––



**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

Spis treści

[Rozdział 1 Wstęp 4](#_Toc184908308)

[1.1 Cel Pracy 4](#_Toc184908309)

[1.2 Zakres pracy 4](#_Toc184908310)

[1.3 Założenia projektowe 5](#_Toc184908311)

[Rozdział 2 Przegląd istniejących rozwiązań 7](#_Toc184908312)

[2.1 LyriQ Text-to-Speech Reader 7](#_Toc184908313)

[2.2 Envision Low Vision Glasses 8](#_Toc184908314)

[2.3 Text to Speech (TTS) 8](#_Toc184908315)

[Rozdział 3 Zagadnienia uczenia maszynowego 9](#_Toc184908316)

[3.1 Sieć neuronowa 9](#_Toc184908317)

[3.2 Transfer Learning 10](#_Toc184908318)

[Rozdział 4 Proces przetwarzania obrazu 12](#_Toc184908319)

[4.1 Obraz RGB i binaryzacja 13](#_Toc184908320)

[4.2 Filtracja i usuwanie szumów 14](#_Toc184908321)

[4.3 Segmentacja 19](#_Toc184908323)

[Rozdział 5 Klasyfikacja przy użyciu sieci neuronowej 23](#_Toc184908324)

[5.1 Baza danych 23](#_Toc184908325)

[5.2 Budowa sieci neuronowej 24](#_Toc184908326)

[Rozdział 6 Klasyfikacja za pomocą *Transfer Learningu* 27](#_Toc184908327)

[Część praktyczna 27](#_Toc184908328)

[Opis software 27](#_Toc184908329)

[Aplikacja do testów 27](#_Toc184908330)

[Projekt obudowy 27](#_Toc184908331)

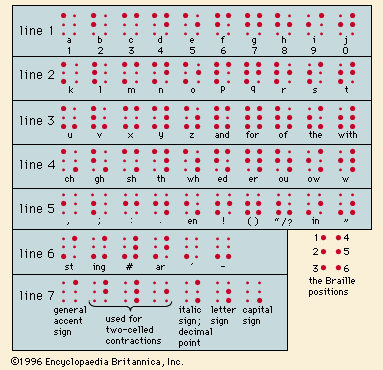
[Wnioski 27](#_Toc184908332)

[Podsumowanie 27](#_Toc184908333)

[Bibliografia 27](#_Toc184908334)

# Rozdział 1 Wstęp [2]

Na przestrzeni dziejów 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 opiera jest uproszczeniem kodu wojskowego wymyślonego przez Charlesa Barbiera, który pozwalał na komunikacje między oficerami w nocy. Alfabet Braille’a składa się z 63 znaków składających się z maksymalnie 6 wypukłych kropek ustawionych w macierzy 3x2.



Rys.1.1 Alfabet Braille’a

## 1.1 Cel Pracy

Celem pracy jest stworzenie urządzenia służącego do rozpoznawania liter i cyfr w języku polskim i angielskim służącego jako pomoc dla osób niewidomych. Narzędzie 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 gotową funkcją środowiska MATLAB, która została rozszerzona o polskie znaki za pomocą metody Transfer Learningu. 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 oraz narzędzia oraz części *hardware* wymagane w jej użyciu.

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

Rozdział 3 wyjaśnia zagadnienia związane z uczeniem maszynowym, które są wykorzystane w pracy oraz omawia ich zastosowanie.

Rozdział 4 jest poświęcony przygotowaniu obrazu. Autor opisuję i porównuje wybrane metody filtracji i segmentacji, które mogą zostać wykorzystane w projekcie oraz wybiera optymalną opcję.

Rozdział 5 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.

## 1.3 Założenia projektowe

Autor zdecydował się na użycie mikrokomputera RaspberryPi 4B 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 RaspberryPi i bardzo ułatwiają programowanie urządzenia. W tabeli 1.1 można odczytać specyfikacje użytego mikrokomputera.

Tabela 1.1 Dokumentacja RaspberryPi 4B

|  |  |
| --- | --- |
| 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 |

Jako źródło obrazu wybrano kamerę 5MP OV5647 ze względu na kompatybilność z platformą RaspberryPi 4. Dane techniczne użytej kamery są widoczne w tabeli 1.2.

