AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

Projekt dyplomowy

System wizyjny do odczytu materiałów drukowanych wspomagający osoby niewidzące
A vision system for reading printed materials supporting blind people

Autor: *Szymon Tokarz*
Kierunek studiów: *Automatyka i Robotyka*
Opiekun pracy: *dr inż. Maciej Rosół*

Kraków, 2025

Oświadczenie

Świadomy odpowiedzialności karnej za poświadczanie nieprawdy oświadczam, że niniejszą inżynierską pracę dyplomową wykonałem/łam osobiście i samodzielnie oraz nie korzystałem/łam ze źródeł innych niż wymienione w pracy.

Jednocześnie oświadczam, że dokumentacja oraz praca nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. z 2006 r. Nr 90 poz. 631 z późniejszymi zmianami) oraz dóbr osobistych chronionych prawem cywilnym. Nie zawiera ona również danych i informacji, które uzyskałem/łam w sposób niedozwolony.

Zaświadczam także, że niniejsza inżynierska praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

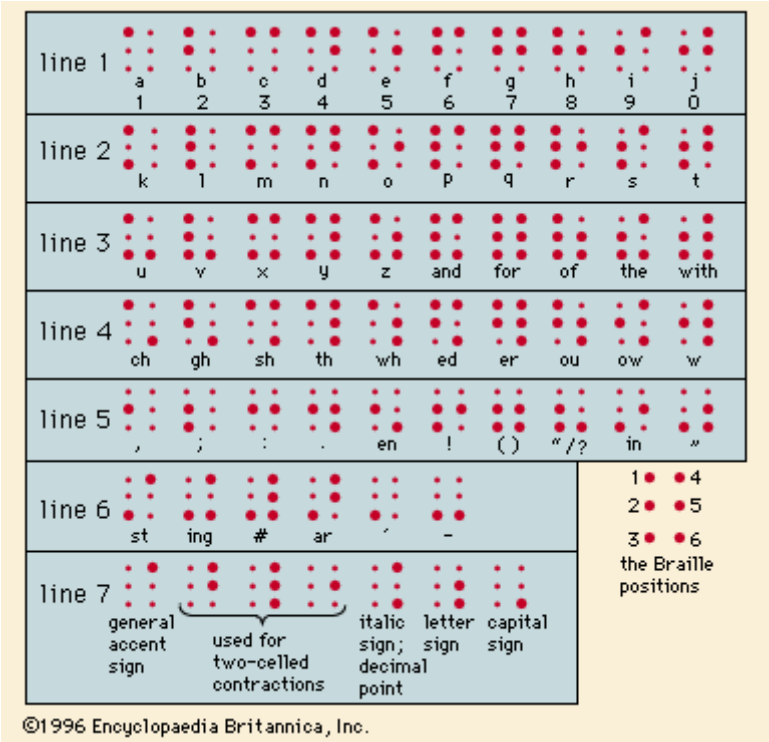
Spis treści

Rozdział 1 Wstęp 5
1.1 Cel Pracy 5
1.2 Zakres pracy 6
Rozdział 2 Przegląd istniejących rozwiązań 8
2.1 LyriQ Text-to-Speech Reader 8
2.2 Envision Low Vision Glasses 9
2.3 Text to Speech (TTS) 10
Rozdział 3 Projekt urządzenia 11
3.1 Raspberry Pi 4B 11
3.2 Kamera 5MP OV5647 13
3.3 Projekt obudowy 13
3.4 Komputer 18
Rozdział 4 Zagadnienia uczenia maszynowego 19
4.1 Sieć neuronowa 19
4.2 Transfer Learning 20
4.3 Konwolucyjne sieci neuronowe (CNN) 21
Rozdział 5 Proces przetwarzania obrazu 22
5.1 Obraz RGB i binaryzacja 24
5.2 Filtracja i usuwanie szumów 25
5.3 Segmentacja 29
Rozdział 6 Klasyfikacja za pomocą sieci neuronowej 33
6.1 Baza danych 33
6.2 Budowa sieci neuronowej 34
6.3 Budowa ResNet50 37
6.4 Modyfikacja sieci ResNet50 38
Rozdział 7 Implementacja programu w Raspberry Pi 39
Rozdział 8 Porównanie wyników 41

Rozdział 9 Testy funkcjonalne 46
Rozdział 10 Podsumowanie 50
Bibliografia 51

Rozdział 1 Wstęp [2]

Na przestrzeni ostatnich kilkudziesięciu lat pojawiały się najróżniejsze rozwiązania mające na celu pomoc osobom niewidzącym w czytaniu. Zdecydowanie najbardziej popularnym i rozpowszechnionym jest alfabet Braille'a, który wspiera w codziennym życiu wiele osób niepełnosprawnych. Zastąpienie typowych liter alfabetu zestawem wypukłych punktów przez Louis Braille'a w 1821 roku umożliwiło niewidomym czytanie i było największym skokiem w historii ich edukacji. Sam system jest uproszczeniem kodu wojskowego wymyślonego przez Charlesa Barbiera, który pozwalał na komunikację w nocy pomiędzy oficerami. Alfabet Braille'a, przedstawiony na rysunku 1.1, składa się z 63 znaków obejmujących maksymalnie 6 wypukłych kropek ustawionych w macierzy 3x2.



Rys.1.1 Alfabet Braille'a [2]

1.1 Cel Pracy

Celem pracy jest projekt i praktyczna realizacja systemu wizyjnego do rozpoznawania liter i cyfr w języku polskim i angielskim z drukowanych materiałów, służącego jako pomoc dla osób niewidomych. Rozpoznane litery i słowa będą następnie przekształcane na sygnał dźwiękowy, który może zostać odsłuchany przez te osoby. Urządzenie to wraz z oprogramowaniem powinno być łatwe oraz intuicyjne w użyciu.

1.2 Zakres pracy

Zakres pracy obejmuje zaprojektowanie oraz wykonanie sieci neuronowej służącej do rozpoznawanie znaków drukowanych, a następnie przekształcenie rozpoznanego ciągu znaków na sygnał dźwiękowy z możliwością jego odsłuchania. Zadana sieć neuronowa jest jednokierunkowa oraz może mieć maksymalnie 2 warstwy wewnętrzne. Oczekuje się przeprowadzania porównania między własnoręczną siecią neuronową, a gotowym modelem, który za pomocą metody *Transfer Learningu* został nauczony rozpoznawać znaki. Całe urządzenie będzie osadzone w obudowie, która zostanie zaprojektowana oraz wydrukowana przez autora.

Rozdział 1 opisuje podstawowe założenia oraz cele pracy dyplomowej. Opisane są również przewidywane wyniki pracy dyplomowej, cele i zakres pracy.

W rozdziale 2 autor porusza temat rynku istniejących rozwiązań sprzętowo-programowych dla osób niewidomych oraz prezentuje wybrane dostępne produkty.

Rozdział 3 opisuje założenia projektowe. Autor przedstawia w nim wybrane podzespoły wykorzystane w projekcie oraz pokazuje jego wstępną logikę. Autor przedstawia wykonany przez niego projekt obudowy wraz z koniecznymi rysunkami technicznymi i modelami 3D.

Rozdział 4 wyjaśnia zagadnienia związane z uczeniem maszynowym, które są wykorzystane w pracy oraz omawia ich zastosowanie.

Rozdział 5 jest poświęcony przygotowaniu obrazu. Autor opisuje i porównuje wybrane metody filtracji i segmentacji, które mogą zostać wykorzystane w projekcie oraz wybiera optymalną opcję.

Rozdział 6 jest poświęcony budowie przygotowanej sieci neuronowej oraz wykonanej bazie danych. Autor porusza w nim tematy z zakresu rodzajów warstw w konwolucyjnych sieciach neuronowych oraz przytacza wartości hiperparametrów. Opisuje również model ResNet 50 wykorzystany do Transfer Learningu oraz omawia, jakie modyfikacje zostały wprowadzone do tej sieci.

W rozdziale 7 przedstawione są rezultaty uzyskane za pomocą obydwóch modeli oraz zostało przeprowadzone porównanie tych wyników. Autor omawia w nim wady i zalety wykorzystanych sieci w kontekście problemu opisywanego w pracy.

Rozdział 8 omawia etapy złożenia oraz połączenia wszystkich części projektu opisywanych w pracy. Przedstawione zostają metody implementacji projektu oraz syntezy tekstu na mowę.

Rozdział 9 opisuje aplikację wykonaną w celu ułatwienia testowania urządzenia. Poruszane są w nim treści z zakresu narzędzi użytych do tworzenia GUI oraz zalety jego stworzenia. Przedstawione są testy funkcjonalne urządzenia, które mają na celu sprawdzenie poprawności działania w różnych warunkach.

Rozdział 2 Przegląd istniejących rozwiązań

Obecnie na rynku istnieje kilka interesujących rozwiązań wspomagających czytanie i rozumienie tekstu osobom niewidomym, jednak każde z nich ma swoje wady i zalety. Narzędzia rozpoznające tekst powinna cechować duża dokładność oraz możliwość identyfikacji znaków drukowanych jak i pisma ręcznego. Problem z dokładnym sklasyfikowaniem konkretnych liter i cyfr pisma ręcznego sprawia różnorodność charakterów pisma i nie tylko, ponieważ znaczenie ma również nawet kolor długopisu i strony, który może powodować zaburzenia odczytu przez urządzenie. Kolejnym ważnym czynnikiem jest cena produktu. Osoby niepełnosprawne, w tym ludzie niewidomi, mają ograniczony budżet związany ze swoją przypadłością, więc kwota, którą powinni zapłacić za urządzenie nie powinna być duża. Jednak z mniejszą ceną naturalnie wiąże się gorsza jakość części *hardware'u,* takich jak kamera bądź skaner. Użycie gorszego sprzętu oznacza również zmniejszoną dokładność rozpoznawania znaków. Z tych względów rynek narzędzi rozpoznających tekst jest dosyć zróżnicowany.

2.1 LyriQ Text-to-Speech Reader [1]

Urządzenie firmy Zyrlo (Rys.2.1), w odróżnieniu od narzędzia wykonanego przez autora, wykorzystuje skaner w celu zwiększenia dokładności rozpoznawania tekstu. Użycie skanera powoduje, że LyriQ jest dosyć sporym i relatywnie ciężkim obiektem. Waży około 1,3 kg (3 lbs), ma możliwość złożenia na płasko, co faktycznie ułatwia transport. Najbardziej odstraszającym czynnikiem w samym produkcie jest niewątpliwie cena, która mieści się w granciach 2300 – 2500 $, czyli w przeliczeniu na polską walutę to około 9200 – 10000 zł. Dodatkowo trzeba wziąć pod uwagę koszty wysyłki. Wysoka cena jest spowodowana jakością produktu. LyriQ Text-to-Speech Reader błyskawicznie rozpoczyna czytanie i jest intuicyjny w użyciu. Urządzenie wykorzystuje sztuczną inteligencję – Lyriq AI, która jest w stanie rozpoznać pismo odręczne jak również materiały drukowane takie jak rachunki czy paragony. Dzięki użyciu skanera dokładność urządzenia Lyriq wynosi około 98%. Posiada także możliwość skanowania kodów kreskowych. Urządzenie ma możliwość czytania w 20 różnych językach, domyślnie zainstalowano angielski i hiszpański.

Rys.2.1 Urządzenie *LyriQ [1]*

2.2 Envision Low Vision Glasses [3]

Ciekawym rozwiązaniem są okulary firmy Envision (Rys.2.2). Zaprojektowane z myślą o komforcie użytkownika, okulary poza rozpoznawaniem teksu zostały wyposażone w szereg różnych funkcji, takich jak, rozpoznawanie kolorów, skanowanie dokumentów, czy nawet rozpoznawanie ludzi. Urządzenie zostało zaprojektowane na podstawie *Google Glass Enterprise Edition 2* oraz wyposażone w lekkie tytanowe oprawki. Największą zaletą produktu firmy Envision jest jego wygoda. Samo urządzenie można zamówić w różnych opcjach, które oczywiście mają inne ceny w zależności od wyposażonych funkcji. Najbardziej podstawowa wersja okularów Envsion kosztuje około 8 500 zł. Okulary firmy Envision wykorzystują sztuczną inteligencję zapewnianą przez Google. Pozwala ona na rozpoznanie tekstu z dokładnością 95% - 99% w dobrych warunkach oświetleniowych. Urządzenie ma nawet możliwość połączenia z Chatem GPT-4. Wykorzystano w nim kamerę 8MP z 80º polem

widzenia, które przetwarza obrazy w czasie rzeczywistym, co pozwala na dokładne rozpoznanie tekstu.

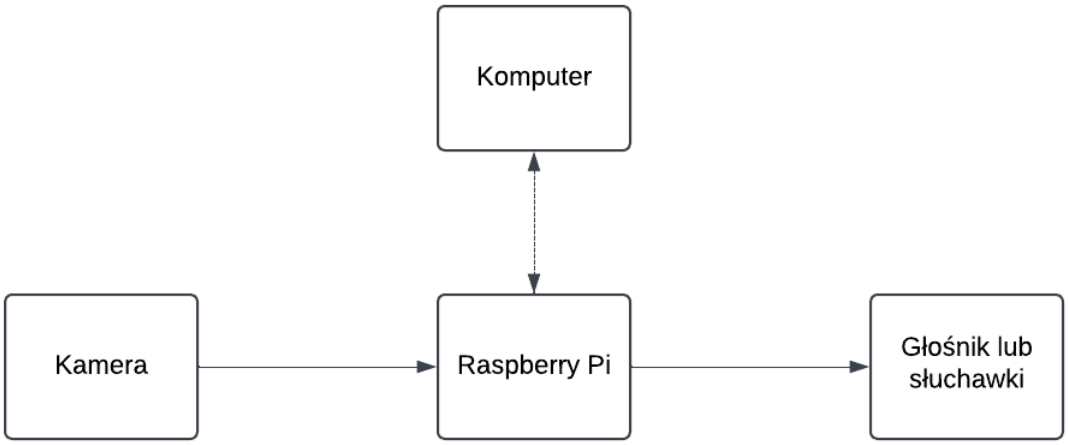
Rys.2.2 *Envision Low Vision Glasses [3]*

2.3 Text to Speech (TTS) [4]

Bardzo popularnym rozwiązaniem jest użycie *Text to Speech (TTS)*. Jest to rozszerzenie wyprodukowane przez Google wykorzystujące sztuczną inteligencję tej samej firmy. Wykorzystanie tej wtyczki jest bardzo proste, wystarczy mieć telefon i zrobić zdjęcie strony. Google zapewnia możliwość czytania w ponad 50 językach operujących na różnych alfabetach. Narzędzie ma również opcję wyboru lektora. Wtyczka sprawia, że użytkownikowi wystarczy telefon komórkowy, aby pomóc osobie niewidomej. Google TTS wykorzystuje technologię WaveNet, stworzoną przez firmę DeepMind, do generacji dźwięku z bardzo naturalnym brzmieniem.

Rozdział 3 Projekt urządzenia

Jak wspomniano we wstępie, zaproponowane urządzenie wspomagające osoby niewidzące powinno być tanie i zapewniać prostotę użytkowania. Stąd założenia projektowe koncentrują się na ogólnodostępnych, budżetowych komponentach. Schemat blokowy przedstawiający koncepcję budowy urządzenia przedstawiono na Rys.3.1. Zawiera on minikomputer Raspberry Pi, kamerę, wyjście audio do podłączenia głośników lub słuchawek oraz komputer klasy PC.



