródło: opracowanie własne) Czas przetwarzania danych (1800 próbek) CNN+LSTM 56 u le d o m ja z d o R CNN 54 53 54 54 55 55 56 56 57 Czas przetwarzania (min) Wykres 2 Czas przetwarzania danych – porównanie CNN – CNN + LSTM 5.1.2. Uczenie modelu Uczenie modelu również zostało porównane w identycznych kryteriach. Z otrzymanych informacji wynika, że czas uczenia maszynowego nie jest stricte związany z ilością próbek. Funkcja przerywania uczenia, która za zadanie ma chronić model przed przeuczeniem, skutecznie skraca czas potrzebny do stworzenia modelu. Program kończy pracę za każdym razem gdy wykryje, że precyzja modelu nie poprawia się co skutkuje utworzeniem modelu w krótszym czasie niżeli z potencjalnie mniejszą ilością próbek. Uczenie maszynowe w stosunku do czasu 140 124 120)s (u 100 le d 76 o m 80 a in 60 e z c u s 40 30 a z C 20 0 600 próbek 1200 próbek 1800 próbek Czas w sekundach 30 124 76 Ilość próbek / Czas Wykres 3 Uczenie modelu CNN O ile nie było widocznej różnicy pomiędzy przetwarzaniem danych pomiędzy modelem wykorzystującym CNN, a modelem hybrydowym CNN + LSTM, o tyle w przypadku uczenia modelu różnica jest bardzo zauważalna. Model z mniejszą ilością warstw uczył się przez 1 minutę i 16 sekund, natomiast model z przetwarzaniem danych wstecz, trenował się przez 6 godzin, 9 minut i 36 sekund. Różnica w czasie nauczania modelu jest ogromna, a co za tym idzie model hybrydowy z zastosowaniem LSTM wydaje się mało wydajnym rozwiązaniem, zważywszy na małą ilość przetwarzanych danych. Program oparty na mniejszej ilości warstw uczył się ze stosunkowo wysoką precyzją, która oscylowała w zakresie 91-95%, oraz 81-88% podczas walidacji, co obrazuje poniższa ilustracja. Rysunek 36 Proces uczenia maszynowego CNN (Źródło: opracowanie własne) Druga wersja programu zatytułowana roboczo 2.1, wykazywała wyższą precyzję uczenia podczas kilku prób uczenia modelu. Dokładność modelu oscylowała pomiędzy 97-99%, a podczas walidacji wynik ten wynosił 93-99%. Wyniki te sugerują, że model utworzony za pomocą CNN oraz LSTM radzi sobie znacznie lepiej niżeli poprzednia wersja. Należy zwrócić uwagę również na wielkość pliku docelowego. Model utworzony w wersji 1.0 zajmuje 25,8 MB miejsca na dysku, natomiast model w wersji 2.1 zajmuje 1,26 GB. Kwestia ta staje się problematyczna, gdyż model reprezentujący niewiele niższą precyzję podczas uczenia 5 maszynowego, zajmuje 50 razy mniej miejsca na dysku, co staje się kuszącym rozwiązaniem w przypadku chęci zaoszczędzenia miejsca w przestrzeni dyskowej. Rysunek 37 Proces uczenia maszynowego CNN + LSTM (Źródło: opracowanie własne) Czas przetwarzania danych (1800 próbek) CNN+LSTM 22176 u le d o m ja z d o R CNN 76 0 5000 10000 15000 20000 25000 Czas przetwarzania (s) Wykres 4 Czas uczenia modelu – porównanie CNN – CNN + LSTM 5.1.3. Testy w czasie rzeczywistym Model z założenia miał zostać przeznaczony do platformy obsługującej rozpoznawanie polskiego języka migowego w czasie rzeczywistym. W związku z powyższą informacją, również ten aspekt został

przetestowany. Niestety w związku z wieloma czynnikami losowymi, nie da się zmierzyć precyzyjnie różnicy pomiędzy rozpoznawaniem gestów w czasie rzeczywistym pomiędzy modelem utworzonym w wersji podstawowej oraz hybrydowej. 5 Jednakże dokonano pewnych obserwacji, które sugerują wykorzystanie jednego spośród dwóch modeli. W przypadku modelu podstawowego, nie zaobserwowano problemów. Model działał stabilnie, precyzja wykrywanych gestów była zadowalająca. W przypadku modelu hybrydowego, uruchomionego w warunkach czasu rzeczywistego zaobserwowano problemy. Model działał niestabilnie, rejestrując ciągłe „przycięcia" obrazu. Zachodziło tam zjawisko klatkowania, a dokładna przyczyna została opisana w punkcie 5.3. 5.2. Skuteczność i dokładność rozpoznawania gestów Każdy z utworzonych modeli został również przetestowany w warunkach odpowiednio wcześniej przygotowanych. Specjalnie zaprojektowany program, odtwarzał trzy nagrania których nie było zamieszonych w próbkach modelu, dla każdego spośród 6 istniejących gestów. Następnie za pomocą dostarczonych modeli klasyfikował wykryte gesty. Ocenie została poddana precyzja rozpoznanych gestów z podziałem procentowym. Spośród testowanych gestów jeden z nich otrzymał najgorszy wynik, rozpoznając tym samym inny gest. Pozostałe tj. 5/6 gestów zostało rozpoznanych poprawnie. Zestawienie wykresów wraz z uśrednieniem wyników, zostało zaprezentowane na wykresie 5. 5 Wykres 5 Zestawienie wykresów Uśredniając wyniki spośród trzech testów, gest „Cześć" został rozpoznany z precyzją 60,70%, program wykrył również możliwość zaistnienia gestu „Dzięki" z wynikiem 38,69% oraz „Prosić" (0,58%). Program podczas analizy nagrań wykrył 2 z 3 wskazanych gestów poprawnie. Podsumowanie analizy gestu „Cześć" zostało przedstawione na poniższym wykresie. 5 Wykres 6 Analiza gestu Cześć - CNN Porównując wyniki z modelem CNN + LSTM, lepiej prezentuje się model CNN. Model składający się z dodatkowej warstwy, błędnie wykrył gest dzięki z precyzją 60,33%, oraz cześć z precyzją 39,67%. 5 W przypadku gestu „Dzięki" model nieprecyzyjnie rozpoznał wskazywany gest. Jedynie na drugiej próbce gest dzięki miał wyższe prawdopodobieństwo wystąpienia niżeli inne opcje. Program błędnie rozpoznał, że wskazanym modelem jest gest Proszę z prawdopodobieństwem 66,13%, drugim możliwym gestem było dzięki z wynikiem 33,44%. Dodatkowo program dostrzegł podobieństwo z gestem prosić (0,24%), Cześć (0,17%) oraz Przepraszać (0,02%). Wykres 7 Analiza gestu Dzięki – CNN W powyższym przypadku lepszy okazał się model z dodatkową warstwą, prezentując następujące wyniki: dzięki (67,62%), proszę (32,11%), cześć (0,26%). 5 W przypadku gestu „Dziękuję" nie było minimalnych wątpliwości. Gest ten został wykryty z precyzją aż 99,98%. Pozostałe 0.02% zostało podzielone po równo pomiędzy gestem „dzięki" i „Cześć". Wykres 8 Analiza gestu Dziękuję - CNN W powyższym przypadku, znacznie lepsza okazała się sieć CNN. Druga sieć wykryła gest: dziękuję (65,22%), dzięki (32,16%), przepraszać (2,38%), cześć (0,18%), prosić (0,05%), proszę (0,01%). 5 Czwarty badany gest (Prosić) został rozpoznany poprawnie z precyzją 75,69%, pozostałe wartości zostały podzielone pomiędzy: cześć (22,34%), proszę (1,88%), dziękuję (0,06%) oraz dzięki (0,03%) Wykres 9 Analiza gestu Prosić – CNN Gest prosić został wykryty poprawnie w każdej konfiguracji sieci, lecz sieć LSTM wykryła gest z większą precyzją, zwracając wynik precyzji: prosić (99,74%), dziękuję (0,24%), dzięki (0,02%), przepraszać (0,01%). 5 Piąty gest, tak jak gest trzeci, został rozpoznany z bardzo wysoką precyzją (99,91%). Pozostałe prawdopodobieństwo zostało podzielone jedynie przez dzięki (0,07%) oraz prosić (0,02%). Wykres 10 Analiza gestu Proszę – CNN W powyższym przypadku sieć CNN okazała się lepsza niżeli LSTM. Druga z wymienionych sieci wykryła gesty: proszę (77,73%), dzięki (21,90%), cześć (0,35%), prosić oraz przepraszać (0,01%). 5 Ostatnim badanym gestem jest gest przepraszać. Gest ten został sumarycznie wykryty z precyzją 49%, możliwe było również wykrycie etykiety dzięki, która miała również wysoki wynik (43,71%). Analizując wyniki zauważyć można także proszę (7,07) oraz cześć (0,22%). Wykres 11 Analiza gestu Przepraszać – CNN Model LSTM wykrył odpowiednio gesty: przepraszać (44,11%), dzięki (38,40%), proszę (17,01%), cześć (0,23%), dziękuję (0,14%) oraz prosić (0,10%). Ogólny wynik precyzji przetwarzanych gestów wskazuje na to, iż model CNN jest bardziej precyzyjny (sumarycznie 417,72%) niżeli model hybrydowy CNN + LSTM (sumarycznie 394,09%). 60 5.3. Analiza błędów i propozycje ulepszeń Platforma, a co za tym idzie również model SI posiada nie tylko swoje zalety ale również wady. W pierwszej kolejności należy wymienić warstwę backendową. Na przykładzie omawianej dysertacji udowodniono, że model hybrydowy w postaci CNN + LSTM, niekoniecznie działa poprawnie z przetwarzaniem obrazów w czasie rzeczywistym. LSTM jest przeznaczony głównie do pracy z sekwencjami danych, które mają wyraźny związek czasowy, takimi jak teksty lub sygnały czasowe. W przypadku obrazu, dane są przestrzenne, a nie czasowe. Do przetwarzania danych obrazowych lepiej nadają się sieci konwolucyjne (CNN), które są zaprojektowane specjalnie do pracy z danymi w formie siatek (takich jak obrazy). Zwiększenie złożoności, w tym przypadku wpłynęło negatywnie na działanie programu. Model wykorzystujący LSTM potrzebował

znacznie więcej czasu na przetworzenie sekwencji, co skutkowało zacinaniem się obrazu z kamery oraz powolną predykcją gestów. Precyzja w przetwarzaniu statycznym, również nie została podniesiona. Zaobserwować można było spadek precyzji modelu na danych testowych. W związku z dokonaną analizą, nie zaleca się wykorzystywania LSTM do przetwarzania obrazu w czasie rzeczywistym. Model CNN, można jednakże rozwinąć o podejście hybrydowe CNN + HMM, które polecają autorzy publikacji „Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition.”. Zastosowanie powyższego rozwiązania potencjalnie mogłoby podnieść poziom precyzji modelu, bez negatywnego wpływu na stabilność programu. Możliwe jest także dodanie Transformatorów takich jak ST- Transformer, które są aktualnym rozwiązaniem na rok 2024. Dodatkowo, model mógłby być wzbogacony o większą ilość próbek, wliczając w to zróżnicowanych modeli motion capture, różne warunki oświetlenia, różnego typu kamery oraz zmienne tła otoczenia. Powyższe elementy z pewnością poprawiłyby precyzję uczenia modelu. Utworzona platforma, mogłaby również zostać ulepszona poprzez dodanie elementów frontendowych, takich jak responsywność, a także możliwości obsługi wielu języków. Posiada ona również przygotowany moduł, który można rozwinąć o logowanie oraz rejestrację, wprowadzając tym samym system abonamentów (tokenów), znany chociażby z konkurencyjnych aplikacji SI. 61 Podsumowanie Platformy edukacyjne stale cieszą się popularnością. W grupie ww. platform znajdują się aplikacje do nauki języków obcych z których każdego dnia korzystają setki tysięcy osób, a ich stały rozwój pozwala na zwiększenie komunikatywności w środowisku międzynarodowym. Sytuacja ta jest analogiczna w przypadku zwiększenia świadomości ludzkiej na wszelkiego typu dysfunkcje organizmu. Strony internetowe coraz częściej dostosowują się do różnego rodzaju dyrektyw, chociażby takiej jak WCAG 2.0, które to w jasny sposób określają jak powinny wyglądać strony przyjazne dla osób z niepełnosprawnościami. Pomimo zwiększenia dostępności stron internetowych dla osób z dysfunkcjami, nadal istnieje problem wykluczenia społecznego w warunkach kontaktu bezpośredniego. Osoby, które są głuchonieme, potrzebują obecności tłumacza, który stale przekazuje informacje pomiędzy osobami pełnosprawnymi, a osobą dotknięta dysfunkcją. Niniejsza dysertacja łączy możliwości platform internetowych oraz tematyki niepełnosprawności, tworząc prototyp platformy do rozpoznawania polskiego języka migowego. Pomimo istnienia takowych aplikacji na rynku anglojęzycznym, nadal nie istnieje stabilne rozwiązanie na polskim rynku. W związku z powyższą informacją, praca ta jest innowacyjna, łącząc ze sobą prostotę obsługi, z zaawansowanymi technologiami takimi jak model sztucznej inteligencji rozpoznający gesty języka migowego w czasie rzeczywistym. W ramach niniejszej pracy dyplomowej, opracowany został projekt platformy internetowej, wraz z jej pełną implementacją przy zastosowaniu języków webowych. Dodatkowo na potrzeby dysertacji, autor utworzył kilka modeli sztucznej inteligencji, składających się z różnej ilości próbek opracowanych przez twórcę. Następnie każdy model został przetestowany pod względem wydajność oraz możliwości wykorzystania ich w aplikacji korzystającej z kamery internetowej w czasie rzeczywistym. Finalnie, utworzonych zostało 1800 nagrań o długości 4 sekund, przedstawiających osobę, wykonującą określony gest polskiego języka migowego. Najlepszym modelem okazał się program oparty na architekturze CNN, który wykazał się wysoką precyzją oraz szybkością działania. Należy również uwzględnić, że wszystkie elementy estetyczne w tym grafiki, wizualizacje oraz wykresy zostały opracowane przez autora pracy. Praca dyplomowa, może również zostać rozwinięta o dodatkowe elementy, takie jak responsywność oraz moduł logowania i rejestracji, pozwalający na wykorzystanie ww. dysertacji w celach materialnych. Sugerowane jest również wzbogacenie opracowanego 62 modelu o większą ilość próbek, opracowanych w różnych warunkach oświetleniowych, zmiennym otoczeniu, a także na innych osobach. Przebudowa modelu, dodając warstwę architektury HMM lub ST-Transformer również może zwiększyć precyzję wykrywanych gestów, a co za tym idzie poprawić działanie całej platformy edukacyjno-translatorycznej. 63 Bibliografia oraz Netografia 1. Camgoz, N. C., Koller, O., Hadfield, S., & Bowden, R. (2017). SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition. IEEE International Conference on Computer Vision (ICCV). 2. Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 7291-7299. 3. Huang, J., Zhou, W., Zhang, Q., Li, H., & Li, W. (2018). Attention-Based 3D-CNNs for Large-Vocabulary Sign Language Recognition. IEEE Transactions on Circuits and Systems for Video Technology, 29(9), 2822-2832. 4. Jamroży D.(2023): Aplikacja internetowa z graficznym interfejsem użytkownika opartym na technologii typu eye-tracking oraz speech recognition. [Praca inżynierska, Uniwersytet Rzeszowski] 5. Koller, O., Zargaran, S., Ney, H., & Bowden, R. (2015). Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition. In Proceedings of the British Machine Vision Conference (BMVC). 6. --- MediaPipe

Documentation (n.d.). Google Developers. Retrieved from https://ai.google.dev/edge/mediapipe/solutions/guide?hl=pl 7. Pandey, A., Mishra, M., & Verma, A. K. (2022). Real-Time Sign Language Recognition Using Machine Learning and Neural Networks. IEEE Access. 8. Patel, K., Gil-González, A.-B., & Corchado, J. M. (2022). Deepsign: Sign Language Detection and Recognition Using Deep Learning. Electronics, 11(11), 1780. 9. Patel, M., & Shah, N. (2021). Real-Time Gesture-Based Sign Language Recognition System. IEEE International Conference on Information Technology and Engineering (ICITE). 10. Pigou, L., Dieleman, S., Kindermans, P. J., & Schrauwen, B. (2015). Sign Language Recognition Using Convolutional Neural Networks. European Conference on Computer Vision Workshops (ECCVW). 11. Zhang, Z., & Liu, C. (2020). Skeleton-Based Sign Language Recognition Using Whole-Hand Features. IEEE Access, 8, 68827-68837. 12. https://www.wsb-nlu.edu.pl/pl/wpisy/wplyw-sztucznej-inteligencji-na-przyszlosc-pracy- nowe-perspektywy-i-wyzwania (22.08.2024) 64 13. https://www.tamoco.com/blog/blog-app-design-app-functionality-ux-ui/ (22.08.2024) 14. https://echoinnovateit.com/ux-or-ui-whats-more-important-in-app-development/ (22.08.2024) 15. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5(4), 115-133. 65 Spis ilustracji, tabel oraz wykresów Spis Ilustracji Rysunek 1 Porównanie technologii (Źródło: opracowanie własne) 11 Rysunek 2 Interfejs programu XAMPP 20 Rysunek 3 Etapy zbierania próbek (Źródło: opracowanie własne) 22 Rysunek 4 Uczenie maszynowe (GUI 1) (Źródło: opracowanie własne) 22 Rysunek 5 Uczenie maszynowe (GUI 2) (Źródło: opracowanie własne) 23 Rysunek 6 Uczenie maszynowe (GUI 2 - Struktura) (Źródło: opracowanie własne) 24 Rysunek 7 Uczenie maszynowe (GUI 3) (Źródło: opracowanie własne) 24 Rysunek 8 Uczenie maszynowe (GUI 3 - Wizualizacja) (Źródło: opracowanie własne) 25 Rysunek 9 Uczenie maszynowe (GUI 3 - Zakończenie przetwarzania danych) (Źródło: opracowanie własne) 26 Rysunek 10 Wzór algorytmu gradientu prostego 28 Rysunek 11 Uczenie maszynowe (GUI 4) (Źródło: opracowanie własne) 29 Rysunek 12 Uczenie maszynowe (GUI 4 - Zakończenie pracy) (Źródło: opracowanie własne) .. 30 Rysunek 13 Diagram aktywności (Uczenie maszynowe) (Źródło: opracowanie własne) 31 Rysunek 14 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 33 Rysunek 15 Interfejs graficzny aplikacji – zakładka Home (Źródło: opracowanie własne) 35 Rysunek 16 Zakładka Home (Źródło: opracowanie własne) 35 Rysunek 17 Slider - strona Index.php (Źródło: opracowanie własne) 36 Rysunek 18 Widok zakładki O projekcie (Project.php) (Źródło: opracowanie własne) 36 Rysunek 19 Widok zakładki O nas (About.php) (Źródło: opracowanie własne) 37 Rysunek 20 Uhonorowanie wsparcia (About.php) (Źródło: opracowanie własne) 38 Rysunek 21 Widok zakładki Dashboard (Dashboard.php) (Źródło: opracowanie własne) 38 Rysunek 22 Działanie systemu rozpoznawania gestu (Źródło: opracowanie własne) 39 Rysunek 23 Widok zakładki Tłumacz (Translator.php) (Źródło: opracowanie własne) 39 Rysunek 24 Wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) 40 Rysunek 25 Głosowe wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) 40 Rysunek 26 Wizualizacja gestu (Źródło: opracowanie własne) 41 Rysunek 27 Prezentacja gestu "Dzięki" (Źródło: opracowanie własne) 42 66 Rysunek 28 Synchronizacja wizualizacji z obrazem w czasie rzeczywistym (Źródło: opracowanie własne) 42 Rysunek 29 Diagram przypadków użycia (Tłumacz) (Źródło: opracowanie własne) 43 Rysunek 30 Diagram aktywności (Tłumacz - kamera / wizualizacja) (Źródło: opracowanie własne) 44 Rysunek 31 Diagram aktywności (Tłumacz – Wizualizacja) (Źródło: opracowanie własne) 45 Rysunek 32 Widok zakładki Nauka Języka Migowego (Edu.php) (Źródło: opracowanie własne) 46 Rysunek 33 Gesty wybranej kategorii (Źródło: opracowanie własne) 46 Rysunek 34 Wykonanie poprawnego gestu (Źródło: opracowanie własne) 47 Rysunek 35 Zestawienie metody CNN oraz CNN + LSTM (Źródło: opracowanie własne) 49 Rysunek 36 Proces uczenia maszynowego CNN (Źródło: opracowanie własne) 51 Rysunek 37 Proces uczenia maszynowego CNN + LSTM (Źródło: opracowanie własne) 52 Spis Tabel Tabela 1 Wymagania systemowe cz1 18 Tabela 2 Wymagania systemowe cz2 18 Spis Wykresów Wykres 1 Przetwarzanie próbek CNN 49 Wykres 2 Czas przetwarzania danych – porównanie CNN – CNN + LSTM 50 Wykres 3 Uczenie modelu CNN 50 Wykres 4 Czas uczenia modelu – porównanie CNN – CNN + LSTM 52 Wykres 5 Zestawienie wykresów 54 Wykres 6 Analiza gestu Cześć - CNN 55 Wykres 7 Analiza gestu Dzięki – CNN 56 Wykres 8 Analiza gestu Dziękuję - CNN 57 Wykres 9 Analiza gestu Prosić – CNN 58 Wykres 10 Analiza gestu Proszę – CNN 59 Wykres 11 Analiza gestu Przepraszać – CNN 60 67 Streszczenie: Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego. Niniejsza dysertacja dotyczy utworzenia prototypu platformy do rozpoznawania polskiego języka migowego (PJM) w czasie rzeczywistym poprzez zastosowanie metod uczenia maszynowego, których efektem jest model sztucznej inteligencji (SI). Głównym celem pracy było opracowanie platformy, która umożliwi osobom niesłyszącym łatwiejszą komunikację z otoczeniem bez potrzeby korzystania z dodatkowego sprzętu czy też usług tłumacza, a także zwiększenie ludzkiej świadomości na temat

dysfunkcji organizmu. W pracy dokonano również przeglądu technologii i metod związanych z rozpoznawaniem gestów, takich jak konwolucyjne sieci neuronowe (CNN) oraz narzędzia przetwarzania obrazu, między innymi MediaPipe i OpenPose. Autor dysertacji, zaprojektował oraz przeanalizował modele SI, które rozpoznawały gesty PJM na podstawie danych wideo, przetwarzanych oraz trenowanych w modelu CNN. W dalszej części prac, opracowano aplikację webową, która korzysta z kamery internetowej w celu rozpoznania gestów użytkownika w czasie rzeczywistym. Przeprowadzone testy, zarówno w czasie rzeczywistym jak i na przygotowanych wcześniej danych wykazały, że aplikacja działa skutecznie, choć istnieją pewne wyzwania związane z precyzją rozpoznawania gestów, wliczając w to ograniczony dostęp do bazy gestów. Utworzona platforma oferuje potencjalne rozwiązania, które mogą wspierać osoby z dysfunkcjami słuchu oraz mowy w codziennej komunikacji. 68 Abstract: Prototype and Analysis of a Real-time Polish Sign Language Gesture Recognition Platform Using Machine Learning Method. This dissertation concerns the creation of a prototype of a platform for real-time recognition of Polish Sign Language (PJM) through the use of machine learning methods, which result in an artificial intelligence (AI) model. The main objective of the work was to develop a platform that would enable deaf people to communicate more easily with their surroundings without the need for additional equipment or interpreter services, as well as to increase human awareness of body dysfunctions. The paper also reviews technologies and methods related to gesture recognition, such as convolutional neural networks (CNNs) and image processing tools, including MediaPipe and OpenPose. Author of the dissertation, he designed and analyzed AI models that recognized PJM gestures based on video data processed and trained in a CNN model. Further on, a web application was developed that uses a webcam to recognize user gestures in real time. Tests, both in real time and on pre-prepared data, have shown that the application works effectively, although there are some challenges related to the precision of gesture recognition, including limited access to the gesture database. The created platform offers potential solutions that can support people with hearing and speech impairments in everyday communication. 69 70

