POLITECHNIKA LUBELSKA

Wydział Elektrotechniki i Informatyki

Kierunek Informatyka

PRACA INŻYNIERSKA

Projekt i implementacja systemu rozpoznawania znaków alfanumerycznych oparty na głębokich sieciach neuronowych

Design and implementation of alphanumeric characters recognition system based on deep neural network

Dyplomant:

Małgorzata Wróbel

nr albumu: 86686

Marek Andrzej Piszczaniuk

nr albumu: 86674

Mikołaj Jędrzej Nowak

nr albumu: 67416

Promotor:

Dr Paweł Powroźnik

Lublin 2020

Spis treści

Streszczenie

Głównym tematem niniejszej pracy jest projekt i implementacja systemu służącego do rozpoznawania znaków alfanumerycznych z wykorzystaniem głębokich sieci neuronowych. Treść tej pracy obejmuje opis użytych technologii oraz narzędzi, wymagań funkcjonalnych i niefunkcjonalnych aplikacji mobilnej jak również diagramy BPMN oraz zaprezentowano działanie aplikacji. Główną jej częścią jest opis implementacji algorytmu rozpoznawania i aplikacji mobilnej zaprojektowanej na system Android oraz przeprowadzone testy.

Abstract

The main topic of the following thesis is project and implementation of a system for alphanumeric characters recognition with use of deep neural network. The thesis contains a description of used technologies and tools, functional and non-functional requirements of mobile application as well as BPMN diagrams and presentation of a working application. The main part of this thesis is the description of implementation of recognition algorithm and mobile application designed for Android system and conducted tests.

Wstęp

Automatyzacja weszła do różnych dziedzin w dość szybkim tempie. Nie ominęło to również przewozu drogowego. Powszechnie są wykorzystywane systemy inteligentnego transportu czyli systemy informacji i komunikacji, które mają na celu świadczenie usług związanych z różnymi rodzajami przewoźnictwa i zarządzaniem ruchem. Pozwala to na lepsze informowanie użytkowników oraz zapewnia bezpieczniejsze, bardziej skoordynowane i inteligentniejsze korzystanie z sieci transportowych. Łączy on telekomunikację, techniki informatyczne i elektronikę z inżynierią transportu w celu obsługi, utrzymywania, projektowania, planowania oraz zarządzania systemami transportu.

Ciągły rozwój motoryzacji i stale wzrastająca liczba pojazdów, które poruszają się po drogach sprawiają, że kolejne rozbudowy infrastruktury, jak i nadzór nad już istniejącą, jest coraz trudniejszy. Aby poprawić tą sytuacje rozpoczęto automatyzację niektórych czynności.

Według danych GUS [26] w roku 2018 liczba zarejestrowanych pojazdów w Polsce wynosiła ponad 30 milionów. Liczba ta oznacza, że automatyzacja pewnych czynności jest jak najbardziej konieczna, a w szczególności tych związanych z ruchem drogowym w celu obsłużenia jego rosnącego natężenia. Dziedziny, które jej najbardziej podlegają to:

- analiza ruchu drogowego,
- naliczanie opłat i kontrola dostępu,
- wykrywanie wykroczeń drogowych.

We wszystkich tych zadaniach potrzebna jest identyfikacja pojazdu. Do czego używa się systemów automatycznego rozpoznawania numerów rejestracyjnych. Są one powszechnie stosowane na granicach państw, głównych szlakach komunikacyjnych, w samochodach operacyjnych służb skarbowych oraz celnych. Również używane na parkingach, na przykład w celu sprawdzenia czy dany pojazd jest uprawniony do wjazdu.

Niniejsza praca stanowi próbę opracowania własnego systemu rozpoznawania tablic rejestracyjnych w oparciu o narzędzia już istniejące oraz samodzielnie opracowane rozwiązania.

1. Cel i zakres pracy

Celem pracy jest zaprojektowanie oraz implementacja systemu do rozpoznawania numerów tablic rejestracyjnych na podstawie zdjęcia. Aplikacja składa się z części mobilnej – pozwalającej na pozyskanie obrazu do analizy oraz części serwerowej - odpowiedzialnej za identyfikację znaków, dzięki wykorzystaniu głębokich sieci neuronowych. Aplikacja mobilna ma za zadanie umożliwić użytkownikom zrobienie zdjęcia lub wybranie go z galerii oraz zaprezentować rozpoznane znaki. Cześć serwerowa odpowiada za wyodrębnienie ze zdjęcia tablicy rejestracyjnej oraz identyfikację znaków znajdujących się na niej.