Rys.3.1 Poglądowy schemat blokowy urządzenia

Obraz będzie pobierany z kamery i przesłany na platformę Raspberry Pi, gdzie zostanie przetworzony oraz klasyfikowany. Raspberry Pi zostanie połączone bezprzewodowo za pomocą sieci WiFi z komputerem osobistym, gdzie będzie można zobaczyć podgląd z kamery oraz dokonać wyboru, który typ modelu wykorzystać do klasyfikacji. Rozpoznany tekst zostanie przekonwertowany na dźwięk i wysłany, za pomocą Bluetooth, do głośnika lub słuchawek.

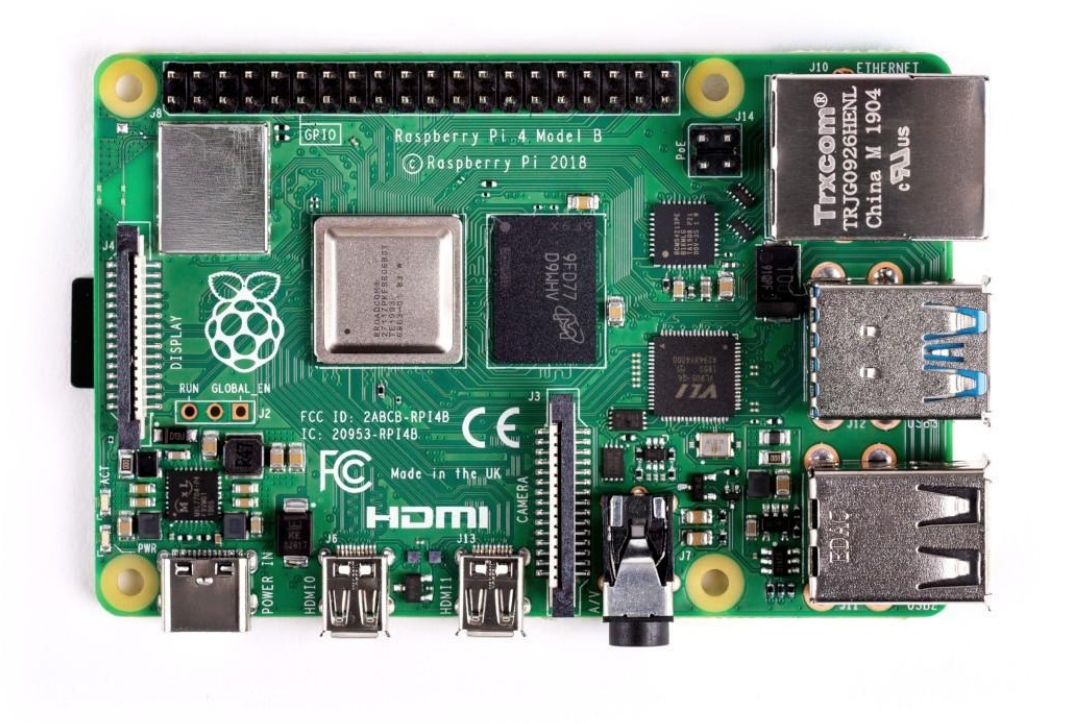
3.1 Raspberry Pi 4B

Do wykonania urządzenia autor zdecydował się na użycie minikomputera Raspberry Pi 4B (Rys.3.2) ze względu na jego kompaktowość oraz niski koszt. Dodatkową zaletą jest również możliwość wykorzystania narzędzi w środowisku Matlab/Simulink takich jak, np. MATLAB Support Package for Raspberry Pi Hardware, które są kompatybilne z platformą Raspberry Pi i bardzo ułatwiają programowanie urządzenia. Wykorzystano system operacyjny

Raspberry Pi OS w wersji *bookworm*. W tabeli 3.1 zawarto specyfikację techniczną użytej platformy sprzętowej.

Tabela 3.1 Podstawowe parametry techniczne Raspberry Pi 4B [22]

Procesor	Cortex-A72
Liczba rdzeni	4
Liczba portów USB 3.0	2
Liczba portów USB 2.0	2
Liczba portów Gigabit Ethernet	1
Bluetooth	5.0
Zasilanie	5V/3A
RAM	4GB



Rys.3.2 Raspberry Pi 4B [22]

3.2 Kamera 5MP OV5647

Do akwizycji obrazów z drukowanych dokumentów tekstowych wybrano kamerę 5MP OV5647 (Rys.3.3) ze względu na kompatybilność z platformą Raspberry Pi 4B oraz wysoką jakość obrazu w porównaniu do ceny. Dane techniczne użytej kamery są widoczne w tabeli 3.2.

Tabela 3.2 Dokumentacja kamery 5MP OV5647 [23]

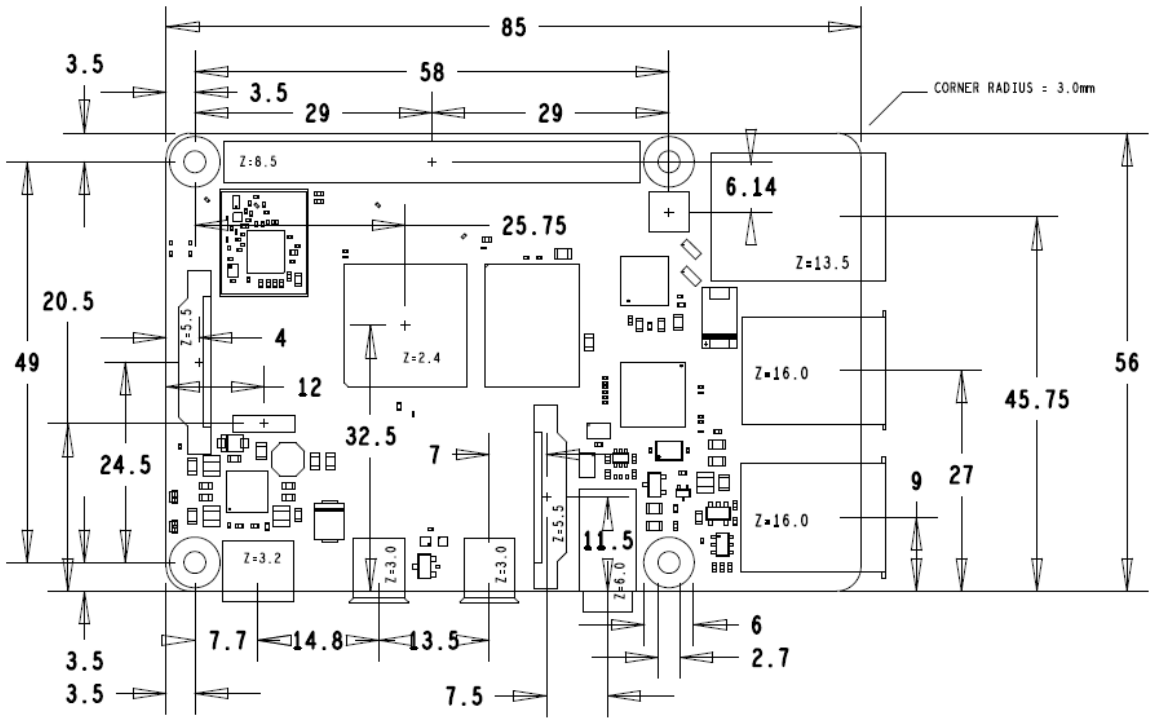
Liczba MP	5
Aktywne piksele	2592x1944
Tryby wideo	1080p30,720p60,640x480p60/90
Ogniskowa	3,60 mm
FoV	54x41
Rozmiar piksela	1,4µm x 1,4µm

Rys.3.3 Kamera 5MP OV5647 [23]

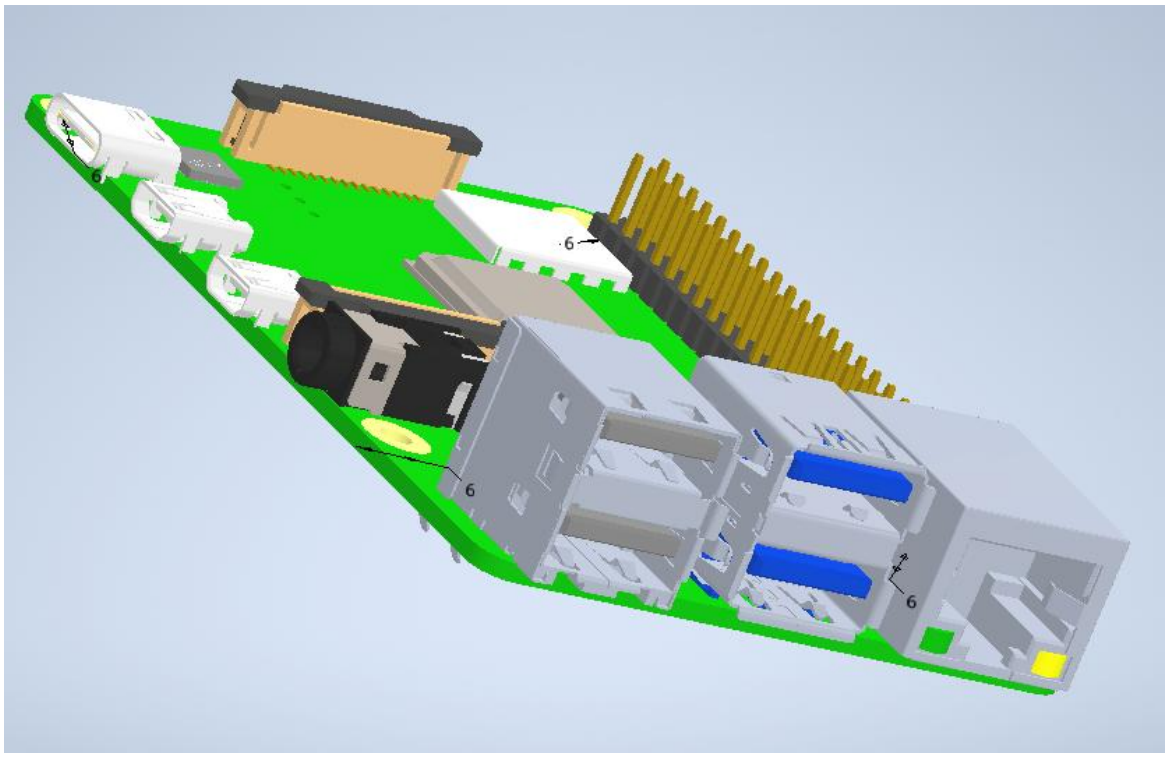
3.3 Projekt obudowy

Komponenty użyte do budowy urządzenia zostały umieszczone w zaprojektowanej obudowie. Na tym etapie użyto program Inventor Professional 2025 firmy Autodesk.

Wykonano w nim model 3D obudowy, który później został użyty do druku 3D. Cała obudowa składa się z części górnej i dolnej, które zostały połączone śrubami M2,6. Do projektu obudowy wykorzystano gotowy model 3D Raspberry Pi 4B, który został pobrany ze strony GrabCad (Rys.3.4) (Rys.3.5).

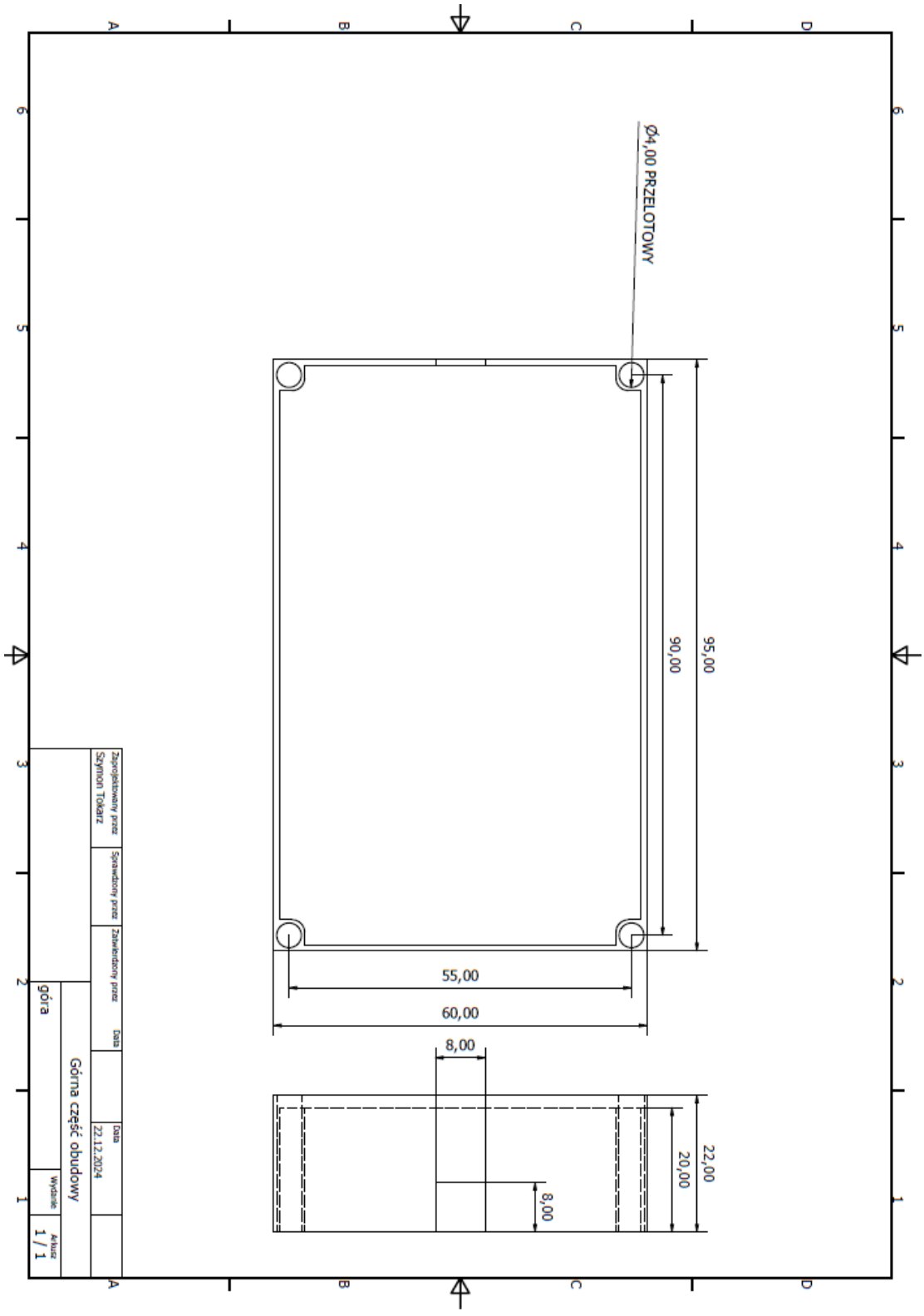


Rys.3.4 Rysunek techniczny modelu Raspberry Pi 4B [20]