Znaki specjalne lub spoza języka pracy

UNIWERSYTET RZESZOWSKI Kolegium Nauk Przyrodniczych Damian Jamroży Nr albumu: 113729 Kierunek: Informatyka Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego Praca magisterska Praca wykonana pod kierunkiem Dr. Inż. Bogusława Twaroga Rzeszów, 2024 Pragnę serdecznie podziękować Panu dr inż. Bogusławowi Twarogowi za nieocenione wsparcie oraz życzliwość okazywaną na każdym etapie mojej edukacji. Jego pomoc i zaangażowanie miały kluczowy wpływ nie tylko na proces powstawania niniejszej pracy magisterskiej, ale także na moją pracę inżynierską i cały tok studiów. Bez Jego merytorycznej wiedzy oraz wsparcia, ukończenie tych etapów byłoby znacznie trudniejsze. Jednocześnie chciałbym serdecznie podziękować moim przyjaciołom z UCI UR. To dzięki ich namowom i wsparciu udało mi się zrealizować ww. etap życia. Spis treści Wstęp 7 Cel i zakres pracy 8 1. Przegląd literatury i analiza istniejących rozwiązań 9 1.1. Przegląd technologii rozpoznawania gestów 9 1.2. Przegląd metod uczenia maszynowego 12 1.3. Wyzwania w rozpoznawaniu gestów języka migowego 13 1.4. Analiza problematyki oraz istniejących aplikacji dotyczących języka migowego 15 2. Techniczne aspekty funkcjonowania aplikacji 17 2.1. Wymagania systemowe 17 2.2. Narzędzia modelowania sztucznej inteligencji 19 3. Budowa modelu rozpoznawania gestów 21 3.1. Przygotowanie danych wejściowych 21 3.2. Preprocessing danych 24 3.3. Architektura modelu oraz algorytm klasyfikacji 26 3.4. Trening, walidacja oraz ocena skuteczności modelu 27 4. Implementacja platformy 32 4.1. Koncepcja wizualna platformy 32 4.2. Mechanizmy komunikacji między frontendem a backendem 33 4.2.1. Komunikacja serwer – użytkownik 33 4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) 34 4.3. Widoki aplikacji 34 5. Ocena działania platformy 48 5.1. Testy w rzeczywistych warunkach 48 5.1.1. Ekstrakcja punktów kluczowych 48 5.1.2. Uczenie modelu 50 .1.3. Testy w czasie rzeczywistym 52 5.2. Skuteczność i dokładność rozpoznawania gestów 53 5.3. Analiza błędów i propozycje ulepszeń 61 Podsumowanie 62 Bibliografia oraz Netografia 64 Spis ilustracji, tabel oraz wykresów 66 Wstęp Obecny rozwój technologii pozwala na automatyzację wielu procesów, w tym procesów finansowych, administracyjnych oraz sprzedażowych. Wymienione elementy są związane z obsługą biznesów, które umożliwiają komercyjny zarobek twórcom oprogramowania. Algorytmy, mają w tym przypadku ułatwić pracę lub całkowicie zastąpić osoby fizyczne w ich obowiązkach. Dzięki takim praktykom, pracodawcy, czy też różnego rodzaju organizacje, zwiększają swoje dochody poprzez przyspieszenie wykonywanej pracy lub w gorszym przypadku, oszczędzają fundusze poprzez zredukowanie etatów. Szerokie zastosowanie sztucznej inteligencji jest widoczne w każdym aspekcie naszego życia. Coraz większa ilość sklepów, banków czy też producentów wszelkiego rodzaju produktów, decyduje się na wdrażanie sztucznej inteligencji. Zgodnie z opinią specjalistów z Wyższej Szkoły Biznesu National-Louis University, AI (Artificial Intelligence) może powodować utratę miejsc pracy oraz restrukturyzację zawodów, jednakże tym samym może zwiększać zapotrzebowanie na specjalistów w branży kreatywnej oraz IT. Autor artykułu, zwraca uwagę na problematykę dotyczącą etyki związanej ze sztuczną inteligencją oraz potrzebę nieustannej nauki i rozwoju[12]. Istnieją również aspekty SI (Sztucznej Inteligencji), które są niezaprzeczalnie pozytywne, chociażby zastosowane w strefach pożytku publicznego, czy też w rozwiązaniach dla osób z dysfunkcjami. Wszelkiego rodzaju protezy, pojazdy, syntezatory mowy, algorytmy analizujące tekst, dźwięk czy też obraz, pozwalają na łatwiejsze funkcjonowanie osób niepełnosprawnych. Niestety obszary te są często pomijane, ze względu na stosunkowo niewielkie grono odbiorców, gotowych zapłacić za wprowadzenie takowych rozwiązań. Podejmując dalszą próbę analizy problemu, możemy zaobserwować wysokie zainteresowanie wadami wzroku oraz problemami ruchowymi, natomiast niskie zainteresowanie dysfunkcją głosową w odniesieniu do osób głuchoniemych. Istnieje wiele rozwiązań, które ułatwiają kontakt wzrokowy, chociażby takie jak regulacja wielkości czcionek we wszelkiego rodzaju aplikacjach, asystenci głosowi, czy też operacyjne korekty wzroku. Funkcje ruchowe, wspierane są przez różnorodne protezy, pojazdy, a także specjalne miejsca dostosowane do ich potrzeb np. miejsca parkingowe. Niestety w odniesieniu do problematyki osób głuchoniemych, nie ma zbyt wielu rozwiązań technologicznych, które mogą ułatwić ich życie w sposób nieinwazyjny. Cel i zakres pracy Celem pracy dyplomowej jest wparcie ww. grupy docelowej poprzez utworzenie oraz analizę oprogramowania do rozpoznawania sekwencji ruchów w czasie rzeczywistym, w oparciu o metody uczenia maszynowego. Zastosowane rozwiązania, wykorzystane zostaną następnie w aplikacji służącej do nauki oraz tłumaczenia polskiego języka

migowego. Oprogramowanie to może służyć jako wsparcie osób z dysfunkcjami w łatwiejszej adaptacji w środowisku osób pełnosprawnych. Aplikacja ta, również może działać w sposób edukacyjny, zwiększając świadomość użytkowników na temat dysfunkcji słuchowych oraz głosowych. Opracowanie autorskich skryptów, pozwalać ma na obsługę pełnego procesu pracy aplikacji, począwszy od rejestrowania próbek nagrań wideo, uczenia modelu SI, testowaniu jego poprawności, a kończąc na obsłudze aplikacji internetowej[4]. Niniejsza dysertacja, składa się z pięciu rozdziałów. Pierwszy z nich omawia zagadnienia teoretyczne związane z funkcjonowaniem aplikacji, takie jak przegląd dostępnych technologii związanych z rozpoznawaniem gestów oraz uczeniem maszynowym, wyzwaniami w rozpoznawaniu gestów oraz analizą istniejących platform dotyczących języka migowego. Następny rozdział omawia aspekty techniczne, które muszą zostać spełnione aby aplikacja działała w pełni poprawnie. W powyższym rozdziale znajdują się takie elementy jak: wymagania systemowe, zalecane oprogramowanie zewnętrzne oraz biblioteki, które należy zainstalować na urządzeniu docelowym. Rozdział trzeci porusza tematykę tworzenia modelu sztucznej inteligencji, przechodząc po każdym etapie jego powstawania, zaczynając od przygotowaniu danych wejściowych, a kończąc na ocenie skuteczności stworzonego modelu. Rozdział czwarty, zatytułowany „Implementacja platformy" skupia się na połączeniu aspektów sztucznej inteligencji wraz z aplikacją internetową. Zaprezentowane tam informacje dotyczą koncepcji wizualnej platformy, integracji modeli AI z aplikacją webową, mechanizmów komunikacji między klientem a serwerem oraz warstwy graficznej interfejsu użytkownika. Piąty, a zarazem ostatni rozdział niniejszej dysertacji omawia ocenę działania platformy, skupiając się na testach w warunkach rzeczywistych oraz symulowanych, ocenie skuteczności pracy aplikacji oraz propozycji ulepszeń oprogramowania. Praca zakończona została podsumowaniem, w którym zaprezentowano ogólny opis aplikacji wraz z jej analizą pod kątem przydatności oraz wydajności w rzeczywistych warunkach. 1. Przegląd literatury i analiza istniejących rozwiązań Rozpoznawanie wszelkiego rodzaju obiektów oraz ich właściwości, zyskały w ostatnich latach na popularności, głównie dzięki postępom w technikach uczenia maszynowego oraz rozwoju głębokich sieci neuronowych. W niniejszym rozdziale, omówione zostaną najważniejsze technologie oraz metody stosowane w tematyce rozpoznawania gestów, ze szczególnym uwzględnieniem języka migowego. 1.1. Przegląd technologii rozpoznawania gestów Technologie rozpoznawania gestów opierają się na stałej analizie ruchów ciała, ze szczególnym uwzględnieniem dłoni i palców, przy pomocy różnych sensorów oraz kamer. Systemy te wykorzystują zarówno dane wideo, jak i informacje trójwymiarowe, uzyskane za pomocą czujników głębi. Interdyscyplinarna dziedzina badań rozpoznawania gestów, łączy ze sobą elementy przetwarzania obrazów, komputerowego widzenia, sztucznej inteligencji oraz interakcji człowiek-komputer. Podstawą rozpoznawania gestów jest przetwarzanie obrazu, które obejmuje szereg technik służących do analizy danych wizualnych. Można wyłonić takie etapy jak: detekcja obrazów, śledzenie ruchu oraz ekstrakcja cech. Detekcja obrazów opiera się na rozpoznaniu dłoni oraz pozostałych istotnych części ciała na przetwarzanym obrazie. Techniki te bazują na algorytmach segmentacji obrazu, które identyfikują obiekty na podstawie ich koloru, kształtu lub ruchu. Algorytmy takie jak YOLO (You Only Look Once) oraz SSD (Single Shot Multibox Detector) są powszechnie stosowane do szybkiej i dokładnej detekcji dłoni w badanych obrazach. Śledzenie ruchu, opiera się na analizie każdej klatki nagrania w celu uchwycenia przesunięć, występujących w punktach wykrytych przez poprzedni proces. W tym celu stosuje się miedzy innymi algorytmy KLT (Kanade-Lucas-Tomasi) oraz Mean Shift, które pozwalają na dokładne monitorowanie trajektorii ruchu dłoni. Ostatnią techniką jest ekstrakcja cech, która umożliwia dokonanie wyodrębnienia z obrazów stawów oraz kości, niezbędnych do identyfikacji wykonanego gestu. W tym celu stosowane są takie narzędzia jak OpenPose oraz MediaPipe[3]. Biblioteka OpenPose została utworzona przez Perceptual Lab na Uniwersytecie Carnegie Mellon (CMU), pod nadzorem profesora Asuyuki Matsumoto oraz doktoranta Zhe Cao, który jest głównym autorem pracy badawczej, opisującej możliwości biblioteki OpenPose. Pierwsza wersja OpenPose została opublikowana w 2017 roku, a jej wyniki były oparte na pracy naukowej pt. "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", której współautorem był wspomniany wcześniej Zhe Cao. Praca ta po raz pierwszy zaprezentowana została na konferencji CVPR (Computer Vision and Pattern Recognition) w 2017 roku. Początkowo OpenPose służył głównie do analizy oraz detekcji pozycji wielu osób w czasie rzeczywistym na obrazach 2D. Pierwsza wersja koncentrowała się na problemie, który dotąd stanowił duże wyzwanie, a mianowicie detekcji pozycji ciała wielu osób na jednym obrazie, w jak najszybszym czasie, z zachowaniem wysokiej precyzji. Istniejące wcześniej rozwiązania, były skoncentrowane głównie na wykryciu jednego człowieka lub działały wolniej, co uniemożliwiało zastosowanie ich w czasie rzeczywistym. Biblioteka ta zyskała spore zainteresowanie, co przyczyniło się do rozszerzenia jej o dodatkowe funkcje, takie jak detekcja szczegółowych ruchów dłoni i mimiki twarzy oraz analizy 3D. Głównym zadaniem obecnej wersji biblioteki jest wykrywanie oraz śledzenie pozycji ciała

człowieka w obrazie 2D oraz 3D, uwzględniając różne części ciała takie jak kończyny, głowa oraz palce rąk i stóp. Podstawą działania OpenPose są techniki głębokiego uczenia, a dokładniej konwolucyjne sieci neuronowe (CNN), które za pomocą punktów kluczowych (tzw. keypoints), rozpoznają położenie ciała, dzieląc je na odpowiednie struktury. W początkowej fazie programu, biblioteka przeprowadza detekcję punktów kluczowych, następnie tworzy mapę ufności, która określa z jakim prawdopodobieństwem dany punkt (np. łokieć) znajduje się w określonej lokalizacji na obrazie. Utworzona mapa jest dwuwymiarową siatką, gdzie wartości w poszczególnych komórkach reprezentują stopień pewności. Po utworzeniu mapy ufności, biblioteka tworzy mapy części ciała, analizując położenie konkretnych elementów, co w dalszym etapie służy do utworzenia szkieletu osoby. W tym celu wykorzystywane są mapy PAF (Part Affinity Fields), czyli wektorowe pola, które wskazują, jak prawdopodobne jest, że dwa punkty kluczowe są ze sobą powiązane. Gdy mapy te zostaną utworzone, biblioteka przystępuje do połączenia punktów tworząc szkielet[2]. Druga spośród wymienionych bibliotek (MediaPipe) została udostępniona jako projekt open-source przez firmę Google w czerwcu 2019 roku. MediaPipe, jest to otwartoźródłowe środowisko, które umożliwia wykonywanie różnych zadań związanych z przetwarzaniem obrazów, wideo oraz danych w czasie rzeczywistym. Zbudowana została na podstawie struktury pipelinów przetwarzania multimodalnych danych, co oznacza, że pozwala na tworzenie i uruchamianie strumieniowych systemów analizy danych, w sposób modułowy oraz wydajny. Pipeline to zbiór skomponowanych kroków przetwarzania (tzw. modułów) w ustalonej kolejności. Każdy z modułów odpowiedzialny jest za określone zadanie np. detekcję obiektów, segmentację czy śledzenie ruchu. Dzięki temu, biblioteka ta jest bardzo elastyczna, gdyż użytkownik może z łatwością tworzyć własne pipeliney, dostosowując je do specyficznych potrzeb. Pierwotnym założeniem MediaPipe było dostarczenie wydajnej biblioteki, która umożliwiałaby wieloplatformowe przetwarzanie strumieni wideo i obrazów w czasie rzeczywistym[6]. Przechodząc do kwestii hardware, najczęściej wykorzystywanymi urządzeniami do rejestracji gestów są kamery RGB. Ich popularność jest uargumentowana stosunkowo niską ceną oraz dostarczanym kolorowym obrazem w wysokiej rozdzielczości, który może być przetwarzany przez algorytmy komputerowego widzenia. Nie są to jednakże jedyne narzędzia, wykorzystywane w procesie rejestrowania ruchów. Elementy takie jak kamery głębi pozwalają na trójwymiarową rejestrację sceny, co umożliwia bardziej precyzyjną analizę ruchów dłoni i ciała. Dużą zaletą kamer głębi jest wysoka dokładność w trudnych warunkach oświetleniowych, gdzie w porównaniu z kamerami RGB, nie mają problemów z dokładnym uchwyceniem gestu. Dodatkowym asortymentem, mogą być również czujniki ruchu, które mierzą przyspieszenie i orientację dłoni. Zastosowanie ww. technologii, pozwala na rejestrowanie subtelnych ruchów i położenia dłoni w przestrzeni. Ostatnia z przestawionych opcji, daje największe możliwości rejestrowania ruchów. Niestety ze względu na koszt zaawansowanych technologii stosowanych w czujnikach ruchu, produkty te są nieprzystępne cenowo dla potencjalnego odbiorcy oprogramowania, omawianego w niniejszej dysertacji[9]. Rysunek 1 Porównanie technologii (Źródło: opracowanie własne) 1.2. Przegląd metod uczenia maszynowego Uczenie maszynowe (ang. Machine Learning (ML)) to dziedzina sztucznej inteligencji, która skupia się na rozwijaniu algorytmów oraz modeli zdolnych do uczenia i automatycznego podejmowania decyzji na podstawie dostarczonych danych. W odniesieniu do platformy tłumaczącej język migowy jest kluczowym rozwiązaniem, umożliwiając przekształcanie danych wideo w rozpoznawalne wzorce, które są możliwe do przetłumaczenia na słowa lub zdania. Pierwsza sieć neuronowa została opracowana przez Warrena McCullocha, wykładowcy z University of Illinois i Waltera Pittsa, niezależnego badacza. Ich artykuł pt. „A Logical Calculus of the Ideas Immanent in Nervous Activity", opublikowany w 1943 roku w czasopiśmie „Bulletin of Mathematical Biophysics" został przyjęty z szerokim entuzjazmem, kładąc tym samym podwaliny pod rozwój sztucznych sieci neuronowych. Praca ta okazała się kamieniem milowym w późniejszych badaniach w dziedzinie informatyki, neurobiologii i kognitywistki. Autorzy, przedstawili w nim jak mogą działać neurony, prezentując swoje pomysły i tworząc prostą sieć neuronową za pomocą obwodów elektrycznych[15]. Obecnie sieci neuronowe są fundamentem współczesnych metod uczenia maszynowego. Składają się z wielu warstw neuronów, które przetwarzają dane wejściowe poprzez szereg operacji matematycznych, co umożliwia uczenie się złożonych wzorców. Konwolucyjne sieci neuronowe (CNN) są szczególnie efektywne w przetwarzaniu danych obrazowych oraz wideo, dzięki swojej zdolności do automatycznego wyodrębnienia cech z surowych danych. Sieci te, składają się z warstw konwolucyjnych, poolingowych oraz gęstych, które współpracują ze sobą, w celu przetwarzania informacji o strukturze przestrzennej obiektów. Autorzy publikacji Sign Language Recognition Using Convolutional Neural Networks wykazali, że CNN mogą być używane do skutecznego przetwarzania i rozpoznawania gestów na podstawie obrazu wideo, umożliwiając automatyczną ekstrakcję cech, co jest

kluczowe dla poprawnej klasyfikacji gestów. Podkreślili również, że ich model nie tylko osiąga wysoką dokładność w rozpoznawaniu gestów, ale także jest skalowalny i może być rozszerzany o nowe gesty lub języki migowe z relatywnie niewielkim wysiłkiem. Badanie to otwiera drogę do tworzenia bardziej zaawansowanych systemów rozpoznawania języka migowego, które mogą być wykorzystywane w aplikacjach takich jak tłumacze języka migowego na mowę w czasie rzeczywistym, tworzenie interfejsów użytkownika dla osób niesłyszących, a także w edukacji[10]. Modele hybrydowe to połączenie różnych algorytmów klasyfikacyjnych, które mogą prowadzić do poprawy dokładności i stabilności rozpoznawania gestów. Przykładem takiego podejścia jest kombinacja CNN oraz HMM, co pozwala na skuteczniejsze modelowanie zależności czasowych w danych sekwencyjnych. Zastosowanie hybrydy modeli CNN oraz HMM opisuje publikacja pt. „SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition". Autorzy publikacji wykazują, że hybrydowe podejście może być skutecznie wykorzystywane do rozpoznawania ciągłych gestów w języku migowym. Ich badania przeprowadzone zostały na dużych zestawach danych, zawierających filmy z gestami języka migowego, przy czym wykorzystali takie techniki jak augmentacja danych, zwiększając tym samym różnorodność przykładów i poprawiając zdolność generalizacji modelu. Wyniki eksperymentu pokazują, że SubUNets osiąga znacznie lepsze wyniki w porównaniu z wcześniejszymi podejściami, zarówno w kontekście rozpoznawania kształtów dłoni, jak i ciągłego rozpoznawania gestów języka migowego. Autorzy podkreślają, że ich architektura nie tylko rozpoznaje gesty z większą dokładnością, ale także lepiej radzi sobie z różnorodnością kształtów dłoni i płynnością sekwencji[1]. 1.3. Wyzwania w rozpoznawaniu gestów języka migowego Rozpoznanie gestów języka migowego stanowi złożone wyzwanie, które wynika z unikalnych cech ww. sposobu komunikacji. W przeciwieństwie do języków mówionych, język migowy wykorzystuje mimikę oraz posturę ciała, co wymaga zaawansowanych algorytmów do skutecznej analizy i rozpoznania. Język ten charakteryzuje się ogromną różnorodnością gestów. Każdy z nich może mieć wiele wariantów, które różnią się w zależności od regionu, kultury, a nawet osoby wykonującej gest. Ponadto, gesty mogą obejmować stosunkowo nieistotne różnice w ruchach dłoni, ułożeniu palców, kierunku spojrzenia, a także w mimice twarzy, co sprawia, że poprawne rozpoznanie jest niezwykle trudne, zwłaszcza jeżeli model zostaje rozszerzany o nowe gesty oraz ich warianty. Badania prowadzone przez Koller i innych naukowców wskazują, że klasyczne podejścia oparte na modelach HMM mogą być niewystarczające do modelowania złożonych i różnorodnych gestów języka migowego. Aby sprostać temu wyzwaniu, autorzy sugerują zastosowanie hybrydowych modeli łączących CNN oraz HMM, co pozwala na skuteczniejsze modelowanie złożoności gestów[5]. Gesty języka migowego często wykonywane są w sposób płynny oraz nieprzerwany co stwarza jeden z największych problemów w przypadku zastosowania algorytmów SI. Kwestia ta jest niezwykle problematyczna dla modeli, które do poprawnego działania potrzebują wyraźnych przerw pomiędzy poszczególnymi znakami, aby określić gdzie badany gest się zaczyna, a gdzie kończy. Kluczowym problemem podczas tworzenia modelu sztucznej inteligencji jest również sama tematyka. W związku z niszowością tematu, utrudniony jest dostęp do dużych, zróżnicowanych zbiorów danych, które są niezbędne dla skutecznego trenowania modeli uczenia maszynowego. W przypadku języka migowego, zbiory danych są ograniczone pod względem wielkości i różnorodności. Brakuje danych pochodzących od różnych użytkowników, wykonanych w zróżnicowanych warunkach oświetleniowych i środowiskach, co może prowadzić do słabszej generalizacji modelu. Problematykę tą porusza w swoich badaniach Patel oraz inni naukowcy, zwracając uwagę na problem ograniczonych zbiorów danych, proponując jednocześnie zastosowanie techniki transfer learningu, które pozwalają na wykorzystanie wiedzy zebranej na większych zbiorach danych do trenowania modeli na mniejszych, specyficznych zbiorach języka migowego[6]. Rozpoznawanie gestów w czasie rzeczywistym wymaga również znacznej mocy obliczeniowej, zwłaszcza w przypadku samego tworzenia modelu oraz ekstrakcji punktów na zbiorach treningowych. Programy wykorzystujące przetwarzanie obrazu w czasie rzeczywistym muszą przetwarzać duże ilości danych wideo o wysokiej rozdzielczości, minimalizując poziom opóźnienia do minimum. Optymalizacja rozwiązań w stosunku do urządzeń o gorszych parametrach również stwarza dodatkowe problemy, zwłaszcza w przypadku urządzeń mobilnych. Ostatnim powszechnie znanym wyzwaniem może być różnica w indywidualnym zachowaniu użytkownika. Każda osoba ma inną długość oraz szerokość kończyn. Inna szybkość, kąt nachylenia czy też indywidualny styl poruszania się każdej osoby sprawia, że opracowany model musi być odporny na takowe odchylenia. Na elastyczność modelu zwraca uwagę między innymi Zhang i Liu, wskazując konieczność opracowania metod, które mogą uwzględnić różnice indywidualne, takie jak ww. style gestykulacji, poprzez wykorzystanie cech całej dłoni, co pozwala na dokładniejsze rozpoznawanie gestów w różnych warunkach[11]. 1.4. Analiza problematyki oraz istniejących aplikacji dotyczących języka migowego W ciągu ostatnich lat rozwój technologii głębokiego uczenia oraz komputerowego

widzenia przyczynił się do powstania wielu aplikacji, służących do rozpoznawania i tłumaczenia języka migowego. Aplikacja opracowana przez Patel i Sahah, pozwala użytkownikom uczyć się języka migowego za pomocą interaktywnych ćwiczeń. System ten, analizuje gesty użytkownika za pomocą technologii rozpoznawania wideo w czasie rzeczywistym, umożliwiając otrzymanie natychmiastowej informacji zwrotnej, co przyczynia się do skuteczniejszej nauki[8]. Aplikacja o nazwie DeepSign, również funkcjonuje w przestrzeni publicznej jako tłumacz języka migowego. Aplikacja ta wykorzystuje CNN do rozpoznawania gestów i przekształcania ich w tekst w czasie rzeczywistym. System ten jest bardzo zaawansowany, co pozwala tłumaczyć gesty na pełne zdania, co znacznie ułatwia komunikację[7]. Niestety każda z wyżej wymienionych aplikacji jest stworzona na rynek anglojęzyczny, a co za tym idzie, nie obsługuje polskiego języka migowego. Zapoznając się z aktualną na dzień 27.08.2024 ofertą na rynek polskojęzyczny, można odnaleźć jedynie rozwiązania, które nie są zautomatyzowane. Specjalne ośrodki czy też firmy prywatne obsługują klientów w ramach spotkań lub połączeń online z tłumaczem języka migowego. Analizując problematykę zastosowania sztucznej inteligencji, można dostrzec, iż nisza ta nie została jeszcze odpowiednio obsłużona w Polsce. Rozwiązania takie jak wideokonferencje, są skuteczną alternatywą, jednakże wymagają osoby fizycznej, która takową rozmowę przetłumaczy. Wiąże się to z kosztami zatrudnienia specjalisty oraz utrzymania infrastruktury, takiej jak stabilne połączenie z Internetem, odpowiedni sprzęt audio-wizualny oraz platformy obsługującej powyższe rozmowy. Zagłębiając się bardziej w poruszaną tematykę, można odnieść wrażenie, że osoby z dysfunkcjami instrumentów głosowych są w pewien sposób wykluczone społecznie, poprzez brak powszechnego narzędzia, które ułatwiałoby im życie codzienne w sferze komunikacyjnej. W przypadku każdego typu rozmów, należy skupić się na zagadnieniu prywatności, do której każdy obywatel ma prawo. Chociażby w taki sferach jak zdrowie czy też finanse. Dzięki istnieniu aplikacji, która byłaby w stanie w pełni przetłumaczyć język migowy, osoby z dysfunkcjami mowy, mogłyby korzystać z większości znanych usług, bez obecności osoby trzeciej, która będzie towarzyszyć w prowadzeniu rozmowy. Zaleca się aby aspekt ten został szerzej zbadany przez specjalistów w dziedzinie badań społecznych, analizując jednocześnie potrzeby osób niepełnosprawnych, a tym samym zwiększając świadomość społeczeństwa na temat poruszany w omawianej dysertacji. 2. Techniczne aspekty funkcjonowania aplikacji Każde oprogramowanie wymaga uprzedniego przygotowania środowiska pracy. Tak też jest w przypadku aplikacji dołączonej do niniejszej dysertacji. Wybrane etapy posiadają własne wymagania odnośnie sprzętu (ang. hardware), oprogramowania (ang. software) oraz środowiska programistycznego, dlatego w poniższych punktach przedstawione zostały kroki do poprawnej konfiguracji środowiska. 2.1. Wymagania systemowe Program podzielony został na dwa etapy główne. Pierwszy etap odpowiada za przygotowanie próbek oraz modelu sztucznej inteligencji, natomiast drugi etap jest stricte związany z platformą, która ma za zadanie odczytywać przygotowany w etapie pierwszym model SI. Każdy program posiada inne wymagania systemowe, dlatego też zostały one zawyżone w odniesieniu do najbardziej wymagającego programu z etapu pierwszego. Komponent Minimalne wymagania Zalecane wymagania Procesor (CPU) Sześciordzeniowy, np. Intel Core i5- Ośmiordzeniowy, np. Intel Core 10600K lub AMD Ryzen 5 3600 i7-9700K lub AMD Ryzen 7 3700X Pamięć RAM 16 GB RAM 32 GB RAM Karta graficzna Zintegrowana Zintegrowana (GPU) Przestrzeń Minimum 20 GB wolnej przestrzeni na SSD NVMe z minimum 100 GB dyskowa dysku SSD wolnej przestrzeni System Windows 10 lub nowszy / Linux (Ubuntu Windows 10 Pro lub nowszy / operacyjny 20.04 lub nowszy) Linux (Ubuntu 20.04 LTS lub nowszy) Inne Python 3.8 lub nowszy Python 3.8 lub nowszy Informacje Procesor ośmiordzeniowy z obsługą wielowątkowości zapewni dobrą wydajność w przetwarzaniu wideo i trenowaniu modeli, choć na nieco niższym poziomie niż wcześniej wspomniane opcje. Zintegrowana karta graficzna wystarczy do obsługi podstawowych zadań związanych z wyświetlaniem i przetwarzaniem obrazu, ponieważ główne obciążenie będzie przeniesione na procesor, aby zmienić obciążenie z procesora na kartę graficzną należy dokonać modyfikacji programu „MachineLearning.py”. 32 GB RAM pozwoli na komfortową pracę z większymi zbiorami danych i bardziej złożonymi modelami Tabela 1 Wymagania systemowe cz1 Druga część programu opiera się głównie na obsłudze aplikacji internetowej oraz przetwarzaniu obrazu z kamery na podstawie dostarczonego już modelu, co nie wymaga od użytkownika zwiększonej mocy obliczeniowej. Komponent Minimalne wymagania Zalecane wymagania Procesor Czterordzeniowy procesor, np. Intel Sześciordzeniowy procesor, np. Intel (CPU) Core i5-8265U lub AMD Ryzen 5 Core i7-10750H lub AMD Ryzen 5 2500U 4600H Pamięć RAM 8 GB RAM 16 GB RAM Karta Zintegrowana, np. Intel UHD Zintegrowana, np. Intel Iris Xe lub graficzna Graphics 620 lub AMD Radeon AMD Radeon Vega 11 (GPU) Vega 8 Przestrzeń Minimum 10 GB wolnej przestrzeni SSD NVMe z minimum 50 GB wolnej dyskowa na dysku SSD przestrzeni System Windows 10 lub

