

**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

Spis treści

[Wstęp 5](#_Toc175250113)

[1. Przegląd literatury i analiza istniejących rozwiązań 7](#_Toc175250114)

[1.1. Przegląd technologii rozpoznawania gestów 7](#_Toc175250115)

[1.2. Przegląd metod uczenia maszynowego 7](#_Toc175250116)

[1.3. Wyzwania w rozpoznawaniu gestów języka migowego 7](#_Toc175250117)

[1.4. Analiza istniejących aplikacji dotyczących języka migowego 7](#_Toc175250118)

[2. Techniczne aspekty funkcjonowania aplikacji 8](#_Toc175250119)

[2.1. Wymagania systemowe 8](#_Toc175250120)

[2.1.1. Wymagania systemowe pierwszej części programu 8](#_Toc175250121)

[2.1.2. Wymagania systemowe drugiej części programu 9](#_Toc175250122)

[2.2. Model sztucznej inteligencji 9](#_Toc175250123)

[2.2.1. Zbieranie danych 9](#_Toc175250124)

[2.2.2. Analiza danych oraz utworzenie modelu 9](#_Toc175250125)

[2.2.3. Testowanie poprawności modelu 9](#_Toc175250126)

[2.3. Obsługa platformy 9](#_Toc175250127)

[2.3.1. Aplikacja webowa 9](#_Toc175250128)

[2.3.2. Obsługa modelu sztucznej inteligencji 10](#_Toc175250129)

[3. Budowa modelu rozpoznawania gestów 11](#_Toc175250130)

[3.1. Przygotowanie danych wejściowych 11](#_Toc175250131)

[3.2. Preprocessing 11](#_Toc175250132)

[3.3. Wybór architektury modelu 11](#_Toc175250133)

[3.4. Sieci neuronowe i ich zastosowanie 11](#_Toc175250134)

[3.5. Algorytmy klasyfikacji 11](#_Toc175250135)

[3.6. Trening i walidacja modelu 11](#_Toc175250136)

[3.7. Proces uczenia modelu 11](#_Toc175250137)

[3.8. Ocena skuteczności modelu 11](#_Toc175250138)

[4. Implementacja platformy 12](#_Toc175250139)

[4.1. Koncepcja wizualna platformy 12](#_Toc175250140)

[4.2. Mechanizmy komunikacji między frontendem a backendem 13](#_Toc175250141)

[4.2.1. Komunikacja serwer – użytkownik 13](#_Toc175250142)

[4.2.2. Integracja modelu AI z aplikacją webową (Mechanizmy komunikacji) 14](#_Toc175250143)

[4.3. Widoki aplikacji 14](#_Toc175250144)

[5. Ocena działania platformy 15](#_Toc175250145)

[5.1. Testy w rzeczywistych warunkach 15](#_Toc175250146)

[5.2. Skuteczność i dokładność rozpoznawania gestów 15](#_Toc175250147)

[5.3. Analiza błędów i propozycje ulepszeń 15](#_Toc175250148)

[Podsumowanie 16](#_Toc175250149)

[Bibliografia oraz Netografia 17](#_Toc175250150)

[Spis ilustracji, tabel oraz wykresów 18](#_Toc175250151)

# Wstęp

Obecny rozwój technologii pozwala na automatyzację wielu procesów, w tym procesów finansowych, administracyjnych oraz sprzedażowych. Wymienione elementy są związane stricte z obsługą biznesów, które umożliwiają komercyjny zarobek dla twórców oprogramowania. Algorytmy mają w tym przypadku ułatwić pracę lub całkowicie zastąpić osoby fizyczne w ich pracy. Dzięki takim praktykom pracodawcy, czy też różnego rodzaju organizacje zwiększają swoje dochody poprzez przyspieszeniu 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. Tendencja wzrostowa popularności rozwiązań opartych na SI jest stale obserwowana przez specjalistów, którzy stawiają różnego rodzaju hipotezy dotyczące przyszłości rozwoju sztucznej inteligencji. Zgodnie z opinią specjalistów z Wyższej Szkoły Biznesu National-Louis University, AI 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, oraz rozwoju.[10]

Istnieją również aspekty SI, 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.