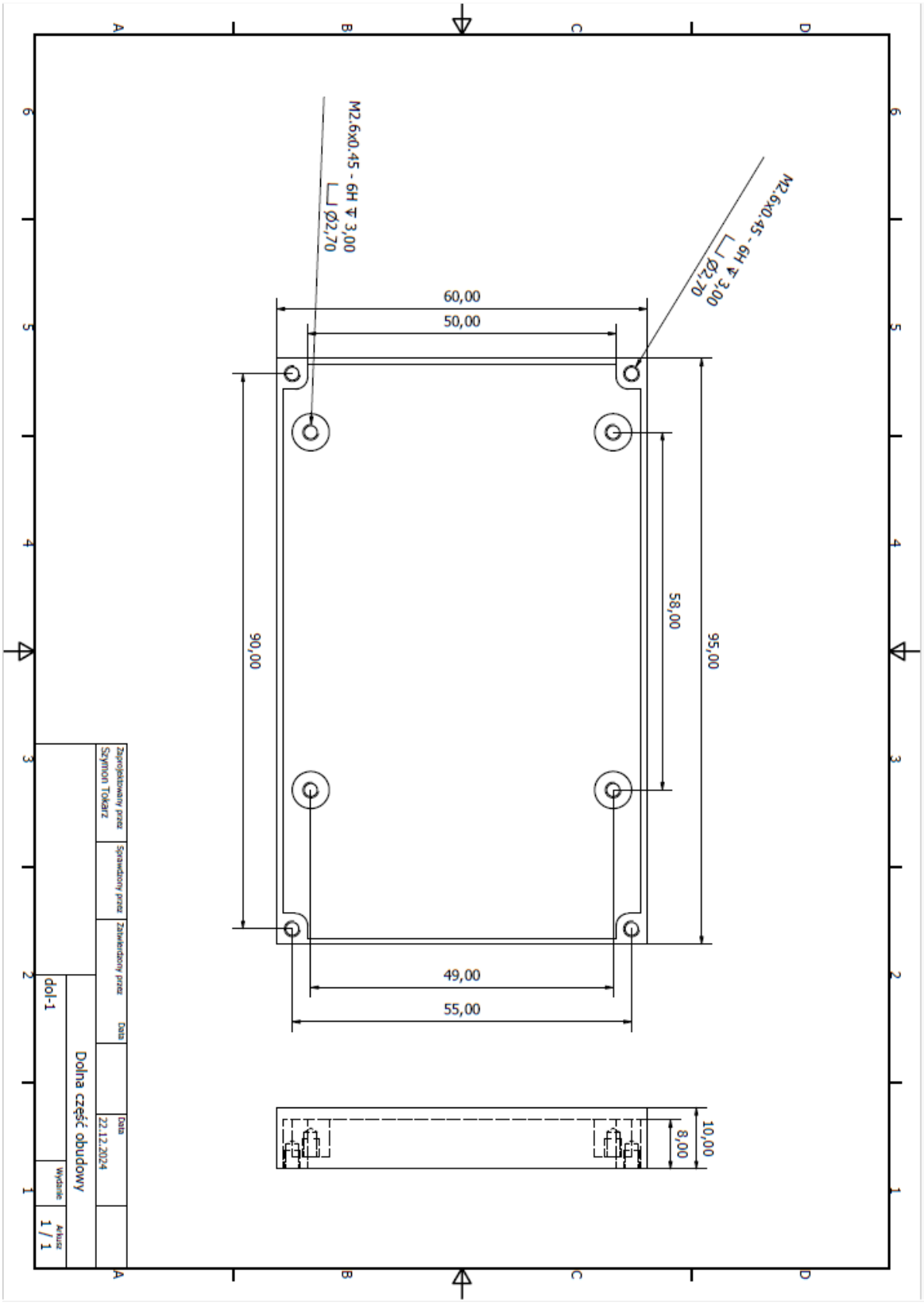


Rys.3.5 Model 3D Raspberry Pi 4B [20]

Po zapoznaniu się z modelem Raspberry Pi i upewnieniu się, że wymiary są poprawne rozpoczęto projektowanie części obudowy. Wykonane modele części górnej i dolnej wraz z rysunkami technicznymi (uwzględniającymi model złożeniowy) przedstawiono odpowiednio na Rys.3.6 i Rys.3.7.



Rys.3.6 Rysunek techniczny górnej części obudowy



Rys.3.7 Rysunek techniczny dolnej części obudowy

3.4 Komputer

W celu wykonania projektu należało wykorzystać komputer osobisty, który został użyty do zaprogramowania minikomputera Raspberry Pi oraz do przeprowadzenia testów funkcjonalnych urządzenia. Do napisania kodu programu, obsługującego urządzenie, wykorzystano głównie środowisko Matlab/Simulink 2024b wraz z następującymi przybornikami:

- Audio Toolbox,
- Computer Vision Toolbox,
- Deep Learning Toolbox,
- Image Processing Toolbox,

oraz dodatkowe pakiety sprzętowe (ang. *Hardware Package*) służące do obsługi Raspberry Pi:

- Matlab Support Package For Raspberry Pi Hardware,
- Simulink Support Package For Raspberry Pi Hardware.

W celu stworzenia bazy danych, gromadzącej dane potrzebne do nauki sieci neuronowej, użyto języka Python 3.9. Jako środowisko rozwojowe tego języka wykorzystano IDE PyCharm 2023.2.2.

Komputer zostanie połączony przez sieć WiFi z urządzeniem i będzie odpowiedzialny za wybór rodzaju modelu do rozpoznawania tekstu. Stworzona będzie również aplikacja, na której zostanie wyświetlony aktualny obraz z kamery.

Rozdział 4 Zagadnienia uczenia maszynowego

Uczenie maszynowe (ang. *Machine Learning*) to dział sztucznej inteligencji, który polega na tym, że komputery mogą zbierać wiedzę z danych i na ich podstawie rozpoznawać wzorce [11]. Rozpoznaje się dwa główne rodzaje uczenia maszynowego: nadzorowane i nienadzorowane. Różnica pomiędzy nimi polega na tym, że w uczeniu nadzorowanym dane są etykietowane, a w nienadzorowanym nie. Taki zabieg sprawia, że uczenie nadzorowane jest dokładniejsze i szybsze w treningu, za to przy uczeniu nienadzorowanym można wykorzystywać niepełne bazy danych.

W uczeniu maszynowym można wykorzystać najróżniejsze algorytmy. Najbardziej popularne to:

- algorytmy regresyjne,
- algorytmy wykorzystujące drzewa decyzyjne,
- algorytm Bayesa,
- sieci neuronowe.

W dalszej części tego rozdziału zostaną omówione podstawowe zagadnienia, takie jak sieci neuronowe i *Transfer Learning,* które to zostały wykorzystane w projekcie.

4.1 Sieć neuronowa [7]

„System przeznaczony do przetwarzania informacji, którego budowa i zasada działania są w pewnym stopniu wzorowane na funkcjonowaniu fragmentów rzeczywistego (biologicznego) systemu nerwowego. Na przesłankach biologicznych oparte są schematy sztucznych neuronów wchodzących w skład sieci oraz (w pewnym stopniu) jej struktura. Jednak schematy połączeń neuronów w sieci neuronowej są wybierane arbitralnie, a nie stanowią modelu rzeczywistych struktur nerwowych.” [8]

W kontekście rozpoznawania znaków drukowanych, sieci neuronowe są wykorzystywane już od prawie 30 lat. Pierwszymi, którzy je zastosowali byli Rost i Kwatra [5][6]. Wytrenowali swoje sieci w celu rozpoznawania angielskich samogłosek. Od tego momentu informatykom z całego świata udało się zaprogramować sieci neuronowe

rozpoznające takie języki jak mandaryński, mongolski, urdu lub nawet staro-cerkiewno-słowiański z bardzo dużą dokładnością.

Najbardziej popularne sieci neuronowe służące do OCR (ang. *Optical Character Recognition*) są kilkuwarstwowe z propagacją wsteczną. Zgodnie z pozycją [8] „Propagacja wsteczna opiera się na koncepcji poprawiania na każdym kroku procesu uczenia wartości korekty wag na podstawie oceny błędu popełnianego przez każdy neuron podczas uczenia sieci".

4.2 Transfer Learning [9]

Transfer Learning jest techniką wykorzystywaną w Machine Learningu polegającą na ponownym użyciu modelu do zwiększenia jakości lub szybkości rozwiązania dla podobnego zadania. Definicja brzmi następująco:

Dla dziedziny źródła D_s i zadania uczącego T_s, dziedzina celu D_t i zadania uczącego T_t, *Transfer Learning* ma na celu poprawę efektów uczenia funkcji przewidywania celu $f_t(\cdot)$ w dziedzinie D_t wykorzystując wiedzę w dziedzinie D_s i zadaniu T_s, gdzie $D_s \neq D_t$ oraz $T_s \neq T_t$.[9][10]

W swojej pracy Pan i Yang dzielą *Transfer Learning* na 3 rodzaje (Tabela 4.1):

- indukcyjny,
- nienadzorowany,
- transdukcyjny.

Tabela 4.1 Rodzaje *Transfer Learningu [9]*

Ustawienia uczenia		**Dziedzina źródła i celu**	**Zadanie źródła i celu**
Tradycyjny *Machine Learning*		Taka sama	Takie same
Transfer Learning	Indukcyjny	Taka sama	Inne, ale spokrewnione
	Nienadzorowany	Inna, ale spokrewniona	Inna, ale spokrewnione
	Transdukcyjny	Inna, ale spokrewniona	Takie same

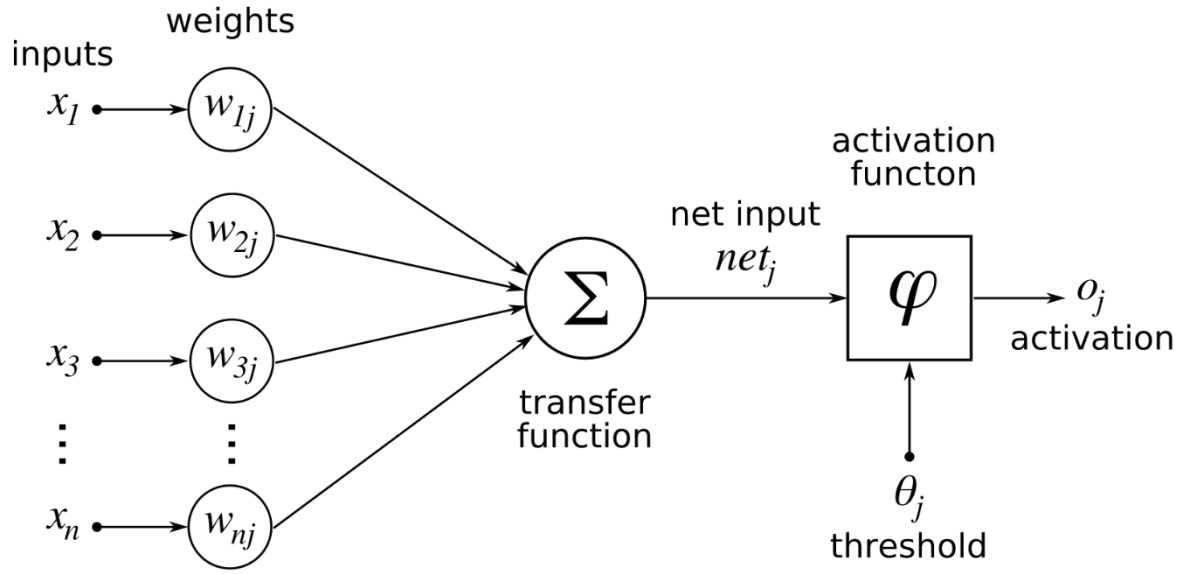
Obecnie *Transfer Learning* jest wykorzystywany głównie przy analizie tekstu oraz obrazów ze względu na łatwość w implementacji, jakość prezentowanych rozwiązań oraz brak potrzeby dużej bazy danych. Kolejną zaletą *Transfer Learningu* jest fakt, że istnieje szeroki wachlarz wytrenowanych sieci neuronowych, które można wykorzystać jako źródło.

4.3 Konwolucyjne sieci neuronowe (CNN) [24]

Konwolucyjna sieć neuronowa jest przykładem sieci typu *feed forward,* co oznacza, że dane przepływają tylko w jednym kierunku. Typowe sieci CNN składają się z różnych rodzajów warstw, jak np.:

- warstwa konwolucji,
- warstwa aktywacji,
- warstwa poolingu,
- warstwa w pełni połączona (ang. *fully connected layer*).

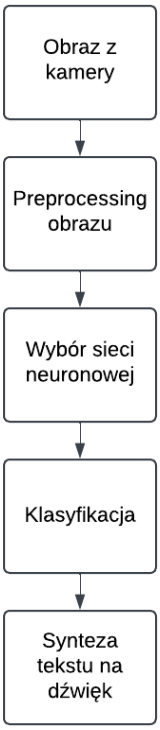
Sieć CNN składa się z połączenia kilku warstw konwolucyjnych, po których znajduje się warstwa poolingu powtórzona kilkukrotnie. Na samym końcu znajduje się warstwa w pełni połączona. Architekturę takiej sieci przedstawiono na rysunku 4.1. Dokładny opis poszczególnych warstw sieci CNN, wykorzystanej w pracy, znajduje się w rozdziale 6.



Rys.4.1 Schemat sieci konwolucyjnej [25]

Rozdział 5 Proces przetwarzania obrazu

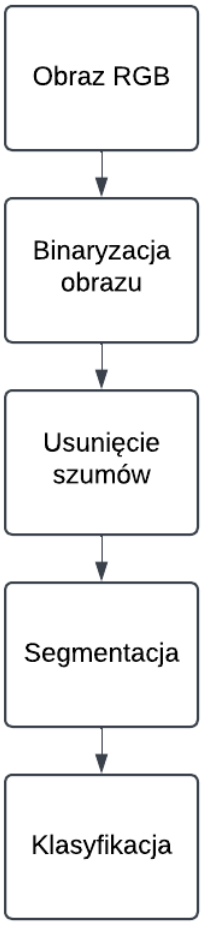
Program uruchamiany na platformie Raspberry Pi uwzględnia wszystkie etapy od pobrania obrazu z kamery do syntezy dźwięku na podstawie rozpoznanego tekstu. Schemat blokowy logiki programu przedstawiono na Rys.5.1.



Rys.5.1 Schemat blokowy logiki programu

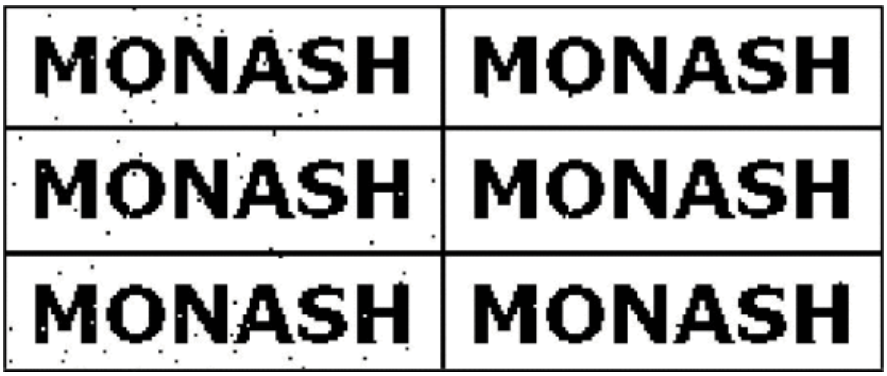
Schemat przedstawia kolejność wykonywanych akcji/operacji wymaganych do poprawnego działania urządzenia. Obraz z kamery jest pobierany i poddawany preprocessingowi, czyli obróbce, tak aby model mógł rozpoznać znaki z lepszą dokładnością. Następnie jest dokonywany wybór modelu, za pomocą którego ma zostać przeprowadzona klasyfikacja. Ostatecznie wykonywana jest synteza tekstu na dźwięk i podany sygnał jest wysłany na odbiornik.