nowszy / Linux Windows 10 Pro lub nowszy / Linux operacyjny (Ubuntu 20.04 lub nowszy) (Ubuntu 20.04 LTS lub nowszy) Inne Python 3.8 lub nowszy Python 3.8 lub nowszy Tabela 2 Wymagania systemowe cz2 2.2. Narzędzia modelowania sztucznej inteligencji Zgodnie z informacjami przedstawionymi w podpunkcie 2.1, każdy program posiada inne wymagania sprzętowe. W tym punkcie przedstawione zostaną indywidualne wymagania dotyczące poszczególnych programów, które nie zostały uwzględnione w poprzednich punktach. Pakiety wymagane do uruchomienia poszczególnych aplikacji, można zainstalować poprzez narzędzie do zarządzania pakietami w Pythonie (pip), wpisując wskazane komendy w terminalu. Program „NagrywanieVideo720p60FPS.py" do poprawnego działania potrzebuje podłączonej kamery internetowej obsługującej jakość 720p oraz przepustowość 60fps, a także importu pakietów takich jak os, time oraz opencv. Dwa pierwsze pakiety są dostępne w podstawowej wersji Pytthona, natomiast pakiet opencv musi zostać doinstalowany. Aby tego dokonać można skorzystać z poniższego polecenia: pip install opencv-python Do obsługi programu „MachineLearning.py" wymagane są pakiety: os, threading, warnings, random, time, webbrowser, multiprocessing, tkinter, cv2, numpy, mediapipe, tensorflow, keras, flatten, scikit-learn oraz pil , osiem pierwszych pakietów to standardowe moduły języka Python, brakujące pakiety można zainstalować inicjując polecenie: pip install opencv-python numpy mediapipe tensorflow keras scikit-learn pillow Program „Prediction Test.py" wymaga takich pakietów jak: cv2, numpy, mediapipe, tensorflow, matplotlib, keras, sklearn, os, tkinter. Brakujące pakiety można zainstalować za pomocą polecenia: pip install opencv-python numpy mediapipe tensorflow keras scikit-learn pillow Do poprawnej obsługi platformy, wymagane są również odpowiednie środki, takie jak pakiety języka python oraz zewnętrzne oprogramowanie inicjujące środowisko serwerowe. Omawiana platforma wymaga utworzenia lokalnego serwera w celu obsługi języka PHP. Przykładem oprogramowania pozwalającego na zainicjowanie ww. środowiska jest XAMPP. Do utworzenia niniejszej aplikacji wykorzystano najnowszą (na dzień 22.08.2024) wersję XAMPP w wersji 3.3.0 oraz PHP w wersji 8.2.12. Po zainstalowaniu aplikacji XAMPP, należy uruchomić opcję Apache, oznaczoną na Rysunku 2 numerem 1. Rysunek 2 Interfejs programu XAMPP Platforma, została projektowana oraz testowana za pomocą przeglądarki internetowej Chrome, która jest zalecana do obsługi ww. aplikacji. Program „app.py" potrzebuje podłączonej kamery internetowej o przepustowości minimum 30 fps oraz zainstalowanych pakietów takich jak: Flask, flask-cors, opencv, numpy, mediapipe, tensorflow, keras, pillow oraz scikit-learn. Elementy te, można zainstalować poprzez komendy w terminalu: pip install Flask flask-cors opencv-python numpy mediapipe tensorflow keras Pillow scikit-learn 3. Budowa modelu rozpoznawania gestów Aplikacja internetowa do właściwego działania potrzebuje wcześniej przygotowanego modelu sztucznej inteligencji, który będzie rozpoznawał wybrane sekwencje ruchów. W tym rozdziale zostanie przedstawiony opis procesu przygotowania ww. modelu. 3.1. Przygotowanie danych wejściowych Przygotowanie danych wejściowych jest kluczowym etapem każdego programu opartego na uczeniu maszynowym. Proces ten obejmuje takie elementy jak wybór odpowiednich źródeł danych, ich organizację, walidację oraz przetworzenie do odpowiedniego formatu. W związku z niszowością tematyki polskiego języka migowego oraz utrudnionym dostępem do próbek badawczych, autor dysertacji utworzył własną bazę gestów, które poprzez algorytm napisany w języku python zostały zarejestrowane za pomocą kamery internetowej. Każde z nagrań musiało zostać ustandaryzowane w celu przystosowania ich do pracy z jednolitym modelem sztucznej inteligencji. Każda z zarejestrowanych próbek posiadała długość 4 sekund, rozdzielczość 720p oraz przepustowość 60 klatek na sekundę. Dodatkowo, autor przyjął metodę 25-50-25, która zaowocowała nagraniem 100 próbek dla każdego gestu. W późniejszym etapie projektu powtórzono ten proces, aby zyskać większą ilość próbek na innych modelach ciał. Metoda ta opierała się na odpowiednim ustawieniu kadru. Pierwszy etap tworzył 25 nagrań od czubka głowy do dolnej części żeber. Taka konfiguracja, pozwalała na najdokładniejsze zarejestrowanie pracy dłoni, wliczając w to układ palców. Drugi etap tworzył 50 nagrań od czubka głowy do górnej części uda. Nagrania te służyły jako baza do całości gestu. Ujęcia te, ujmowały każdą fazę ruchu, od fazy startowej do fazy końcowej, uwzględniając wszystkie niezbędne elementy ludzkiego działa. Ostatni etap tworzył 25 nagrań od czubka głowy do dolnej części kolan. Proces ten miał za zadanie zwiększyć obszar widzenia programu, uwzględniając dalsze położenie szukanych punktów, dzięki czemu utworzony w późniejszym etapie model, mógł z większą precyzją rozpoznawać pozycję statyczną oraz ruch odpowiednich fragmentów ciała człowieka. Ustawienie każdego z etapów zostało zobrazowane na ilustracji 3. Rysunek 3 Etapy zbierania próbek (Źródło: opracowanie własne) Dzięki zastosowaniu ww. rozwiązania, późniejsze algorytmy przetwarzania danych, mogły z większą łatwością zlokalizować wybrane punkty na ludzkim ciele, a następnie przetworzyć je w ramach sekwencji ruchu. Każdy z etapów dostarczał inną ilość punktów, dzięki czemu model uczył się rozpoznawania gestów ze zróżnicowanej perspektywy. W ramach części praktycznej pracy magisterskiej, został dołączony program „NagrywanieVideo720p60FPS.py", który

realizuje wyżej wymienione etapy. Po zakończeniu procesu przygotowywania próbek badawczych autor dysertacji opracował program „MachineLearning.py", który realizuje wszystkie pozostałe kroki do uzyskania działającego modeli SI. Po uruchomieniu ww. programu, użytkownik ma możliwość skorzystania z graficznego interfejsu programu, utworzonego poprzez bibliotekę tkinter. Biblioteka ta jest standardowym pakietem w języku Python. Początkowo, program weryfikuje czy w obecnym katalogu istnieje plik z przetworzonymi punktami o nazwie „preprocessed_data.npz". Jeżeli takowy plik istnieje, zostaje wyświetlone okno z odpowiednim komunikatem (Rys.4). Rysunek 4 Uczenie maszynowe (GUI 1) (Źródło: opracowanie własne) Użytkownik może wybrać jedną z trzech dostępnych opcji. Pierwsza z nich prowadzi do pliku PDF, w którym opisane zostały wymagania sprzętowe dotyczące obsługi programu. Przycisk po prawej stronie, rozpoczyna proces uczenia maszynowego na obecnie wykrytych danych, natomiast przycisk „Wróć do wyboru danych" pozwala na otworzenie widoku okna startowego. Okno to pojawia się również, jeżeli program w początkowej fazie działania nie odnajdzie utworzonego pliku z wyekstraktowanymi punktami. Rysunek 5 Uczenie maszynowe (GUI 2) (Źródło: opracowanie własne) W ww. oknie, użytkownik może wybrać ścieżkę do katalogu, w którym znajdują się wcześniej przygotowane nagrania. Nagrania te, muszą być zgrupowane w odpowiedni sposób. Aby program działał poprawnie, struktura plików powinna wyglądać następująco: Katalog wybrany przez użytkownika / Nazwa sekwencji ruchu / pliki o rozszerzeniu *.avi. Jeżeli program wykryje błędną strukturę plików, zwróci komunikat w kolorze czerwonym. W przypadku poprawnie przetworzonych plików, wyświetli stosowne powiadomienie w kolorze zielonym, tworzy listę wszystkich ścieżek do plików wideo, a następnie odblokuje przycisk „Rozpocznij analizę". Każdy z powyższych przykładów został zobrazowany, za pomocą poniższej grafiki. Rysunek 6 Uczenie maszynowe (GUI 2 - Struktura) (Źródło: opracowanie własne) 3.2. Preprocessing danych Preprocessing polega na wstępnym przetwarzaniu danych, a dokładniej na przekształcaniu surowych danych w formę, która jest bardziej odpowiednia i optymalna dla modelu uczenia maszynowego. Celem tego fragmentu kodu, jest utrzymanie spójności oraz kompatybilności, a także poprawa jakości danych, potrzebnych do uczenia modelu. Usuwa on wszystkie zbędne informacje, czyli tzw. szumy, pozostawiając jedynie potrzebne dane, wymagane przez model SI. Po rozpoczęciu analizy nagrań, program wyświetla nowe okno z paskiem postępów oraz możliwością rozszerzenia widoku o szczegóły. Na rysunku 7, po lewej stronie przedstawiono podstawowe okno przetwarzania danych, po prawej stronie natomiast jest widoczne okno po rozszerzeniu szczegółów. Rysunek 7 Uczenie maszynowe (GUI 3) (Źródło: opracowanie własne) W trakcie przetwarzania danych, algorytm wykorzystuje bibliotekę MediaPipe, opracowaną przez Google. Za jej pomocą, przeprowadza ekstrakcję punktów kluczowych z dostarczonych nagrań, analizując każdą z 240 klatek, a następnie przypisuje do nich odpowiednie etykiety. W ten sposób tworzy pełen spis punktowy, przedstawionego na nagraniu gestu. Zastosowanie powyższego rozwiązania, skutkuje zmniejszeniem ilości danych przetwarzanych przez etap uczenia maszynowego. W przypadku standardowego podejścia, program przetwarzałby każdy pikseli wideo, co mogłoby wprowadzić dodatkowy szum. Skutkowałoby to mniej precyzyjnym uczeniem modelu, a co za tym idzie gorszą precyzją rozpoznawania gestów. Program, posiada również możliwość podglądu wizualizacji procesu ekstrakcji punktów w czasie rzeczywistym. Gdy użytkownik po rozwinięciu szczegółów, uruchomi przycisk „Wyświetl wizualizację" otworzy się nowe okno, które zobrazuje przetwarzanie danych w sposób graficzny. Rysunek 8 Uczenie maszynowe (GUI 3 - Wizualizacja) (Źródło: opracowanie własne) Podczas przetwarzania informacji, program normalizuje dane poprzez pomijanie wideo o dłuższym czasie trwania lub mniejszej ilości klatek na sekundę, jeżeli natomiast nagranie jest krótsze, brakująca część czasu jest dodawana poprzez funkcję uzupełniania sekwencji pad_sequences. Funkcja ta, tworzy macierz wypełnioną zerami o wymiarach zdefiniowanych w kodzie programu, następnie dla każdej sekwencji kopiuje jej wartość do nowej macierzy, a jeżeli sekwencja jest krótsza niż maksymalna długość, resztę wypełnia zerami. Program, ze względu na konieczność przetwarzania dużej ilości danych oraz potencjalnie długi okres trwania ww. procesu, posiada obsługę wieloprocesowości. Wieloprocesowe podejście pozwala na jednoczesne przetwarzanie wielu nagrań na rdzeniach procesora, co znacząco redukuje czas potrzebny na preprocessowanie danych. Funkcje, takie jak blokada oraz kolejka zapewniają synchronizację między procesami, aby zapobiec ewentualnym kolizjom podczas dostępu do wspólnych zasobów, takich jak zapis do listy danych, czy aktualizacja wskaźnika postępu. Gdy dane zostaną poprawnie przetworzone, program tworzy plik „preprocessed_data.npz", w którym zapisuje przetworzone dane oraz „classes.npy" w którym przechowuje nazwy etykiet. Następnie pasek postępu dobiegnie końca, a na ekranie użytkownika wyświetla się komunikat, który umożliwia przejście do procesu uczenia modelu SI lub zamknięcie programu. Rysunek 9 Uczenie maszynowe (GUI 3 - Zakończenie przetwarzania danych) (Źródło: opracowanie

własne) 3.3. Architektura modelu oraz algorytm klasyfikacji Architektura modelu odnosi się do struktury sieci neuronowej. To ona determinuje sposób, w jaki dane są przetwarzane, jakie operacje są wykonywane oraz jakie mechanizmy są stosowane do obsługi modelu. W omawianym oprogramowaniu, autor dysertacji zastosował model oparty na konwolucyjnych sieciach neuronowych (CNN), co miało na celu uchwycenie wzorców w danych sekwencyjnych, takich jak punkty kluczowe wyodrębnione z nagrań wideo. W kontekście przetwarzania sekwencji, warstwy konwolucyjne (Conv1D) przetwarzają dane jednokierunkowo. Filtrują sekwencje punktów kluczowych, tym samym wykrywając wzorce przestrzenno-czasowe. Filtry, inaczej zwane jądrami, są przesuwane wzdłuż sekwencji, stosując operację splotu. Czynność ta, pozwala na uchwycenie lokalnych wzorców danych. Po każdej warstwie konwolucyjnej zastosowano warstwę MaxPooling, która redukuje wymiarowość danych, zachowując najbardziej istotne cechy. MaxPooling wybiera maksymalną wartość z każdej lokalnej grupy wyników, co pomaga w uodpornieniu modelu na przesunięcia w danych. Równocześnie dokonuje redukcji liczby parametrów, co skutkuje zmniejszeniem ryzyka wystąpienia zjawiska przeuczenia modelu. Model został zaimplementowany przy pomocy biblioteki Keras, która działa na bazie TensorFlow. Zastosowane rozwiązanie oferuje szeroką gamę narzędzi do pracy z przetwarzaniem danych, budowaniem architektury sieci, a także optymalizacji modeli. Omawiany program, tworzy model w sposób sekwencyjny, co oznacza, że warstwy są dodawane jedna po drugiej, tworząc stos warstw. Każda z warstw przyjmuje jako argumenty liczbę filtrów o wymiarze 64, rozmiar jądra równy 3 oraz funkcję aktywacji „relu”. Liczba filtrów określa, ile różnych cech model będzie próbował wyodrębnić. Rozmiar jądra to zakres, na którym filtr jest stosowany w danej chwili. Funkcja aktywacji relu, poprzez zamianę wszystkich wartości ujemnych na zero, zapobiega problemowi zanikania gradientu i przyspiesza proces nauki, umożliwiając sieci efektywne uczenie się złożonych wzorców. Warstwa Flatten w niniejszym programie, używana jest do przekształcenia danych z formatu wielowymiarowego do jednowymiarowego, który jest wymagany przez kolejne, w pełni połączone warstwy takie jak warstwa gęsta Dense. Warstwa ta, używana jest zarówno do przekształcania wyodrębnionych cech w postać o mniejszej wymiarowości, jak i do generowania ostatecznego wyniku klasyfikacji. Units, czyli liczba neuronów w warstwie została ustawiona na 256, co pozwala na modelowanie bardziej złożonych zależności, jednakże może prowadzić do przeuczenia. W ostatnim kroku, warstwa Dense generuje wyniki klasyfikacji za pomocą funkcji aktywacji softmax. Funkcja ta, przekształca wyniki w prawdopodobieństwa, które sumują się do 1, co pozwala na przypisanie prawdopodobieństwa do klasy. 3.4. Trening, walidacja oraz ocena skuteczności modelu Trening modelu to proces optymalizacji jego parametrów tak, aby jak najlepiej uczył się rozpoznawania wzorców dostarczonych przy pomocy danych wejściowych, a następnie właściwie przypisywał je do odpowiednich etykiet. W kontekście sieci neuronowych, trening polega na minimalizacji funkcji straty, przy użyciu danych treningowych oraz odpowiedniego optymalizatora. W pierwszej fazie treningu modelu, zastosowany został tzw. forwadr propagation, czyli etap w którym dane przekazywane są do modelu, a każdy neuron przekształca wejścia, oblicza wyjścia i przekazuje je do następnej warstwy. Na końcu tego procesu, model generuje wynik, który jest porównywany z rzeczywistą etykietą danych przy użyciu funkcji straty. Następnie występuje backpropagation, czyli kluczowy mechanizm w treningu sieci neuronowych. Polega on na propagacji błędu wstecz przez sieć neuronową. Błąd ten jest obliczany jako różnica pomiędzy przewidywaniami modelu, a rzeczywistymi wynikami. Proces ten, aktualizuje wagi w sieci tak, aby minimalizować funkcję straty. W tym przypadku zastosowany został algorytm Adam, który jest odmianą algorytmu gradientu prostego (Stochastic Gradient Descent, SGD). Algorytm ten stosuje poniższy wzór matematyczny: Rysunek 10 Wzór algorytmu gradientu prostego gdzie: - w to wagi, które są optymalizowane, η (eta) to współczynnik uczenia się (ang. learning rate), który określa, jak duży krok robimy w kierunku minimalizacji funkcji kosztu (ang. Loss function), natomiast pozostała część to to pochodna funkcji kosztu Loss względem wag w, czyli gradient funkcji kosztu. Pokazuje kierunek i szybkość zmiany funkcji kosztu w zależności od zmian wag. Późniejszy etap obejmuje tworzenie tzw. epok. W każdej epoce, model przetwarza cały zbiór treningowy. Aby zwiększyć efektywność aktualizacji wag, zastosowane zostało porcjowanie (batch), które pozwala na przetwarzanie mniejszej ilości informacji jednocześnie. Podczas treningu, model uczy się na danych treningowych, jednakże istnieje pewne ryzyko, że nauczy się on wzorców specyficznych dla tych danych. Aby zapobiec ww. problematyce, zastosowano techniki regularyzacji takie jak Dropout oraz L2 Regularization (Ridge). Pierwsza z nich polega na losowym wyłączeniu części neuronów w warstwie podczas treningu. W omawianym programie wartość ta została ustawiona na 0.5, co oznacza, że połowa neuronów w warstwie jest wyłączana w każdej iteracji. Druga zaś, dodaje karę do funkcji straty za duże wartości wag, co ogranicza złożoność modelu i zapobiega jego przeuczeniu. Walidacja modelu jest niezbędna w procesie uczenia maszynowego. W jego trakcie program ocenia, jak dobrze model generalizuje, czyli jak radzi sobie z danymi, których nie uwzględnił podczas treningu.

Autor pracy dyplomowej, przyjął złożenie 20% / 80%, gdzie 20% danych jest wykorzystywanych do walidacji, natomiast 80% do trenowania modelu. W trakcie uczenia, stosowane są callbacki. EarlyStopping, czyli wczesne zatrzymanie uczenia maszynowego, wykorzystane jest w celu zabezpieczenia modelu przed ewentualnym przeuczeniem. Proces ten stale monitoruje wydajność modelu na zbiorze walidacyjnym i przerywa trening w przypadku braku poprawy wyników. Istnieje również callback nazwany ModelCheckpoint, który zapisuje najlepsze wagi modelu na podstawie wybranej metryki. Cały proces uczenia maszynowego, został zobrazowany poprzez interfejs graficzny użytkownika, który pojawia się po zatwierdzeniu przycisku dostępnego po przetworzeniu nagrań. Rysunek 11 Uczenie maszynowe (GUI 4) (Źródło: opracowanie własne) Gdy uczenie maszynowe dobiegnie końca, program tworzy plik o nazwie best_model.h5, w którym zapisane są wszystkie informacje opracowywane przez powyższe algorytmy. Użytkownik, otrzymuje również powiadomienie dotyczące zakończenia pracy programu. Na tym etapie, może on zweryfikować poprawność uczenia maszynowego, a także sprawdzić ile epok przetworzył program w celu utworzenia modelu. Zatwierdzając przycisk „Zakończ", użytkownik kończy pracę programu, otrzymując tym samym dostęp do wszystkich niezbędnych plików do obsługi platformy, opisanej w rozdziale czwartym. Rysunek 12 Uczenie maszynowe (GUI 4 - Zakończenie pracy) (Źródło: opracowanie własne) Działanie powyższego programu obrazuje diagram aktywności zawarty na rysunku 13. Rysunek 13 Diagram aktywności (Uczenie maszynowe) (Źródło: opracowanie własne) 4. Implementacja platformy Implementacja platformy opiera się na utworzeniu aplikacji internetowej wraz z odpowiednim połączeniem jej z modelem sztucznej inteligencji. Implementację możemy podzielić w tym przypadku na dwie części. Część frontendową, która opiera się na wdrożeniu wersji graficznej, z której będzie korzystać użytkownik, a także część backendową, która będzie obsługiwać komunikację z serwerem oraz modelem SI. 4.1. Koncepcja wizualna platformy Każda aplikacja weebowa, która będzie posiadać interfejs graficzny (GUI), powinna być uprzednio zaprojektowana przez osobę, która dobrze rozumie zasady korzystania z interfejsów aplikacji internetowych. Doświadczenie w projektowaniu platform przyjaznym użytkownikom pozwala na zdobycie szerszego grona odbiorców poprzez pozytywne odczucia z korzystania z aplikacji. Funkcjonalności wybranego oprogramowania stanowią dobrą podstawę, natomiast design może wzmocnić i uatrakcyjnić aplikację, czyniąc ją bardziej przyjazną i angażującą dla użytkowników[13]. W obecnej dobie Internetu, gdzie panuje przesyt informacji, a użytkownicy są często przebodźcowani, bardzo istotne jest utworzenie odpowiedniej warstwy graficznej. Użytkownicy zazwyczaj oceniają aplikację w ciągu kilku sekund od jej uruchomienia. Atrakcyjna i spójna oprawa graficzna może przyciągnąć uwagę odbiorcy i zachęcić go do dalszej eksploracji. Niezależnie od tego, jak zaawansowane oprogramowanie zostało zaoferowane użytkownikowi, pierwsze wrażenie bazuje na wyglądzie. Intuicyjny interfejs, który jest łatwy w nawigacji, ułatwia korzystanie z aplikacji, pozostawiając pozytywne wrażenia. Wygląd aplikacji nie tylko zwiększa jej zainteresowanie wśród odbiorców ale również pozytywnie wpływa na wzmocnienie marki, poprzez identyfikację marki z wybraną nazwą, kolorystyką czy też utworzonym logotypem[14]. W związku z powyższymi informacjami, omawiana aplikacja również posiada swój projekt wizualny, który został utworzony na początku procesu projektowego w oprogramowaniu figma, a plik projektowy został umieszczony pod nazwą „Projekt koncepcyjny.fig" oraz „Projekt koncepcyjny.pdf" w części praktycznej pracy dyplomowej. Rysunek 14 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 4.2. Mechanizmy komunikacji między frontendem a backendem Większość zaawansowanych aplikacji, korzystających z graficznego interfejsu użytkownika, a zwłaszcza aplikacje webowe, korzystają z warstwy frontendowej oraz backendowej w celu optymalizacji obsługi dużej ilości poleceń lub wykorzystania niedostępnych z poziomu wybranej warstwy narzędzi. 4.2.1. Komunikacja serwer – użytkownik Omawiana aplikacja napisana jest głównie w języku PHP, JavaScript oraz Python co wymaga stałej komunikacji pomiędzy frontendem, a backendem. Kod w języku PHP dzięki stałemu połączeniu z serwerem, może w sposób dynamiczny, kontrolować dostępem do witryny oraz zawartości, która jest wyświetlana na ekranie użytkownika. Poprzez zarządzanie witryną kod PHP umożliwia wczytanie odpowiedniego kodu CSS oraz JavaScript w wybranych zakładkach, dzięki funkcji rozpoznawania adresów URL. Oprócz aspektów wizualnych, PHP obsługuje również wczytywanie wizualizacji dla gestów, poprzez przeszukiwanie katalogów zamieszonych na serwerze w poszukiwaniu nazwy wprowadzonej w warstwie frontendowej oraz przekazanej odpowiednio przez skrypt napisany w języku JavaScript. 4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) Integracja modelu AI z aplikacją internetową jest realizowana poprzez połączenie części backendu, napisanego w języku programowania Python z częścią frontendową platformy napisaną w języku JavaScript. Wspomniana część backendowa jest zapisana w pliku o nazwie „app.py". Program ten uruchamia model sztucznej inteligencji o nazwie „best_model.h5" oraz listę etykiet „classes.npy", które