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 stosunku do problematyki osób głuchoniemych nie ma zbyt wielu rozwiązań technologicznych, które mogą ułatwić ich życie w sposób nieinwazyjny.

Celem pracy dyplomowej jest utworzenie oraz analiza oprogramowania do rozpoznawania sekwencji ruchów w czasie rzeczywistym w oparciu o metody uczenia maszynowego, bez konieczności używania specjalistycznego sprzętu komputerowego. Zastosowane rozwiązania, wykorzystane zostaną w aplikacji do nauki oraz tłumaczenia polskiego języka migowego, które może wspierać osoby z dysfunkcjami w łatwiejszej adaptacji w środowisku osób pełnosprawnych, umożliwiając im łatwiejszą komunikację. 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.[3]

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 rozdziale tym znajdują się takie elementy jak wymagania systemowe, zalecane oprogramowanie zewnętrzne oraz wymagane biblioteki, które należy zainstalować na urządzeniu docelowym.

Rozdział trzeci porusza tematykę tworzenia modelu sztucznej inteligencji, przechodząc po każdym etapie jego tworzenia, zaczynając na 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 omawia ocenę działania platformy, skupiając się na testach w warunkach rzeczywistych, 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.

# Przegląd literatury i analiza istniejących rozwiązań

Rozdział ten porusza teoretyczną część pracy dyplomowej, tłumacząc podstawowe pojęcia, a także analizując istniejące rozwiązania.

## Przegląd technologii rozpoznawania gestów

## Przegląd metod uczenia maszynowego

## Wyzwania w rozpoznawaniu gestów języka migowego

## Analiza istniejących aplikacji dotyczących języka migowego

# Techniczne aspekty funkcjonowania aplikacji

Zaleca się, aby przed uruchomieniem aplikacji, zadbać o odpowiednie przygotowanie środowiska pracy. Każdy etap funkcjonowania oprogramowania posiada własne wymagania odnośnie hardware, software oraz środowiska programistycznego.

## 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.

### Wymagania systemowe pierwszej części programu

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 i5-10600K lub AMD Ryzen 5 3600 | Ośmiordzeniowy, np. Intel Core i7-9700K lub AMD Ryzen 7 3700X |
| Pamięć RAM | 16 GB RAM | 32 GB RAM |
| Karta graficzna (GPU) | Zintegrowana | Zintegrowana |
| Przestrzeń dyskowa | Minimum 20 GB wolnej przestrzeni na dysku SSD | SSD NVMe z minimum 100 GB wolnej przestrzeni |
| System operacyjny | Windows 10 lub nowszy / Linux (Ubuntu 20.04 lub nowszy) | Windows 10 Pro 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

### Wymagania systemowe drugiej części programu

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 (CPU) | Czterordzeniowy procesor, np. Intel Core i5-8265U lub AMD Ryzen 5 2500U | Sześciordzeniowy procesor, np. Intel Core i7-10750H lub AMD Ryzen 5 4600H |
| Pamięć RAM | 8 GB RAM | 16 GB RAM |
| Karta graficzna (GPU) | Zintegrowana, np. Intel UHD Graphics 620 lub AMD Radeon Vega 8 | Zintegrowana, np. Intel Iris Xe lub AMD Radeon Vega 11 |
| Przestrzeń dyskowa | Minimum 10 GB wolnej przestrzeni na dysku SSD | SSD NVMe z minimum 50 GB wolnej przestrzeni |
| System operacyjny | Windows 10 lub nowszy / Linux (Ubuntu 20.04 lub nowszy) | Windows 10 Pro lub nowszy / Linux (Ubuntu 20.04 LTS lub nowszy) |
| Inne | Python 3.8 lub nowszy | Python 3.8 lub nowszy |

Tabela 2 Wymagania systemowe cz2

## Model sztucznej inteligencji

Zgodnie z informacjami przedstawionymi w punkcie 2.1.1, 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.