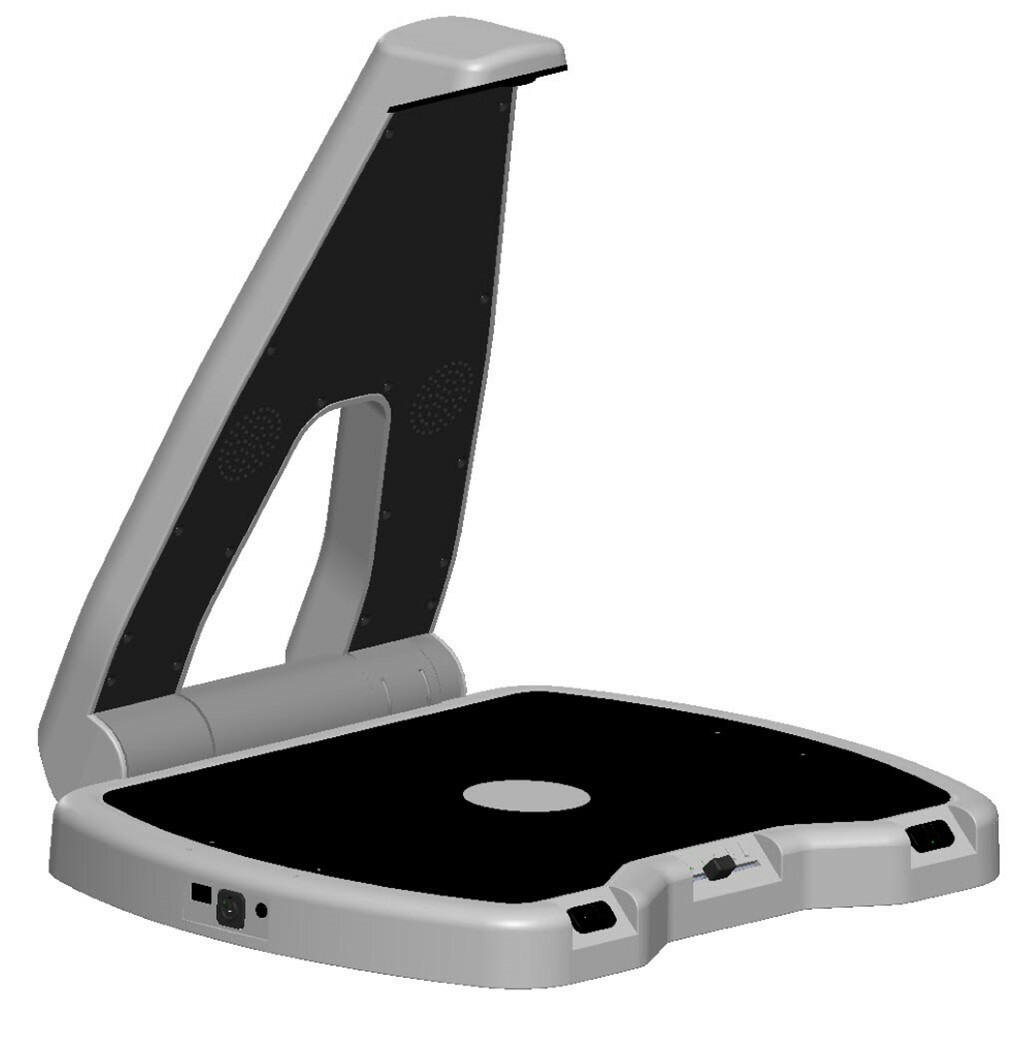
Tabela 1.2 Dokumentacja kamery 5MP OV5647

|  |  |
| --- | --- |
| 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 |

# Rozdział 2 Przegląd istniejących rozwiązań

Obecnie na rynku istnieje kilka ciekawych rozwiązań pomagających 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 to różnorodność charakterów pisma i nie tylko, ponieważ znaczenie ma również nawet kolor długopisu i strony, który może powodować zaburzenia w odczycie urządzenia. 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 ogromna. Jednak z mniejszą ceną naturalnie wiąże się gorsza jakość części *hardware’u,* takich jak kamera bądź skaner. Użycie gorszego sprzętu oznacza również zmniejszoną dokładność rozpoznawanie znaków. Z tych względów rynek narzędzi rozpoznających tekst jest dosyć zróżnicowany.