są wykorzystywane do przetwarzania danych w czasie rzeczywistym. Dane z kamery użytkownika są przetwarzane, a wyodrębnione punkty kluczowe są przekazywane do modelu, który dokonuje predykcji gestów. Wyniki te są następnie udostępniane poprzez endpointy, które zwracają szczegóły dotyczące rozpoznanych gestów w formacie JSON. Frontend aplikacji komunikuje się z backendem za pomocą zapytań http, korzystając z fetch API do wysyłania zapytań do serwera Flask. Za komunikację z serwerową częścią oprogramowania, odpowiedzialne są trzy skrypty napisane w języku JavaScript, a są nimi: „edu,js", „translator.js" oraz „sign_video_loader.js". Każdy z trzech wymienionych skryptów, korzysta z wywołań endpointów poprzez serwer lokalny o porcie 5000 „http://localhost:5000/gesture_details", które służą do pobierania informacji o rozpoznanych gestach i aktualizowania interfejsu użytkownika. W pliku app.py, endpoint dostarcza strumień wideo z kamery internetowej, który następnie jest przetwarzany przez model SI. Frontend inicjuje kamerę oraz przesyła obraz do backendu gdzie obraz jest przetwarzany, a następnie odtwarza przesyłany strumień, umożliwiając tym samym interaktywną analizę gestów w czasie rzeczywistym. 4.3. Widoki aplikacji Omawiana platforma internetowa posiada tzw. widoki, czyli warstwę frontendową, która jest odpowiednio ostylowana poprzez arkusze stylów CSS, starając się jak najdokładniej odwzorować utworzony w początkowych fazach pracy projekt wizualizacji aplikacji. Rysunek 15 Interfejs graficzny aplikacji – zakładka Home (Źródło: opracowanie własne) Platforma składa się z siedmiu widoków. Trzy pierwsze widoki odnoszą się do informacji wstępnych dotyczących realizowanego projektu. Są to zakładki „Home", „O projekcie" oraz „O nas". Każda z zakładek posiada statyczne elementy informacyjne oraz dekoracyjne, które mają zachęcić użytkownika do dalszej interakcji z platformą. Poniżej opisane zostały bardziej interesujące elementy, każdej z zakładek. Rysunek 16 Zakładka Home (Źródło: opracowanie własne) Na stronie Index.php (Home) zostały przedstawione podstawowe informacje dotyczące platformy. Zastosowany został również slider, który wyświetla logotyp Uniwersytetu Rzeszowskiego, przechodząc od prawej strony do lewej, a następnie cofając się do początku. Rysunek 17 Slider - strona Index.php (Źródło: opracowanie własne) Zakładka O projekcie (Project.php) zawiera informacje dotyczące funkcjonowania utworzonej aplikacji. Można znaleźć tam między innymi informacje o zastosowanych rozwiązaniach technologicznych, genezie powstania dysertacji, poruszanej problematyce, skrócony opis działania aplikacji, a także linię czasu, która w sposób skrótowy ukazuje etapy powstawania aplikacji. Rysunek 18 Widok zakładki O projekcie (Project.php) (Źródło: opracowanie własne) Zakładka O nas (About.php) skupia się na opisie osób zaangażowanych w powstanie projektu. W związku ze stałym rozwojem tworzonego oprogramowania oraz wysoką czasochłonnością prac, a także innymi losowymi aspektami, w projekt były zaangażowane 4 osoby. Oprócz autora oprogramowania oraz niniejszej dysertacji, w projekt zaangażował się również Dr. Inż. Wojciech Kozioł, który był pomysłodawcą tematu omawianej pracy, Dr. Inż. Bogusław Twaróg, który prowadził nadzór merytoryczny nad powstającą pracą oraz inż. Patryk Arendt, który służył jako model motion capture dla 5 gestów (tj. 500 próbek nagrań wideo). Rysunek 19 Widok zakładki O nas (About.php) (Źródło: opracowanie własne) W ramach uhonorowania wkładu, jaki włożyły ww. osoby w powstanie pracy dyplomowej, zostały one wymienione w wersji aplikacyjnej w zakładce O nas. Rysunek 20 Uhonorowanie wsparcia (About.php) (Źródło: opracowanie własne) Klikniecie przycisku „Przejdź do aplikacji" powoduje wczytanie ostatniego statycznego widoku jakim jest „Dashboard.php". W widoku tym opisane są podstawowe informacje dotyczące ilości gestów, ilości wykorzystanych próbek, sumarycznego czasu nagrań, a także wiele innych. Rysunek 21 Widok zakładki Dashboard (Dashboard.php) (Źródło: opracowanie własne) W zakładce tej przedstawiona została również animacja, która w trzech etapach prezentuje w jaki sposób program rozpoznaje sekwencje ruchów ludzkiego ciała, na przykładzie gestu z etykietą „Dzięki". Rysunek 22 Działanie systemu rozpoznawania gestu (Źródło: opracowanie własne) Pozostałe trzy widoki, odpowiadają pośrednio lub bezpośrednio za komunikację z systemem rozpoznawania gestów. Działają one na podobnej zasadzie, lecz skupiają się na innym rodzaju wykonywanego zadania przez użytkownika. Rysunek 23 Widok zakładki Tłumacz (Translator.php) (Źródło: opracowanie własne) Zakładka Tłumacz, umożliwia użytkownikowi tłumaczenie słów w języku migowym w obie strony. Może on skorzystać z funkcji udostępnienia obrazu z kamery co pozwoli na tłumaczenie języka migowego na język naturalny lub wyszukać gest w języku migowym, dostępny w systemie poprzez polecenie pisemne lub komendę głosową. W celu wyszukanie gestu języka migowego, wystarczy, że użytkownik uzupełni pole tekstowe odpowiednią nazwą gestu i zatwierdzi zmianę klikając przycisk po prawej stronie lub wciskając klawisz „Enter". Rysunek 24 Wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) Istnieje również alternatywny sposób wyszukiwania gestu w systemie. Użytkownik może skorzystać z przycisku, znajdującego się po lewej stronie okna, aby rozpocząć nasłuchiwanie mikrofonu. Po wypowiedzeniu frazy np. „Cześć", możemy

użyć polecenia głosowego „... stop szukaj”, co automatycznie zakończy nasłuchiwanie mikrofonu i rozpocznie proces szukania gestu o etykiecie „Cześć”. Nasłuchiwanie mikrofonu można także wyłączyć w sposób ręczny, klikając ponownie na ikonę mikrofonu. Wyszukiwanie głosowe, oparte zostało na polskiej wersji językowej Web Speech API dla języka JavaScript. Po aktywacji funkcji nasłuchiwania program sprawdza dostępność API w przeglądarce i konfiguruje obiekt rozpoznawania mowy. Podczas nasłuchiwania mikrofonu, program przetwarza rozpoznane fragmenty, a następnie wstawia je do pola tekstowego. Jeżeli program wykryje określoną komendę tj. „stop szukaj”, przerywa nasłuchiwanie i uruchamia proces wyszukiwania nagrania. Program obsługuje najczęściej pojawiające się błędy, a także automatycznie restartuje nasłuchiwanie, jeżeli nie zostało ono przerwane ręcznie. Rysunek 25 Głosowe wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) Gdy gest nie zostanie odnaleziony, na ekranie pojawia się stosowny komunikat. Gdy gest zostanie wyszukany poprawnie, oczom użytkownika ukaże się wizualizacja gestu odtwarzana w postaci nagrania. Przykład takowej wizualizacji został ukazany na grafice poniżej. Rysunek 26 Wizualizacja gestu (Źródło: opracowanie własne) W drugim przypadku użycia, użytkownik może udostępnić obraz z kamery, a następnie wykonać wybrany przez siebie gest. Na wykrytą postać zostanie naniesiona siatka punktów, która będzie obrazować przetwarzane dane. Program w czasie rzeczywistym, w lewym górnym rogu ekranu wyświetlać będzie prawdopodobne gesty pokazane przez użytkownika. Gdy gest zostanie wykryty wystarczającą liczbę razy, oczom użytkownika ukaże się również przycisk „Wyświetl wizualizację pokazywanych gestów”. Rysunek 27 Prezentacja gestu "Dzięki" (Źródło: opracowanie własne) Gdy użytkownik kliknie na komunikat „Wyświetl wizualizację pokazywanych gestów” po prawej stronie, automatycznie w sposób płynny wczyta się nagranie z wizualizacją gestu. Wizualizacja ta będzie się zmieniać za każdym razem, gdy program wykryje pokazanie innego gestu z odpowiednią pewnością. Rysunek 28 Synchronizacja wizualizacji z obrazem w czasie rzeczywistym (Źródło: opracowanie własne) Po przejściu na inną zakładkę, w przeglądarce internetowej klienta, śledzenie kamery zostaje zatrzymane, zarówno przez warstwę forntendową jak i backendową, a algorytmy wstrzymują swoją pracę oczekując na wznowienie połączenia. Poniżej przedstawiono diagram przypadków użycia dla zakładki Tłumacz: Rysunek 29 Diagram przypadków użycia (Tłumacz) (Źródło: opracowanie własne) Dla powyższego widoku został również przygotowany diagram aktywności, który pozwala na jeszcze bardziej precyzyjne zapoznanie się z architekturą działania oprogramowania. Poniższa grafika obrazuje proces przetwarzania obrazu z kamery użytkownika, wraz z obsługą wizualizacji, która jest aktualizowana w czasie rzeczywistym. Rysunek 30 Diagram aktywności (Tłumacz - kamera / wizualizacja) (Źródło: opracowanie własne) Zobrazowany został również diagram aktywności, odpowiedzialny za funkcję wyszukiwania wizualizacji za pomocą poleceń tekstowych oraz głosowych. Diagram UML ww. procesu znajduje się poniżej. Rysunek 31 Diagram aktywności (Tłumacz – Wizualizacja) (Źródło: opracowanie własne) Ostatnie dwa widoki aplikacji omawiają działanie zakładki „Nauka języka migowego”. Oprogramowanie, dzięki zaawansowanym algorytmom wspiera proces uczenia języka migowego. W tym celu stworzone zostały kategorie oraz „kafelki”, które umożliwiają wybór gestu, którego chce się nauczyć użytkownik. Funkcja ta skierowana jest dla osób początkujących, które dopiero rozpoczynają swoją przygodę z PJM. Rysunek 32 Widok zakładki Nauka Języka Migowego (Edu.php) (Źródło: opracowanie własne) Na stronie podstronie edu.pl możemy wybrać kategorię, która nas interesuje, a następnie odpowiedni dla nas gest. Po wybraniu odpowiedniej kategorii użytkownik będzie mógł zaobserwować zmianę w wyświetlanych elementach. Zamiast kategorii, na ekranie pojawią się gesty, które są z nią powiązane. Rysunek 33 Gesty wybranej kategorii (Źródło: opracowanie własne) Po wybraniu jednego z dostępnych gestów pojawia się nowy widok (Edu-progress.php), który wizualnie jest zbliżony do widoku tłumacza. Tutaj również możemy uruchomić podgląd z kamery, natomiast po prawej stronie odtwarzane jest w zapętleniu nagranie z wybranym przez nas gestem. W widoku tym, niedostępny jest input do wyszukiwania gestów, czy też zmiany wizualizacji. Po uruchomieniu kamery i poprawnym wykonaniu gestu, program zatrzyma udostępnianie obrazu z kamery, a użytkownik otrzyma odpowiedni komunikat na ekranie. Rysunek 34 Wykonanie poprawnego gestu (Źródło: opracowanie własne) 5. Ocena działania platformy Utworzona platforma została przetestowana pod względem działania modelu sztucznej inteligencji oraz kilku powiązanych z nią czynników. Przebadana została precyzja modeli CNN, skonstruowanych z różnych ilości próbek, w konfiguracji 100 próbek na 1 gest, 200 próbek na 1 gest oraz 300 próbek na 1 gest. Wyszczególnione zostały etapy przetwarzania danych, uczenia maszynowego oraz ogólnej pracy modelu w czasie rzeczywistym, a także na bazie przygotowanych wcześniej nagrań. Następnie, wyniki te zostały zestawione z utworzonym hybrydowym modelem CNN + LSTM. 5.1. Testy w rzeczywistych warunkach Testy związane z wydajnością w rzeczywistych warunkach, a co za tym idzie w czasie rzeczywistym są stosunkowo ciężkie do realizacji. Wpływ ma na to rozbieżność pomiędzy wykonywanymi gestami, które ciężko odtworzyć w

sposób identyczny, zróżnicowane oświetlenie, ułożenie ciała oraz wykorzystywany hardware. Istnieje wiele zmiennych, które mogą zakłócić obiektywną ocenę modelu, zwłaszcza jeżeli ten został przygotowany na niewielkiej ilości próbek. W omawianym punkcie, przedstawione zostaną zarówno obiektywne informacje, podparte dowodami, jak i subiektywne odczucia autora dysertacji. 5.1.1. Ekstrakcja punktów kluczowych Pierwszym elementem, który zostanie poddany analizie jest proces przetwarzania informacji, potrzebnych do utworzenia modelu. Cały proces uczenia maszynowego czyli ekstrakcja punktów kluczowych oraz nauka modelu odbywała się na maszynie obliczeniowej z zamontowanym procesorem AMD EPYC 7702 64-Core, który posiadał 128 procesorów wirtualnych o szybkości 2GHz, pamięcią ram 256GB oraz dysku HDD. Przetwarzanie danych w modelu CNN, zastosowane na 6 gestach po 100 próbek każdy, co daje wynik 600 próbek, trwało 19 minut w zaokrągleniu czasu do pełnych minut. Czas trwania tego samego procesu na 1200 próbkach trwało 35 minut, natomiast finalny model omawiany w dysertacji, potrzebował 54 minut na przetworzenie wszystkich danych (1800 próbek). Po zestawieniu ww. informacji w postaci wykresu, możemy zaobserwować jak wygląda złożoność czasowa przetwarzania danych. Przetwarzanie próbek w stosunku do czasu 60 54 50) n im (a 40 35 in a z r a 30 w t e 19 z r p 20 s a z C 10 0 600 próbek 1200 próbek 1800 próbek Czas w minutach 19 35 54 Ilość próbek / Czas Wykres 1 Przetwarzanie próbek CNN Uśredniając czas przetwarzania każdej próbki wynosi 1,81 sekundy. Rozbijając na poszczególne partie: przy 600 próbkach, program potrzebował 1,9 sekundy na przetworzenie jednego nagrania, 1,75 sekundy dla 1200 próbek oraz 1,8 sekundy w przypadku 1800 próbek. Zestawiając dane przetwarzane przez program wykorzystujący CNN oraz CNN w połączeniu z LSTM, nie widać jednoznacznych różnic. Program w podobnym czasie przetworzył dane zarówno dla modelu opartym o CNN jak i modelu hybrydowym, przy czym mniej złożony model zakończył proces ponad 2 minuty wcześniej. Rysunek 35 Zestawienie metody CNN oraz CNN + LSTM (Źródło: opracowanie własne) Czas przetwarzania danych (1800 próbek) CNN+LSTM 56 u le d o m ja z d o R CNN 54 53 54 54 55 55 56 56 57 Czas przetwarzania (min) Wykres 2 Czas przetwarzania danych – porównanie CNN – CNN + LSTM 5.1.2. Uczenie modelu Uczenie modelu również zostało porównane w identycznych kryteriach. Z otrzymanych informacji wynika, że czas uczenia maszynowego nie jest stricte związany z ilością próbek. Funkcja przerywania uczenia, która za zadanie ma chronić model przed przeuczeniem, skutecznie skraca czas potrzebny do stworzenia modelu. Program kończy pracę za każdym razem gdy wykryje, że precyzja modelu nie poprawia się co skutkuje utworzeniem modelu w krótszym czasie niżeli z potencjalnie mniejszą ilością próbek. Uczenie maszynowe w stosunku do czasu 140 124 120)s (u 100 le d 76 o m 80 a in 60 e z c u s 40 30 a z C 20 0 600 próbek 1200 próbek 1800 próbek Czas w sekundach 30 124 76 Ilość próbek / Czas Wykres 3 Uczenie modelu CNN O ile nie było widocznej różnicy pomiędzy przetwarzaniem danych pomiędzy modelem wykorzystującym CNN, a modelem hybrydowym CNN + LSTM, o tyle w przypadku uczenia modelu różnica jest bardzo zauważalna. Model z mniejszą ilością warstw uczył się przez 1 minutę i 16 sekund, natomiast model z przetwarzaniem danych wstecz, trenował się przez 6 godzin, 9 minut i 36 sekund. Różnica w czasie nauczania modelu jest ogromna, a co za tym idzie model hybrydowy z zastosowaniem LSTM wydaje się mało wydajnym rozwiązaniem, zważywszy na małą ilość przetwarzanych danych. Program oparty na mniejszej ilości warstw uczył się ze stosunkowo wysoką precyzją, która oscylowała w zakresie 91-95%, oraz 81-88% podczas walidacji, co obrazuje poniższa ilustracja. Rysunek 36 Proces uczenia maszynowego CNN (Źródło: opracowanie własne) Druga wersja programu zatytułowana roboczo 2.1, wykazywała wyższą precyzję uczenia podczas kilku prób uczenia modelu. Dokładność modelu oscylowała pomiędzy 97-99%, a podczas walidacji wynik ten wynosił 93-99%. Wyniki te sugerują, że model utworzony za pomocą CNN oraz LSTM radzi sobie znacznie lepiej niżeli poprzednia wersja. Należy zwrócić uwagę również na wielkość pliku docelowego. Model utworzony w wersji 1.0 zajmuje 25,8 MB miejsca na dysku, natomiast model w wersji 2.1 zajmuje 1,26 GB. Kwestia ta staje się problematyczna, gdyż model reprezentujący niewiele niższą precyzję podczas uczenia 5 maszynowego, zajmuje 50 razy mniej miejsca na dysku, co staje się kuszącym rozwiązaniem w przypadku chęci zaoszczędzenia miejsca w przestrzeni dyskowej. Rysunek 37 Proces uczenia maszynowego CNN + LSTM (Źródło: opracowanie własne) Czas przetwarzania danych (1800 próbek) CNN+LSTM 22176 u le d o m ja z d o R CNN 76 0 5000 10000 15000 20000 25000 Czas przetwarzania (s) Wykres 4 Czas uczenia modelu – porównanie CNN – CNN + LSTM 5.1.3. Testy w czasie rzeczywistym Model z założenia miał zostać przeznaczony do platformy obsługującej rozpoznawanie polskiego języka migowego w czasie rzeczywistym. W związku z powyższą informacją, również ten aspekt został przetestowany. Niestety w związku z wieloma czynnikami losowymi, nie da się zmierzyć precyzyjnie różnicy pomiędzy rozpoznawaniem gestów w czasie rzeczywistym pomiędzy modelem utworzonym w wersji podstawowej oraz

hybrydowej. 5 Jednakże dokonano pewnych obserwacji, które sugerują wykorzystanie jednego spośród dwóch modeli. W przypadku modelu podstawowego, nie zaobserwowano problemów. Model działał stabilnie, precyzja wykrywanych gestów była zadowalająca. W przypadku modelu hybrydowego, uruchomionego w warunkach czasu rzeczywistego zaobserwowano problemy. Model działał niestabilnie, rejestrując ciągłe „przycięcia" obrazu. Zachodziło tam zjawisko klatkowania, a dokładna przyczyna została opisana w punkcie 5.3. 5.2. Skuteczność i dokładność rozpoznawania gestów Każdy z utworzonych modeli został również przetestowany w warunkach odpowiednio wcześniej przygotowanych. Specjalnie zaprojektowany program, odtwarzał trzy nagrania których nie było zamieszonych w próbkach modelu, dla każdego spośród 6 istniejących gestów. Następnie za pomocą dostarczonych modeli klasyfikował wykryte gesty. Ocenie została poddana precyzja rozpoznanych gestów z podziałem procentowym. Spośród testowanych gestów jeden z nich otrzymał najgorszy wynik, rozpoznając tym samym inny gest. Pozostałe tj. 5/6 gestów zostało rozpoznanych poprawnie. Zestawienie wykresów wraz z uśrednieniem wyników, zostało zaprezentowane na wykresie 5. 5 Wykres 5 Zestawienie wykresów Uśredniając wyniki spośród trzech testów, gest „Cześć" został rozpoznany z precyzją 60,70%, program wykrył również możliwość zaistnienia gestu „Dzięki" z wynikiem 38,69% oraz „Prosić" (0,58%). Program podczas analizy nagrań wykrył 2 z 3 wskazanych gestów poprawnie. Podsumowanie analizy gestu „Cześć" zostało przedstawione na poniższym wykresie. 5 Wykres 6 Analiza gestu Cześć - CNN Porównując wyniki z modelem CNN + LSTM, lepiej prezentuje się model CNN. Model składający się z dodatkowej warstwy, błędnie wykrył gest dzięki z precyzją 60,33%, oraz cześć z precyzją 39,67%. 5 W przypadku gestu „Dzięki" model nieprecyzyjnie rozpoznał wskazywany gest. Jedynie na drugiej próbce gest dzięki miał wyższe prawdopodobieństwo wystąpienia niżeli inne opcje. Program błędnie rozpoznał, że wskazanym modelem jest gest Proszę z prawdopodobieństwem 66,13%, drugim możliwym gestem było dzięki z wynikiem 33,44%. Dodatkowo program dostrzegł podobieństwo z gestem prosić (0,24%), Cześć (0,17%) oraz Przepraszać (0,02%). Wykres 7 Analiza gestu Dzięki – CNN W powyższym przypadku lepszy okazał się model z dodatkową warstwą, prezentując następujące wyniki: dzięki (67,62%), proszę (32,11%), cześć (0,26%). 5 W przypadku gestu „Dziękuję" nie było minimalnych wątpliwości. Gest ten został wykryty z precyzją aż 99,98%. Pozostałe 0.02% zostało podzielone po równo pomiędzy gestem „dzięki" i „Cześć". Wykres 8 Analiza gestu Dziękuję - CNN W powyższym przypadku, znacznie lepsza okazała się sieć CNN. Druga sieć wykryła gest: dziękuję (65,22%), dzięki (32,16%), przepraszać (2,38%), cześć (0,18%), prosić (0,05%), proszę (0,01%). 5 Czwarty badany gest (Prosić) został rozpoznany poprawnie z precyzją 75,69%, pozostałe wartości zostały podzielone pomiędzy: cześć (22,34%), proszę (1,88%), dziękuję (0,06%) oraz dzięki (0,03%) Wykres 9 Analiza gestu Prosić – CNN Gest prosić został wykryty poprawnie w każdej konfiguracji sieci, lecz sieć LSTM wykryła gest z większą precyzją, zwracając wynik precyzji: prosić (99,74%), dziękuję (0,24%), dzięki (0,02%), przepraszać (0,01%). 5 Piąty gest, tak jak gest trzeci, został rozpoznany z bardzo wysoką precyzją (99,91%). Pozostałe prawdopodobieństwo zostało podzielone jedynie przez dzięki (0,07%) oraz prosić (0,02%). Wykres 10 Analiza gestu Proszę – CNN W powyższym przypadku sieć CNN okazała się lepsza niżeli LSTM. Druga z wymienionych sieci wykryła gesty: proszę (77,73%), dzięki (21,90%), cześć (0,35%), prosić oraz przepraszać (0,01%). 5 Ostatnim badanym gestem jest gest przepraszać. Gest ten został sumarycznie wykryty z precyzją 49%, możliwe było również wykrycie etykiety dzięki, która miała również wysoki wynik (43,71%). Analizując wyniki zauważyć można także proszę (7,07) oraz cześć (0,22%). Wykres 11 Analiza gestu Przepraszać – CNN Model LSTM wykrył odpowiednio gesty: przepraszać (44,11%), dzięki (38,40%), proszę (17,01%), cześć (0,23%), dziękuję (0,14%) oraz prosić (0,10%). Ogólny wynik precyzji przetwarzanych gestów wskazuje na to, iż model CNN jest bardziej precyzyjny (sumarycznie 417,72%) niżeli model hybrydowy CNN + LSTM (sumarycznie 394,09%). 60 5.3. Analiza błędów i propozycje ulepszeń Platforma, a co za tym idzie również model SI posiada nie tylko swoje zalety ale również wady. W pierwszej kolejności należy wymienić warstwę backendową. Na przykładzie omawianej dysertacji udowodniono, że model hybrydowy w postaci CNN + LSTM, niekoniecznie działa poprawnie z przetwarzaniem obrazów w czasie rzeczywistym. LSTM jest przeznaczony głównie do pracy z sekwencjami danych, które mają wyraźny związek czasowy, takimi jak teksty lub sygnały czasowe. W przypadku obrazu, dane są przestrzenne, a nie czasowe. Do przetwarzania danych obrazowych lepiej nadają się sieci konwolucyjne (CNN), które są zaprojektowane specjalnie do pracy z danymi w formie siatek (takich jak obrazy). Zwiększenie złożoności, w tym przypadku wpłynęło negatywnie na działanie programu. Model wykorzystujący LSTM potrzebował znacznie więcej czasu na przetworzenie sekwencji, co skutkowało zacinaniem się obrazu z kamery oraz powolną predykcją gestów. Precyzja w przetwarzaniu statycznym, również nie została podniesiona. Zaobserwować można było spadek precyzji modelu na danych testowych. W związku z dokonaną analizą, nie zaleca się wykorzystywania LSTM do przetwarzania obrazu w czasie rzeczywistym. Model

CNN, można jednakże rozwinąć o podejście hybrydowe CNN + HMM, które polecają autorzy publikacji „Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition.”. Zastosowanie powyższego rozwiązania potencjalnie mogłoby podnieść poziom precyzji modelu, bez negatywnego wpływu na stabilność programu. Możliwe jest także dodanie Transformatorów takich jak ST- Transformer, które są aktualnym rozwiązaniem na rok 2024. Dodatkowo, model mógłby być wzbogacony o większą ilość próbek, wliczając w to zróżnicowanych modeli motion capture, różne warunki oświetlenia, różnego typu kamery oraz zmienne tła otoczenia. Powyższe elementy z pewnością poprawiłyby precyzję uczenia modelu. Utworzona platforma, mogłaby również zostać ulepszona poprzez dodanie elementów frontendowych, takich jak responsywność, a także możliwości obsługi wielu języków. Posiada ona również przygotowany moduł, który można rozwinąć o logowanie oraz rejestrację, wprowadzając tym samym system abonamentów (tokenów), znany chociażby z konkurencyjnych aplikacji SI. 61 Podsumowanie Platformy edukacyjne stale cieszą się popularnością. W grupie ww. platform znajdują się aplikacje do nauki języków obcych z których każdego dnia korzystają setki tysięcy osób, a ich stały rozwój pozwala na zwiększenie komunikatywności w środowisku międzynarodowym. Sytuacja ta jest analogiczna w przypadku zwiększenia świadomości ludzkiej na wszelkiego typu dysfunkcje organizmu. Strony internetowe coraz częściej dostosowują się do różnego rodzaju dyrektyw, chociażby takiej jak WCAG 2.0, które to w jasny sposób określają jak powinny wyglądać strony przyjazne dla osób z niepełnosprawnościami. Pomimo zwiększenia dostępności stron internetowych dla osób z dysfunkcjami, nadal istnieje problem wykluczenia społecznego w warunkach kontaktu bezpośredniego. Osoby, które są głuchonieme, potrzebują obecności tłumacza, który stale przekazuje informacje pomiędzy osobami pełnosprawnymi, a osobą dotknięta dysfunkcją. Niniejsza dysertacja łączy możliwości platform internetowych oraz tematyki niepełnosprawności, tworząc prototyp platformy do rozpoznawania polskiego języka migowego. Pomimo istnienia takowych aplikacji na rynku anglojęzycznym, nadal nie istnieje stabilne rozwiązanie na polskim rynku. W związku z powyższą informacją, praca ta jest innowacyjna, łącząc ze sobą prostotę obsługi, z zaawansowanymi technologiami takimi jak model sztucznej inteligencji rozpoznający gesty języka migowego w czasie rzeczywistym. W ramach niniejszej pracy dyplomowej, opracowany został projekt platformy internetowej, wraz z jej pełną implementacją przy zastosowaniu języków webowych. Dodatkowo na potrzeby dysertacji, autor utworzył kilka modeli sztucznej inteligencji, składających się z różnej ilości próbek opracowanych przez twórcę. Następnie każdy model został przetestowany pod względem wydajność oraz możliwości wykorzystania ich w aplikacji korzystającej z kamery internetowej w czasie rzeczywistym. Finalnie, utworzonych zostało 1800 nagrań o długości 4 sekund, przedstawiających osobę, wykonującą określony gest polskiego języka migowego. Najlepszym modelem okazał się program oparty na architekturze CNN, który wykazał się wysoką precyzją oraz szybkością działania. Należy również uwzględnić, że wszystkie elementy estetyczne w tym grafiki, wizualizacje oraz wykresy zostały opracowane przez autora pracy. Praca dyplomowa, może również zostać rozwinięta o dodatkowe elementy, takie jak responsywność oraz moduł logowania i rejestracji, pozwalający na wykorzystanie ww. dysertacji w celach materialnych. Sugerowane jest również wzbogacenie opracowanego 62 modelu o większą ilość próbek, opracowanych w różnych warunkach oświetleniowych, zmiennym otoczeniu, a także na innych osobach. Przebudowa modelu, dodając warstwę architektury HMM lub ST-Transformer również może zwiększyć precyzję wykrywanych gestów, a co za tym idzie poprawić działanie całej platformy edukacyjno-translatorycznej. 63 Bibliografia oraz Netografia 1. Camgoz, N. C., Koller, O., Hadfield, S., & Bowden, R. (2017). SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition. IEEE International Conference on Computer Vision (ICCV). 2. Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 7291-7299. 3. Huang, J., Zhou, W., Zhang, Q., Li, H., & Li, W. (2018). Attention-Based 3D-CNNs for Large- Vocabulary Sign Language Recognition. IEEE Transactions on Circuits and Systems for Video Technology, 29(9), 2822-2832. 4. Jamroży D.(2023): Aplikacja internetowa z graficznym interfejsem użytkownika opartym na technologii typu eye-tracking oraz speech recognition. [Praca inżynierska, Uniwersytet Rzeszowski] 5. Koller, O., Zargaran, S., Ney, H., & Bowden, R. (2015). Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition. In Proceedings of the British Machine Vision Conference (BMVC). 6. --- MediaPipe Documentation (n.d.). Google Developers. Retrieved from https://ai.google.dev/edge/mediapipe/solutions/guide?hl=pl 7. Pandey, A., Mishra, M., & Verma, A. K. (2022). Real-Time Sign Language Recognition Using Machine Learning and Neural Networks. IEEE Access. 8. Patel, K., Gil-González, A.-B., & Corchado, J. M. (2022). Deepsign: Sign Language Detection and Recognition Using Deep Learning. Electronics, 11(11), 1780. 9. Patel, M., & Shah, N. (2021). Real-Time Gesture-Based Sign