### Zbieranie danych

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

### Analiza danych oraz utworzenie modelu

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

### Testowanie poprawności modelu

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

## Obsługa platformy

### Aplikacja webowa

### Obsługa modelu sztucznej inteligencji

Program „app.py” do poprawnego działania potrzebuje 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

# 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 odpowiednich danych.

## Przygotowanie danych wejściowych

Źródła danych

## Preprocessing

## Wybór architektury modelu

## Sieci neuronowe i ich zastosowanie

## Algorytmy klasyfikacji

## Trening i walidacja modelu

## Proces uczenia modelu

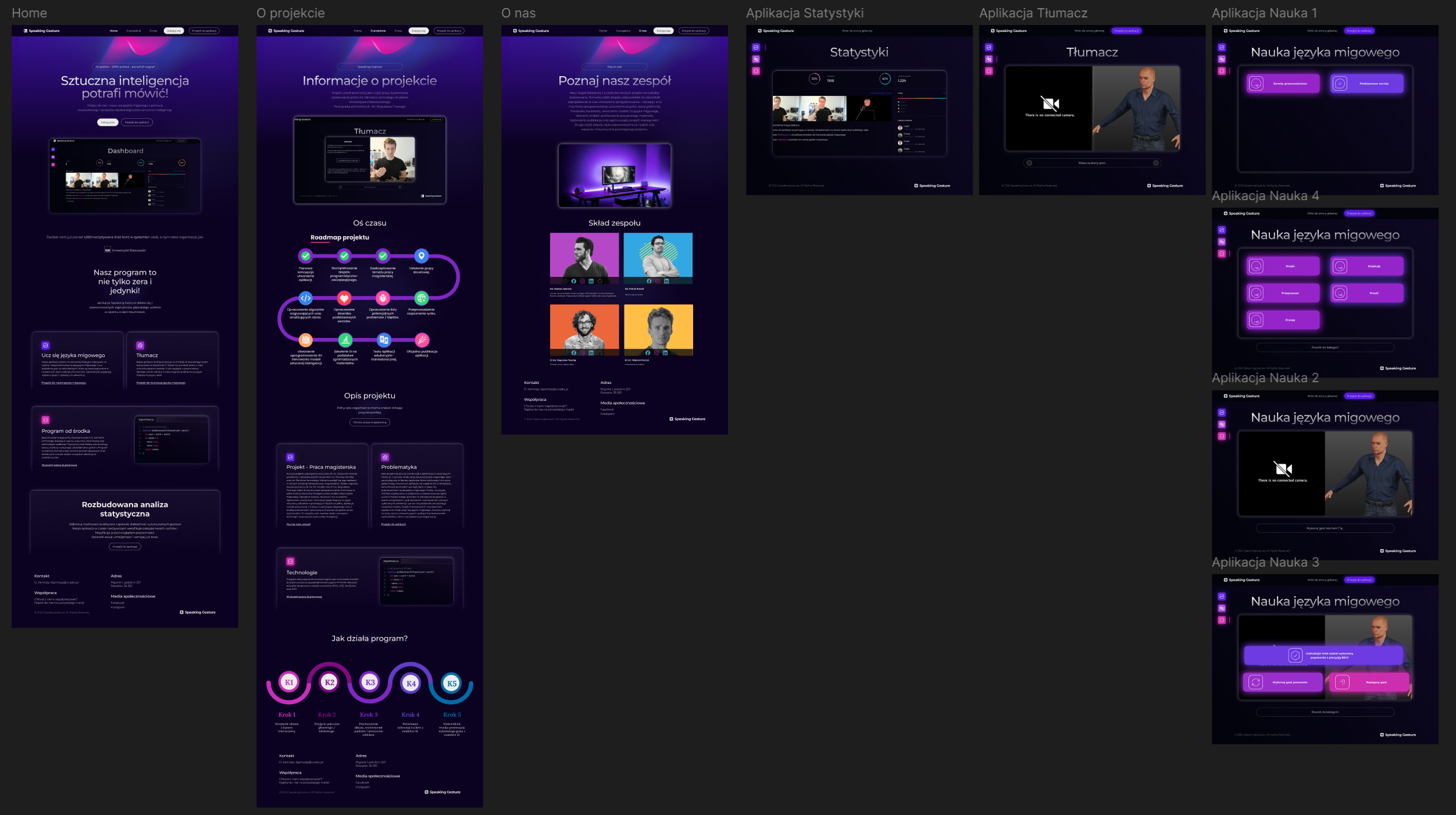
## Ocena skuteczności modelu

# 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.

## 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.[11] 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.[12] 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 1 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne)

## 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.

### 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 wymaga stałego połączenia z serwerem, dzięki czemu w sposób dynamiczny, zarządza dostępem do witryny oraz zawartością, 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.

### 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. 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.

## Widoki aplikacji

# Ocena działania platformy

## Testy w rzeczywistych warunkach

Wydajność w czasie rzeczywistym

## Skuteczność i dokładność rozpoznawania gestów

## Analiza błędów i propozycje ulepszeń

# Podsumowanie

Główne osiągnięcia pracy

Wnioski z przeprowadzonych badań

Propozycje dalszych badań i rozwoju platformy