## 2.1 LyriQ Text-to-Speech Reader [1]



*Rys.2.1 Urządzenie LyriQ*

Urządzenie firmy Zyrlo (Rys.2.1), w odróżnieniu od narzędzia wykonanego przez autora, wykorzystuje skaner w celu zwiększenia dokładności rozpoznawanie tekstu. Użycie skanera powoduje, że LyriQ jest dosyć sporym i relatywnie ciężkim obiektem. Waży około 1,3 kg (3 lbs) i ma możliwość złożenia na płasko, co faktycznie ułatwia transport. Najbardziej odstraszającym czynnikiem w samym produkcie jest niewątpliwie cena, która waha się między 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.

## 2.2 Envision Low Vision Glasses [3]



*Rys.2.2 Envison Low Vison Glasses*

Ciekawym rozwiązaniem są okulary firmy Envision. Zaprojektowane z myślą o komforcie użytkownika okulary poza rozpoznawaniem teksu zostały wyposażone w szereg różnych funkcjonalności takich jak, np. rozpoznawanie kolorów, skanowanie dokumentów czy nawet rozpoznawanie ludzi. Urządzenie zostało zaprojektowane na podstawie *Google Glass Enterprise Edition 2* oraz wyposażone w lekki 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 funkcjonalności. Najbardziej podstawowa wersja okularów Envsion kosztuje około 8 500 zł.

## 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.

# Rozdział 3 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.

Do *machine learningu* można wykorzystać najróżniejsze algorytmy. Najbardziej popularne to:

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

W tym rozdziale autor skupi się na opisie sieci neuronowych, ponieważ to właśnie one są wykorzystane w projekcie.

## 3.1 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], którzy wytrenowali 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 zadziwiającą dokładnością.

Najbardziej popularne sieci neuronowe służące do OCR (*Optical Character Recognition*) są kilkuwarstwowe z propagacją wsteczną. „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” [8].

## 3.2 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 *Ds*i zadania uczącego *Ts*, dziedzina celu *Dt*i zadania uczącego *Tt*, *transfer learning* ma na celu poprawę efektów uczenia funkcji przewidywania celu *ft ()* w dziedzinie *Dt* wykorzystując wiedzę w dziedzinie *Ds*i zadaniu *Ts*, gdzie *Ds*≠ *Dt*oraz *Ts* ≠ *Tt*.[9][10]

W swojej pracy Pan i Yang dzielą transfer learning na 3 rodzaje (Tabela 3.1):

* indukcyjny,
* nienadzorowany,
* transdukcyjny.

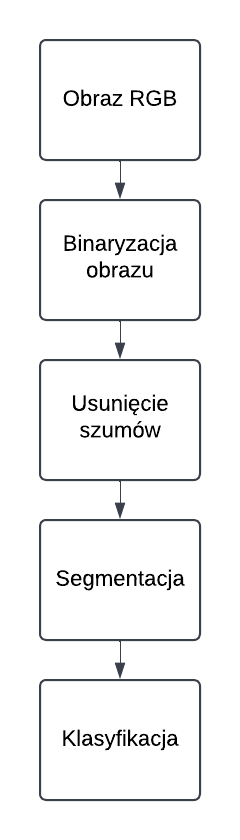
Tabela 3.1 Rodzaje *transfer learningu*

|  |  |  |  |
| --- | --- | --- | --- |
| 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.

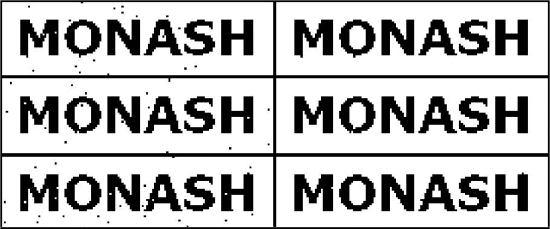
# Rozdział 4 Proces przetwarzania obrazu

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 (Rys.3.1).