Language Recognition System. IEEE International Conference on Information Technology and Engineering (ICITE). 10. Pigou, L., Dieleman, S., Kindermans, P. J., & Schrauwen, B. (2015). Sign Language Recognition Using Convolutional Neural Networks. European Conference on Computer Vision Workshops (ECCVW). 11. Zhang, Z., & Liu, C. (2020). Skeleton-Based Sign Language Recognition Using Whole-Hand Features. IEEE Access, 8, 68827-68837. 12. https://www.wsb-nlu.edu.pl/pl/wpisy/wplyw-sztucznej-inteligencji-na-przyszlosc-pracy- nowe-perspektywy-i-wyzwania (22.08.2024) 64 13. https://www.tamoco.com/blog/blog-app-design-app-functionality-ux-ui/ (22.08.2024) 14. https://echoinnovateit.com/ux-or-ui-whats-more-important-in-app-development/ (22.08.2024) 15. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5(4), 115-133. 65 Spis ilustracji, tabel oraz wykresów Spis Ilustracji Rysunek 1 Porównanie technologii (Źródło: opracowanie własne) 11 Rysunek 2 Interfejs programu XAMPP 20 Rysunek 3 Etapy zbierania próbek (Źródło: opracowanie własne) 22 Rysunek 4 Uczenie maszynowe (GUI 1) (Źródło: opracowanie własne) 22 Rysunek 5 Uczenie maszynowe (GUI 2) (Źródło: opracowanie własne) 23 Rysunek 6 Uczenie maszynowe (GUI 2 - Struktura) (Źródło: opracowanie własne) 24 Rysunek 7 Uczenie maszynowe (GUI 3) (Źródło: opracowanie własne) 24 Rysunek 8 Uczenie maszynowe (GUI 3 - Wizualizacja) (Źródło: opracowanie własne) 25 Rysunek 9 Uczenie maszynowe (GUI 3 - Zakończenie przetwarzania danych) (Źródło: opracowanie własne) 26 Rysunek 10 Wzór algorytmu gradientu prostego 28 Rysunek 11 Uczenie maszynowe (GUI 4) (Źródło: opracowanie własne) 29 Rysunek 12 Uczenie maszynowe (GUI 4 - Zakończenie pracy) (Źródło: opracowanie własne) .. 30 Rysunek 13 Diagram aktywności (Uczenie maszynowe) (Źródło: opracowanie własne) 31 Rysunek 14 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 33 Rysunek 15 Interfejs graficzny aplikacji – zakładka Home (Źródło: opracowanie własne) 35 Rysunek 16 Zakładka Home (Źródło: opracowanie własne) 35 Rysunek 17 Slider - strona Index.php (Źródło: opracowanie własne) 36 Rysunek 18 Widok zakładki O projekcie (Project.php) (Źródło: opracowanie własne) 36 Rysunek 19 Widok zakładki O nas (About.php) (Źródło: opracowanie własne) 37 Rysunek 20 Uhonorowanie wsparcia (About.php) (Źródło: opracowanie własne) 38 Rysunek 21 Widok zakładki Dashboard (Dashboard.php) (Źródło: opracowanie własne) 38 Rysunek 22 Działanie systemu rozpoznawania gestu (Źródło: opracowanie własne) 39 Rysunek 23 Widok zakładki Tłumacz (Translator.php) (Źródło: opracowanie własne) 39 Rysunek 24 Wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) 40 Rysunek 25 Głosowe wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) 40 Rysunek 26 Wizualizacja gestu (Źródło: opracowanie własne) 41 Rysunek 27 Prezentacja gestu "Dzięki" (Źródło: opracowanie własne) 42 66 Rysunek 28 Synchronizacja wizualizacji z obrazem w czasie rzeczywistym (Źródło: opracowanie własne) 42 Rysunek 29 Diagram przypadków użycia (Tłumacz) (Źródło: opracowanie własne) 43 Rysunek 30 Diagram aktywności (Tłumacz - kamera / wizualizacja) (Źródło: opracowanie własne) 44 Rysunek 31 Diagram aktywności (Tłumacz – Wizualizacja) (Źródło: opracowanie własne) 45 Rysunek 32 Widok zakładki Nauka Języka Migowego (Edu.php) (Źródło: opracowanie własne) 46 Rysunek 33 Gesty wybranej kategorii (Źródło: opracowanie własne) 46 Rysunek 34 Wykonanie poprawnego gestu (Źródło: opracowanie własne) 47 Rysunek 35 Zestawienie metody CNN oraz CNN + LSTM (Źródło: opracowanie własne) 49 Rysunek 36 Proces uczenia maszynowego CNN (Źródło: opracowanie własne) 51 Rysunek 37 Proces uczenia maszynowego CNN + LSTM (Źródło: opracowanie własne) 52 Spis Tabel Tabela 1 Wymagania systemowe cz1 18 Tabela 2 Wymagania systemowe cz2 18 Spis Wykresów Wykres 1 Przetwarzanie próbek CNN 49 Wykres 2 Czas przetwarzania danych – porównanie CNN – CNN + LSTM 50 Wykres 3 Uczenie modelu CNN 50 Wykres 4 Czas uczenia modelu – porównanie CNN – CNN + LSTM 52 Wykres 5 Zestawienie wykresów 54 Wykres 6 Analiza gestu Cześć - CNN 55 Wykres 7 Analiza gestu Dzięki – CNN 56 Wykres 8 Analiza gestu Dziękuję - CNN 57 Wykres 9 Analiza gestu Prosić – CNN 58 Wykres 10 Analiza gestu Proszę – CNN 59 Wykres 11 Analiza gestu Przepraszać – CNN 60 67 Streszczenie: Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego. Niniejsza dysertacja dotyczy utworzenia prototypu platformy do rozpoznawania polskiego języka migowego (PJM) w czasie rzeczywistym poprzez zastosowanie metod uczenia maszynowego, których efektem jest model sztucznej inteligencji (SI). Głównym celem pracy było opracowanie platformy, która umożliwi osobom niesłyszącym łatwiejszą komunikację z otoczeniem bez potrzeby korzystania z dodatkowego sprzętu czy też usług tłumacza, a także zwiększenie ludzkiej świadomości na temat dysfunkcji organizmu. W pracy dokonano również przeglądu technologii i metod związanych z rozpoznawaniem gestów, takich jak konwolucyjne sieci neuronowe (CNN) oraz narzędzia przetwarzania obrazu, między innymi MediaPipe i OpenPose. Autor dysertacji, zaprojektował oraz przeanalizował modele SI, które rozpoznawały gesty PJM na podstawie danych wideo, przetwarzanych oraz trenowanych w modelu CNN. W dalszej części prac, opracowano aplikację webową, która korzysta z

kamery internetowej w celu rozpoznania gestów użytkownika w czasie rzeczywistym. Przeprowadzone testy, zarówno w czasie rzeczywistym jak i na przygotowanych wcześniej danych wykazały, że aplikacja działa skutecznie, choć istnieją pewne wyzwania związane z precyzją rozpoznawania gestów, wliczając w to ograniczony dostęp do bazy gestów. Utworzona platforma oferuje potencjalne rozwiązania, które mogą wspierać osoby z dysfunkcjami słuchu oraz mowy w codziennej komunikacji. 68 Abstract: Prototype and Analysis of a Real-time Polish Sign Language Gesture Recognition Platform Using Machine Learning Method. This dissertation concerns the creation of a prototype of a platform for real-time recognition of Polish Sign Language (PJM) through the use of machine learning methods, which result in an artificial intelligence (AI) model. The main objective of the work was to develop a platform that would enable deaf people to communicate more easily with their surroundings without the need for additional equipment or interpreter services, as well as to increase human awareness of body dysfunctions. The paper also reviews technologies and methods related to gesture recognition, such as convolutional neural networks (CNNs) and image processing tools, including MediaPipe and OpenPose. Author of the dissertation, he designed and analyzed AI models that recognized PJM gestures based on video data processed and trained in a CNN model. Further on, a web application was developed that uses a webcam to recognize user gestures in real time. Tests, both in real time and on pre-prepared data, have shown that the application works effectively, although there are some challenges related to the precision of gesture recognition, including limited access to the gesture database. The created platform offers potential solutions that can support people with hearing and speech impairments in everyday communication. 69 70

Fragmenty innego stylu

UNIWERSYTET RZESZOWSKI Kolegium Nauk Przyrodniczych Damian Jamroży Nr albumu: 113729 Kierunek: Informatyka Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego Praca magisterska Praca wykonana pod kierunkiem Dr. Inż. Bogusława Twaroga Rzeszów, 2024 Pragnę serdecznie podziękować Panu dr inż. Bogusławowi Twarogowi za nieocenione wsparcie oraz życzliwość okazywaną na każdym etapie mojej edukacji. Jego pomoc i zaangażowanie miały kluczowy wpływ nie tylko na proces powstawania niniejszej pracy magisterskiej, ale także na moją pracę inżynierską i cały tok studiów. Bez Jego merytorycznej wiedzy oraz wsparcia, ukończenie tych etapów byłoby znacznie trudniejsze. Jednocześnie chciałbym serdecznie podziękować moim przyjaciołom z UCI UR. To dzięki ich namowom i wsparciu udało mi się zrealizować ww. etap życia. Spis treści Wstęp 7 Cel i zakres pracy 8 1. Przegląd literatury i analiza istniejących rozwiązań 9 1.1. Przegląd technologii rozpoznawania gestów 9 1.2. Przegląd metod uczenia maszynowego 12 1.3. Wyzwania w rozpoznawaniu gestów języka migowego 13 1.4. Analiza problematyki oraz istniejących aplikacji dotyczących języka migowego 15 2. Techniczne aspekty funkcjonowania aplikacji 17 2.1. Wymagania systemowe 17 2.2. Narzędzia modelowania sztucznej inteligencji 19 3. Budowa modelu rozpoznawania gestów 21 3.1. Przygotowanie danych wejściowych 21 3.2. Preprocessing danych 24 3.3. Architektura modelu oraz algorytm klasyfikacji 26 3.4. Trening, walidacja oraz ocena skuteczności modelu 27 4. Implementacja platformy 32 4.1. Koncepcja wizualna platformy 32 4.2. Mechanizmy komunikacji między frontendem a backendem 33 4.2.1. Komunikacja serwer – użytkownik 33 4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) 34 4.3. Widoki aplikacji 34 5. Ocena działania platformy 48 5.1. Testy w rzeczywistych warunkach 48 5.1.1. Ekstrakcja punktów kluczowych 48 5.1.2. Uczenie modelu 50 .1.3. Testy w czasie rzeczywistym 52 5.2. Skuteczność i dokładność rozpoznawania gestów 53 5.3. Analiza błędów i propozycje ulepszeń 61 Podsumowanie 62 Bibliografia oraz Netografia 64 Spis ilustracji, tabel oraz wykresów 66 Wstęp Obecny rozwój technologii pozwala na automatyzację wielu procesów, w tym procesów finansowych, administracyjnych oraz sprzedażowych. Wymienione elementy są związane z obsługą biznesów, które umożliwiają komercyjny zarobek twórcom oprogramowania. Algorytmy, mają w tym przypadku ułatwić pracę lub całkowicie zastąpić osoby fizyczne w ich obowiązkach. Dzięki takim praktykom, pracodawcy, czy też różnego rodzaju organizacje, zwiększają swoje dochody poprzez przyspieszenie wykonywanej pracy lub w gorszym przypadku, oszczędzają fundusze poprzez zredukowanie etatów. Szerokie zastosowanie sztucznej inteligencji jest widoczne w każdym aspekcie naszego życia. Coraz większa ilość sklepów, banków czy też producentów wszelkiego rodzaju produktów, decyduje się na wdrażanie sztucznej inteligencji. Zgodnie z opinią specjalistów z Wyższej Szkoły Biznesu National-Louis University, AI (Artificial Intelligence) może powodować utratę miejsc pracy oraz restrukturyzację zawodów, jednakże tym samym może zwiększać zapotrzebowanie na specjalistów w branży kreatywnej oraz IT. Autor artykułu, zwraca uwagę na problematykę dotyczącą etyki związanej ze sztuczną inteligencją oraz potrzebę nieustannej nauki i rozwoju[12]. Istnieją również aspekty SI (Sztucznej Inteligencji), które są niezaprzeczalnie pozytywne, chociażby zastosowane w strefach pożytku publicznego, czy też w rozwiązaniach dla osób z dysfunkcjami. Wszelkiego rodzaju protezy, pojazdy, syntezatory mowy, algorytmy analizujące tekst, dźwięk czy też obraz, pozwalają na łatwiejsze funkcjonowanie osób niepełnosprawnych. Niestety obszary te są często pomijane, ze względu na stosunkowo niewielkie grono odbiorców, gotowych zapłacić za wprowadzenie takowych rozwiązań. Podejmując dalszą próbę analizy problemu, możemy zaobserwować wysokie zainteresowanie wadami wzroku oraz problemami ruchowymi, natomiast niskie zainteresowanie dysfunkcją głosową w odniesieniu do osób głuchoniemych. Istnieje wiele rozwiązań, które ułatwiają kontakt wzrokowy, chociażby takie jak regulacja wielkości czcionek we wszelkiego rodzaju aplikacjach, asystenci głosowi, czy też operacyjne korekty wzroku. Funkcje ruchowe, wspierane są przez różnorodne protezy, pojazdy, a także specjalne miejsca dostosowane do ich potrzeb np. miejsca parkingowe. Niestety w odniesieniu do problematyki osób głuchoniemych, nie ma zbyt wielu rozwiązań technologicznych, które mogą ułatwić ich życie w sposób nieinwazyjny. Cel i zakres pracy Celem pracy dyplomowej jest wparcie ww. grupy docelowej poprzez utworzenie oraz analizę oprogramowania do rozpoznawania sekwencji ruchów w czasie rzeczywistym, w oparciu o metody uczenia maszynowego. Zastosowane rozwiązania, wykorzystane zostaną następnie w aplikacji służącej do nauki oraz tłumaczenia polskiego języka

migowego. Oprogramowanie to może służyć jako wsparcie osób z dysfunkcjami w łatwiejszej adaptacji w środowisku osób pełnosprawnych. Aplikacja ta, również może działać w sposób edukacyjny, zwiększając świadomość użytkowników na temat dysfunkcji słuchowych oraz głosowych. Opracowanie autorskich skryptów, pozwalać ma na obsługę pełnego procesu pracy aplikacji, począwszy od rejestrowania próbek nagrań wideo, uczenia modelu SI, testowaniu jego poprawności, a kończąc na obsłudze aplikacji internetowej[4]. Niniejsza dysertacja, składa się z pięciu rozdziałów. Pierwszy z nich omawia zagadnienia teoretyczne związane z funkcjonowaniem aplikacji, takie jak przegląd dostępnych technologii związanych z rozpoznawaniem gestów oraz uczeniem maszynowym, wyzwaniami w rozpoznawaniu gestów oraz analizą istniejących platform dotyczących języka migowego. Następny rozdział omawia aspekty techniczne, które muszą zostać spełnione aby aplikacja działała w pełni poprawnie. W powyższym rozdziale znajdują się takie elementy jak: wymagania systemowe, zalecane oprogramowanie zewnętrzne oraz biblioteki, które należy zainstalować na urządzeniu docelowym. Rozdział trzeci porusza tematykę tworzenia modelu sztucznej inteligencji, przechodząc po każdym etapie jego powstawania, zaczynając od przygotowaniu danych wejściowych, a kończąc na ocenie skuteczności stworzonego modelu. Rozdział czwarty, zatytułowany „Implementacja platformy" skupia się na połączeniu aspektów sztucznej inteligencji wraz z aplikacją internetową. Zaprezentowane tam informacje dotyczą koncepcji wizualnej platformy, integracji modeli AI z aplikacją webową, mechanizmów komunikacji między klientem a serwerem oraz warstwy graficznej interfejsu użytkownika. Piąty, a zarazem ostatni rozdział niniejszej dysertacji omawia ocenę działania platformy, skupiając się na testach w warunkach rzeczywistych oraz symulowanych, ocenie skuteczności pracy aplikacji oraz propozycji ulepszeń oprogramowania. Praca zakończona została podsumowaniem, w którym zaprezentowano ogólny opis aplikacji wraz z jej analizą pod kątem przydatności oraz wydajności w rzeczywistych warunkach. 1. Przegląd literatury i analiza istniejących rozwiązań Rozpoznawanie wszelkiego rodzaju obiektów oraz ich właściwości, zyskały w ostatnich latach na popularności, głównie dzięki postępom w technikach uczenia maszynowego oraz rozwoju głębokich sieci neuronowych. W niniejszym rozdziale, omówione zostaną najważniejsze technologie oraz metody stosowane w tematyce rozpoznawania gestów, ze szczególnym uwzględnieniem języka migowego. 1.1. Przegląd technologii rozpoznawania gestów Technologie rozpoznawania gestów opierają się na stałej analizie ruchów ciała, ze szczególnym uwzględnieniem dłoni i palców, przy pomocy różnych sensorów oraz kamer. Systemy te wykorzystują zarówno dane wideo, jak i informacje trójwymiarowe, uzyskane za pomocą czujników głębi. Interdyscyplinarna dziedzina badań rozpoznawania gestów, łączy ze sobą elementy przetwarzania obrazów, komputerowego widzenia, sztucznej inteligencji oraz interakcji człowiek-komputer. Podstawą rozpoznawania gestów jest przetwarzanie obrazu, które obejmuje szereg technik służących do analizy danych wizualnych. Można wyłonić takie etapy jak: detekcja obrazów, śledzenie ruchu oraz ekstrakcja cech. Detekcja obrazów opiera się na rozpoznaniu dłoni oraz pozostałych istotnych części ciała na przetwarzanym obrazie. Techniki te bazują na algorytmach segmentacji obrazu, które identyfikują obiekty na podstawie ich koloru, kształtu lub ruchu. Algorytmy takie jak YOLO (You Only Look Once) oraz SSD (Single Shot Multibox Detector) są powszechnie stosowane do szybkiej i dokładnej detekcji dłoni w badanych obrazach. Śledzenie ruchu, opiera się na analizie każdej klatki nagrania w celu uchwycenia przesunięć, występujących w punktach wykrytych przez poprzedni proces. W tym celu stosuje się miedzy innymi algorytmy KLT (Kanade-Lucas-Tomasi) oraz Mean Shift, które pozwalają na dokładne monitorowanie trajektorii ruchu dłoni. Ostatnią techniką jest ekstrakcja cech, która umożliwia dokonanie wyodrębnienia z obrazów stawów oraz kości, niezbędnych do identyfikacji wykonanego gestu. W tym celu stosowane są takie narzędzia jak OpenPose oraz MediaPipe[3]. Biblioteka OpenPose została utworzona przez Perceptual Lab na Uniwersytecie Carnegie Mellon (CMU), pod nadzorem profesora Asuyuki Matsumoto oraz doktoranta Zhe Cao, który jest głównym autorem pracy badawczej, opisującej możliwości biblioteki OpenPose. Pierwsza wersja OpenPose została opublikowana w 2017 roku, a jej wyniki były oparte na pracy naukowej pt. "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", której współautorem był wspomniany wcześniej Zhe Cao. Praca ta po raz pierwszy zaprezentowana została na konferencji CVPR (Computer Vision and Pattern Recognition) w 2017 roku. Początkowo OpenPose służył głównie do analizy oraz detekcji pozycji wielu osób w czasie rzeczywistym na obrazach 2D. Pierwsza wersja koncentrowała się na problemie, który dotąd stanowił duże wyzwanie, a mianowicie detekcji pozycji ciała wielu osób na jednym obrazie, w jak najszybszym czasie, z zachowaniem wysokiej precyzji. Istniejące wcześniej rozwiązania, były skoncentrowane głównie na wykryciu jednego człowieka lub działały wolniej, co uniemożliwiało zastosowanie ich w czasie rzeczywistym. Biblioteka ta zyskała spore zainteresowanie, co przyczyniło się do rozszerzenia jej o dodatkowe funkcje, takie jak detekcja szczegółowych ruchów dłoni i mimiki twarzy oraz analizy 3D. Głównym zadaniem obecnej wersji biblioteki jest wykrywanie oraz śledzenie pozycji ciała

człowieka w obrazie 2D oraz 3D, uwzględniając różne części ciała takie jak kończyny, głowa oraz palce rąk i stóp. Podstawą działania OpenPose są techniki głębokiego uczenia, a dokładniej konwolucyjne sieci neuronowe (CNN), które za pomocą punktów kluczowych (tzw. keypoints), rozpoznają położenie ciała, dzieląc je na odpowiednie struktury. W początkowej fazie programu, biblioteka przeprowadza detekcję punktów kluczowych, następnie tworzy mapę ufności, która określa z jakim prawdopodobieństwem dany punkt (np. łokieć) znajduje się w określonej lokalizacji na obrazie. Utworzona mapa jest dwuwymiarową siatką, gdzie wartości w poszczególnych komórkach reprezentują stopień pewności. Po utworzeniu mapy ufności, biblioteka tworzy mapy części ciała, analizując położenie konkretnych elementów, co w dalszym etapie służy do utworzenia szkieletu osoby. W tym celu wykorzystywane są mapy PAF (Part Affinity Fields), czyli wektorowe pola, które wskazują, jak prawdopodobne jest, że dwa punkty kluczowe są ze sobą powiązane. Gdy mapy te zostaną utworzone, biblioteka przystępuje do połączenia punktów tworząc szkielet[2]. Druga spośród wymienionych bibliotek (MediaPipe) została udostępniona jako projekt open-source przez firmę Google w czerwcu 2019 roku. MediaPipe, jest to otwartoźródłowe środowisko, które umożliwia wykonywanie różnych zadań związanych z przetwarzaniem obrazów, wideo oraz danych w czasie rzeczywistym. Zbudowana została na podstawie struktury pipelinów przetwarzania multimodalnych danych, co oznacza, że pozwala na tworzenie i uruchamianie strumieniowych systemów analizy danych, w sposób modułowy oraz wydajny. Pipeline to zbiór skomponowanych kroków przetwarzania (tzw. modułów) w ustalonej kolejności. Każdy z modułów odpowiedzialny jest za określone zadanie np. detekcję obiektów, segmentację czy śledzenie ruchu. Dzięki temu, biblioteka ta jest bardzo elastyczna, gdyż użytkownik może z łatwością tworzyć własne pipeline'y, dostosowując je do specyficznych potrzeb. Pierwotnym założeniem MediaPipe było dostarczenie wydajnej biblioteki, która umożliwiałaby wieloplatformowe przetwarzanie strumieni wideo i obrazów w czasie rzeczywistym[6]. Przechodząc do kwestii hardware, najczęściej wykorzystywanymi urządzeniami do rejestracji gestów są kamery RGB. Ich popularność jest uargumentowana stosunkowo niską ceną oraz dostarczanym kolorowym obrazem w wysokiej rozdzielczości, który może być przetwarzany przez algorytmy komputerowego widzenia. Nie są to jednakże jedyne narzędzia, wykorzystywane w procesie rejestrowania ruchów. Elementy takie jak kamery głębi pozwalają na trójwymiarową rejestrację sceny, co umożliwia bardziej precyzyjną analizę ruchów dłoni i ciała. Dużą zaletą kamer głębi jest wysoka dokładność w trudnych warunkach oświetleniowych, gdzie w porównaniu z kamerami RGB, nie mają problemów z dokładnym uchwyceniem gestu. Dodatkowym asortymentem, mogą być również czujniki ruchu, które mierzą przyspieszenie i orientację dłoni. Zastosowanie ww. technologii, pozwala na rejestrowanie subtelnych ruchów i położenia dłoni w przestrzeni. Ostatnia z przestawionych opcji, daje największe możliwości rejestrowania ruchów. Niestety ze względu na koszt zaawansowanych technologii stosowanych w czujnikach ruchu, produkty te są nieprzystępne cenowo dla potencjalnego odbiorcy oprogramowania, omawianego w niniejszej dysertacji[9]. Rysunek 1 Porównanie technologii (Źródło: opracowanie własne) 1.2. Przegląd metod uczenia maszynowego Uczenie maszynowe (ang. Machine Learning (ML)) to dziedzina sztucznej inteligencji, która skupia się na rozwijaniu algorytmów oraz modeli zdolnych do uczenia i automatycznego podejmowania decyzji na podstawie dostarczonych danych. W odniesieniu do platformy tłumaczącej język migowy jest kluczowym rozwiązaniem, umożliwiając przekształcanie danych wideo w rozpoznawalne wzorce, które są możliwe do przetłumaczenia na słowa lub zdania. Pierwsza sieć neuronowa została opracowana przez Warrena McCullocha, wykładowcy z University of Illinois i Waltera Pittsa, niezależnego badacza. Ich artykuł pt. „A Logical Calculus of the Ideas Immanent in Nervous Activity", opublikowany w 1943 roku w czasopiśmie „Bulletin of Mathematical Biophysics" został przyjęty z szerokim entuzjazmem, kładąc tym samym podwaliny pod rozwój sztucznych sieci neuronowych. Praca ta okazała się kamieniem milowym w późniejszych badaniach w dziedzinie informatyki, neurobiologii i kognitywistki. Autorzy, przedstawili w nim jak mogą działać neurony, prezentując swoje pomysły i tworząc prostą sieć neuronową za pomocą obwodów elektrycznych[15]. Obecnie sieci neuronowe są fundamentem współczesnych metod uczenia maszynowego. Składają się z wielu warstw neuronów, które przetwarzają dane wejściowe poprzez szereg operacji matematycznych, co umożliwia uczenie się złożonych wzorców. Konwolucyjne sieci neuronowe (CNN) są szczególnie efektywne w przetwarzaniu danych obrazowych oraz wideo, dzięki swojej zdolności do automatycznego wyodrębnienia cech z surowych danych. Sieci te, składają się z warstw konwolucyjnych, poolingowych oraz gęstych, które współpracują ze sobą, w celu przetwarzania informacji o strukturze przestrzennej obiektów. Autorzy publikacji Sign Language Recognition Using Convolutional Neural Networks wykazali, że CNN mogą być używane do skutecznego przetwarzania i rozpoznawania gestów na podstawie obrazu wideo, umożliwiając automatyczną ekstrakcję cech, co jest