Aplikacja oraz algorytm zostały wykonane z zastosowaniem nowoczesnych narzędzi i technologii takich jak: język programowania Python w wersji 3.7, biblioteka OpenCV w wersji 4.1, biblioteka NumPy 1.17.4, biblioteka Pandas 0.25.3, biblioteka Seaborn 0.9, bilbioteka Matplotlib 3.1.1, biblioteka Tensorflow 2.0, środowisko Jupyter Notebook, szkielet programistyczny NativeScript, szkielet programistyczny Angular, język programowania Typescript, język programowania PHP 7.2, edytor Visual Studio Code, system kontroli wersji Git.

Zakres pracy

Zakres pracy obejmuje:

- przedstawienie wykorzystanych technologii oraz narzędzi,
- prezentacja wymagań i założeń projektowych aplikacji,
- opis działania aplikacji mobilnej,
- zaprezentowanie sposobu wykrywania tablicy rejestracyjnej,
- prezentacja metody segmentacji znaków,
- przedstawienie zagadnienia sieci neuronowych,
- opisanie sposobu działania sieci neuronowych,
- zaprezentowanie metody pozwalającej na rozpoznawanie znaków alfanumerycznych,
- opis procesu implementacji sieci neuronowej,
- przedstawienie etapów uczenia sieci neuronowej,
- określenie stopnia zgodności otrzymanych wyników z początkowym obrazem.

Podział pracy

Podczas realizacji pracy dokonano podziału obowiązków wg Tabeli 1.1

Tabela 1.1. Podział obowiązków

Małgorzata Wróbel	4.3.1, 4.3.2, 4.3.3, 4.3.4, 6, 7, 12.2, 12.3
Marek Piszczaniuk	3, 4.1, 4.2, 5, 12.1
Mikołaj Nowak	4.3.5, 8, 9, 10, 11
Wspólnie	1, 2

2. Projekt systemu

W tym rozdziale przybliżone zostały wymagania funkcjonalne i niefunkcjonalne opracowanej aplikacji. Dodatkowo w fazie projektowania powstały różne diagramy takie jak diagramy BPMN.

Wymagania niefunkcjonalne

System rozpoznawania tablic rejestracyjnych składa się z 4 elementów: aplikacji mobilnej, programu lokalizującego tablicę i segmentującego znaki, sieci neuronowej przetwarzającej rozpoznane znaki z tablicy na tekst oraz serwera obliczeniowego odpowiedzialnego za komunikację z aplikacją i działanie sieci neuronowej. Aplikacja mobilna ma za zadanie pozyskać obraz, przekazać go na serwer i uzyskać wynik w postaci ciągu znaków alfanumerycznych.

Wymagania funkcjonalne dla aplikacji mobilnej :

- aplikacja ma być prosta w użyciu,
- program ma pozwolić nadać uprawnienia do plików oraz aparatu,
- program ma dać możliwość wyboru sposobu pozyskania zdjęcia z urządzenia,
- aplikacja ma informować użytkownika o błędach,
- aplikacja ma się łączyć z serwerem obliczeniowym, a w przypadku problemów z komunikacją – informować o tym,
- aplikacja ma przekazać pozyskany obraz na serwer.

Wymagania funkcjonalne serwera:

- serwer ma odbierać i przechowywać zdjęcia w pamięci,
- połączenie z serwerem musi odbywać się za pomocą szyfrowanego protokołu,
- serwer musi analizować obraz i przekazywać rezultat działania do aplikacji mobilnej w możliwie jak najkrótszym czasie,
- system operacyjny musi pozwolić na uruchomienie skryptów w języku Python w wersji 3.7.

Wymagania funkcjonalne programu lokalizującego tablicę i segmentującego znaki:

- funkcja musi otrzymać obraz potrzebny do przetworzenia,

- algorytm powinien umożliwić znalezienie tablicy rejestracyjnej na zdjęciu, a następnie segmentację znaków na zlokalizowanej tablicy,
- funkcja powinna zwrócić wyodrębnione znaki tablicy,
- jeśli tablica nie zostanie zlokalizowana funkcja powinna zwrócić odpowiedni komunikat.

Wymagania funkcjonalne sieci neuronowej:

- sieć neuronowa ma być w stanie odebrać dane wejściowe z modułu rozpoznającego znaki,
- algorytm ma utworzyć model o z góry zdefiniowanych parametrach,
- program ma przetrenować model na dostarczonych danych treningowych,
- sieć ma zbadać prawdopodobieństwo zgodności dostarczonego obrazu wejściowego z danymi treningowymi,
- program ma zwrócić otrzymany wynik i określić czy jest prawidłowy lub nie,
- program ma przekazać odpowiedź do aplikacji mobilnej.