W trakcie procesu przetwarzania obrazu dla sieci neuronowej, powinny zostać usunięte zakłócenia i szumy wynikające z błędów oraz niedoskonałości obrazu. W tym celu obraz należy poddać odpowiedniej obróbce graficznej, w kolejności pokazanej na Rys.5.2.



Rys.5.2 Schemat blokowy dla przetwarzania obrazów

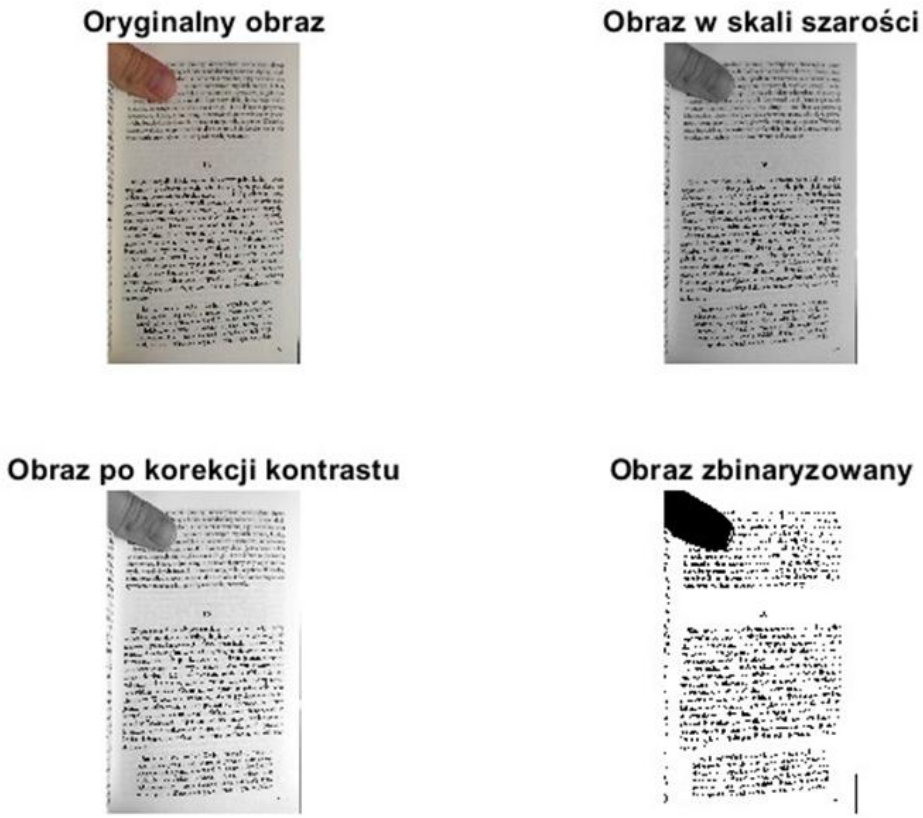
Najpierw obraz RGB musi zostać poddany binaryzacji, czyli konwersji do reprezentacji czarno-białej. Następnym etapem jest usunięcie szumów. Proces polega na analizie obrazu binarnego i znalezieniu pikseli, które nie pasują do ich otoczenia. Na Rys.5.3 widać, że kilkanaście pikseli nie należy do liter, ale istnieją w obrazie. Piksele te trzeba usunąć, ponieważ zaburzają klasyfikację sieci neuronowej. Kolejną etapem jest odpowiednia segmentacja obrazu na poszczególne litery. Ostatnim krokiem jest klasyfikacja obrazu do odpowiedniej litery.



Rys.5.3 Przykład usuwania szumów [7]

5.1 Obraz RGB i binaryzacja

Urządzenie jest obsługiwane przez kamerę o rozdzielczości 5 MP, więc obraz może być nieznacznie zamazany. W tym celu, aby uzyskać jak najlepszą binaryzację całego obrazu, autor najpierw przekształca go do odcieni szarości, po czym dokonuje korekcji kontrastu (Rys.5.4). Dzięki temu zabiegowi łatwiej jest dobrać uniwersalny próg binaryzacji, ponieważ różnice pomiędzy tłem, a tekstem są zdecydowanie większe. Przeprowadzenie binaryzacji jest konieczne, ponieważ ułatwia analizę obrazu przez sieć neuronową. Użycie tej operacji zwiększa dokładność i szybkość klasyfikacji.



Rys.5.4 Operacje przeprowadzone na obrazie

5.2 Filtracja i usuwanie szumów [12]

Po binaryzacji obrazu należy przefiltrować go ze względu na występujące szumy. W tym celu autor rozważał użycie następujących filtrów:

- filtr uśredniający,
- filtr medianowy,
- filtr Wienera,
- filtr korzystający z operacji morfologicznych.

Filtr uśredniający poprawia wartość piksela na podstawie średniej wartości pikseli w jego otoczeniu. R_{xy} reprezentuje sąsiedztwo okna podobrazu o wymiarach $m \times n$, o środku w punkcie (x, y). Dla poziomu szarości piksela (x, y), filtr zmienia poziom szarości $J(x, y)$ ze szczegółami otaczającego piksela. Opisuje to równanie:

$$J(x,y) = \frac{1}{M \cdot N} \sum_{(u,v)\in R_{xy}} K(u,v)$$

Filtr medianowy jest nieliniową metodą wykorzystującą wartość mediany wartości szarości otaczających pikseli. Ten filtr jest w stanie usunąć zaburzenia bez rozmazywania krawędzi. Mediana jest obliczana po posortowaniu wartości pikseli i wybierana jest z nich środkowa wartość. Filtr może wybierać różne rozmiary otoczenia, jednak najbardziej popularne są 3 × 3 lub 5 × 5. Filtr ten jest zdefiniowany równaniem:

$$J(x,y) = \text{median}_{(u,v)\in R_{xy}}\{K(u,v)\}$$

Filtr Wienera jest przykładem adaptacyjnego filtru, który działa w oparciu o minimalizację błędu średniokwadratowego (MSE). Jego zaletą jest lepsze działanie niż np. filtr średniej zwłaszcza w obszarach o zróżnicowanej intensywności, jednak wymaga większego nakładu czasowego. Równanie opisujące filtr Wienera ma następującą postać:

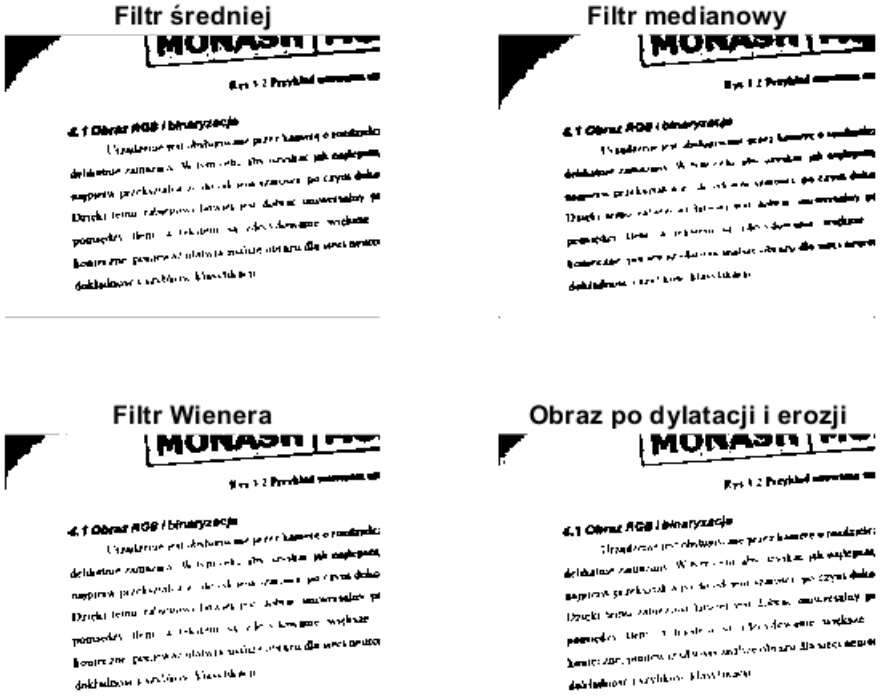
$$\hat{I}(x,y) = \frac{\sigma_I^2}{\sigma_I^2+\sigma_N^2} \cdot I(x,y),$$

gdzie:

- $\hat{I}(x,y)$: Wyjściowy (oczyszczony) obraz,
- $I(x,y)$: Sygnał wejściowy z szumem,
- σ_I^2: Wariancja obrazu oryginalnego,
- σ_N^2: Wariancja szumu.

Istnieje również możliwość przefiltrowania obrazu działając wyłącznie na operacjach morfologicznych. Wymaga to użycia funkcji *bwareaopen*, która jest dostępna w pakiecie Image Processing Toolbox programu MATLAB. Ta funkcja znajduje obiekty o rozmiarach zadanych jako argument i je usuwa. Tę operację należy wykonać na obrazie binarnym oraz trzeba zadać stały próg wielkości obiektów. Ta metoda jest bardzo użyteczna przy obrazach, na których występują litery o dużym rozmiarze i nie nachodzą na siebie, ponieważ bezpośrednio usuwa niechciane artefakty. Wykorzystanie funkcji *bwareaopen* posiada wadę w postaci długiego czasu wykonania, spowodowanego skomplikowanym algorytmem.

Autor pracy wykonał porównanie rezultatów po filtracji (Rys. 5.5) oraz przeprowadził testy czasowe w celu wybrania najlepszej opcji dla problemu rozpoznawania znaków drukowanych.



Rys.5.5 Obrazy po różnych rodzajach filtracji

Z powyższych obrazów można zauważyć, że filtr Wienera daje najlepsze rezultaty pod względem wyglądu obrazu. Usunął więcej małych artefaktów u góry obrazu, przy czym nie zniekształcił znacząco liter. Warto zauważyć, że bardzo dobry wynik daje również użycie funkcji *bwareaopen.* Przy dużych literach właśnie użycie tej metody najlepiej wyodrębnia litery. Wyniki testów czasu wykonywania poszczególnych algorytmów filtracji przedstawiono w Tabeli 5.1.

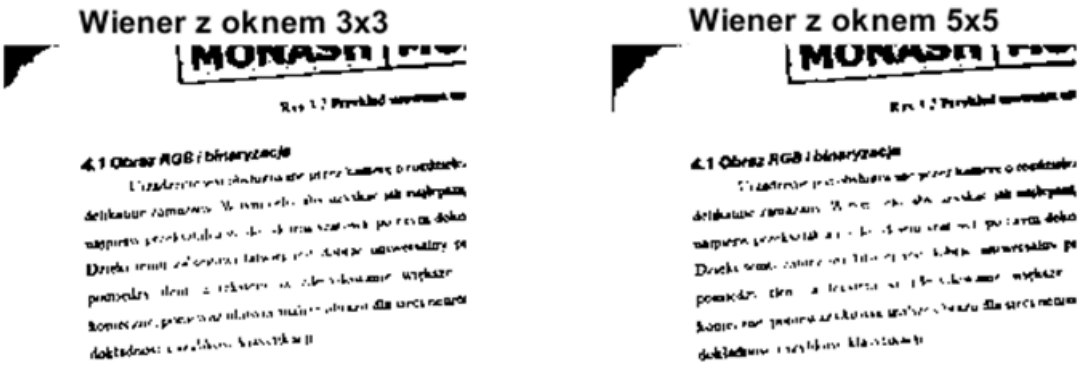
Tabela 5.1 Porównanie testów czasowych

Rodzaj filtru	**Średni czas przetwarzania obrazu [s]**
Filtr średniej	0.2673
Filtr medianowy	0.3246
Filtr Wienera	0.4625
Funkcja *bwareaopen*	3.1932

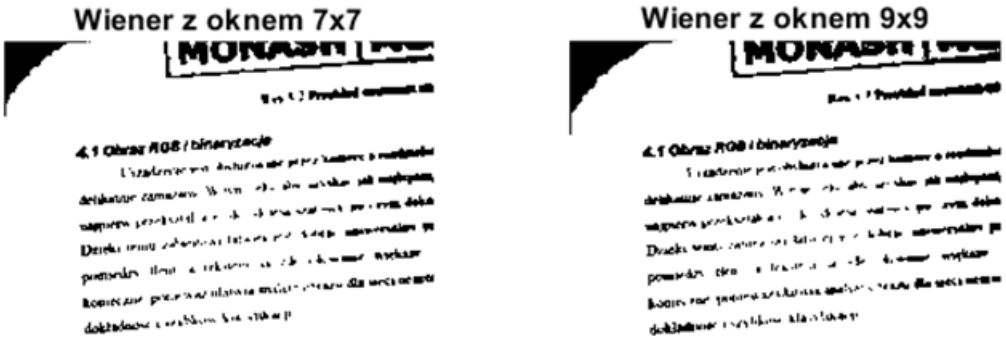
Czasy potrzebne do przefiltrowania obrazu (Tabela 5.1) są porównywalne dla filtrów średniej, medianowego i Wienera. Jak przewidywano funkcja *bwareaopen* zajmuje dużo więcej czasu, aby wykonać to zadanie. Biorąc pod uwagę testy czasowe oraz dokładność filtracji autor zdecydował się użyć filtru Wienera.

Należy również dokonać wyboru rozmiaru okna używanego przez filtr Wienera. Wszystkie poprzednie testy wykonano dla rozmiaru okna 3 × 3 (tak samo, jak dla filtru medianowego). Analizie poddano trzy rodzaje okien:

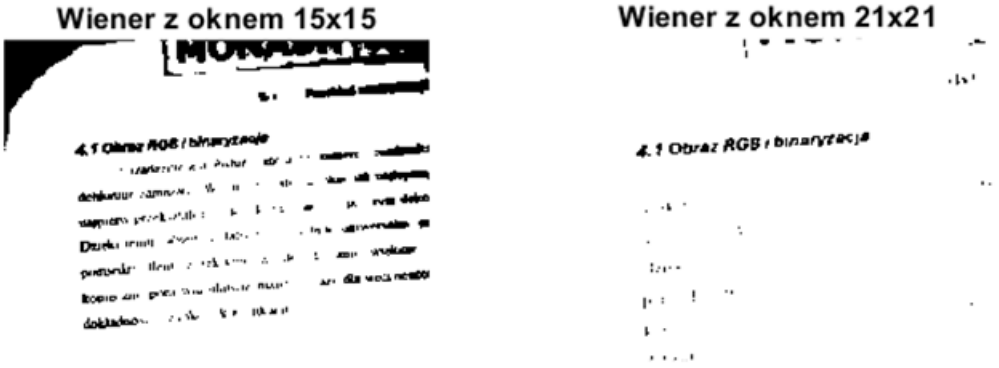
- Małe - 3×3, 5×5;
- Średnie - 7×7, 9×9;
- Duże - 15×15, 21×21.



Rys.5.6 Obrazy przy użyciu małych okien dla filtra Wienera



Rys.5.7 Obrazy przy użyciu średnich okien dla filtra Wienera



Rys.5.8 Obrazy przy użyciu dużych okien dla filtra Wienera

Z obrazów przedstawionych na: Rys.5.6, Rys.5.7 i Rys.5.8 można wywnioskować, że duże okna 15×15 i 21×21, ze względu na swój rozmiar, filtrują zbyt dużą część obrazu. Z kolei filtry o średnich oknach dobrze usuwają małe obiekty, które nie powinny być poddane ocenie, ale zbytnio zniekształcają krawędzie liter. Najlepszym rozwiązaniem będzie użycie okna o wymiarach 5×5, ze względu na lepszą zdolność usuwania niepożądanych szumów niż okno 3×3 przy zachowaniu odporności na zmianę wyglądu liter.