kluczowe dla poprawnej klasyfikacji gestów. Podkreślili również, że ich model nie tylko osiąga wysoką dokładność w rozpoznawaniu gestów, ale także jest skalowalny i może być rozszerzany o nowe gesty lub języki migowe z relatywnie niewielkim wysiłkiem. Badanie to otwiera drogę do tworzenia bardziej zaawansowanych systemów rozpoznawania języka migowego, które mogą być wykorzystywane w aplikacjach takich jak tłumacze języka migowego na mowę w czasie rzeczywistym, tworzenie interfejsów użytkownika dla osób niesłyszących, a także w edukacji[10]. Modele hybrydowe to połączenie różnych algorytmów klasyfikacyjnych, które mogą prowadzić do poprawy dokładności i stabilności rozpoznawania gestów. Przykładem takiego podejścia jest kombinacja CNN oraz HMM, co pozwala na skuteczniejsze modelowanie zależności czasowych w danych sekwencyjnych. Zastosowanie hybrydy modeli CNN oraz HMM opisuje publikacja pt. „SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition". Autorzy publikacji wykazują, że hybrydowe podejście może być skutecznie wykorzystywane do rozpoznawania ciągłych gestów w języku migowym. Ich badania przeprowadzone zostały na dużych zestawach danych, zawierających filmy z gestami języka migowego, przy czym wykorzystali takie techniki jak augmentacja danych, zwiększając tym samym różnorodność przykładów i poprawiając zdolność generalizacji modelu. Wyniki eksperymentu pokazują, że SubUNets osiąga znacznie lepsze wyniki w porównaniu z wcześniejszymi podejściami, zarówno w kontekście rozpoznawania kształtów dłoni, jak i ciągłego rozpoznawania gestów języka migowego. Autorzy podkreślają, że ich architektura nie tylko rozpoznaje gesty z większą dokładnością, ale także lepiej radzi sobie z różnorodnością kształtów dłoni i płynnością sekwencji[1]. 1.3. Wyzwania w rozpoznawaniu gestów języka migowego Rozpoznanie gestów języka migowego stanowi złożone wyzwanie, które wynika z unikalnych cech ww. sposobu komunikacji. W przeciwieństwie do języków mówionych, język migowy wykorzystuje mimikę oraz posturę ciała, co wymaga zaawansowanych algorytmów do skutecznej analizy i rozpoznania. Język ten charakteryzuje się ogromną różnorodnością gestów. Każdy z nich może mieć wiele wariantów, które różnią się w zależności od regionu, kultury, a nawet osoby wykonującej gest. Ponadto, gesty mogą obejmować stosunkowo nieistotne różnice w ruchach dłoni, ułożeniu palców, kierunku spojrzenia, a także w mimice twarzy, co sprawia, że poprawne rozpoznanie jest niezwykle trudne, zwłaszcza jeżeli model zostaje rozszerzany o nowe gesty oraz ich warianty. Badania prowadzone przez Koller i innych naukowców wskazują, że klasyczne podejścia oparte na modelach HMM mogą być niewystarczające do modelowania złożonych i różnorodnych gestów języka migowego. Aby sprostać temu wyzwaniu, autorzy sugerują zastosowanie hybrydowych modeli łączących CNN oraz HMM, co pozwala na skuteczniejsze modelowanie złożoności gestów[5]. Gesty języka migowego często wykonywane są w sposób płynny oraz nieprzerwany co stwarza jeden z największych problemów w przypadku zastosowania algorytmów SI. Kwestia ta jest niezwykle problematyczna dla modeli, które do poprawnego działania potrzebują wyraźnych przerw pomiędzy poszczególnymi znakami, aby określić gdzie badany gest się zaczyna, a gdzie kończy. Kluczowym problemem podczas tworzenia modelu sztucznej inteligencji jest również sama tematyka. W związku z niszowością tematu, utrudniony jest dostęp do dużych, zróżnicowanych zbiorów danych, które są niezbędne dla skutecznego trenowania modeli uczenia maszynowego. W przypadku języka migowego, zbiory danych są ograniczone pod względem wielkości i różnorodności. Brakuje danych pochodzących od różnych użytkowników, wykonanych w zróżnicowanych warunkach oświetleniowych i środowiskach, co może prowadzić do słabszej generalizacji modelu. Problematykę tą porusza w swoich badaniach Patel oraz inni naukowcy, zwracając uwagę na problem ograniczonych zbiorów danych, proponując jednocześnie zastosowanie techniki transfer learningu, które pozwalają na wykorzystanie wiedzy zebranej na większych zbiorach danych do trenowania modeli na mniejszych, specyficznych zbiorach języka migowego[6]. Rozpoznawanie gestów w czasie rzeczywistym wymaga również znacznej mocy obliczeniowej, zwłaszcza w przypadku samego tworzenia modelu oraz ekstrakcji punktów na zbiorach treningowych. Programy wykorzystujące przetwarzanie obrazu w czasie rzeczywistym muszą przetwarzać duże ilości danych wideo o wysokiej rozdzielczości, minimalizując poziom opóźnienia do minimum. Optymalizacja rozwiązań w stosunku do urządzeń o gorszych parametrach również stwarza dodatkowe problemy, zwłaszcza w przypadku urządzeń mobilnych. Ostatnim powszechnie znanym wyzwaniem może być różnica w indywidualnym zachowaniu użytkownika. Każda osoba ma inną długość oraz szerokość kończyn. Inna szybkość, kąt nachylenia czy też indywidualny styl poruszania się każdej osoby sprawia, że opracowany model musi być odporny na takowe odchylenia. Na elastyczność modelu zwraca uwagę między innymi Zhang i Liu, wskazując konieczność opracowania metod, które mogą uwzględnić różnice indywidualne, takie jak ww. style gestykulacji, poprzez wykorzystanie cech całej dłoni, co pozwala na dokładniejsze rozpoznawanie gestów w różnych warunkach[11]. 1.4. Analiza problematyki oraz istniejących aplikacji dotyczących języka migowego W ciągu ostatnich lat rozwój technologii głębokiego uczenia oraz komputerowego

widzenia przyczynił się do powstania wielu aplikacji, służących do rozpoznawania i tłumaczenia języka migowego. Aplikacja opracowana przez Patel i Sahah, pozwala użytkownikom uczyć się języka migowego za pomocą interaktywnych ćwiczeń. System ten, analizuje gesty użytkownika za pomocą technologii rozpoznawania wideo w czasie rzeczywistym, umożliwiając otrzymanie natychmiastowej informacji zwrotnej, co przyczynia się do skuteczniejszej nauki[8]. Aplikacja o nazwie DeepSign, również funkcjonuje w przestrzeni publicznej jako tłumacz języka migowego. Aplikacja ta wykorzystuje CNN do rozpoznawania gestów i przekształcania ich w tekst w czasie rzeczywistym. System ten jest bardzo zaawansowany, co pozwala tłumaczyć gesty na pełne zdania, co znacznie ułatwia komunikację[7]. Niestety każda z wyżej wymienionych aplikacji jest stworzona na rynek anglojęzyczny, a co za tym idzie, nie obsługuje polskiego języka migowego. Zapoznając się z aktualną na dzień 27.08.2024 ofertą na rynek polskojęzyczny, można odnaleźć jedynie rozwiązania, które nie są zautomatyzowane. Specjalne ośrodki czy też firmy prywatne obsługują klientów w ramach spotkań lub połączeń online z tłumaczem języka migowego. Analizując problematykę zastosowania sztucznej inteligencji, można dostrzec, iż nisza ta nie została jeszcze odpowiednio obsłużona w Polsce. Rozwiązania takie jak wideokonferencje, są skuteczną alternatywą, jednakże wymagają osoby fizycznej, która takową rozmowę przetłumaczy. Wiąże się to z kosztami zatrudnienia specjalisty oraz utrzymania infrastruktury, takiej jak stabilne połączenie z Internetem, odpowiedni sprzęt audio-wizualny oraz platformy obsługującej powyższe rozmowy. Zagłębiając się bardziej w poruszaną tematykę, można odnieść wrażenie, że osoby z dysfunkcjami instrumentów głosowych są w pewien sposób wykluczone społecznie, poprzez brak powszechnego narzędzia, które ułatwiałoby im życie codzienne w sferze komunikacyjnej. W przypadku każdego typu rozmów, należy skupić się na zagadnieniu prywatności, do której każdy obywatel ma prawo. Chociażby w taki sferach jak zdrowie czy też finanse. Dzięki istnieniu aplikacji, która byłaby w stanie w pełni przetłumaczyć język migowy, osoby z dysfunkcjami mowy, mogłyby korzystać z większości znanych usług, bez obecności osoby trzeciej, która będzie towarzyszyć w prowadzeniu rozmowy. Zaleca się aby aspekt ten został szerzej zbadany przez specjalistów w dziedzinie badań społecznych, analizując jednocześnie potrzeby osób niepełnosprawnych, a tym samym zwiększając świadomość społeczeństwa na temat poruszany w omawianej dysertacji. 2. Techniczne aspekty funkcjonowania aplikacji Każde oprogramowanie wymaga uprzedniego przygotowania środowiska pracy. Tak też jest w przypadku aplikacji dołączonej do niniejszej dysertacji. Wybrane etapy posiadają własne wymagania odnośnie sprzętu (ang. hardware), oprogramowania (ang. software) oraz środowiska programistycznego, dlatego w poniższych punktach przedstawione zostały kroki do poprawnej konfiguracji środowiska. 2.1. Wymagania systemowe Program podzielony został na dwa etapy główne. Pierwszy etap odpowiada za przygotowanie próbek oraz modelu sztucznej inteligencji, natomiast drugi etap jest stricte związany z platformą, która ma za zadanie odczytywać przygotowany w etapie pierwszym model SI. Każdy program posiada inne wymagania systemowe, dlatego też zostały one zawyżone w odniesieniu do najbardziej wymagającego programu z etapu pierwszego. Komponent Minimalne wymagania Zalecane wymagania Procesor (CPU) Sześciordzeniowy, np. Intel Core i5- Ośmiordzeniowy, np. Intel Core 10600K lub AMD Ryzen 5 3600 i7-9700K lub AMD Ryzen 7 3700X Pamięć RAM 16 GB RAM 32 GB RAM Karta graficzna Zintegrowana Zintegrowana (GPU) Przestrzeń Minimum 20 GB wolnej przestrzeni na SSD NVMe z minimum 100 GB dyskowa dysku SSD wolnej przestrzeni System Windows 10 lub nowszy / Linux (Ubuntu Windows 10 Pro lub nowszy / operacyjny 20.04 lub nowszy) Linux (Ubuntu 20.04 LTS lub nowszy) Inne Python 3.8 lub nowszy Python 3.8 lub nowszy Informacje Procesor ośmiordzeniowy z obsługą wielowątkowości zapewni dobrą wydajność w przetwarzaniu wideo i trenowaniu modeli, choć na nieco niższym poziomie niż wcześniej wspomniane opcje. Zintegrowana karta graficzna wystarczy do obsługi podstawowych zadań związanych z wyświetlaniem i przetwarzaniem obrazu, ponieważ główne obciążenie będzie przeniesione na procesor, aby zmienić obciążenie z procesora na kartę graficzną należy dokonać modyfikacji programu „MachineLearning.py”. 32 GB RAM pozwoli na komfortową pracę z większymi zbiorami danych i bardziej złożonymi modelami Tabela 1 Wymagania systemowe cz1 Druga część programu opiera się głównie na obsłudze aplikacji internetowej oraz przetwarzaniu obrazu z kamery na podstawie dostarczonego już modelu, co nie wymaga od użytkownika zwiększonej mocy obliczeniowej. Komponent Minimalne wymagania Zalecane wymagania Procesor Czterordzeniowy procesor, np. Intel Sześciordzeniowy procesor, np. Intel (CPU) Core i5-8265U lub AMD Ryzen 5 Core i7-10750H lub AMD Ryzen 5 2500U 4600H Pamięć RAM 8 GB RAM 16 GB RAM Karta Zintegrowana, np. Intel UHD Zintegrowana, np. Intel Iris Xe lub graficzna Graphics 620 lub AMD Radeon AMD Radeon Vega 11 (GPU) Vega 8 Przestrzeń Minimum 10 GB wolnej przestrzeni SSD NVMe z minimum 50 GB wolnej dyskowa na dysku SSD przestrzeni System Windows 10 lub

nowszy / Linux Windows 10 Pro lub nowszy / Linux operacyjny (Ubuntu 20.04 lub nowszy) (Ubuntu 20.04 LTS lub nowszy) Inne Python 3.8 lub nowszy Python 3.8 lub nowszy Tabela 2 Wymagania systemowe cz2 2.2. Narzędzia modelowania sztucznej inteligencji Zgodnie z informacjami przedstawionymi w podpunkcie 2.1, każdy program posiada inne wymagania sprzętowe. W tym punkcie przedstawione zostaną indywidualne wymagania dotyczące poszczególnych programów, które nie zostały uwzględnione w poprzednich punktach. Pakiety wymagane do uruchomienia poszczególnych aplikacji, można zainstalować poprzez narzędzie do zarządzania pakietami w Pythonie (pip), wpisując wskazane komendy w terminalu. Program „NagrywanieVideo720p60FPS.py" do poprawnego działania potrzebuje podłączonej kamery internetowej obsługującej jakość 720p oraz przepustowość 60fps, a także importu pakietów takich jak os, time oraz opencv. Dwa pierwsze pakiety są dostępne w podstawowej wersji Pytthona, natomiast pakiet opencv musi zostać doinstalowany. Aby tego dokonać można skorzystać z poniższego polecenia: pip install opencv-python Do obsługi programu „MachineLearning.py" wymagane są pakiety: os, threading, warnings, random, time, webbrowser, multiprocessing, tkinter, cv2, numpy, mediapipe, tensorflow, keras, flatten, scikit-learn oraz pil , osiem pierwszych pakietów to standardowe moduły języka Python, brakujące pakiety można zainstalować inicjując polecenie: pip install opencv-python numpy mediapipe tensorflow keras scikit-learn pillow Program „Prediction Test.py" wymaga takich pakietów jak: cv2, numpy, mediapipe, tensorflow, matplotlib, keras, sklearn, os, tkinter. Brakujące pakiety można zainstalować za pomocą polecenia: pip install opencv-python numpy mediapipe tensorflow keras scikit-learn pillow Do poprawnej obsługi platformy, wymagane są również odpowiednie środki, takie jak pakiety języka python oraz zewnętrzne oprogramowanie inicjujące środowisko serwerowe. Omawiana platforma wymaga utworzenia lokalnego serwera w celu obsługi języka PHP. Przykładem oprogramowania pozwalającego na zainicjowanie ww. środowiska jest XAMPP. Do utworzenia niniejszej aplikacji wykorzystano najnowszą (na dzień 22.08.2024) wersję XAMPP w wersji 3.3.0 oraz PHP w wersji 8.2.12. Po zainstalowaniu aplikacji XAMPP, należy uruchomić opcję Apache, oznaczoną na Rysunku 2 numerem 1. Rysunek 2 Interfejs programu XAMPP Platforma, została projektowana oraz testowana za pomocą przeglądarki internetowej Chrome, która jest zalecana do obsługi ww. aplikacji. Program „app.py" potrzebuje podłączonej kamery internetowej o przepustowości minimum 30 fps oraz zainstalowanych pakietów takich jak: Flask, flask-cors, opencv, numpy, mediapipe, tensorflow, keras, pillow oraz scikit-learn. Elementy te, można zainstalować poprzez komendy w terminalu: pip install Flask flask-cors opencv-python numpy mediapipe tensorflow keras Pillow scikit-learn 3. Budowa modelu rozpoznawania gestów Aplikacja internetowa do właściwego działania potrzebuje wcześniej przygotowanego modelu sztucznej inteligencji, który będzie rozpoznawał wybrane sekwencje ruchów. W tym rozdziale zostanie przedstawiony opis procesu przygotowania ww. modelu. 3.1. Przygotowanie danych wejściowych Przygotowanie danych wejściowych jest kluczowym etapem każdego programu opartego na uczeniu maszynowym. Proces ten obejmuje takie elementy jak wybór odpowiednich źródeł danych, ich organizację, walidację oraz przetworzenie do odpowiedniego formatu. W związku z niszowością tematyki polskiego języka migowego oraz utrudnionym dostępem do próbek badawczych, autor dysertacji utworzył własną bazę gestów, które poprzez algorytm napisany w języku python zostały zarejestrowane za pomocą kamery internetowej. Każde z nagrań musiało zostać ustandaryzowane w celu przystosowania ich do pracy z jednolitym modelem sztucznej inteligencji. Każda z zarejestrowanych próbek posiadała długość 4 sekund, rozdzielczość 720p oraz przepustowość 60 klatek na sekundę. Dodatkowo, autor przyjął metodę 25-50-25, która zaowocowała nagraniem 100 próbek dla każdego gestu. W późniejszym etapie projektu powtórzono ten proces, aby zyskać większą ilość próbek na innych modelach ciał. Metoda ta opierała się na odpowiednim ustawieniu kadru. Pierwszy etap tworzył 25 nagrań od czubka głowy do dolnej części żeber. Taka konfiguracja, pozwalała na najdokładniejsze zarejestrowanie pracy dłoni, wliczając w to układ palców. Drugi etap tworzył 50 nagrań od czubka głowy do górnej części uda. Nagrania te służyły jako baza do całości gestu. Ujęcia te, ujmowały każdą fazę ruchu, od fazy startowej do fazy końcowej, uwzględniając wszystkie niezbędne elementy ludzkiego działa. Ostatni etap tworzył 25 nagrań od czubka głowy do dolnej części kolan. Proces ten miał za zadanie zwiększyć obszar widzenia programu, uwzględniając dalsze położenie szukanych punktów, dzięki czemu utworzony w późniejszym etapie model, mógł z większą precyzją rozpoznawać pozycję statyczną oraz ruch odpowiednich fragmentów ciała człowieka. Ustawienie każdego z etapów zostało zobrazowane na ilustracji 3. Rysunek 3 Etapy zbierania próbek (Źródło: opracowanie własne) Dzięki zastosowaniu ww. rozwiązania, późniejsze algorytmy przetwarzania danych, mogły z większą łatwością zlokalizować wybrane punkty na ludzkim ciele, a następnie przetworzyć je w ramach sekwencji ruchu. Każdy z etapów dostarczał inną ilość punktów, dzięki czemu model uczył się rozpoznawania gestów ze zróżnicowanej perspektywy. W ramach części praktycznej pracy magisterskiej, został dołączony program „NagrywanieVideo720p60FPS.py", który

realizuje wyżej wymienione etapy. Po zakończeniu procesu przygotowywania próbek badawczych autor dysertacji opracował program „MachineLearning.py", który realizuje wszystkie pozostałe kroki do uzyskania działającego modeli SI. Po uruchomieniu ww. programu, użytkownik ma możliwość skorzystania z graficznego interfejsu programu, utworzonego poprzez bibliotekę tkinter. Biblioteka ta jest standardowym pakietem w języku Python. Początkowo, program weryfikuje czy w obecnym katalogu istnieje plik z przetworzonymi punktami o nazwie „preprocessed_data.npz". Jeżeli takowy plik istnieje, zostaje wyświetlone okno z odpowiednim komunikatem (Rys.4). Rysunek 4 Uczenie maszynowe (GUI 1) (Źródło: opracowanie własne) Użytkownik może wybrać jedną z trzech dostępnych opcji. Pierwsza z nich prowadzi do pliku PDF, w którym opisane zostały wymagania sprzętowe dotyczące obsługi programu. Przycisk po prawej stronie, rozpoczyna proces uczenia maszynowego na obecnie wykrytych danych, natomiast przycisk „Wróć do wyboru danych" pozwala na otworzenie widoku okna startowego. Okno to pojawia się również, jeżeli program w początkowej fazie działania nie odnajdzie utworzonego pliku z wyekstraktowanymi punktami. Rysunek 5 Uczenie maszynowe (GUI 2) (Źródło: opracowanie własne) W ww. oknie, użytkownik może wybrać ścieżkę do katalogu, w którym znajdują się wcześniej przygotowane nagrania. Nagrania te, muszą być zgrupowane w odpowiedni sposób. Aby program działał poprawnie, struktura plików powinna wyglądać następująco: Katalog wybrany przez użytkownika / Nazwa sekwencji ruchu / pliki o rozszerzeniu *.avi. Jeżeli program wykryje błędną strukturę plików, zwróci komunikat w kolorze czerwonym. W przypadku poprawnie przetworzonych plików, wyświetli stosowne powiadomienie w kolorze zielonym, tworzy listę wszystkich ścieżek do plików wideo, a następnie odblokuje przycisk „Rozpocznij analizę". Każdy z powyższych przykładów został zobrazowany, za pomocą poniższej grafiki. Rysunek 6 Uczenie maszynowe (GUI 2 - Struktura) (Źródło: opracowanie własne) 3.2. Preprocessing danych Preprocessing polega na wstępnym przetwarzaniu danych, a dokładniej na przekształcaniu surowych danych w formę, która jest bardziej odpowiednia i optymalna dla modelu uczenia maszynowego. Celem tego fragmentu kodu, jest utrzymanie spójności oraz kompatybilności, a także poprawa jakości danych, potrzebnych do uczenia modelu. Usuwa on wszystkie zbędne informacje, czyli tzw. szumy, pozostawiając jedynie potrzebne dane, wymagane przez model SI. Po rozpoczęciu analizy nagrań, program wyświetla nowe okno z paskiem postępów oraz możliwością rozszerzenia widoku o szczegóły. Na rysunku 7, po lewej stronie przedstawiono podstawowe okno przetwarzania danych, po prawej stronie natomiast jest widoczne okno po rozszerzeniu szczegółów. Rysunek 7 Uczenie maszynowe (GUI 3) (Źródło: opracowanie własne) W trakcie przetwarzania danych, algorytm wykorzystuje bibliotekę MediaPipe, opracowaną przez Google. Za jej pomocą, przeprowadza ekstrakcję punktów kluczowych z dostarczonych nagrań, analizując każdą z 240 klatek, a następnie przypisuje do nich odpowiednie etykiety. W ten sposób tworzy pełen spis punktowy, przedstawionego na nagraniu gestu. Zastosowanie powyższego rozwiązania, skutkuje zmniejszeniem ilości danych przetwarzanych przez etap uczenia maszynowego. W przypadku standardowego podejścia, program przetwarzałby każdy pikseli wideo, co mogłoby wprowadzić dodatkowy szum. Skutkowałoby to mniej precyzyjnym uczeniem modelu, a co za tym idzie gorszą precyzją rozpoznawania gestów. Program, posiada również możliwość podglądu wizualizacji procesu ekstrakcji punktów w czasie rzeczywistym. Gdy użytkownik po rozwinięciu szczegółów, uruchomi przycisk „Wyświetl wizualizację" otworzy się nowe okno, które zobrazuje przetwarzanie danych w sposób graficzny. Rysunek 8 Uczenie maszynowe (GUI 3 - Wizualizacja) (Źródło: opracowanie własne) Podczas przetwarzania informacji, program normalizuje dane poprzez pomijanie wideo o dłuższym czasie trwania lub mniejszej ilości klatek na sekundę, jeżeli natomiast nagranie jest krótsze, brakująca część czasu jest dodawana poprzez funkcję uzupełniania sekwencji pad_sequences. Funkcja ta, tworzy macierz wypełnioną zerami o wymiarach zdefiniowanych w kodzie programu, następnie dla każdej sekwencji kopiuje jej wartość do nowej macierzy, a jeżeli sekwencja jest krótsza niż maksymalna długość, resztę wypełnia zerami. Program, ze względu na konieczność przetwarzania dużej ilości danych oraz potencjalnie długi okres trwania ww. procesu, posiada obsługę wieloprocesowości. Wieloprocesowe podejście pozwala na jednoczesne przetwarzanie wielu nagrań na rdzeniach procesora, co znacząco redukuje czas potrzebny na preprocessowanie danych. Funkcje, takie jak blokada oraz kolejka zapewniają synchronizację między procesami, aby zapobiec ewentualnym kolizjom podczas dostępu do wspólnych zasobów, takich jak zapis do listy danych, czy aktualizacja wskaźnika postępu. Gdy dane zostaną poprawnie przetworzone, program tworzy plik „preprocessed_data.npz", w którym zapisuje przetworzone dane oraz „classes.npy" w którym przechowuje nazwy etykiet. Następnie pasek postępu dobiegnie końca, a na ekranie użytkownika wyświetla się komunikat, który umożliwia przejście do procesu uczenia modelu SI lub zamknięcie programu. Rysunek 9 Uczenie maszynowe (GUI 3 - Zakończenie przetwarzania danych) (Źródło: opracowanie