Wymagania funkcjonalne

Wymagania te nie dotyczą funkcjonalności tylko cech oprogramowania. Określają ograniczenia, które należy spełnić w celu uruchomienia danego programu. Opisywana w tej pracy aplikacja posiada następujące wymagania:

- urządzenie mobilne z systemem Android w wersji 6.0 Marshmallow lub wyższej,
- łatwy w obsłudze interfejs,
- dostęp do Internetu, aparatu i pamięci wewnętrznej telefonu,
- serwer obliczeniowy obsługujący język Python w wersji 3.7 wraz z zainstalowanymi bibliotekami : OpenCV 4.1, imutils 0.5.3, NumPy 1.17.4, Tensorflow 2.0.

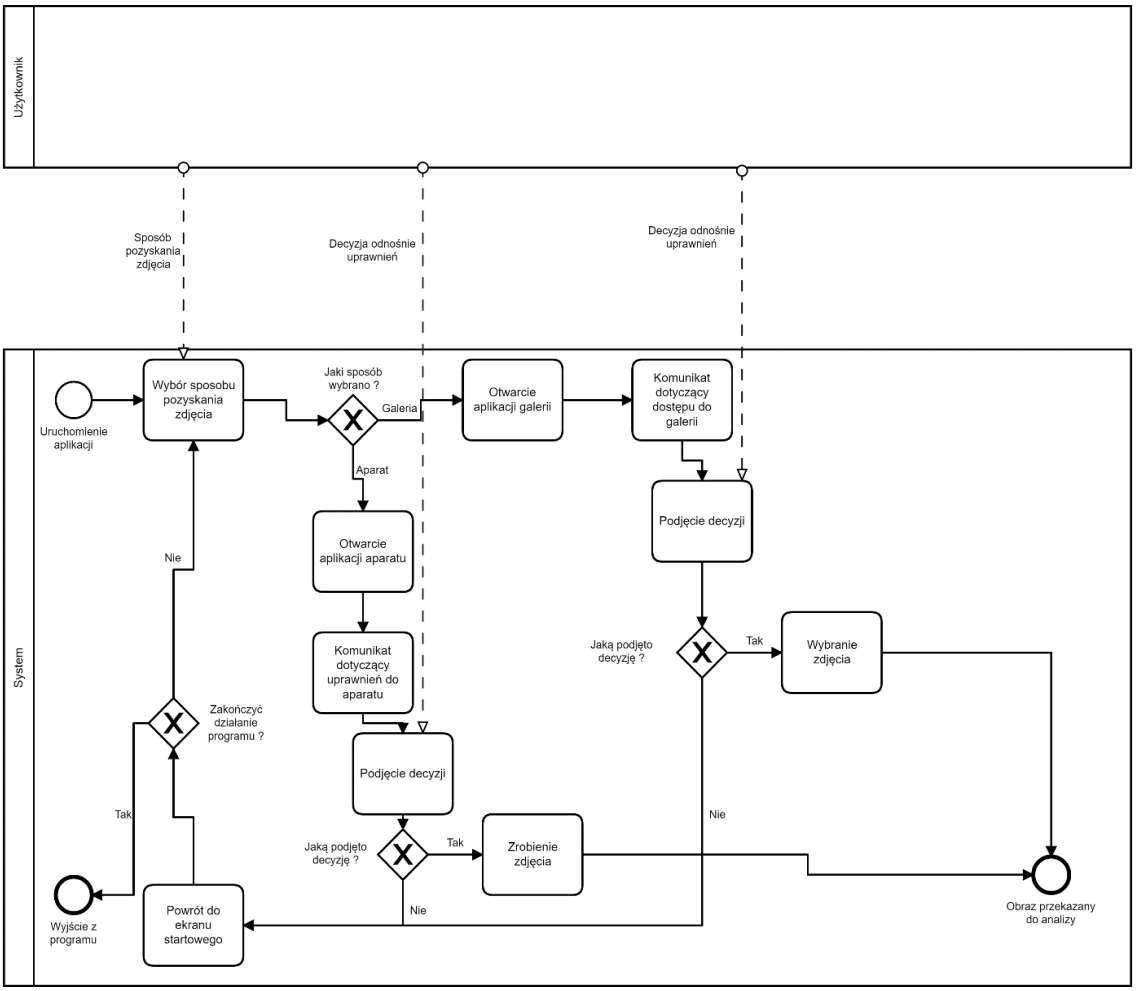
Diagramy BPMN

BPMN (z angielskiego Business Process Model and Notation) jest to notacja graficzna służąca do opisywania procesów biznesowych. Schematy te przedstawiają przepływ informacji i procesy zachodzące w danej części projektu czy też aplikacji. Za pomocą diagramów BPMN można zaplanować działanie systemu od początku do końca, za pomocą symboli, które definiują początkowy etap, zdarzenia pośrednie i zakończenie procesu.