Rys.3.1 Schemat blokowy dla przetwarzania obrazów

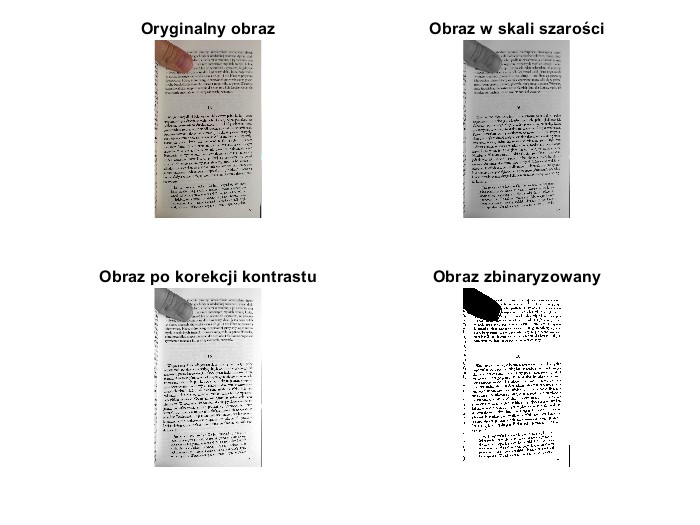
Najpierw obraz RGB musi zostać poddany binaryzacji, czyli konwersji do formy binarnej (czarno-białej). Następnym etapem jest usunięcie szumów. Proces polega na analizie obrazu binarnego i znalezieniu pikseli, które nie pasują do ich otoczenia. Na Rys.3.2 widać, że kilkanaście pikseli nie należą do liter, ale istnieją w obrazie, więc trzeba je usunąć, ponieważ zaburzyłoby to klasyfikację sieci neuronowej. Kolejną częścią jest odpowiednia segmentacja obrazu na poszczególne litery. Ostatnim etapem jest klasyfikacja obrazu do odpowiedniej litery.



Rys.3.2 Przykład usuwania szumów

## 4.1 Obraz RGB i binaryzacja

Urządzenie jest obsługiwane przez kamerę o rozdzielczości 3 MP, więc obraz może być delikatnie zamazany. W tym celu, aby uzyskać jak najlepszą binaryzację całego obrazu, autor najpierw przekształca go do odcieni szarości, po czym dokonuje korekcji kontrastu (Rys.4.1). 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 dla sieci neuronowej. Użycie tej operacji zwiększa dokładność i szybkość klasyfikacji.

Rys.4.1 Operacje przeprowadzono na obrazie

## 4.2 Filtracja i usuwanie szumów [12]

Po binaryzacji obrazu należy przefiltrować go pod względem szumów. W tym celu autor rozważał użycie kilku filtrów:

* filtr uśredniający,
* filtr medianowy,
* filtr Wienera,
* filtr przy użyciu operacji morfologicznych.

Filtr uśredniający poprawia wartość piksela na podstawie średniej wartości pikseli w jego otoczeniu. Rxy reprezentuje sąsiedztwo okno podobrazu o wymiarach m × n, o środku w punkcie (x, y). Dla poziomu szarości piksela (x, y) w K, filtr zmienia poziom szarości J(x, y) z szczegółami otaczającego piksela. Równanie znajduje się poniżej:

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. Definicja tego filtru wygląda następująco:

# 

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. Jego równanie wygląda następująco:

,

gdzie:

* : Wyjściowy (oczyszczony) obraz,
* : Sygnał wejściowy z szumem,
* : Wariancja obrazu oryginalnego,
* : 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. Ta funkcja znajduje obiekty o rozmiarach zadanych jako argument i je usuwa. Tą operacje należy wykonać na obrazie binarnym oraz trzeba zadać stały próg wielkości obiektów. Ta metoda jest bardzo użyteczna przy obrazach, gdzie litery są wielki i nie nachodzą na siebie, ponieważ bezpośrednio niechciane artefakty. Wykorzystanie bwareaopen posiada wadę w postaci długiego czasu wykonania, spowodowanego skomplikowanym algorytmem.