własne) 3.3. Architektura modelu oraz algorytm klasyfikacji Architektura modelu odnosi się do struktury sieci neuronowej. To ona determinuje sposób, w jaki dane są przetwarzane, jakie operacje są wykonywane oraz jakie mechanizmy są stosowane do obsługi modelu. W omawianym oprogramowaniu, autor dysertacji zastosował model oparty na konwolucyjnych sieciach neuronowych (CNN), co miało na celu uchwycenie wzorców w danych sekwencyjnych, takich jak punkty kluczowe wyodrębnione z nagrań wideo. W kontekście przetwarzania sekwencji, warstwy konwolucyjne (Conv1D) przetwarzają dane jednokierunkowo. Filtrują sekwencje punktów kluczowych, tym samym wykrywając wzorce przestrzenno-czasowe. Filtry, inaczej zwane jądrami, są przesuwane wzdłuż sekwencji, stosując operację splotu. Czynność ta, pozwala na uchwycenie lokalnych wzorców danych. Po każdej warstwie konwolucyjnej zastosowano warstwę MaxPooling, która redukuje wymiarowość danych, zachowując najbardziej istotne cechy. MaxPooling wybiera maksymalną wartość z każdej lokalnej grupy wyników, co pomaga w uodpornieniu modelu na przesunięcia w danych. Równocześnie dokonuje redukcji liczby parametrów, co skutkuje zmniejszeniem ryzyka wystąpienia zjawiska przeuczenia modelu. Model został zaimplementowany przy pomocy biblioteki Keras, która działa na bazie TensorFlow. Zastosowane rozwiązanie oferuje szeroką gamę narzędzi do pracy z przetwarzaniem danych, budowaniem architektury sieci, a także optymalizacji modeli. Omawiany program, tworzy model w sposób sekwencyjny, co oznacza, że warstwy są dodawane jedna po drugiej, tworząc stos warstw. Każda z warstw przyjmuje jako argumenty liczbę filtrów o wymiarze 64, rozmiar jądra równy 3 oraz funkcję aktywacji „relu”. Liczba filtrów określa, ile różnych cech model będzie próbował wyodrębnić. Rozmiar jądra to zakres, na którym filtr jest stosowany w danej chwili. Funkcja aktywacji relu, poprzez zamianę wszystkich wartości ujemnych na zero, zapobiega problemowi zanikania gradientu i przyspiesza proces nauki, umożliwiając sieci efektywne uczenie się złożonych wzorców. Warstwa Flatten w niniejszym programie, używana jest do przekształcenia danych z formatu wielowymiarowego do jednowymiarowego, który jest wymagany przez kolejne, w pełni połączone warstwy takie jak warstwa gęsta Dense. Warstwa ta, używana jest zarówno do przekształcania wyodrębnionych cech w postać o mniejszej wymiarowości, jak i do generowania ostatecznego wyniku klasyfikacji. Units, czyli liczba neuronów w warstwie została ustawiona na 256, co pozwala na modelowanie bardziej złożonych zależności, jednakże może prowadzić do przeuczenia. W ostatnim kroku, warstwa Dense generuje wyniki klasyfikacji za pomocą funkcji aktywacji softmax. Funkcja ta, przekształca wyniki w prawdopodobieństwa, które sumują się do 1, co pozwala na przypisanie prawdopodobieństwa do klasy. 3.4. Trening, walidacja oraz ocena skuteczności modelu Trening modelu to proces optymalizacji jego parametrów tak, aby jak najlepiej uczył się rozpoznawania wzorców dostarczonych przy pomocy danych wejściowych, a następnie właściwie przypisywał je do odpowiednich etykiet. W kontekście sieci neuronowych, trening polega na minimalizacji funkcji straty, przy użyciu danych treningowych oraz odpowiedniego optymalizatora. W pierwszej fazie treningu modelu, zastosowany został tzw. forwadr propagation, czyli etap w którym dane przekazywane są do modelu, a każdy neuron przekształca wejścia, oblicza wyjścia i przekazuje je do następnej warstwy. Na końcu tego procesu, model generuje wynik, który jest porównywany z rzeczywistą etykietą danych przy użyciu funkcji straty. Następnie występuje backpropagation, czyli kluczowy mechanizm w treningu sieci neuronowych. Polega on na propagacji błędu wstecz przez sieć neuronową. Błąd ten jest obliczany jako różnica pomiędzy przewidywaniami modelu, a rzeczywistymi wynikami. Proces ten, aktualizuje wagi w sieci tak, aby minimalizować funkcję straty. W tym przypadku zastosowany został algorytm Adam, który jest odmianą algorytmu gradientu prostego (Stochastic Gradient Descent, SGD). Algorytm ten stosuje poniższy wzór matematyczny: Rysunek 10 Wzór algorytmu gradientu prostego gdzie: - w to wagi, które są optymalizowane, η (eta) to współczynnik uczenia się (ang. learning rate), który określa, jak duży krok robimy w kierunku minimalizacji funkcji kosztu (ang. Loss function), natomiast pozostała część to to pochodna funkcji kosztu Loss względem wag w, czyli gradient funkcji kosztu. Pokazuje kierunek i szybkość zmiany funkcji kosztu w zależności od zmian wag. Późniejszy etap obejmuje tworzenie tzw. epok. W każdej epoce, model przetwarza cały zbiór treningowy. Aby zwiększyć efektywność aktualizacji wag, zastosowane zostało porcjowanie (batch), które pozwala na przetwarzanie mniejszej ilości informacji jednocześnie. Podczas treningu, model uczy się na danych treningowych, jednakże istnieje pewne ryzyko, że nauczy się on wzorców specyficznych dla tych danych. Aby zapobiec ww. problematyce, zastosowano techniki regularyzacji takie jak Dropout oraz L2 Regularization (Ridge). Pierwsza z nich polega na losowym wyłączeniu części neuronów w warstwie podczas treningu. W omawianym programie wartość ta została ustawiona na 0.5, co oznacza, że połowa neuronów w warstwie jest wyłączana w każdej iteracji. Druga zaś, dodaje karę do funkcji straty za duże wartości wag, co ogranicza złożoność modelu i zapobiega jego przeuczeniu. Walidacja modelu jest niezbędna w procesie uczenia maszynowego. W jego trakcie program ocenia, jak dobrze model generalizuje, czyli jak radzi sobie z danymi, których nie uwzględnił podczas treningu.

Autor pracy dyplomowej, przyjął złożenie 20% / 80%, gdzie 20% danych jest wykorzystywanych do walidacji, natomiast 80% do trenowania modelu. W trakcie uczenia, stosowane są callback'i. EarlyStopping, czyli wczesne zatrzymanie uczenia maszynowego, wykorzystane jest w celu zabezpieczenia modelu przed ewentualnym przeuczeniem. Proces ten stale monitoruje wydajność modelu na zbiorze walidacyjnym i przerywa trening w przypadku braku poprawy wyników. Istnieje również callback nazwany ModelCheckpoint, który zapisuje najlepsze wagi modelu na podstawie wybranej metryki. Cały proces uczenia maszynowego, został zobrazowany poprzez interfejs graficzny użytkownika, który pojawia się po zatwierdzeniu przycisku dostępnego po przetworzeniu nagrań. Rysunek 11 Uczenie maszynowe (GUI 4) (Źródło: opracowanie własne) Gdy uczenie maszynowe dobiegnie końca, program tworzy plik o nazwie best_model.h5, w którym zapisane są wszystkie informacje opracowywane przez powyższe algorytmy. Użytkownik, otrzymuje również powiadomienie dotyczące zakończenia pracy programu. Na tym etapie, może on zweryfikować poprawność uczenia maszynowego, a także sprawdzić ile epok przetworzył program w celu utworzenia modelu. Zatwierdzając przycisk „Zakończ", użytkownik kończy pracę programu, otrzymując tym samym dostęp do wszystkich niezbędnych plików do obsługi platformy, opisanej w rozdziale czwartym. Rysunek 12 Uczenie maszynowe (GUI 4 - Zakończenie pracy) (Źródło: opracowanie własne) Działanie powyższego programu obrazuje diagram aktywności zawarty na rysunku 13. Rysunek 13 Diagram aktywności (Uczenie maszynowe) (Źródło: opracowanie własne) 4. Implementacja platformy Implementacja platformy opiera się na utworzeniu aplikacji internetowej wraz z odpowiednim połączeniem jej z modelem sztucznej inteligencji. Implementację możemy podzielić w tym przypadku na dwie części. Część frontendową, która opiera się na wdrożeniu wersji graficznej, z której będzie korzystać użytkownik, a także część backendową, która będzie obsługiwać komunikację z serwerem oraz modelem SI. 4.1. Koncepcja wizualna platformy Każda aplikacja weebowa, która będzie posiadać interfejs graficzny (GUI), powinna być uprzednio zaprojektowana przez osobę, która dobrze rozumie zasady korzystania z interfejsów aplikacji internetowych. Doświadczenie w projektowaniu platform przyjaznym użytkownikom pozwala na zdobycie szerszego grona odbiorców poprzez pozytywne odczucia z korzystania z aplikacji. Funkcjonalności wybranego oprogramowania stanowią dobrą podstawę, natomiast design może wzmocnić i uatrakcyjnić aplikację, czyniąc ją bardziej przyjazną i angażującą dla użytkowników[13]. W obecnej dobie Internetu, gdzie panuje przesyt informacji, a użytkownicy są często przebodźcowani, bardzo istotne jest utworzenie odpowiedniej warstwy graficznej. Użytkownicy zazwyczaj oceniają aplikację w ciągu kilku sekund od jej uruchomienia. Atrakcyjna i spójna oprawa graficzna może przyciągnąć uwagę odbiorcy i zachęcić go do dalszej eksploracji. Niezależnie od tego, jak zaawansowane oprogramowanie zostało zaoferowane użytkownikowi, pierwsze wrażenie bazuje na wyglądzie. Intuicyjny interfejs, który jest łatwy w nawigacji, ułatwia korzystanie z aplikacji, pozostawiając pozytywne wrażenia. Wygląd aplikacji nie tylko zwiększa jej zainteresowanie wśród odbiorców ale również pozytywnie wpływa na wzmocnienie marki, poprzez identyfikację marki z wybraną nazwą, kolorystyką czy też utworzonym logotypem[14]. W związku z powyższymi informacjami, omawiana aplikacja również posiada swój projekt wizualny, który został utworzony na początku procesu projektowego w oprogramowaniu figma, a plik projektowy został umieszczony pod nazwą „Projekt koncepcyjny.fig" oraz „Projekt koncepcyjny.pdf" w części praktycznej pracy dyplomowej. Rysunek 14 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 4.2. Mechanizmy komunikacji między frontendem a backendem Większość zaawansowanych aplikacji, korzystających z graficznego interfejsu użytkownika, a zwłaszcza aplikacje webowe, korzystają z warstwy frontendowej oraz backendowej w celu optymalizacji obsługi dużej ilości poleceń lub wykorzystania niedostępnych z poziomu wybranej warstwy narzędzi. 4.2.1. Komunikacja serwer – użytkownik Omawiana aplikacja napisana jest głównie w języku PHP, JavaScript oraz Python co wymaga stałej komunikacji pomiędzy frontendem, a backendem. Kod w języku PHP dzięki stałemu połączeniu z serwerem, może w sposób dynamiczny, kontrolować dostępem do witryny oraz zawartości, która jest wyświetlana na ekranie użytkownika. Poprzez zarządzanie witryną kod PHP umożliwia wczytanie odpowiedniego kodu CSS oraz JavaScript w wybranych zakładkach, dzięki funkcji rozpoznawania adresów URL. Oprócz aspektów wizualnych, PHP obsługuje również wczytywanie wizualizacji dla gestów, poprzez przeszukiwanie katalogów zamieszonych na serwerze w poszukiwaniu nazwy wprowadzonej w warstwie frontendowej oraz przekazanej odpowiednio przez skrypt napisany w języku JavaScript. 4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) Integracja modelu AI z aplikacją internetową jest realizowana poprzez połączenie części backendu, napisanego w języku programowania Python z częścią frontendową platformy napisaną w języku JavaScript. Wspomniana część backendowa jest zapisana w pliku o nazwie „app.py". Program ten uruchamia model sztucznej inteligencji o nazwie „best_model.h5" oraz listę etykiet „classes.npy", które

są wykorzystywane do przetwarzania danych w czasie rzeczywistym. Dane z kamery użytkownika są przetwarzane, a wyodrębnione punkty kluczowe są przekazywane do modelu, który dokonuje predykcji gestów. Wyniki te są następnie udostępniane poprzez endpointy, które zwracają szczegóły dotyczące rozpoznanych gestów w formacie JSON. Frontend aplikacji komunikuje się z backendem za pomocą zapytań http, korzystając z fetch API do wysyłania zapytań do serwera Flask. Za komunikację z serwerową częścią oprogramowania, odpowiedzialne są trzy skrypty napisane w języku JavaScript, a są nimi: „edu,js", „translator.js" oraz „sign_video_loader.js". Każdy z trzech wymienionych skryptów, korzysta z wywołań endpointów poprzez serwer lokalny o porcie 5000 „http://localhost:5000/gesture_details", które służą do pobierania informacji o rozpoznanych gestach i aktualizowania interfejsu użytkownika. W pliku app.py, endpoint dostarcza strumień wideo z kamery internetowej, który następnie jest przetwarzany przez model SI. Frontend inicjuje kamerę oraz przesyła obraz do backendu gdzie obraz jest przetwarzany, a następnie odtwarza przesyłany strumień, umożliwiając tym samym interaktywną analizę gestów w czasie rzeczywistym. 4.3. Widoki aplikacji Omawiana platforma internetowa posiada tzw. widoki, czyli warstwę frontendową, która jest odpowiednio ostylowana poprzez arkusze stylów CSS, starając się jak najdokładniej odwzorować utworzony w początkowych fazach pracy projekt wizualizacji aplikacji. Rysunek 15 Interfejs graficzny aplikacji – zakładka Home (Źródło: opracowanie własne) Platforma składa się z siedmiu widoków. Trzy pierwsze widoki odnoszą się do informacji wstępnych dotyczących realizowanego projektu. Są to zakładki „Home", „O projekcie" oraz „O nas". Każda z zakładek posiada statyczne elementy informacyjne oraz dekoracyjne, które mają zachęcić użytkownika do dalszej interakcji z platformą. Poniżej opisane zostały bardziej interesujące elementy, każdej z zakładek. Rysunek 16 Zakładka Home (Źródło: opracowanie własne) Na stronie Index.php (Home) zostały przedstawione podstawowe informacje dotyczące platformy. Zastosowany został również slider, który wyświetla logotyp Uniwersytetu Rzeszowskiego, przechodząc od prawej strony do lewej, a następnie cofając się do początku. Rysunek 17 Slider - strona Index.php (Źródło: opracowanie własne) Zakładka O projekcie (Project.php) zawiera informacje dotyczące funkcjonowania utworzonej aplikacji. Można znaleźć tam między innymi informacje o zastosowanych rozwiązaniach technologicznych, genezie powstania dysertacji, poruszanej problematyce, skrócony opis działania aplikacji, a także linię czasu, która w sposób skrótowy ukazuje etapy powstawania aplikacji. Rysunek 18 Widok zakładki O projekcie (Project.php) (Źródło: opracowanie własne) Zakładka O nas (About.php) skupia się na opisie osób zaangażowanych w powstanie projektu. W związku ze stałym rozwojem tworzonego oprogramowania oraz wysoką czasochłonnością prac, a także innymi losowymi aspektami, w projekt były zaangażowane 4 osoby. Oprócz autora oprogramowania oraz niniejszej dysertacji, w projekt zaangażował się również Dr. Inż. Wojciech Kozioł, który był pomysłodawcą tematu omawianej pracy, Dr. Inż. Bogusław Twaróg, który prowadził nadzór merytoryczny nad powstającą pracą oraz inż. Patryk Arendt, który służył jako model motion capture dla 5 gestów (tj. 500 próbek nagrań wideo). Rysunek 19 Widok zakładki O nas (About.php) (Źródło: opracowanie własne) W ramach uhonorowania wkładu, jaki włożyły ww. osoby w powstanie pracy dyplomowej, zostały one wymienione w wersji aplikacyjnej w zakładce O nas. Rysunek 20 Uhonorowanie wsparcia (About.php) (Źródło: opracowanie własne) Klikniecie przycisku „Przejdź do aplikacji" powoduje wczytanie ostatniego statycznego widoku jakim jest „Dashboard.php". W widoku tym opisane są podstawowe informacje dotyczące ilości gestów, ilości wykorzystanych próbek, sumarycznego czasu nagrań, a także wiele innych. Rysunek 21 Widok zakładki Dashboard (Dashboard.php) (Źródło: opracowanie własne) W zakładce tej przedstawiona została również animacja, która w trzech etapach prezentuje w jaki sposób program rozpoznaje sekwencje ruchów ludzkiego ciała, na przykładzie gestu z etykietą „Dzięki". Rysunek 22 Działanie systemu rozpoznawania gestu (Źródło: opracowanie własne) Pozostałe trzy widoki, odpowiadają pośrednio lub bezpośrednio za komunikację z systemem rozpoznawania gestów. Działają one na podobnej zasadzie, lecz skupiają się na innym rodzaju wykonywanego zadania przez użytkownika. Rysunek 23 Widok zakładki Tłumacz (Translator.php) (Źródło: opracowanie własne) Zakładka Tłumacz, umożliwia użytkownikowi tłumaczenie słów w języku migowym w obie strony. Może on skorzystać z funkcji udostępnienia obrazu z kamery co pozwoli na tłumaczenie języka migowego na język naturalny lub wyszukać gest w języku migowym, dostępny w systemie poprzez polecenie pisemne lub komendę głosową. W celu wyszukanie gestu języka migowego, wystarczy, że użytkownik uzupełni pole tekstowe odpowiednią nazwą gestu i zatwierdzi zmianę klikając przycisk po prawej stronie lub wciskając klawisz „Enter". Rysunek 24 Wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) Istnieje również alternatywny sposób wyszukiwania gestu w systemie. Użytkownik może skorzystać z przycisku, znajdującego się po lewej stronie okna, aby rozpocząć nasłuchiwanie mikrofonu. Po wypowiedzeniu frazy np. „Cześć", możemy użyć polecenia głosowego „... stop szukaj", co automatycznie zakończy nasłuchiwanie mikrofonu i rozpocznie proces szukania gestu o etykiecie „Cześć".

Nasłuchiwanie mikrofonu można także wyłączyć w sposób ręczny, klikając ponownie na ikonę mikrofonu. Wyszukiwanie głosowe, oparte zostało na polskiej wersji językowej Web Speech API dla języka JavaScript. Po aktywacji funkcji nasłuchiwania program sprawdza dostępność API w przeglądarce i konfiguruje obiekt rozpoznawania mowy. Podczas nasłuchiwania mikrofonu, program przetwarza rozpoznane fragmenty, a następnie wstawia je do pola tekstowego. Jeżeli program wykryje określoną komendę tj. „stop szukaj", przerywa nasłuchiwanie i uruchamia proces wyszukiwania nagrania. Program obsługuje najczęściej pojawiające się błędy, a także automatycznie restartuje nasłuchiwanie, jeżeli nie zostało ono przerwane ręcznie. Rysunek 25 Głosowe wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) Gdy gest nie zostanie odnaleziony, na ekranie pojawia się stosowny komunikat. Gdy gest zostanie wyszukany poprawnie, oczom użytkownika ukaże się wizualizacja gestu odtwarzana w postaci nagrania. Przykład takowej wizualizacji został ukazany na grafice poniżej. Rysunek 26 Wizualizacja gestu (Źródło: opracowanie własne) W drugim przypadku użycia, użytkownik może udostępnić obraz z kamery, a następnie wykonać wybrany przez siebie gest. Na wykrytą postać zostanie naniesiona siatka punktów, która będzie obrazować przetwarzane dane. Program w czasie rzeczywistym, w lewym górnym rogu ekranu wyświetlać będzie prawdopodobne gesty pokazane przez użytkownika. Gdy gest zostanie wykryty wystarczającą liczbę razy, oczom użytkownika ukaże się również przycisk „Wyświetl wizualizację pokazywanych gestów". Rysunek 27 Prezentacja gestu "Dzięki" (Źródło: opracowanie własne) Gdy użytkownik kliknie na komunikat „Wyświetl wizualizację pokazywanych gestów" po prawej stronie, automatycznie w sposób płynny wczyta się nagranie z wizualizacją gestu. Wizualizacja ta będzie się zmieniać za każdym razem, gdy program wykryje pokazanie innego gestu z odpowiednią pewnością. Rysunek 28 Synchronizacja wizualizacji z obrazem w czasie rzeczywistym (Źródło: opracowanie własne) Po przejściu na inną zakładkę, w przeglądarce internetowej klienta, śledzenie kamery zostaje zatrzymane, zarówno przez warstwę forntendową jak i backendową, a algorytmy wstrzymują swoją pracę oczekując na wznowienie połączenia. Poniżej przedstawiono diagram przypadków użycia dla zakładki Tłumacz: Rysunek 29 Diagram przypadków użycia (Tłumacz) (Źródło: opracowanie własne) Dla powyższego widoku został również przygotowany diagram aktywności, który pozwala na jeszcze bardziej precyzyjne zapoznanie się z architekturą działania oprogramowania. Poniższa grafika obrazuje proces przetwarzania obrazu z kamery użytkownika, wraz z obsługą wizualizacji, która jest aktualizowana w czasie rzeczywistym. Rysunek 30 Diagram aktywności (Tłumacz - kamera / wizualizacja) (Źródło: opracowanie własne) Zobrazowany został również diagram aktywności, odpowiedzialny za funkcję wyszukiwania wizualizacji za pomocą poleceń tekstowych oraz głosowych. Diagram UML ww. procesu znajduje się poniżej. Rysunek 31 Diagram aktywności (Tłumacz – Wizualizacja) (Źródło: opracowanie własne) Ostatnie dwa widoki aplikacji omawiają działanie zakładki „Nauka języka migowego". Oprogramowanie, dzięki zaawansowanym algorytmom wspiera proces uczenia języka migowego. W tym celu stworzone zostały kategorie oraz „kafelki", które umożliwiają wybór gestu, którego chce się nauczyć użytkownik. Funkcja ta skierowana jest dla osób początkujących, które dopiero rozpoczynają swoją przygodę z PJM. Rysunek 32 Widok zakładki Nauka Języka Migowego (Edu.php) (Źródło: opracowanie własne) Na stronie podstronie edu.pl możemy wybrać kategorię, która nas interesuje, a następnie odpowiedni dla nas gest. Po wybraniu odpowiedniej kategorii użytkownik będzie mógł zaobserwować zmianę w wyświetlanych elementach. Zamiast kategorii, na ekranie pojawią się gesty, które są z nią powiązane. Rysunek 33 Gesty wybranej kategorii (Źródło: opracowanie własne) Po wybraniu jednego z dostępnych gestów pojawia się nowy widok (Edu-progress.php), który wizualnie jest zbliżony do widoku tłumacza. Tutaj również możemy uruchomić podgląd z kamery, natomiast po prawej stronie odtwarzane jest w zapętleniu nagranie z wybranym przez nas gestem. W widoku tym, niedostępny jest input do wyszukiwania gestów, czy też zmiany wizualizacji. Po uruchomieniu kamery i poprawnym wykonaniu gestu, program zatrzyma udostępnianie obrazu z kamery, a użytkownik otrzyma odpowiedni komunikat na ekranie. Rysunek 34 Wykonanie poprawnego gestu (Źródło: opracowanie własne) 5. Ocena działania platformy Utworzona platforma została przetestowana pod względem działania modelu sztucznej inteligencji oraz kilku powiązanych z nią czynników. Przebadana została precyzja modeli CNN, skonstruowanych z różnych ilości próbek, w konfiguracji 100 próbek na 1 gest, 200 próbek na 1 gest oraz 300 próbek na 1 gest. Wyszczególnione zostały etapy przetwarzania danych, uczenia maszynowego oraz ogólnej pracy modelu w czasie rzeczywistym, a także na bazie przygotowanych wcześniej nagrań. Następnie, wyniki te zostały zestawione z utworzonym hybrydowym modelem CNN + LSTM. 5.1. Testy w rzeczywistych warunkach Testy związane z wydajnością w rzeczywistych warunkach, a co za tym idzie w czasie rzeczywistym są stosunkowo ciężkie do realizacji. Wpływ ma na to rozbieżność pomiędzy wykonywanymi gestami, które ciężko odtworzyć w sposób identyczny,