5.3 Segmentacja

Po wstępnej obróbce obrazu należy rozpoznać znajdujący się na nim tekst z uwzględnieniem podziału na pojedyncze litery, tak aby sieć neuronowa mogła

je sklasyfikować. Istnieją różne metody segmentacji, jednak przy rozpoznawaniu tekstu najbardziej popularne to [26]:

- segmentacja oparta na progowaniu,
- segmentacja oparta na połączeniu pikseli,
- segmentacja oparta na liniach konturowych,
- segmentacja oparta na projekcjach, itd.

Segmentacja oparta na progowaniu jest techniką wykorzystywaną ze względu na swoją prostotę i łatwość w implementacji, często przy obrazach z małą entropią. Ta metoda dzieli obraz na regiony na podstawie wartości pikseli w skali szarości. Wartość progu może być wybrana automatycznie, np. za pomocą funkcji *graythresh* w środowisku MATLAB, która wykorzystuje metodę *Otsu*. Ten algorytm opiera się na wyborze optymalnego progu na podstawie maksymalizacji wariancji pomiędzy klasami obrazu. Sama metoda może być zaimplementowana globalnie, gdzie separowane jest tło i cały obraz zostaje podzielony na dwa regiony. Znacznie bardziej efektywną metodą jest zastosowanie lokalnego progowania, gdzie progi są wybierane dla poszczególnych części obrazu z wykorzystaniem metody *Otsu*. Można również dobrać próg ręcznie.[13]

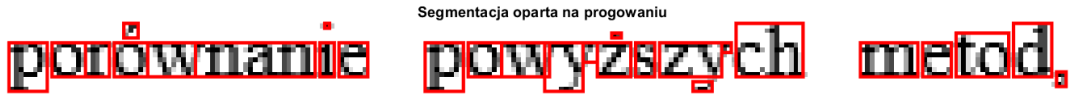
Algorytm polegający na łączeniu pikseli (ang. *Connected Component Labeling)* opiera się na zasadzie mówiącej, że piksele można nazwać połączonymi, gdy spełniają założenia przylegania albo bliskości piksela. Każdy piksel z połączeniem jest przylegający, ponieważ ma relację z sąsiadem. Ponadto istnieje także stałość pomiędzy pikselami przylegającymi, która wynosi 1 jednostkę długości bez żadnego pośrednika. Tą metodę można wykorzystywać przy obrazach monochromatycznych jak i binarnych. [14]

Segmentacja wykorzystująca detekcję krawędzi polega na analizie zestawu pikseli, gdzie otaczające go piksele w skali szarości lub zmiany struktury sugerują zmianę regionu. Krawędź obiektu w obrazie jest odzwierciedlona poprzez różnice w skali szarości, więc pierwszym krokiem w tym algorytmie jest znalezienie punktów, które utworzą kontur regionu. Istotą działania tej metody jest wykrywanie nieciągłości w różnych częściach obrazu za pomocą operatorów różnicowych pierwszego lub drugiego rzędu. [15]

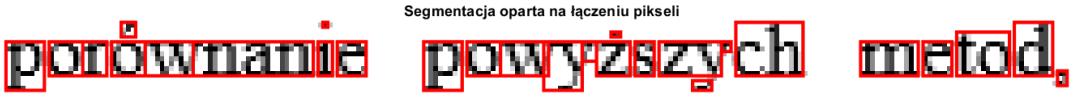
Segmentacja oparta na projekcjach to technika wykorzystująca tzw. „projekcje obrazu”, czyli sumy wartości pikseli. Ta metoda może wystąpić w dwóch wariantach działający na tej

samej zasadzie. Projekcje poziome wykorzystują sumy pikseli z poszczególnych wierszy, za to projekcje pionowe wykorzystują sumy pojawiające się w kolumnach. Ideą tego algorytmu jest sprowadzenie informacji do jednowymiarowych profili, które ułatwiają odnalezienie konturów obiektów. [16]

W ramach pracy wykonano porównanie powyższych metod, w celu wybrania najlepszego rozwiązania dla poruszanego problemu. Samo porównanie obejmuje jakość dokonanej segmentacji oraz szybkość działania algorytmu.



Rys.5.9 Segmentacja oparta na progowaniu



Rys.5.10 Segmentacja oparta na łączeniu pikseli

Rys.5.11 Segmentacja oparta na liniach konturowych

Rys.5.12 Segmentacja oparta o projekcje pionowe

Największym problemem w skuteczności działania ww. algorytmów jest fakt, że litery, w szczególności te małe, mogą nachodzić na siebie, co powoduje, że przy segmentacji często pojawia się kilka liter w wyodrębnionym fragmencie. Można zaobserwować to zachowanie przy segmentacji polegającej na progowaniu (Rys.5.9) i łączeniu pikseli (Rys.5.10). Obydwie metody dają podobne rezultaty, ponieważ wykorzystano w nich do wyznaczenia progu metodę *Otsu*. Algorytm używający projekcji (Rys.5.12) nie jest w stanie dobrze podzielić obrazu, ze względu na jego zbyt słabą jakość i niewielki odstęp pomiędzy literami. Najlepszą segmentację udało się uzyskać korzystając z algorytmu opartego na liniach konturowych (Rys.5.11), który poprawnie rozdzielił wszystkie litery w obrazie testowym.

Tabela 5.2 Porównanie testów czasowych

Rodzaj segmentacji	**Średni czas przetwarzania obrazu [s]**
Progowanie	0.1821
Łączenie pikseli	0.2075
Linie konturowe	0.2625
Projekcje	0.1772

Wyniki czasu wykonania poszczególnych algorytmów segmentacji zebrano w Tabeli 5.2. Na ich podstawie można stwierdzić, że najlepiej wypadł algorytm wykorzystujący projekcje, a najgorzej ten oparty o linie konturowe. Jednakże biorąc pod uwagę dokładność wszystkich metod autor zdecydował się wybrać segmentacje opartą o linie konturowe.

Przy użyciu tej metody segmentacji należy zadbać o to, aby znaki interpunkcyjne nie zostały poddane klasyfikacji. Nie wystarczy jedynie nie brać ich pod uwagę jako małych obiektów, ponieważ w tym wypadku usunięte zostałyby również części znaków „ó”, „ż” oraz „ź”. W tym celu wykorzystano dylatację, aby połączyć znaki diakrytyczne (np. kropka na „i”) z resztą litery w jeden region.

Rozdział 6 Klasyfikacja za pomocą sieci neuronowej

Ostatnim etapem rozpoznawania tekstu jest sklasyfikowanie poszczególnych liter oraz ich połączenie w słowa. Jednakże, aby móc odpowiednio rozpoznać znaki należy przygotować sieć neuronową. Dokładny opis i definicję, czym jest sieć neuronowa, znajduje się w rozdziale 4.1. W tym rozdziale autor skupi się na opisie budowy, uczenia i testów zaproponowanej sieci oraz wyjaśni działania wybranych funkcji i parametrów.

6.1 Baza danych

W celu wytrenowania sieci neuronowej należało wykorzystać gotową bazę danych lub przygotować własną. Autor postanowił wykonać samodzielnie bazę danych ze względu na brak odpowiedniej, która byłaby zgodna z narzuconymi kryteriami. W tym celu przygotowano skrypt w języku Python, który generował litery polskiego alfabetu i cyfry używając czcionki Times New Roman, grupował je oraz przygotował osobny plik, gdzie każdy obraz był skategoryzowany. Zadbano również, aby obrazy były poddane odpowiedniej augmentacji, co oznacza, że litery były losowo obracane o kąt ± 15º oraz pogrubiane (Rys.6.1). Ten zabieg sprawia, że zbiór danych jest bardziej zróżnicowany, przez co sieć neuronowa lepiej rozpoznaje cechy danego obrazu.

Rys.6.1 Przykłady liter z przygotowanej bazy danych

W efekcie podjętych działań uzyskano zbiór 32 znaków (dużych i małych), przy czym każdy znak alfabetu był reprezentowany w liczbie 5000. Cała baza danych została podzielona na 3 zbiory:

- uczący (80%),
- walidacyjny (10%),
- testowy (10%).

Zbiór uczący był wykorzystany do wytrenowania modelu, walidacyjny do oceny jakości modelu w trakcie procesu trenowania. Zbiór testowy został użyty po treningu, aby ocenić ogólną wydajność modelu.

6.2 Budowa sieci neuronowej

Do wykonania sieci neuronowej zostało wykorzystane narzędzie *Deep Network Designer*, które jest dostępne w *Deep Learning Toolbox* programu MATLAB. *Deep Network Designer* to aplikacja, w której można tworzyć blokowe schematy sieci neuronowych. W tym celu można wykorzystać gotowe funkcje służące do np. optymalizacji lub normalizacji sieci. Autor wybrał to narzędzie ze względu na prostotę w obsłudze oraz łatwość implementacji wybranej struktury sieci neuronowej. Na Rys.6.2 przedstawiono schemat blokowy zaprojektowanej, konwolucyjnej sieci neuronowej (CNN). Na wejście sieci podawany jest obraz o wymiarach 28×28 jednokanałowy, czyli binarny. Sama sieć składa się z dwóch warstw konwolucyjnych, zgodnie z założeniami projektu dyplomowego. Każda warstwa składa się z:

- warstwy normalizacji *batcha*,
- warstwy aktywacji *ReLu*,
- warstwy *dropout*,
- warstwy *poolingu* 2D.

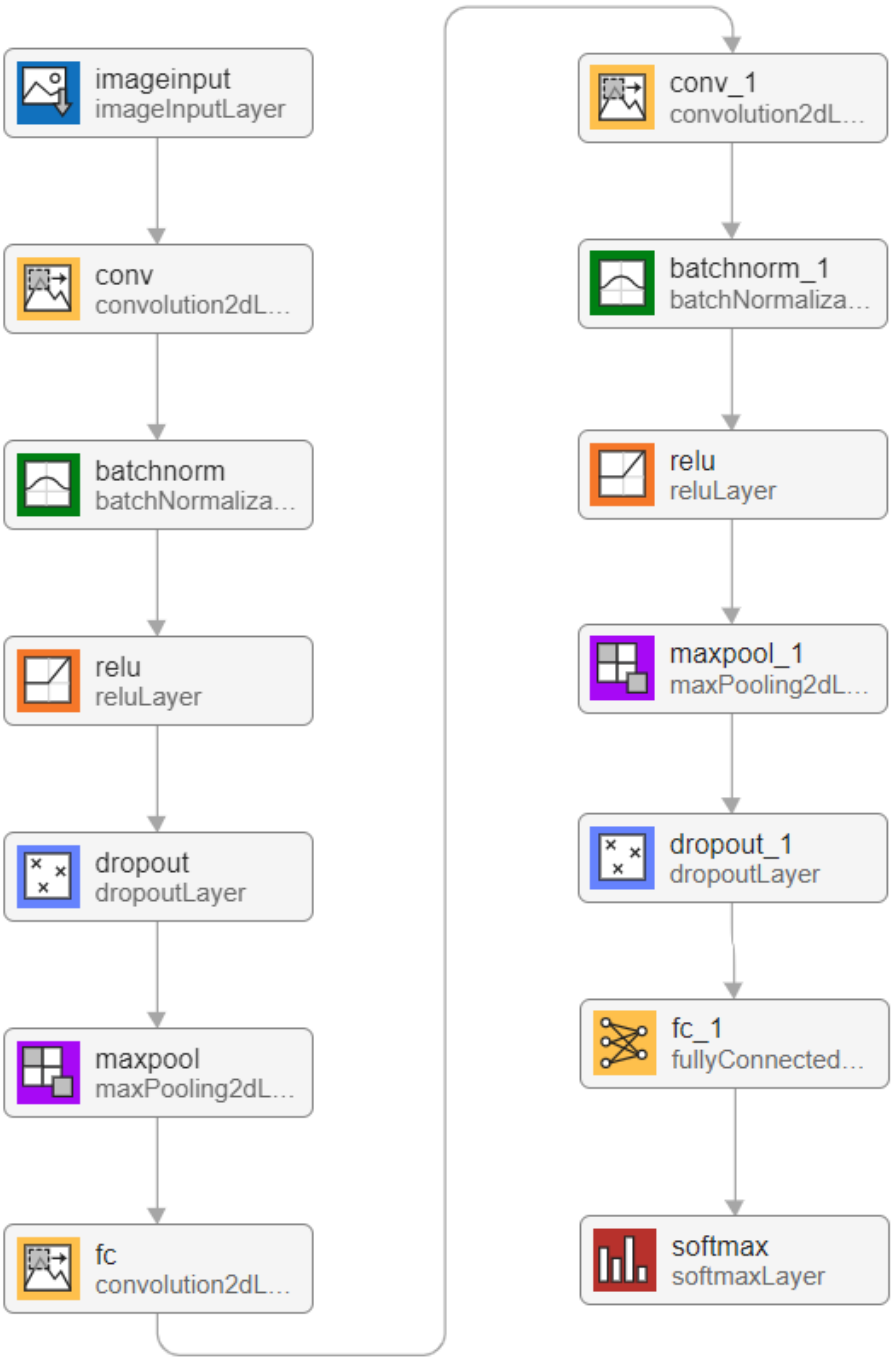
Warstwa normalizacji *batcha* jest używana w głębokich sieciach neuronowych ze względu na przyspieszenie procesu uczenia, stabilizacji gradientów oraz zmniejszenie szansy na przeuczenie (ang. *overfitting*). Normalizacja sprawia, że dla każdej partii (ang. *batch*) każda cecha ma średnią równą 0 i odchylenie standardowe równe 1, zgodnie z poniższym wzorem:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

gdzie:

- $\hat{x}$ to znormalizowana wartość cechy,
- x to pojedyncza wartość wejściowa,

- μ to średnia arytmetyczna wartości wejściowych,
- σ^2 to wariancja wartości wejściowych
- ϵ to bardzo mała liczba dodawana do mianownika, aby uniknąć dzielenia przez 0.



Rys. 6.2 Schemat blokowy sieci neuronowej

Warstwa *ReLu* (ang. *Rectified Linear Unit*) to funkcja aktywacji mająca na celu wprowadzenie nieliniowości, aby model mógł nauczyć się bardziej złożonych zależności. Sama funkcja jest definiowa jako:

$$f(x) = \max(0, x)$$

Funkcja *ReLu* jest bardzo popularna ze względu na łatwość w implementacji, szybkość działania oraz jest odporna na problem zanikających gradientów.

Warstwa *dropout* sprawia, że neurony zostają losowo wyłączane, czyli ich wartość zostaje ustawiana na 0, w celu ograniczenia przeuczenia. Dzięki użyciu tej warstwy sieć nie polega na zbyt małej liczbie neuronów, przez co model nie jest zależny od pojedynczych cech, co powoduje lepszą generalizację. Wartość parametru *dropout* użytego w powyższej sieci wynosi 0,5, czyli 50% neuronów zostaje wyłączonych.

Warstwa *MaxPooling2D* pozwala na zmniejszenie wymiarowości danych wejściowych. Operacja działa na zasadzie maksymalnej selekcji w małych regionach obrazu, czyli wybierana jest największa wartość w danym oknie. Dzięki tej funkcji model jest w stanie uczyć się szybciej, przez mniejszą ilość danych oraz jest bardziej odporny na przeuczenie. Wielkość okna dla funkcji *maxPooling2D* wynosi 5×5, a krok o jaki przesuwane jest okno to (1,1).