Autor wykonał porównanie rezultatów po filtracji (Rys. 4.2) oraz przeprowadził testy czasowe w celu wybrania najlepiej pasującej opcji dla problemu rozpoznawania znaków drukowanych.

Obraz zawierający tekst, Czcionka, zrzut ekranu, Drukowanie

Opis wygenerowany automatycznieRys.4.2 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 liter znacząco. Warto zauważyć, że bardzo dobry wynik daje również użycie funkcji *bwareaopen.* Przy dużych literach właśnie użycie tej metody najlepiej wyodrębnia litery.

Przeprowadzono również testy czasowe, których rezultaty przedstawiono w Tabeli 4.1.

Tabela 4.1 Porównanie testów czasowych

|  |  |
| --- | --- |
| Rodzaj filtru | Średni czas potrzebny, aby przetworzyć obraz [s] |
| Filtr średniej | 0.2673 |
| Filtr medianowy | 0.3246 |
| Filtr Wienera | 0.4625 |
| Funkcja *bwareaopen* | 3.1932 |

Czasy potrzebne do przefiltrowania obrazu (Tabela 4.1) są porównywalne dla filtrów średniej, medianowego i Wienera. Jak przewidywano funkcja *bwareaopen* zajmuję dużo więcej czasu, aby wykonać zadanie. Biorąc pod uwagę testy czasowe oraz dokładność w 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.

Obraz zawierający tekst, Czcionka, atrament, list

Opis wygenerowany automatycznie

Rys.4.3 Obrazy przy użyciu małych okien dla filtra Wienera

Obraz zawierający tekst, Czcionka, Drukowanie, atrament

Opis wygenerowany automatycznie

Rys.4.4 Obrazy przy użyciu średnich okien dla filtra Wienera

Obraz zawierający tekst, Czcionka, atrament, zrzut ekranu

Opis wygenerowany automatycznie

Rys.4.5 Obrazy przy użyciu dużych okien dla filtra Wienera

Z powyższych rysunków można wywnioskować, że duże okna 15×15 i 21×21 ewidentnie są zbyt wielkie i filtrują zbyt znacząco 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 niechcianych szumów niż okno 3×3 przy zachowaniu odporności na zmianę wyglądu liter.

## 4.3 Segmentacja

Po obróbce obraz należy podzielić na pojedyncze litery, aby sieć neuronowa mogła je sklasyfikować. Istnieją różne metody segmentacji, jednak przy rozpoznawaniu tekstu najbardziej popularne są.:

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

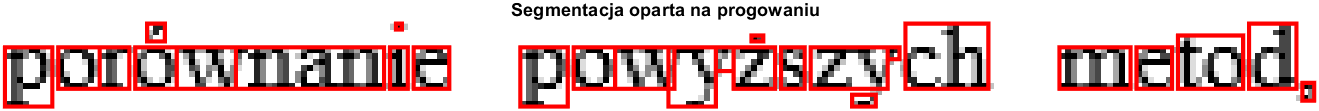
Segmentacja oparta na progowaniu jest techniką wykorzystywaną ze względu na swoją prostotę i łatwość w implementacji, często przy obrazach z małą entropią. Ta metoda dzieli obraz na regiony na podstawie wartości pikseli w skali szarości. Wartość progu może być wybrana automatycznie, np. za pomocą funkcji *graythresh* w środowisku Matlab, która opiera się na metodzie Otsu, lub ręcznie. 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. Ten algorytm opiera się na wyborze optymalnego progu na podstawie maksymalizacji wariancji pomiędzy klasami obrazu. [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]

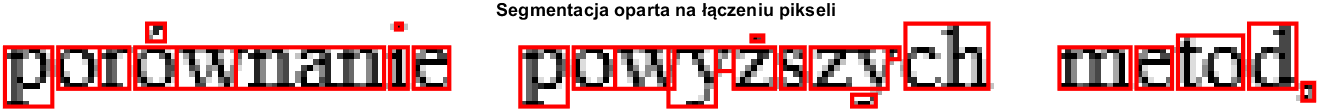
Segmentacja wykorzystująca detekcję krawędzi polega na analizie zestawu pikseli, gdzie otaczające go piksele w skali szarości lub zmiany struktury sugerują zmianę regionu. Krawędź obiektu w obrazie jest odzwierciedlona poprzez różnice w skali szarości, więc pierwszym krokiem w tym algorytmie jest znalezienie 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]

Autor wykonał porównanie powyższych metod, w celu wybrania najlepszego rozwiązania dla poruszanego problemu. Samo porównanie obejmuje jakość dokonanej segmentacji oraz szybkość działania algorytmu.