zróżnicowane oświetlenie, ułożenie ciała oraz wykorzystywany hardware. Istnieje wiele zmiennych, które mogą zakłócić obiektywną ocenę modelu, zwłaszcza jeżeli ten został przygotowany na niewielkiej ilości próbek. W omawianym punkcie, przedstawione zostaną zarówno obiektywne informacje, podparte dowodami, jak i subiektywne odczucia autora dysertacji. 5.1.1. Ekstrakcja punktów kluczowych Pierwszym elementem, który zostanie poddany analizie jest proces przetwarzania informacji, potrzebnych do utworzenia modelu. Cały proces uczenia maszynowego czyli ekstrakcja punktów kluczowych oraz nauka modelu odbywała się na maszynie obliczeniowej z zamontowanym procesorem AMD EPYC 7702 64-Core, który posiadał 128 procesorów wirtualnych o szybkości 2GHz, pamięcią ram 256GB oraz dysku HDD. Przetwarzanie danych w modelu CNN, zastosowane na 6 gestach po 100 próbek każdy, co daje wynik 600 próbek, trwało 19 minut w zaokrągleniu czasu do pełnych minut. Czas trwania tego samego procesu na 1200 próbkach trwało 35 minut, natomiast finalny model omawiany w dysertacji, potrzebował 54 minut na przetworzenie wszystkich danych (1800 próbek). Po zestawieniu ww. informacji w postaci wykresu, możemy zaobserwować jak wygląda złożoność czasowa przetwarzania danych. Przetwarzanie próbek w stosunku do czasu 60 54 50) n im (a 40 35 in a z r a 30 w t e 19 z r p 20 s a z C 10 0 600 próbek 1200 próbek 1800 próbek Czas w minutach 19 35 54 Ilość próbek / Czas Wykres 1 Przetwarzanie próbek CNN Uśredniając czas przetwarzania każdej próbki wynosi 1,81 sekundy. Rozbijając na poszczególne partie: przy 600 próbkach, program potrzebował 1,9 sekundy na przetworzenie jednego nagrania, 1,75 sekundy dla 1200 próbek oraz 1,8 sekundy w przypadku 1800 próbek. Zestawiając dane przetwarzane przez program wykorzystujący CNN oraz CNN w połączeniu z LSTM, nie widać jednoznacznych różnic. Program w podobnym czasie przetworzył dane zarówno dla modelu opartym o CNN jak i modelu hybrydowym, przy czym mniej złożony model zakończył proces ponad 2 minuty wcześniej. Rysunek 35 Zestawienie metody CNN oraz CNN + LSTM (Źródło: opracowanie własne) Czas przetwarzania danych (1800 próbek) CNN+LSTM 56 u le d o m ja z d o R CNN 54 53 54 54 55 55 56 56 57 Czas przetwarzania (min) Wykres 2 Czas przetwarzania danych – porównanie CNN – CNN + LSTM 5.1.2. Uczenie modelu Uczenie modelu również zostało porównane w identycznych kryteriach. Z otrzymanych informacji wynika, że czas uczenia maszynowego nie jest stricte związany z ilością próbek. Funkcja przerywania uczenia, która za zadanie ma chronić model przed przeuczeniem, skutecznie skraca czas potrzebny do stworzenia modelu. Program kończy pracę za każdym razem gdy wykryje, że precyzja modelu nie poprawia się co skutkuje utworzeniem modelu w krótszym czasie niżeli z potencjalnie mniejszą ilością próbek. Uczenie maszynowe w stosunku do czasu 140 124 120)s (u 100 le d 76 o m 80 a in 60 e z c u s 40 30 a z C 20 0 600 próbek 1200 próbek 1800 próbek Czas w sekundach 30 124 76 Ilość próbek / Czas Wykres 3 Uczenie modelu CNN O ile nie było widocznej różnicy pomiędzy przetwarzaniem danych pomiędzy modelem wykorzystującym CNN, a modelem hybrydowym CNN + LSTM, o tyle w przypadku uczenia modelu różnica jest bardzo zauważalna. Model z mniejszą ilością warstw uczył się przez 1 minutę i 16 sekund, natomiast model z przetwarzaniem danych wstecz, trenował się przez 6 godzin, 9 minut i 36 sekund. Różnica w czasie nauczania modelu jest ogromna, a co za tym idzie model hybrydowy z zastosowaniem LSTM wydaje się mało wydajnym rozwiązaniem, zważywszy na małą ilość przetwarzanych danych. Program oparty na mniejszej ilości warstw uczył się ze stosunkowo wysoką precyzją, która oscylowała w zakresie 91-95%, oraz 81-88% podczas walidacji, co obrazuje poniższa ilustracja. Rysunek 36 Proces uczenia maszynowego CNN (Źródło: opracowanie własne) Druga wersja programu zatytułowana roboczo 2.1, wykazywała wyższą precyzję uczenia podczas kilku prób uczenia modelu. Dokładność modelu oscylowała pomiędzy 97-99%, a podczas walidacji wynik ten wynosił 93-99%. Wyniki te sugerują, że model utworzony za pomocą CNN oraz LSTM radzi sobie znacznie lepiej niżeli poprzednia wersja. Należy zwrócić uwagę również na wielkość pliku docelowego. Model utworzony w wersji 1.0 zajmuje 25,8 MB miejsca na dysku, natomiast model w wersji 2.1 zajmuje 1,26 GB. Kwestia ta staje się problematyczna, gdyż model reprezentujący niewiele niższą precyzję podczas uczenia 5 maszynowego, zajmuje 50 razy mniej miejsca na dysku, co staje się kuszącym rozwiązaniem w przypadku chęci zaoszczędzenia miejsca w przestrzeni dyskowej. Rysunek 37 Proces uczenia maszynowego CNN + LSTM (Źródło: opracowanie własne) Czas przetwarzania danych (1800 próbek) CNN+LSTM 22176 u le d o m ja z d o R CNN 76 0 5000 10000 15000 20000 25000 Czas przetwarzania (s) Wykres 4 Czas uczenia modelu – porównanie CNN – CNN + LSTM 5.1.3. Testy w czasie rzeczywistym Model z założenia miał zostać przeznaczony do platformy obsługującej rozpoznawanie polskiego języka migowego w czasie rzeczywistym. W związku z powyższą informacją, również ten aspekt został przetestowany. Niestety w związku z wieloma czynnikami losowymi, nie da się zmierzyć precyzyjnie różnicy pomiędzy rozpoznawaniem gestów w czasie rzeczywistym pomiędzy modelem utworzonym w wersji podstawowej oraz hybrydowej. 5 Jednakże dokonano pewnych obserwacji, które sugerują wykorzystanie jednego spośród dwóch modeli. W przypadku modelu podstawowego, nie zaobserwowano

problemów. Model działał stabilnie, precyzja wykrywanych gestów była zadowalająca. W przypadku modelu hybrydowego, uruchomionego w warunkach czasu rzeczywistego zaobserwowano problemy. Model działał niestabilnie, rejestrując ciągłe „przycięcia" obrazu. Zachodziło tam zjawisko klatkowania, a dokładna przyczyna została opisana w punkcie 5.3. 5.2. Skuteczność i dokładność rozpoznawania gestów Każdy z utworzonych modeli został również przetestowany w warunkach odpowiednio wcześniej przygotowanych. Specjalnie zaprojektowany program, odtwarzał trzy nagrania których nie było zamieszonych w próbkach modelu, dla każdego spośród 6 istniejących gestów. Następnie za pomocą dostarczonych modeli klasyfikował wykryte gesty. Ocenie została poddana precyzja rozpoznanych gestów z podziałem procentowym. Spośród testowanych gestów jeden z nich otrzymał najgorszy wynik, rozpoznając tym samym inny gest. Pozostałe tj. 5/6 gestów zostało rozpoznanych poprawnie. Zestawienie wykresów wraz z uśrednieniem wyników, zostało zaprezentowane na wykresie 5. 5 Wykres 5 Zestawienie wykresów Uśredniając wyniki spośród trzech testów, gest „Cześć" został rozpoznany z precyzją 60,70%, program wykrył również możliwość zaistnienia gestu „Dzięki" z wynikiem 38,69% oraz „Prosić" (0,58%). Program podczas analizy nagrań wykrył 2 z 3 wskazanych gestów poprawnie. Podsumowanie analizy gestu „Cześć" zostało przedstawione na poniższym wykresie. 5 Wykres 6 Analiza gestu Cześć - CNN Porównując wyniki z modelem CNN + LSTM, lepiej prezentuje się model CNN. Model składający się z dodatkowej warstwy, błędnie wykrył gest dzięki z precyzją 60,33%, oraz cześć z precyzją 39,67%. 5 W przypadku gestu „Dzięki" model nieprecyzyjnie rozpoznał wskazywany gest. Jedynie na drugiej próbce gest dzięki miał wyższe prawdopodobieństwo wystąpienia niżeli inne opcje. Program błędnie rozpoznał, że wskazanym modelem jest gest Proszę z prawdopodobieństwem 66,13%, drugim możliwym gestem było dzięki z wynikiem 33,44%. Dodatkowo program dostrzegł podobieństwo z gestem prosić (0,24%), Cześć (0,17%) oraz Przepraszać (0,02%). Wykres 7 Analiza gestu Dzięki – CNN W powyższym przypadku lepszy okazał się model z dodatkową warstwą, prezentując następujące wyniki: dzięki (67,62%), proszę (32,11%), cześć (0,26%). 5 W przypadku gestu „Dziękuję" nie było minimalnych wątpliwości. Gest ten został wykryty z precyzją aż 99,98%. Pozostałe 0.02% zostało podzielone po równo pomiędzy gestem „dzięki" i „Cześć". Wykres 8 Analiza gestu Dziękuję - CNN W powyższym przypadku, znacznie lepsza okazała się sieć CNN. Druga sieć wykryła gest: dziękuję (65,22%), dzięki (32,16%), przepraszać (2,38%), cześć (0,18%), prosić (0,05%), proszę (0,01%). 5 Czwarty badany gest (Prosić) został rozpoznany poprawnie z precyzją 75,69%, pozostałe wartości zostały podzielone pomiędzy: cześć (22,34%), proszę (1,88%), dziękuję (0,06%) oraz dzięki (0,03%) Wykres 9 Analiza gestu Prosić – CNN Gest prosić został wykryty poprawnie w każdej konfiguracji sieci, lecz sieć LSTM wykryła gest z większą precyzją, zwracając wynik precyzji: prosić (99,74%), dziękuję (0,24%), dzięki (0,02%), przepraszać (0,01%). 5 Piąty gest, tak jak gest trzeci, został rozpoznany z bardzo wysoką precyzją (99,91%). Pozostałe prawdopodobieństwo zostało podzielone jedynie przez dzięki (0,07%) oraz prosić (0,02%). Wykres 10 Analiza gestu Proszę – CNN W powyższym przypadku sieć CNN okazała się lepsza niżeli LSTM. Druga z wymienionych sieci wykryła gesty: proszę (77,73%), dzięki (21,90%), cześć (0,35%), prosić oraz przepraszać (0,01%). 5 Ostatnim badanym gestem jest gest przepraszać. Gest ten został sumarycznie wykryty z precyzją 49%, możliwe było również wykrycie etykiety dzięki, która miała również wysoki wynik (43,71%). Analizując wyniki zauważyć można także proszę (7,07) oraz cześć (0,22%). Wykres 11 Analiza gestu Przepraszać – CNN Model LSTM wykrył odpowiednio gesty: przepraszać (44,11%), dzięki (38,40%), proszę (17,01%), cześć (0,23%), dziękuję (0,14%) oraz prosić (0,10%). Ogólny wynik precyzji przetwarzanych gestów wskazuje na to, iż model CNN jest bardziej precyzyjny (sumarycznie 417,72%) niżeli model hybrydowy CNN + LSTM (sumarycznie 394,09%). 60 5.3. Analiza błędów i propozycje ulepszeń Platforma, a co za tym idzie również model SI posiada nie tylko swoje zalety ale również wady. W pierwszej kolejności należy wymienić warstwę backendową. Na przykładzie omawianej dysertacji udowodniono, że model hybrydowy w postaci CNN + LSTM, niekoniecznie działa poprawnie z przetwarzaniem obrazów w czasie rzeczywistym. LSTM jest przeznaczony głównie do pracy z sekwencjami danych, które mają wyraźny związek czasowy, takimi jak teksty lub sygnały czasowe. W przypadku obrazu, dane są przestrzenne, a nie czasowe. Do przetwarzania danych obrazowych lepiej nadają się sieci konwolucyjne (CNN), które są zaprojektowane specjalnie do pracy z danymi w formie siatek (takich jak obrazy). Zwiększenie złożoności, w tym przypadku wpłynęło negatywnie na działanie programu. Model wykorzystujący LSTM potrzebował znacznie więcej czasu na przetworzenie sekwencji, co skutkowało zacinaniem się obrazu z kamery oraz powolną predykcją gestów. Precyzja w przetwarzaniu statycznym, również nie została podniesiona. Zaobserwować można było spadek precyzji modelu na danych testowych. W związku z dokonaną analizą, nie zaleca się wykorzystywania LSTM do przetwarzania obrazu w czasie rzeczywistym. Model CNN, można jednakże rozwinąć o

podejście hybrydowe CNN + HMM, które polecają autorzy publikacji „Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition.”. Zastosowanie powyższego rozwiązania potencjalnie mogłoby podnieść poziom precyzji modelu, bez negatywnego wpływu na stabilność programu. Możliwe jest także dodanie Transformatorów takich jak ST- Transformer, które są aktualnym rozwiązaniem na rok 2024. Dodatkowo, model mógłby być wzbogacony o większą ilość próbek, wliczając w to zróżnicowanych modeli motion capture, różne warunki oświetlenia, różnego typu kamery oraz zmienne tła otoczenia. Powyższe elementy z pewnością poprawiłyby precyzję uczenia modelu. Utworzona platforma, mogłaby również zostać ulepszona poprzez dodanie elementów frontendowych, takich jak responsywność, a także możliwości obsługi wielu języków. Posiada ona również przygotowany moduł, który można rozwinąć o logowanie oraz rejestrację, wprowadzając tym samym system abonamentów (tokenów), znany chociażby z konkurencyjnych aplikacji SI. 61 Podsumowanie Platformy edukacyjne stale cieszą się popularnością. W grupie ww. platform znajdują się aplikacje do nauki języków obcych z których każdego dnia korzystają setki tysięcy osób, a ich stały rozwój pozwala na zwiększenie komunikatywności w środowisku międzynarodowym. Sytuacja ta jest analogiczna w przypadku zwiększenia świadomości ludzkiej na wszelkiego typu dysfunkcje organizmu. Strony internetowe coraz częściej dostosowują się do różnego rodzaju dyrektyw, chociażby takiej jak WCAG 2.0, które to w jasny sposób określają jak powinny wyglądać strony przyjazne dla osób z niepełnosprawnościami. Pomimo zwiększenia dostępności stron internetowych dla osób z dysfunkcjami, nadal istnieje problem wykluczenia społecznego w warunkach kontaktu bezpośredniego. Osoby, które są głuchonieme, potrzebują obecności tłumacza, który stale przekazuje informacje pomiędzy osobami pełnosprawnymi, a osobą dotknięta dysfunkcją. Niniejsza dysertacja łączy możliwości platform internetowych oraz tematyki niepełnosprawności, tworząc prototyp platformy do rozpoznawania polskiego języka migowego. Pomimo istnienia takowych aplikacji na rynku anglojęzycznym, nadal nie istnieje stabilne rozwiązanie na polskim rynku. W związku z powyższą informacją, praca ta jest innowacyjna, łącząc ze sobą prostotę obsługi, z zaawansowanymi technologiami takimi jak model sztucznej inteligencji rozpoznający gesty języka migowego w czasie rzeczywistym. W ramach niniejszej pracy dyplomowej, opracowany został projekt platformy internetowej, wraz z jej pełną implementacją przy zastosowaniu języków webowych. Dodatkowo na potrzeby dysertacji, autor utworzył kilka modeli sztucznej inteligencji, składających się z różnej ilości próbek opracowanych przez twórcę. Następnie każdy model został przetestowany pod względem wydajność oraz możliwości wykorzystania ich w aplikacji korzystającej z kamery internetowej w czasie rzeczywistym. Finalnie, utworzonych zostało 1800 nagrań o długości 4 sekund, przedstawiających osobę, wykonującą określony gest polskiego języka migowego. Najlepszym modelem okazał się program oparty na architekturze CNN, który wykazał się wysoką precyzją oraz szybkością działania. Należy również uwzględnić, że wszystkie elementy estetyczne w tym grafiki, wizualizacje oraz wykresy zostały opracowane przez autora pracy. Praca dyplomowa, może również zostać rozwinięta o dodatkowe elementy, takie jak responsywność oraz moduł logowania i rejestracji, pozwalający na wykorzystanie ww. dysertacji w celach materialnych. Sugerowane jest również wzbogacenie opracowanego 62 modelu o większą ilość próbek, opracowanych w różnych warunkach oświetleniowych, zmiennym otoczeniu, a także na innych osobach. Przebudowa modelu, dodając warstwę architektury HMM lub ST-Transformer również może zwiększyć precyzję wykrywanych gestów, a co za tym idzie poprawić działanie całej platformy edukacyjno-translatorycznej. 63 Bibliografia oraz Netografia 1. Camgoz, N. C., Koller, O., Hadfield, S., & Bowden, R. (2017). SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition. IEEE International Conference on Computer Vision (ICCV). 2. Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 7291-7299. 3. Huang, J., Zhou, W., Zhang, Q., Li, H., & Li, W. (2018). Attention-Based 3D-CNNs for Large- Vocabulary Sign Language Recognition. IEEE Transactions on Circuits and Systems for Video Technology, 29(9), 2822-2832. 4. Jamroży D.(2023): Aplikacja internetowa z graficznym interfejsem użytkownika opartym na technologii typu eye-tracking oraz speech recognition. [Praca inżynierska, Uniwersytet Rzeszowski] 5. Koller, O., Zargaran, S., Ney, H., & Bowden, R. (2015). Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition. In Proceedings of the British Machine Vision Conference (BMVC). 6. --- MediaPipe Documentation (n.d.). Google Developers. Retrieved from https://ai.google.dev/edge/mediapipe/solutions/guide?hl=pl 7. Pandey, A., Mishra, M., & Verma, A. K. (2022). Real-Time Sign Language Recognition Using Machine Learning and Neural Networks. IEEE Access. 8. Patel, K., Gil-González, A.-B., & Corchado, J. M. (2022). Deepsign: Sign Language Detection and Recognition Using Deep Learning. Electronics, 11(11), 1780. 9. Patel, M., & Shah, N. (2021). Real-Time Gesture-Based Sign

Language Recognition System. IEEE International Conference on Information Technology and Engineering (ICITE). 10. Pigou, L., Dieleman, S., Kindermans, P. J., & Schrauwen, B. (2015). Sign Language Recognition Using Convolutional Neural Networks. European Conference on Computer Vision Workshops (ECCVW). 11. Zhang, Z., & Liu, C. (2020). Skeleton-Based Sign Language Recognition Using Whole-Hand Features. IEEE Access, 8, 68827-68837. 12. https://www.wsb-nlu.edu.pl/pl/wpisy/wplyw-sztucznej-inteligencji-na-przyszlosc-pracy- nowe-perspektywy-i-wyzwania (22.08.2024) 64 13. https://www.tamoco.com/blog/blog-app-design-app-functionality-ux-ui/ (22.08.2024) 14. https://echoinnovateit.com/ux-or-ui-whats-more-important-in-app-development/ (22.08.2024) 15. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5(4), 115-133. 65 Spis ilustracji, tabel oraz wykresów Spis Ilustracji Rysunek 1 Porównanie technologii (Źródło: opracowanie własne) 11 Rysunek 2 Interfejs programu XAMPP 20 Rysunek 3 Etapy zbierania próbek (Źródło: opracowanie własne) 22 Rysunek 4 Uczenie maszynowe (GUI 1) (Źródło: opracowanie własne) 22 Rysunek 5 Uczenie maszynowe (GUI 2) (Źródło: opracowanie własne) 23 Rysunek 6 Uczenie maszynowe (GUI 2 - Struktura) (Źródło: opracowanie własne) 24 Rysunek 7 Uczenie maszynowe (GUI 3) (Źródło: opracowanie własne) 24 Rysunek 8 Uczenie maszynowe (GUI 3 - Wizualizacja) (Źródło: opracowanie własne) 25 Rysunek 9 Uczenie maszynowe (GUI 3 - Zakończenie przetwarzania danych) (Źródło: opracowanie własne) 26 Rysunek 10 Wzór algorytmu gradientu prostego 28 Rysunek 11 Uczenie maszynowe (GUI 4) (Źródło: opracowanie własne) 29 Rysunek 12 Uczenie maszynowe (GUI 4 - Zakończenie pracy) (Źródło: opracowanie własne) .. 30 Rysunek 13 Diagram aktywności (Uczenie maszynowe) (Źródło: opracowanie własne) 31 Rysunek 14 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 33 Rysunek 15 Interfejs graficzny aplikacji – zakładka Home (Źródło: opracowanie własne) 35 Rysunek 16 Zakładka Home (Źródło: opracowanie własne) 35 Rysunek 17 Slider - strona Index.php (Źródło: opracowanie własne) 36 Rysunek 18 Widok zakładki O projekcie (Project.php) (Źródło: opracowanie własne) 36 Rysunek 19 Widok zakładki O nas (About.php) (Źródło: opracowanie własne) 37 Rysunek 20 Uhonorowanie wsparcia (About.php) (Źródło: opracowanie własne) 38 Rysunek 21 Widok zakładki Dashboard (Dashboard.php) (Źródło: opracowanie własne) 38 Rysunek 22 Działanie systemu rozpoznawania gestu (Źródło: opracowanie własne) 39 Rysunek 23 Widok zakładki Tłumacz (Translator.php) (Źródło: opracowanie własne) 39 Rysunek 24 Wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) 40 Rysunek 25 Głosowe wyszukiwanie gestu (Translator.php) (Źródło: opracowanie własne) 40 Rysunek 26 Wizualizacja gestu (Źródło: opracowanie własne) 41 Rysunek 27 Prezentacja gestu "Dzięki" (Źródło: opracowanie własne) 42 66 Rysunek 28 Synchronizacja wizualizacji z obrazem w czasie rzeczywistym (Źródło: opracowanie własne) 42 Rysunek 29 Diagram przypadków użycia (Tłumacz) (Źródło: opracowanie własne) 43 Rysunek 30 Diagram aktywności (Tłumacz - kamera / wizualizacja) (Źródło: opracowanie własne) 44 Rysunek 31 Diagram aktywności (Tłumacz – Wizualizacja) (Źródło: opracowanie własne) 45 Rysunek 32 Widok zakładki Nauka Języka Migowego (Edu.php) (Źródło: opracowanie własne) 46 Rysunek 33 Gesty wybranej kategorii (Źródło: opracowanie własne) 46 Rysunek 34 Wykonanie poprawnego gestu (Źródło: opracowanie własne) 47 Rysunek 35 Zestawienie metody CNN oraz CNN + LSTM (Źródło: opracowanie własne) 49 Rysunek 36 Proces uczenia maszynowego CNN (Źródło: opracowanie własne) 51 Rysunek 37 Proces uczenia maszynowego CNN + LSTM (Źródło: opracowanie własne) 52 Spis Tabel Tabela 1 Wymagania systemowe cz1 18 Tabela 2 Wymagania systemowe cz2 18 Spis Wykresów Wykres 1 Przetwarzanie próbek CNN 49 Wykres 2 Czas przetwarzania danych – porównanie CNN – CNN + LSTM 50 Wykres 3 Uczenie modelu CNN 50 Wykres 4 Czas uczenia modelu – porównanie CNN – CNN + LSTM 52 Wykres 5 Zestawienie wykresów 54 Wykres 6 Analiza gestu Cześć - CNN 55 Wykres 7 Analiza gestu Dzięki – CNN 56 Wykres 8 Analiza gestu Dziękuję - CNN 57 Wykres 9 Analiza gestu Prosić – CNN 58 Wykres 10 Analiza gestu Proszę – CNN 59 Wykres 11 Analiza gestu Przepraszać – CNN 60 67 Streszczenie: Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego. Niniejsza dysertacja dotyczy utworzenia prototypu platformy do rozpoznawania polskiego języka migowego (PJM) w czasie rzeczywistym poprzez zastosowanie metod uczenia maszynowego, których efektem jest model sztucznej inteligencji (SI). Głównym celem pracy było opracowanie platformy, która umożliwi osobom niesłyszącym łatwiejszą komunikację z otoczeniem bez potrzeby korzystania z dodatkowego sprzętu czy też usług tłumacza, a także zwiększenie ludzkiej świadomości na temat dysfunkcji organizmu. W pracy dokonano również przeglądu technologii i metod związanych z rozpoznawaniem gestów, takich jak konwolucyjne sieci neuronowe (CNN) oraz narzędzia przetwarzania obrazu, między innymi MediaPipe i OpenPose. Autor dysertacji, zaprojektował oraz przeanalizował modele SI, które rozpoznawały gesty PJM na podstawie danych wideo, przetwarzanych oraz trenowanych w modelu CNN. W dalszej części prac, opracowano aplikację webową, która korzysta z

kamery internetowej w celu rozpoznania gestów użytkownika w czasie rzeczywistym. Przeprowadzone testy, zarówno w czasie rzeczywistym jak i na przygotowanych wcześniej danych wykazały, że aplikacja działa skutecznie, choć istnieją pewne wyzwania związane z precyzją rozpoznawania gestów, wliczając w to ograniczony dostęp do bazy gestów. Utworzona platforma oferuje potencjalne rozwiązania, które mogą wspierać osoby z dysfunkcjami słuchu oraz mowy w codziennej komunikacji. 68 Abstract: Prototype and Analysis of a Real-time Polish Sign Language Gesture Recognition Platform Using Machine Learning Method. This dissertation concerns the creation of a prototype of a platform for real-time recognition of Polish Sign Language (PJM) through the use of machine learning methods, which result in an artificial intelligence (AI) model. The main objective of the work was to develop a platform that would enable deaf people to communicate more easily with their surroundings without the need for additional equipment or interpreter services, as well as to increase human awareness of body dysfunctions. The paper also reviews technologies and methods related to gesture recognition, such as convolutional neural networks (CNNs) and image processing tools, including MediaPipe and OpenPose. Author of the dissertation, he designed and analyzed AI models that recognized PJM gestures based on video data processed and trained in a CNN model. Further on, a web application was developed that uses a webcam to recognize user gestures in real time. Tests, both in real time and on pre-prepared data, have shown that the application works effectively, although there are some challenges related to the precision of gesture recognition, including limited access to the gesture database. The created platform offers potential solutions that can support people with hearing and speech impairments in everyday communication. 69 70

Definicje

JSA – oznaczenie (skrót) określający Jednolity System Antyplagiatowy.

PRP – oznaczenie (skrót) określający Procentowy Rozmiar Podobieństwa. Jest to stosunek rozmiaru tekstu z uwzględnionymi fragmentami podobieństwa do całego rozmiaru tekstu pracy badanej wyrażony w procentach.

ORPPD – oznaczenie (skrót) określający Ogólnopolskie Repozytorium Pisemnych Prac Dyplomowych.

Analiza tekstu – jest to rozbiór tekstu na zestaw danych, który wyodrębnia określoną cechę np. znaki specjalne w tekście. Celem analizy tekstu jest pomoc przy wykryciu fałszowania i manipulacji w tekście badanej pracy.

Fragmenty innego stylu / Stylometria – jest to rodzaj analizy tekstu i polega na wykryciu fragmentów, które potencjalnie mógł napisać ktoś inny niż główny autor tekstu. Fragmenty tekstu o stylu odmiennym niż główny zostaną wykryte i zaznaczone, przy spełnionym założeniu, że min 70% tekstu napisane jest jednym głównym stylem.

Rozkład długości wyrazów – zależność wykazana na histogramie jako procentowa wartość stosunku liczby słów o określonej długości do liczby wszystkich słów w pracy badanej.

Najdłuższa fraza – długość frazy, podana w znakach, stanowi wielkość najdłuższej podobnej frazy, znalezionej we wskazanym dokumencie porównawczym.

Liczba znalezionych fraz o zadanej długości – liczba fraz, które mają długość nie krótszą niż podana liczba wyrazów w nagłówku.

❞ – podobieństwo może być kopią lub prawidłowym cytowaniem.

ⓘ – podobieństwo nie zostało wliczone do wyników badania antyplagiatowego.

Wnioski

Promotor (opiekun) po zapoznaniu się z wynikiem szczegółowych porównań stwierdza, że:

☑ praca nie wskazuje istotnych podobieństw wykrytych w ramach weryfikacji przez Jednolity System Antyplagiatowy;

☐ praca zawiera istotne podobieństwa wykryte w ramach weryfikacji przez Jednolity System Antyplagiatowy;

Uwagi:

.....

.....

.....

.....

☐ w pracy zastosowano manipulacje utrudniające badanie antyplagiatowe;

Promotor (opiekun) wnioskuje o:

☑ dopuszczenie pracy do obrony;

☐ niedopuszczenie pracy do obrony

....

data podpis promotora / opiekuna pracy