# 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. 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.
3. Jamroży D.(2023): Aplikacja internetowa z graficznym interfejsem użytkownika opartym na technologii typu eye-tracking oraz speech recognition. [Praca inżynierska, Uniwersytet Rzeszowski]
4. 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).
5. Pandey, A., Mishra, M., & Verma, A. K. (2022). Real-Time Sign Language Recognition Using Machine Learning and Neural Networks. IEEE Access.
6. Patel, K., Gil-González, A.-B., & Corchado, J. M. (2022). Deepsign: Sign Language Detection and Recognition Using Deep Learning. Electronics, 11(11), 1780.
7. Patel, M., & Shah, N. (2021). Real-Time Gesture-Based Sign Language Recognition System. IEEE International Conference on Information Technology and Engineering (ICITE).
8. Pigou, L., Dieleman, S., Kindermans, P. J., & Schrauwen, B. (2015). Sign Language Recognition Using Convolutional Neural Networks. European Conference on Computer Vision Workshops (ECCVW).
9. Zhang, Z., & Liu, C. (2020). Skeleton-Based Sign Language Recognition Using Whole-Hand Features. IEEE Access, 8, 68827-68837.
10. <https://www.wsb-nlu.edu.pl/pl/wpisy/wplyw-sztucznej-inteligencji-na-przyszlosc-pracy-nowe-perspektywy-i-wyzwania> (22.08.2024)
11. <https://www.tamoco.com/blog/blog-app-design-app-functionality-ux-ui/> (22.08.2024)
12. <https://echoinnovateit.com/ux-or-ui-whats-more-important-in-app-development/> (22.08.2024)

# Spis ilustracji, tabel oraz wykresów

[Rysunek 1 Projekt interfejsu użytkownika (figma) (Źródło: opracowanie własne) 10](#_Toc175245829)

[Tabela 1 Wymagania systemowe cz1 8](#_Toc175250152)

[Tabela 2 Wymagania systemowe cz2 9](#_Toc175250153)

**Streszczenie:**

**Prototyp i analiza platformy do rozpoznawania gestów polskiego języka migowego w czasie rzeczywistym z zastosowaniem metod uczenia maszynowego.**

**Abstract:**

**Prototype and Analysis of a Real-time Polish Sign Language Gesture Recognition Platform Using Machine Learning Method.**

****