Rys.4.6 Segmentacja oparta na progowaniu



Rys.4.7 Segmentacja oparta na łączeniu pikseli



Rys.4.8 Segmentacja oparta na liniach konturowych



Rys.4.9 Segmentacja oparta o projekcje pionowe

Największym problemem 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 to zaobserwować to zachowanie przy segmentacji polegającej na progowaniu (Rys.4.6) i łączeniu pikseli (Rys.4.7). Obydwie metody dają podobne rezultaty, ponieważ wykorzystano w nich do wyznaczenia progu metodę Otsu. Algorytm używający projekcji (Rys.4.9) nie jest w stanie dobrze podzielić obrazu, ze względu na jego zbyt słabą jakość i zbyt dużą bliskość liter. Najlepszą segmentacje zdecydowanie przeprowadził algorytm oparty na liniach konturowych (Rys.4.8), który poprawnie rozdzielił wszystkie litery w obrazie testowym.

Tabela 4.2 Porównanie testów czasowych

|  |  |
| --- | --- |
| Rodzaj segmentacji | Średni czas potrzebny, aby przetworzyć obraz [s] |
| Progowanie | 0.1821 |
| Łączenie pikseli | 0.2075 |
| Linie konturowe | 0.2625 |
| Projekcje | 0.1772 |

W testach czasowych (Tabela 4.2) najlepiej wypadł algorytm wykorzystujący projekcje, a najgorzej ten oparty o linie konturowe. Jednakże biorąc pod uwagę dokładność wszystkich metod autor zdecydował się wybrać segmentacje używającej 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 delikatną dylatacje, aby połączyć znaki diakrytyczne (np. kropka na „i”) z resztą litery w jeden region.

# Rozdział 5 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 3.1. W tym rozdziale autor skupi się na budowie przygotowanej sieci oraz wyjaśni działania wybranych funkcji i parametrów.

## 5.1 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 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. Ten zabieg sprawia, że zbiór danych jest bardziej zróżnicowany, przez co sieć neuronowa lepiej rozpoznaje cechy danego obrazu.

Obraz zawierający szkic, biały, czarne i białe, design

Opis wygenerowany automatycznie

Rys.5.1 Przykłady liter z przygotowanej bazy danych

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.

## 5.2 Budowa sieci neuronowej

Do wykonania sieci neuronowej zostało wykorzystane narzędzie *Deep Network Designer*, które jest dostępne w *Deep Learning Toolbox.* Deep Network Designer to aplikacja, w której można tworzyć blokowe schematy sieci neuronowych. W tym celu można wykorzystać gotowe funkcję służące do np. optymalizacji lub normalizacji sieci. Autor wybrał to narzędzie ze względu na prostotę w obsłudze oraz łatwość w zaimplementowaniu w środowisku Matlab. Wykonano poniższy schemat blokowy konwolucyjnej sieci neuronowej (Rys.5.2). Jako wejście zadawany 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,

gdzie:

* to znormalizowana wartość cechy,
* to pojedyncza wartość wejściowa,
* to średnia arytmetyczna wartości wejściowych,
* to wariancja wartości wejściowych
* to bardzo mała liczba dodawana do mianownika, aby uniknąć dzielenia przez 0.

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Rys. 5.2 Schemat blokowy sieci neuronowej

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

Funkcja ReLu jest bardzo popularna ze względu na łatwość w implementacji, szybkość działania oraz nie cierpi 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 zależy 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 wybiera największą 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 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 przepuszczane 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:

gdzie:

* to logit, czyli surowy wynik, dla klasy i,
* to liczba klas,
* to eksponenta logitu,
* 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.