Kluczową częścią wybranej sieci neuronowej są warstwy konwolucyjne, które odpowiadają za wykrywanie cech. Warstwa konwolucyjna przesuwa filtry, które są małymi macierzami po całym obrazie. Przy każdym przesunięciu filtr dokonuje operacji splotu, której wynik jest dodawany do macierzy cech. [17] W pierwszej warstwie przygotowanej sieci znajdują się 32 filtry o rozmiarze 3×3, które są przesuwane o krok (1,1), natomiast w drugiej są 64 filtry o wymiarach 5×5, które przemieszczają się z tym samym krokiem. Taki zabieg powoduje, że model łatwiej jest w stanie rozpoznać bardziej złożone cechy oraz pozwala mu na bardziej szczegółową analizę danych wejściowych.

Na sam koniec działania sieci dane są analizowane przez warstwę *Softmax*. Jej zadaniem jest przekształcanie surowych wyników z wyjścia sieci w rozkład prawdopodobieństwa, który będzie sumował wartości wszystkich klas do 1. Wzór tej warstwy wygląda następująco:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

gdzie:

- z_i to logit, czyli surowy wynik, dla klasy *i*,
- K to liczba klas,
- e^{z_j} to eksponenta logitu,
- p_i prawdopodobieństwo dla klasy i.

Warstwę *Softmax* wykorzystuje się jako warstwę wyjściową ze względu na możliwość interpretacji rozkładu prawdopodobieństwa do klasyfikacji obrazu. Na podstawie tej warstwy można jednoznacznie wskazać, która klasa ma największą szansę na poprawną klasyfikację.

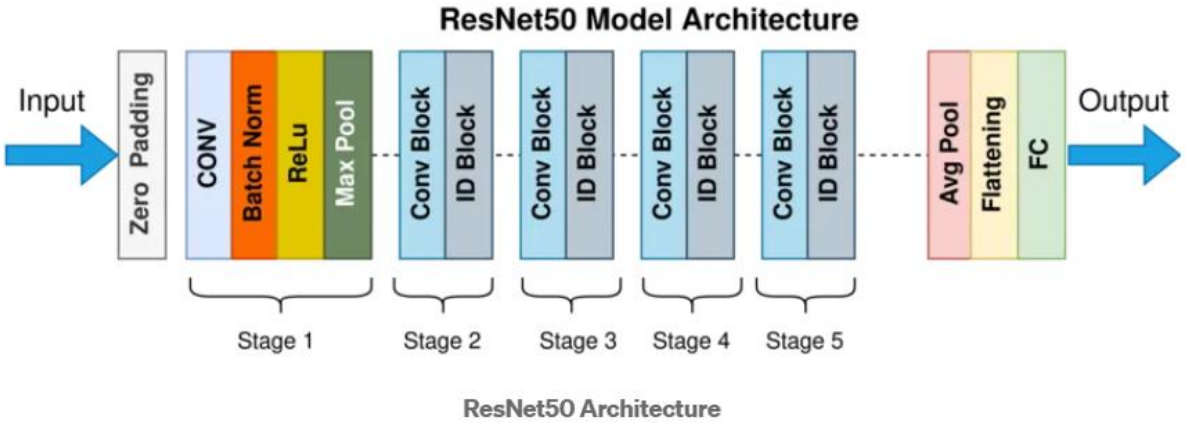
6.3 Budowa ResNet50

Istotą *Transfer Learningu* jest douczenie już wytrenowanej sieci, co dokładniej autor opisuje w rozdziale 3.2. Kluczowym aspektem staje się wybór odpowiedniej sieci neuronowej. Autor postanowił wybrać sieć *ResNet50* ze względu na jej popularność i zastosowanie w dziedzinie detekcji obiektów. W środowisku MATLAB istnieje gotowa funkcja rozpoznająca tekst (*ocr ()*), która bazuje na silniku Tesseract OCR (*open-source*), obsługującym ponad 100 języków, w tym polski. Sam silnik jest wspierany przez Google.

ResNet50 to głęboka sieć neuronowa z rodziny Residual Networks, składająca się z 50 warstw konwolucyjnych połączonych za pomocą bloków resztkowych (Rys.6.3). Te bloki wykorzystują tzw. *shortcut connections,* co unika problemu zanikającego gradientu, spowodowanego przez dużą liczbę warstw. Architektura sieci *ResNet50* jest podzielona na 4 sekcje:

- warstwy konwolucyjne,
- blok tożsamościowy (ang. *identity block*),
- blok konwolucyjny,
- w pełni połączone warstwy (ang. *fully connected layers*).

Warstwy konwolucyjne odpowiadają za ekstrakcje cech z obrazu wejściowego, a bloki tożsamościowe i konwolucji przetwarzają i przekształcają te cechy. Do klasyfikacji wykorzystane są w pełni połączone warstwy. [18 - 19]



Rys.6.3 Architektura sieci *ResNet 50* [19]

Najbardziej kluczowym aspektem sieci z rodziny *ResNet* jest wykorzystanie tzw. *shortcut connections.* Te połączenia pozwalają danym przemieszczać się po sieci z pominięciem kilku warstw. Ten zabieg umożliwia sieci nauczenie się funkcji resztkowych, które mapują dane wejściowe, zamiast uczenie się mapowania od zera. *Shortcut connections* są używane w blokach konwolucyjnych i tożsamościowych. [19]

6.4 Modyfikacja sieci ResNet50

W celu wykorzystania metody Transfer Learningu należało zmodyfikować sieć *ResNet50*. Na początku zmodyfikowano dane wejściowe, ponieważ ta sieć wymaga, aby były one rozmiaru 224×224 oraz wyrażone w kolorach RGB. Po tym zabiegu usunięto ostatnią warstwę i zastąpiono ją nową, służącą do klasyfikacji, tak aby zgadzała się w niej liczba klas. Model jest zdecydowanie bardziej skomplikowany niż przygotowana sieć neuronowa z poprzedniego rozdziału, więc autor zdecydował, aby trenować sieć ustalając mniejszą liczbę epok.

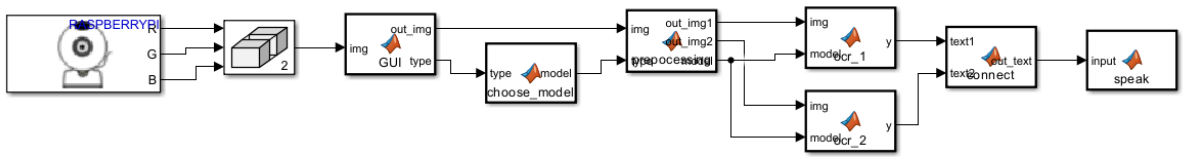
Rozdział 7 Implementacja programu w Raspberry Pi

Zaimplementowano obydwa modele rozpoznawania tekstu, obsługę kamery oraz algorytmu przetwarzania obrazu, czyli binaryzację, filtrację oraz segmentację. Cały program został podzielony na odpowiednie funkcje obsługujące poszczególne zadania.

Ostatnim etapem było zaprogramowanie funkcjonalności, która pozwoli urządzeniu komunikować się poprzez Bluetooth ze słuchawkami bądź głośnikiem, zaimplementowanie całego programu na platformę Raspberry Pi oraz złożenie obudowy.

W celu zapewnienia konwersji tekstu na dźwięk użyto najpierw syntezatora mowy *espeak*. Narzędzie *espeak* jest bardzo wygodną opcją ze względu na fakt, że jest darmowe oraz działa na wielu platformach, w tym na Raspberry Pi. Obsługuje ponad 50 języków i dialektów oraz ma możliwość dostosowania tempa i tonu głosu. Przygotowano również skrypt w programie MATLAB, który pozwala na automatyczne parowanie urządzeń z Bluetooth, aby można było usłyszeć rozpoznawany tekst.

Cały program został podzielony na funkcje, które ułatwiły wykonanie diagramu blokowego w programie Simulink (Rys.7.1). Jak można zauważyć pierwszym blokiem w łańcuchu jest blok odczytu obrazu z kamery w formacie RGB. Następnie obraz RGB jest odczytywany przez blok *GUI*, gdzie jest przedstawiany użytkownikowi. Użytkownik ma możliwość zrobienia zdjęcia i wyboru modelu, który przeprowadzi rozpoznawanie tekstu. Gdy to się wydarzy przeprowadzany jest proces przetwarzania obrazu w bloku *preprocessing*, gdzie dokonywana jest filtracja filtrem Wienera oraz segmentacja za pomocą metody używającej linie konturowe. Zbiór obrazów jest klasyfikowany przez funkcję *ocr_1* i *ocr_2*. Ostatecznie wykonywana jest konwersja tekstu na dźwięk w bloku *speak* i następuje wysłanie komunikatu do urządzenia audio.



Rys.7.1 Schemat blokowy wykonany w Simulinku

Na podstawie ww. diagramu wygenerowano kod języka C++, który został przesłany, skompilowany i uruchomiony na minikomputerze. Użyto do tego przybornika Simulink Support Package for Raspberry Pi.

Na platformie Raspberry Pi, w celu poprawy wydajności urządzenia, wykorzystano możliwość użycia większej liczby rdzeni. Dzięki tej metodzie można dokonać równoległej klasyfikacji obrazów. Wykorzystany minikomputer Raspberry Pi 4B posiada 4 rdzenie, a do klasyfikacji użyto dwóch. W tym celu zbiór obrazów otrzymanych po segmentacji podzielono na dwie części. Użyto opcji *Concurrent Execution* i zmapowano funkcję *ocr_1* i *ocr_2* do dwóch różnych tasków. Funkcja *connect* łączy w jeden spójny tekst wyniki klasyfikacji i zapewnia, że są połączone w odpowiedniej kolejności.

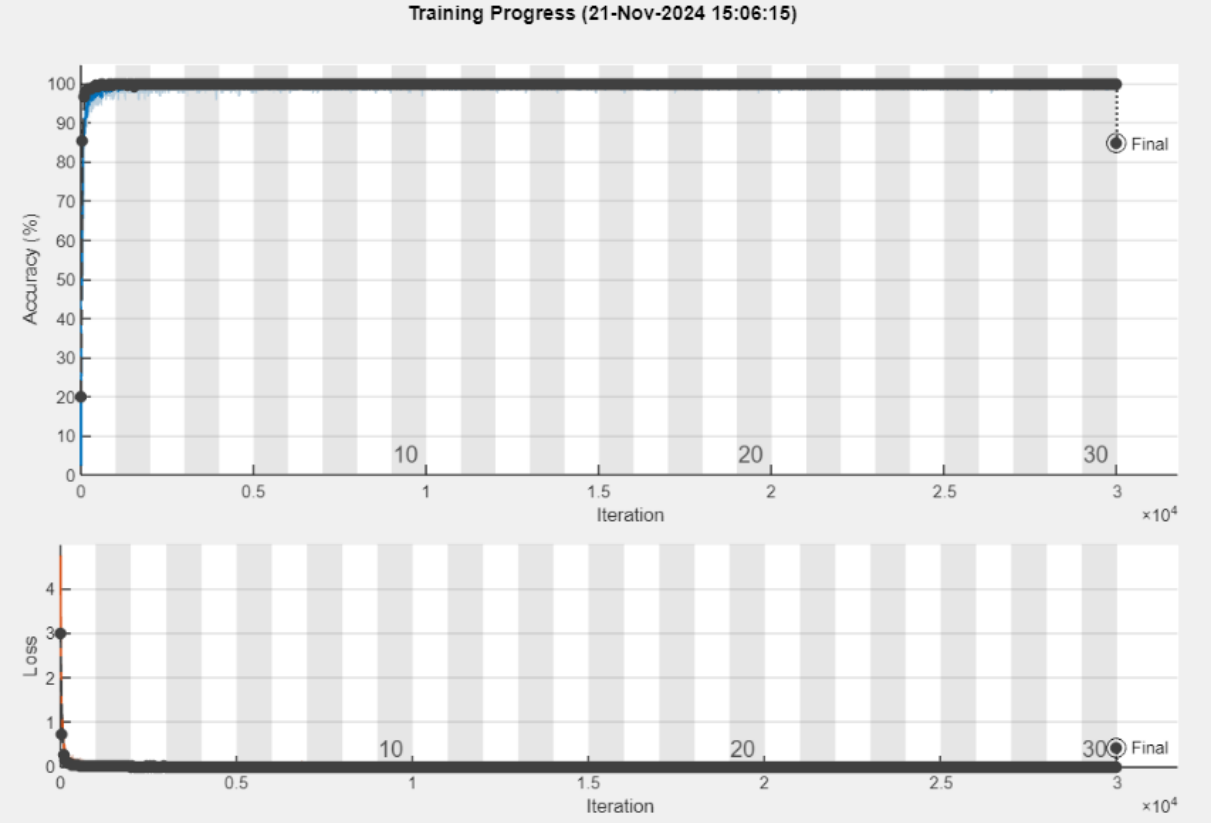
Ostatnim etapem projektu było zamontowanie w obudowie minikomputera Raspberry Pi wraz z kamerą. Końcowy wygląd urządzenia można zobaczyć na rysunku 7.2.

Rys.7.2 Końcowy wygląd urządzenia

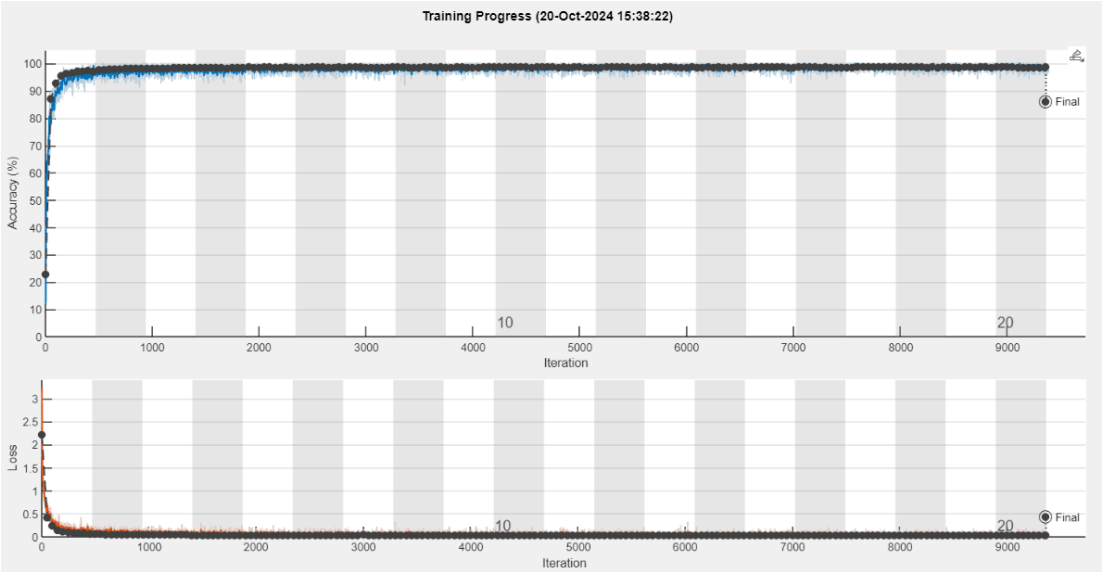
Rozdział 8 Porównanie wyników

W celu sprawdzenia poprawnego działania dwóch rozwiązań, czyli autorskiej sieci neuronowej oraz modelu opartego na *ResNet50*, przeprowadzono testy dokładności na platformie Raspberry Pi z wykorzystaniem przygotowanej aplikacji. Spodziewanym rezultatem było uzyskanie lepszych wyników za pomocą douczonej sieci *ResNet50* ze względu na większą liczbę warstw konwolucyjnych.

Po wytrenowaniu dwóch sieci wykonano wykresy obrazujące ich dokładność na zbiorze walidacyjnym w trakcie procesu uczeniu.

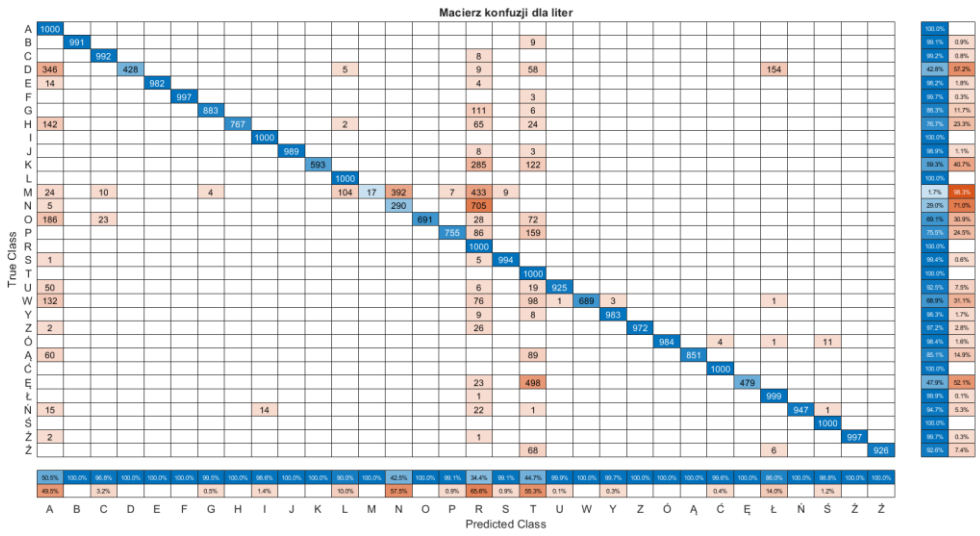


Rys. 8.1 Dokładność przygotowanej sieci



Rys.8.2 Dokładność sieci ResNet50

Można zauważyć, że sieć *ResNet50* (Rys.8.2) dużo szybciej zbiega do 100 % dokładności niż przygotowana przez autora sieć (Rys.8.1) w trakcie uczenia. Przeprowadzono również sprawdzenie dokładności na zbiorze testowym, który jest taki sam dla obydwóch sieci. Dla przygotowanej sieci ta dokładność wynosiła 79,14 %, a dla sieci *ResNet50* 87,24%. Trzeba zaznaczyć, że sieć *ResNet50* była trenowana na mniejszej liczbie epok, ze względu na bardzo długi czas treningu.

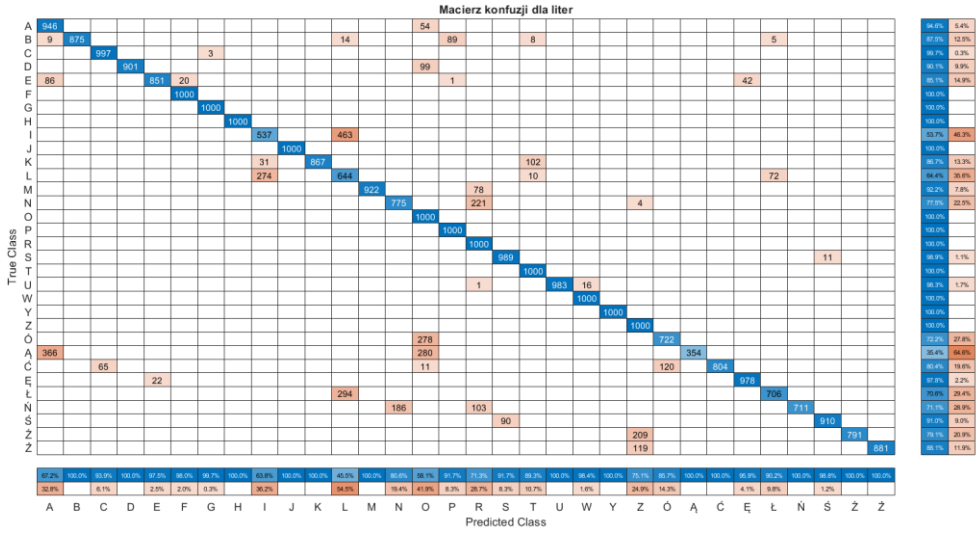


Rys. 8.3 Macierz pomyłek dla przygotowanej sieci

Przygotowano również macierz pomyłek (ang. *confusion matrix*) przedstawioną na Rys.8.3, dzięki której można zaobserwować, które litery były najczęściej mylone.

Najrzadziej poprawnie sklasyfikowane litery to „m” i „n” i były one mylone z literą „r”. Dzieje się tak ze względu na dużo podobieństwo tych znaków do siebie. Również często myloną literą jest „ę”, gdzie najczęściej była klasyfikowana jako litera „e”. Pozostałe polskie znaki były w większości klasyfikowane poprawnie.

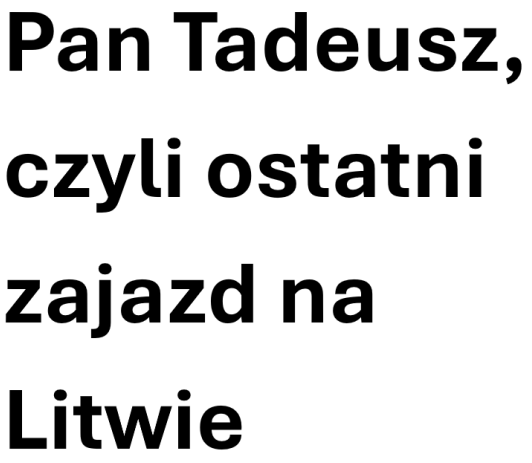
Taką samą macierz wyznaczono również dla sieci *ResNet50* i pokazano na Rys.8.4.



Rys.8.4 Macierz pomyłek dla sieci ResNet 50

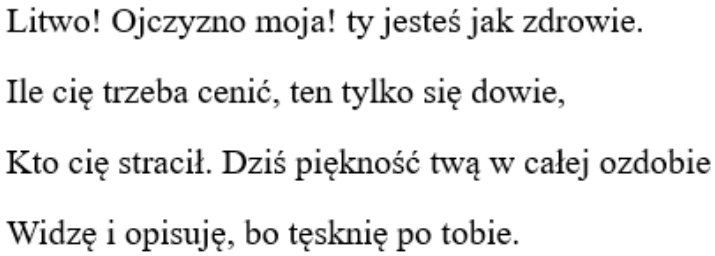
Jak się spodziewano wyniki otrzymane z użyciem sieci *ResNet50* są dokładniejsze i klasyfikacja jest istotnie lepsza. Jednak co zaskakujące model *ResNet* bardziej myli się przy polskich znakach niż autorska sieć neuronowa. Nie wykazuje jednak cech nadmiernej klasyfikacji poszczególnych liter. Najrzadziej rozpoznawaną literą jest „ą” i głównie jest mylone z „a” oraz „o”.

Następnie sprawdzono działanie całego urządzenia bazując na dwóch modelach wykorzystując platformę Raspberry Pi. W tym celu przygotowano 2 rodzaje tekstów: jeden, gdzie tekst jest dobrze widoczny z wyraźnymi przerwami (Rys. 8.5) oraz fragment, gdzie litery są mniejsze bez dużych odstępów (Rys.8.6).



Rys.8.5 Fragment testowy z dużymi odstępami [21]

Wyraźny tekst nie sprawił problemu modelowi *ResNet50*. Sieć rozpoznała cały tekst i prawidłowo go przeczytała. Autorskiej sieci neuronowej problem sprawiły jedynie litery „n". W słowie „ostatni" oraz „na" zostały rozpoznane jako litera „r", przez co przekaz fragmentu jest trochę zaburzony. W przypadku segmentacji obraz został podzielony prawidłowo, a znak interpunkcyjny w postaci przecinka nie został poddany klasyfikacji.



Rys.8.6 Fragment testowy z małymi odstępami [21]

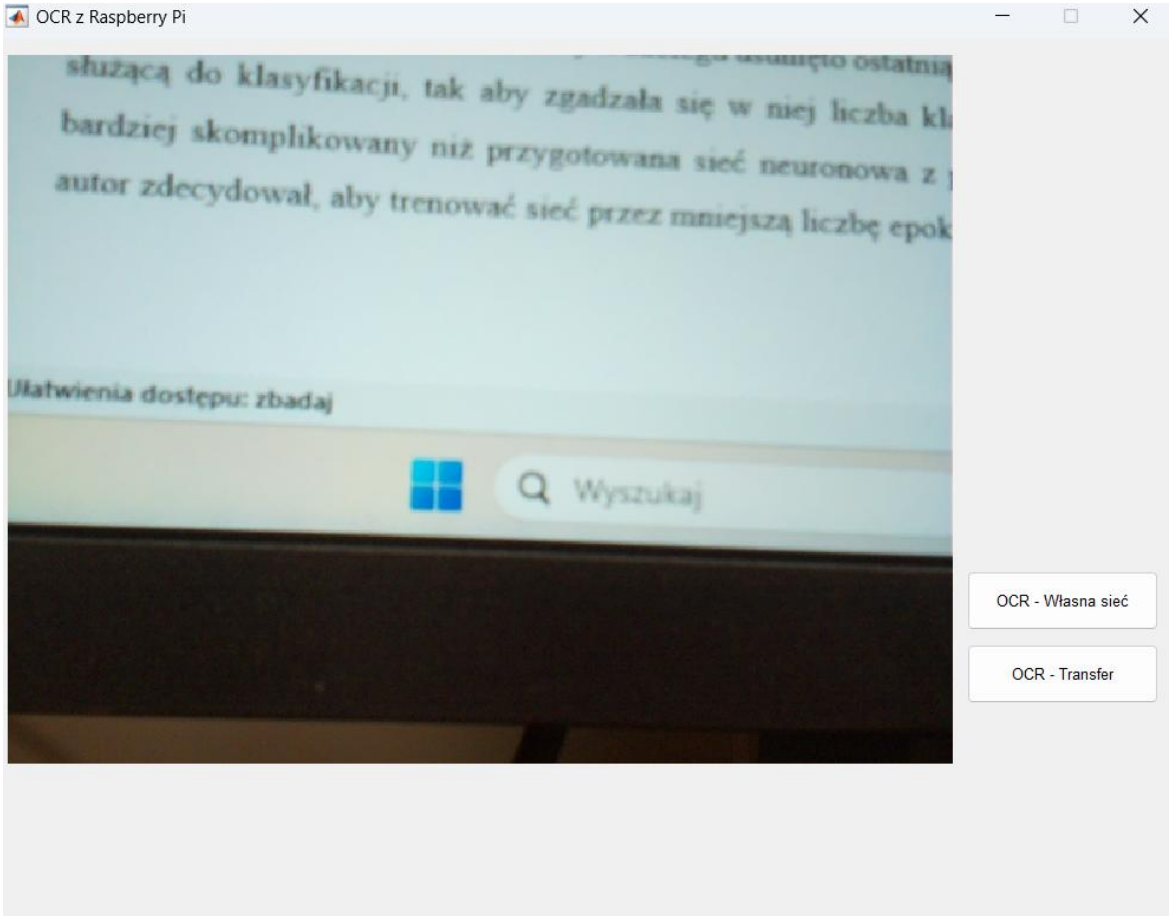
Początek inwokacji „Pana Tadeusza" autorstwa Adam Mickiewicza sprawił więcej problemów obydwu sieciom. Segmentacja obrazu została przeprowadzona prawie idealnie, jednak za jedną literę zostały uznane „st" w wyrazie „stracił", „rz" w wyrazie „trzeba", „zd" w wyrazie „ozdobie" oraz „kn" w wyrazie „tęsknię". Z tego powodu te połączenia liter nie mogły być sklasyfikowane poprawnie przez obydwa modele. Autorska sieć neuronowa znowu nie była w stanie sklasyfikować poprawnie wszystkich liter „n" oraz „m", które myliła z literą „r". Dodatkowo raz litera „p" została pomylona z „t", obydwa wykrzykniki zostały sklasyfikowane jako litera „i", dwie litery „ę" oceniono jako „e" oraz kilka innych pomyłek.

Autorski model miał więc dokładność 83,55% na podanym fragmencie. Model *ResNet50,* oprócz złej klasyfikacji liter, związanej z nieudaną segmentacją, ocenił zdecydowaną liczbę liter poprawnie. Nie miał takich problemów z klasyfikacją liter „n" i „m" jak autorska sieć, jednak napotkał problem przy ocenie niektórych polskich znaków. Mimo tego model miał dokładność 92,76%. Wysokie dokładności obydwóch modeli są spowodowane wykorzystaniem zrzutów ekranu przy sprawdzeniu jakości klasyfikacji sieci, a nie zdjęć z kamery jak to będzie miało miejsce w ostateczniej wersji projektu.

Zgodnie z oczekiwaniami rezultaty otrzymane dzięki wykorzystaniu modelu używającego sieć *ResNet50* są dokładniejsze szczególnie dla tekstu o mniejszym rozmiarze znaków. Autorska sieć neuronowa natomiast nie rozpoznała tylko dwóch liter dla tekstu, gdzie ostępy między literami były większe. Użycie segmentacji opartej o linie konturowe okazało się bardzo trafne, ponieważ obraz, gdzie odstępy były małe został podzielony bardzo dokładnie. Główną różnicą pomiędzy obydwoma modelami jest rozmiar obrazu wejściowego, gdzie w modelu *ResNet50* jest dużo większy niż w autorskiej sieci. To sprawia, że model *ResNet50* jest w stanie dużo dokładniej rozpoznawać cechy danego obrazu.

Rozdział 9 Testy funkcjonalne

W celu usprawnienia procesu testowania urządzenia autor stworzył prostą aplikację, której główne okno pokazano na Rys.9.1. Interfejs graficzny składa się z aktualnego obrazu przekazywanego z kamery oraz ma możliwość wyboru rodzaju modelu przeznaczonego do rozpoznawania tekstu.