# Rozdział 6 Klasyfikacja za pomocą *Transfer Learningu*

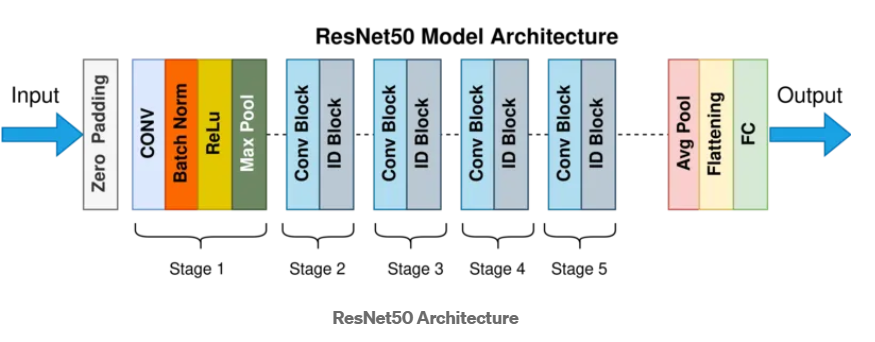
Istotą *Transfer Learingu* jest douczenie już wytrenowanej sieci, co dokładniej autor opisuje w rozdziale 3.2. Kluczowym aspektem staje się wybór odpowiedniej sieci neuronowej. Autor postanowił wybrać sieć ResNet50 ze względu na jej popularność i jej zastosowania w dziedzinie detekcji obiektów. W środowisku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

## 6.1 Budowa ResNet50

ResNet-50 to głęboka sieć neuronowa z rodziny Residual Networks, składająca się z 50 warstw konwolucyjnych połączonych za pomocą bloków resztkowych. Te bloki wykorzystują tzw. *shortcut connections,* co unika problemu zanikającego gradientu, spowodowanego przez dużą liczbę warstw.Architektura sieciResNet50 jest podzielona na 4 sekcje:

* warstwy konwolucyjne,
* blok tożsamościowy (ang. *identity block*),
* blok konwolucyjny,
* pełnie 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.6.1 Architektura sieci ResNet 50 [19]

Najbardziej kluczowym aspektem sieci z rodziny ResNet jest wykorzystanie *shortcut connections.* Te połączenia pozwalają informacją 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. *Shotcut connections* są używane w blokach konwolucyjnych i tożsamościowych.[19]

## 6.2 Modyfikacja sieci ResNet 50

W celu wykorzystania metody Transfer Learningu należało zmodyfikować sieć ResNet 50. Na początku zmodyfikowano dane wejściowe, ponieważ ta sieć wymaga, aby były one rozmiaru 224×224 oraz w 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 przez mniejszą liczbę epok.

# Rozdział 7 Porównanie wyników

W celu sprawdzenia poprawnego działania obydwóch rozwiązań przeprowadzono testy dokładności na platformie RaspberryPi z wykorzystaniem przygotowanej aplikacji. Spodziewanym rezultatem było uzyskanie lepszych wyników za pomocą douczonej sieci ResNet 50 ze względu na większą liczbę warstw konwolucyjnych.

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

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

Rys. 7.1 Dokładność przygotowanej sieci

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

Rys.7.2 Dokładność sieci ResNet50

Można zauważyć, że sieć ResNet 50 (Rys.7.2) dużo szybciej zbiega do 100 % dokładności niż przygotowana sieć (Rys.7.1) w trakcie uczenia. Przeprowadzono również sprawdzenie dokładności na zbiorze testowym, który jest taki sam dla obydwu sieci. Dla przygotowanej sieci ta dokładność wynosiła 79,14 %, a dla sieci ResNet 50 98,24%. Trzeba zaznaczyć, że sieć ResNet 50 była trenowana na mniejszej liczbie epok, ze względu na bardzo długi czas treningu.

Przygotowano również macierz konfuzji (ang*. confusion matrix*) (Rys.7.3), dzięki której można zaobserwować, które litery były najczęściej mylone.

Obraz zawierający linia, diagram, Wykres, Równolegle

Opis wygenerowany automatycznie

Rys. 7.3 Macierz konfuzji dla przygotowanej sieci

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 operację przeprowadzono również dla sieci ResNet 50 (Rys.7.4).