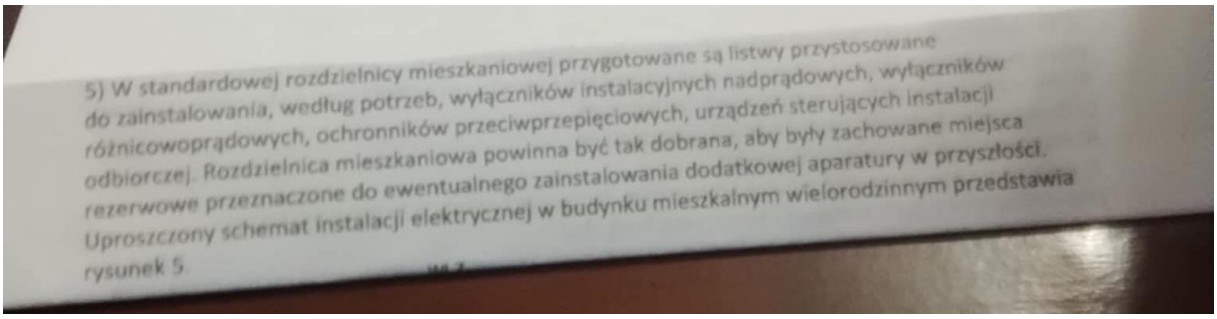


Rys.9.1 Interfejs aplikacji do testowania urządzenia

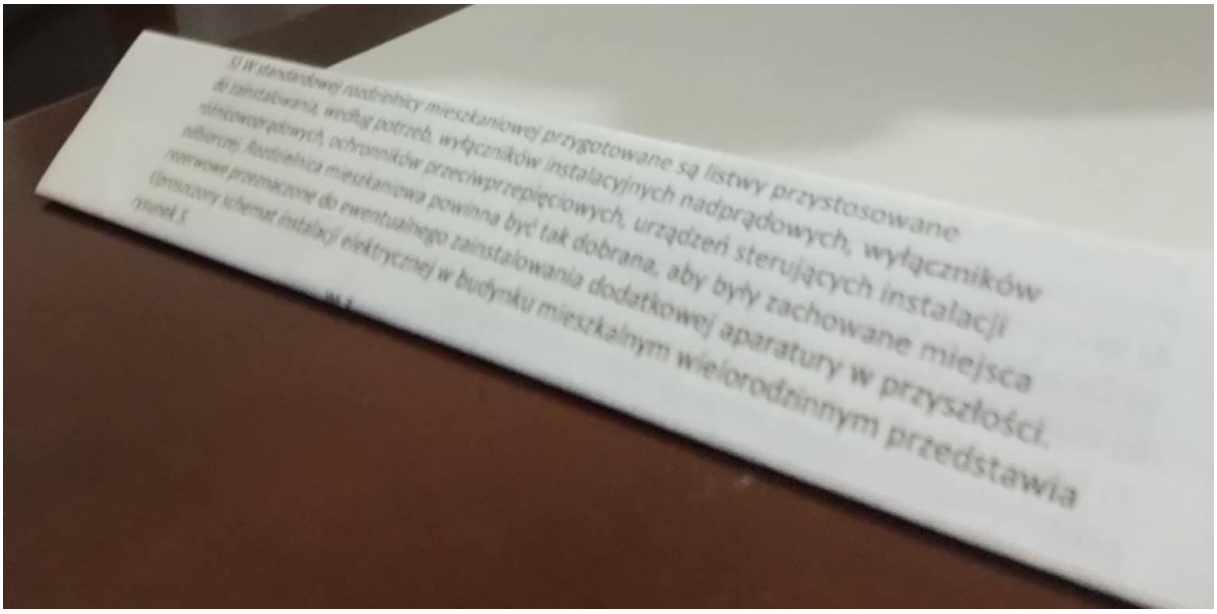
Aplikacja otrzymuje dane z Raspberry Pi poprzez lokalną sieć Wi-Fi i służy tylko do wizualizacji, co oznacza, że cała logika projektu jest wykonywana na minikomputerze. Autor wykorzystuje również syntezator systemowy Windows w celu sprawdzenia czy tekst faktycznie jest dobrze rozpoznawany. GUI zostało całkowicie wykonane w programie MATLAB.

Przeprowadzono testy funkcjonalne działania urządzenia. W tym celu autor wykonał zdjęcia fragmentu tekstu pod różnym kątem i oświetleniem (Rys.9.2 – 9.4), aby sprawdzić dokładność działania urządzenia. Wykonano również testy czasowe, aby oszacować średni czas

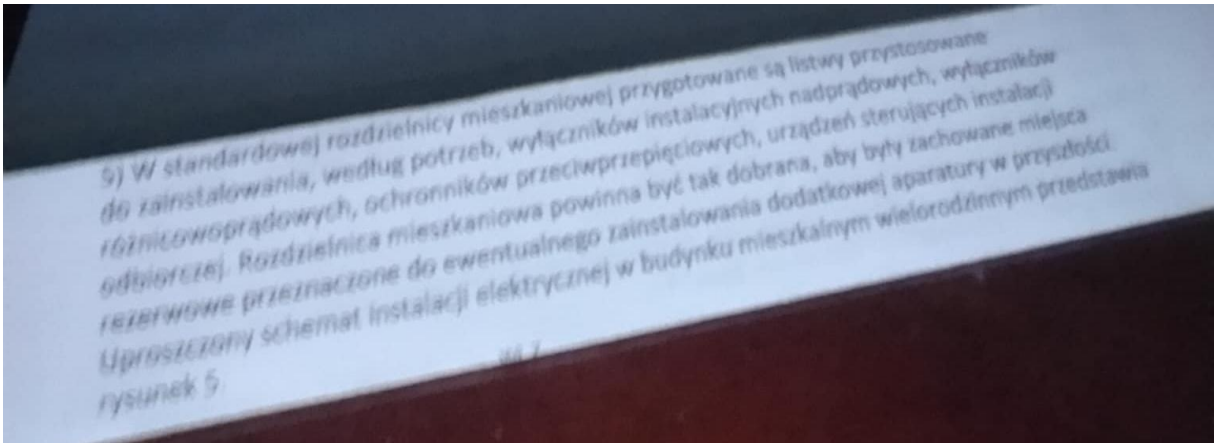
potrzebny na analizę jednego fragmentu. Wszystkie te testy zostały przeprowadzone dla obydwóch modeli wykonanych w projekcie.



Rys.9.2 Zdjęcie wykonane pod kątem prostym z dobrym oświetleniem



Rys.9.3 Zdjęcie wykonane pod kątem z dobrym oświetleniem



Rys.9.4 Zdjęcie wykonane pod kątem prostym z słabym oświetleniem

Wykonano porównanie działania modelu autorskiego oraz wykorzystującego sieć *ResNet50*. Rezultaty, w formie rozpoznanego tekstu, można zobaczyć w Tabeli 9.1.

Tabela 9.1 Porównanie działania modeli przy użyciu Rys.9.2

Rodzaj modelu	
Autorski	*ResNet50*
Sl W slardardowej razdzielricy rieszkariowej przygotaw tistwy przystosoware do zainstalowaria wedtug patrzeb wytączników instalacyjrych radprądowych wyłącznikow aprądowych, ochromików pwaciwprzeaitciowych urządzen aerujących irstalacji odbiorczej Rozazietnica rieszkaniowa powirna być tak aobrana aby były zactowane riejsca rezarwowe przeznaczore da ewentuatrego zairstalowaria aodatkowej aparatury w przysztości Uproszczory scherat irstalacji etektryczrej w budyrku rieszkalnyr wielorodzirnyr przeastawia rysurek S	SO W standordowej rozdźelnicy mieszkariowej przygotow sa listwy przystosoware do zalnstalowania wedtug potrzeb wyłoczników instalacyjnych nadprodowych wyłoczników opradowych, ochronników pmeciworzepieciowyoh, urzodzeń sterujacych instalacji odbiorczej Pozdzielnica rieszkaniowa powinna być tak dobrana aby byty zachoware riejsca reżerwowe przeznoczone do ewentualrego zainstalowania dodatkowej ąparatury w przyszłosć Upromczony schemat instalacjl eleitrycznej w budynku mieszkalryr wielorodzimym przedstawla rysunek S

Autorski model osiągnął dokładność 74,84%, gdzie jedynymi relatywnie długimi słowami, które rozpoznał poprawnie są „aparatury” oraz „odbiorczej”. W pozostałych przypadkach błędy klasyfikacji to zazwyczaj od 1 do 3 liter na słowo, gdzie najczęściej mylonymi literami jest „n” z „r”. Model wykorzystujący sieć *ResNet50* osiągnął dokładność 86,89%. Ten model rozpoznał poprawnie dużo więcej wyrazów, niż pierwszy, ale za to miał więcej problemów z klasyfikacją polskich znaków. Dla obydwóch rozpoznanych tekstów przeprowadzono ich syntezę na dźwięk. Nie wychwycono żadnych niezbieżności między rozpoznanymi tekstami, a ich reprezentacjami dźwiękowymi.

Przeprowadzono również analizę pozostałych dwóch zdjęć, lecz dla nich rezultaty były niezadowalające. Dokładność klasyfikacji dla Rys.9.3 wynosiła dla autorskiego modelu 21,14%, a dla modelu z siecią *ResNet50* 24,1%. Rozpoznawane były jedynie znaki znajdujące się blisko obiektywu kamery. Dla Rys.9.4 obydwa modele rozpoznały jedynie pojedyncze znaki. Problemem była także segmentacja obrazu, ponieważ większość znaków nie została wyodrębniona.

Na podstawie serii zdjęć wykonanych pod kątem prostym i z dobrym oświetleniem wykonano również testy czasowe. Średni czas potrzebny do przetworzenia obrazu i klasyfikacji wszystkich znaków na nim zajął:

- 12,1 s autorskiemu modelowi,
- 20,5 s modelowi używającego sieć *ResNet50*.

Mimo, że średnio model używający sieci *ResNet50* zajmował więcej czasu, jego dokładność jest o ponad 10 % lepsza od drugiego modelu. Z tego powodu zalecane jest używanie właśnie modelu *ResNet50*.

Rozdział 10 Podsumowanie

Podsumowując uważam, że projekt okazał się zasadniczym sukcesem. Ukończono wszystkie cele zawarte w opisie pracy. W projekcie zawarto opis rzeczywistych rozwiązań wspomagających osoby niewidome lub niedowidzące, przytoczono definicję zagadnień związanych z *Machine Learningiem.* W części poświęconej projektowi dokonano analizy różnych rodzajów filtrów i metod segmentacji obrazu i wybrano, zdaniem autora, najlepsze z nich. Wykonano również dwa modele sieci neuronowych do klasyfikacji znaków tekstowych, na podstawie obrazu z kamery i porównano wyniki ich klasyfikacji. Autor zrealizował również aplikację ułatwiającą testowanie urządzenia i wykonał obudowę używając druku 3D.

Efektem projektu jest działające urządzenie, które może zostać wykorzystane jako pomoc dla osoby niewidomej. Jednakże urządzenie w dalszym ciągu może być rozwijane, aby dało się je zastosować komercyjnie, np. poprzez dodanie szyfrowania kodu, użycie bardziej trwałego materiału do produkcji obudowy albo wykorzystanie kamery o wyższej rozdzielczości. Można również dodać moduł oświetleniowy, który pomógłby robić zdjęcia lepszej jakości. Sam algorytm sieci neuronowej też może zostać ulepszony poprzez dodanie kolejnych warstw konwolucyjnych i zwiększenie złożoności przeprowadzanych operacji. Model oparty o *Transfer Learning* działa dobrze, aczkolwiek autor uważa, że on też mógłby działać lepiej.

W trakcie pracy nad projektem autor rozwinął swoje umiejętności i wiedzę z zakresu sztucznej inteligencji, programowania, systemów wbudowanych i wizyjnych oraz projektowania konstrukcji w 3D. Z tego powodu wybór tematu pracy okazał się być bardzo dobry uwzględniając kierunek studiów autora.

Bibliografia

[1] Zyrlo LLC, The LyriQ Assistive Reader, https://www.zyrlo.com/ , Ostatnia wizyta: 23.12.2024.

[2] Brian R. Miller, History of the blind, https://www.britannica.com/science/history-of-the-blind, Publikacja: 17.05.2023.

[3] Envision, Envision Glasses, https://www.letsenvision.com/glasses/home , Ostatnia wizyta 23.12.2024.

[4] Google Text-To-Speech (TTS), https://cloud.google.com/text-to-speech, Ostatnia wizyta: 27.12.2024.

[5] S. Rost, "Character Recognition: Image and Video Available: Project One by Stanislav Rost", 1998, [online] Available: http://web.mit.edu/stanrost/www/cs585pl/pl.html.

[6] V. Kwatra, "Optical Character Recognition", [online] Available: http://www.cc.gatech.edu/-kwatra/computer_vision/ocr/OCR.html.

[7] V. Ganapathy and C. C. H. Lean, "Optical Character Recognition Program for Images of Printed Text using a Neural Network," *2006 IEEE International Conference on Industrial Technology*, Mumbai, India, 2006

[8] Ryszard Tadeusiewicz, Maciej Szaleniec; „Leksykon sieci neuronowych”, https://www.researchgate.net/publication/294578763_LEKSYKON_SIECI_NEURONOWYCH_Lexicon_on_Neural_Networks, Wydawnictwo Fundacji „Projekt Nauka”, Wydanie I Wrocław, Publikacja: 01.2015.

[9] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, Oct. 2010,

[10] J. Hao, "Deep learning-based medical image analysis with explainable transfer learning," *2023 International Conference on Computer Engineering and Distance Learning (CEDL)*, Shanghai, China, 2023

[11] Abhishek, A. Dhankar and N. Gupta, "A Systematic Review of Techniques, Tools and Applications of Machine Learning," *2021 Third International Conference on Intelligent*

Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, 2021,

[12] C. M. Maheshan & H. Prasanna Kumar „Performance of image pre-processing filters for noise removal in transformer oil images at different temperatures", https://link.springer.com/article/10.1007/s42452-019-1800-x, Publikacja: 11.12.2019.

[13] T. Singh and S. Karanchery, "Universal Image Segmentation Technique for Cancer Detection in Medical Images," *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kanpur, India, 2019

[14] P. Mimboro, A. Sunyoto and R. S. Kharisma, "Segmentation of Brain Tumor Objects in Magnetic Resonance Imaging (MRI) Image using Connected Component Label Algorithm," *2021 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA)*, Surabaya, Indonesia, 2021,

[15] H. Qu and W. Zhao, "Image Segmentation Edge Detection Technology Based on Optimization Model," *2023 International Seminar on Computer Science and Engineering Technology (SCSET)*, New York, NY, USA, 2023,

[16] Yawei Ma, Jieru Chi, Ran Hu and Guowei Yang, "A new algorithm for characters segmentation of license plate based on variance projection and mean filter," *2011 IEEE 5th International Conference on Cybernetics and Intelligent Systems (CIS)*, Qingdao, China, 2011 [17] Y. Kinoshita and H. Kiya, "Fixed Smooth Convolutional Layer for Avoiding Checkerboard Artifacts in CNNS," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020

[18] Z. H. Salsabila, R. R. Nurmalasari and L. Kamelia, "Indonesian Sign Language Translation System Using ResNet-50 Architecture-Based Convolutional Neural Network," *2024 10th International Conference on Wireless and Telematics (ICWT)*, Batam, Indonesia, 2024

[19] Nitish Kundu „Exploring ResNet50: An In-Depth Look at the Model Architecture and Code Implementation", https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f, Publikacja: 23.01.2023.

[20] Model Raspberry Pi 4B, https://grabcad.com/library/raspberry-pi-4-model-b-1, Ostatnio odwiedzona: 26.12.2024.

[21] Adam Mickiewicz „Pan Tadeusz”

[22] Dokumentacja Raspberry Pi model 4B, https://botland.com.pl/moduly-i-zestawy-raspberry-pi-4b/14647-raspberry-pi-4-model-b-wifi-dualband-bluetooth-4gb-ram-18ghz-5056561800349.html, Ostatnio odwiedzona: 26.12.2024.

[23] Dokumentacja kamery 5MP OV5647, https://botland.com.pl/kamery-do-raspberry-pi/6850-kamera-arducam-ov5647-5mpx-dla-raspberry-pi-zgodna-z-wersja-oryginalna-5904422375324.html, Ostatnio odwiedzona: 26.12.2024.

[24] G. Kumar, P. Kumar and D. Kumar, "Brain Tumor Detection Using Convolutional Neural Network," *2021 IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, Tumkur, Karnataka, India, 2021

[25] Schemat konwolucyjnej sieci neuronowej, https://bdtechtalks.com/wp-content/uploads/2019/08/Artificial-Neuron.png, Ostatnio odwiedzona: 04.01.2025.

[26] Y. Cheng and B. Li, "Image Segmentation Technology and Its Application in Digital Image Processing," *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, Dalian, China, 2021