Obraz zawierający linia, diagram, Wykres, Równolegle

Opis wygenerowany automatycznie

Rys.7.4 Macierz konfuzji dla sieci ResNet 50

Jak się spodziewano wyniki otrzymane dzięki sieci ResNet 50 są dokładniejsze i klasyfikacja jest bardziej poprawna.

Następnie sprawdzono działanie całego urządzenia bazując na obydwóch modelach wykorzystując platformę RaspberryPi. W tym celu przygotowano 2 rodzaje tekstów: jeden, gdzie tekst jest dobrze widoczny z wyraźnymi przerwami oraz fragment, gdzie litery są mniejsze bez dużych odstępów.

# Rozdział 8 Aplikacja przeznaczona do testów

W celu poprawienia jakości testowania urządzenia autor stworzył prostą aplikację (Rys.8.1). Interfejs graficzny składa się z aktualnego obrazu przekazywanego z kamery oraz ma możliwość wyboru rodzaju modelu przeznaczonego do rozpoznawania tekstu.

Obraz zawierający tekst, zrzut ekranu, komputer, oprogramowanie

Opis wygenerowany automatycznie

Rys.8.1 Interfejs aplikacji

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

# Rozdział 9 Projekt obudowy

Ostatnim etapem projektu było zaprojektowanie obudowy do urządzenia. W tym celu użyto programu Inventor Professional 2025 firmy Autodesk. Wykonano w nim model 3D obudowy, aby wykorzystać technikę druku 3D

# Wnioski

# Podsumowanie

# Bibliografia

[1] https://www.zyrlo.com/ - dokumentacja i opis lyriq

[2] History of the blind – Brian R. Miller

[3] https://www.letsenvision.com/glasses/home - strona producenta

[4] https://cloud.google.com/text-to-speech - opis tts

[5] S. Rost, "Character Recognition: Image and Video Available: Project One by Stanislav Rost"

[6] V. Kwatra, "Optical Character Recognition"

[7] [Velappa Ganapathy](https://ieeexplore-1ieee-1org-1000047a700a9.wbg2.bg.agh.edu.pl/author/37426447700); [Charles C. H. Lean](https://ieeexplore-1ieee-1org-1000047a700a9.wbg2.bg.agh.edu.pl/author/37697796800) „Optical Character Recognition Program for Images of Printed Text using a Neural Network”

[8] Ryszard Tadeusiewicz, Maciej Szaleniec: Leksykon sieci neuronowych

[9] Sinno Jialin Pan and Qiang Yang „A Survey on Transfer Learning”

[10] Jingbo Hao „Deep learning-based medical image analysis with explainable transfer learning”

[11] Abhishek; Abhishek Dhankar; Neha Gupta „A Systematic Review of Techniques, Tools and Applications of Machine Learning”

[12] C. M. Maheshan & H. Prasanna Kumar „Performance of image pre-processing filters for noise removal in transformer oil images at different temperatures”

[13] Tripty Singh; Siddharth Karanchery „Universal Image Segmentation Technique for Cancer Detection in Medical Images”

[14] Prasetyo Mimboro; Andi Sunyoto; Rizqi Sukma Kharisma „Segmentation of Brain Tumor Objects in Magnetic Resonance Imaging (MRI) Image using Connected Component Label Algorithm”

[15] Huiyan Qu; Wei Zhao „Image Segmentation Edge Detection Technology Based on Optimization Model”

[16] Yawei Ma; Jieru Chi; Ran Hu; Guowei Yang A new algorithm for characters segmentation of license plate based on variance projection and mean filter

[17] Yuma Kinoshita; Hitoshi Kiya „Fixed Smooth Convolutional Layer for Avoiding Checkerboard Artifacts in CNNS”

[18] Zalfa Humaira Salsabila; Rin Rin Nurmalasari; Lia Kamelia „Indonesian Sign Language Translation System Using ResNet-50 Architecture-Based Convolutional Neural Network”

[19] Nitish Kundu „Exploring ResNet50: An In-Depth Look at the Model Architecture and Code Implementation”