W pracy zostało opracowanych kilka diagramów BPMN, które przedstawiają procesy zachodzące w tej aplikacji, takie jak:

- proces pozyskania zdjęcia i przekazania go do analizy,
- proces wyodrębniana tablicy rejestracyjnej,
- proces segmentacji znaków.

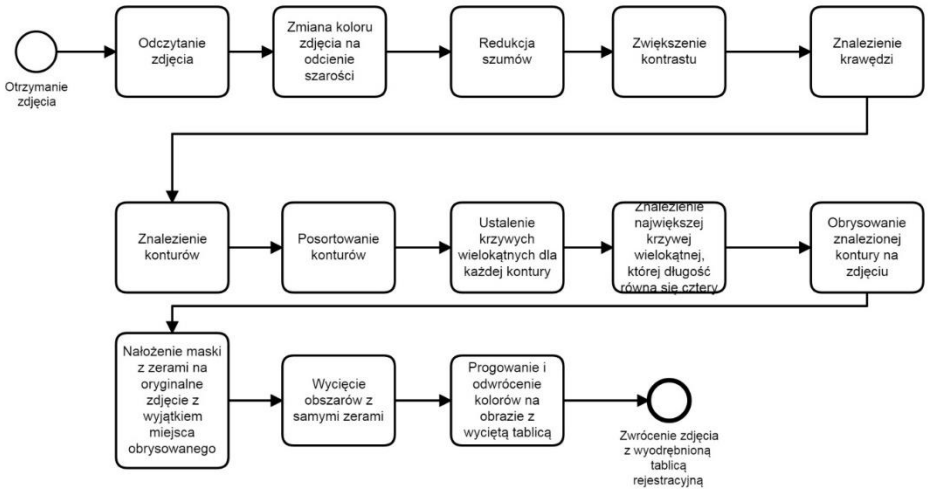
W aplikacji mobilnej zostają wyświetlone przyciski do wyboru sposobu pozyskania obrazu, czyli z wbudowanego aparatu fotograficznego bądź wybranego z galerii. Podczas pierwszego uruchomienia wybranych funkcji wymagane jest nadanie odpowiednich uprawnień. W kolejnym kroku obraz zostaje pobrany i przekazany do analizy. W przypadku braku praw do galerii lub aparatu, następuje powrót aplikacji do ekranu startowego. Cały ten proces ukazano na Rys.2.1.



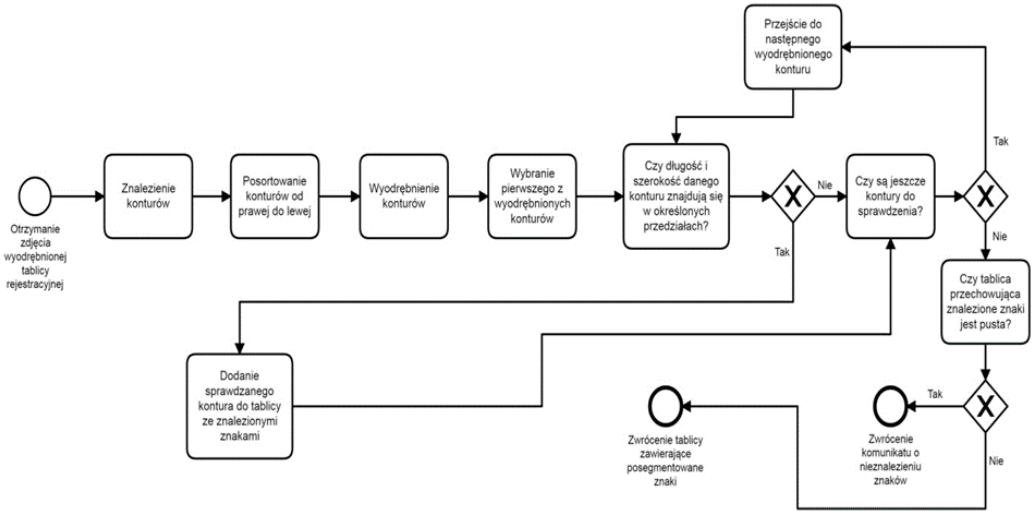
Rys. 2.1. Proces pozyskania obrazu

Aplikacja otrzymuje zdjęcie, które następnie zostaje poddane procesowi wyodrębniana tablicy. Schemat przebiegu powyższego etapu został przedstawiony na rysunku 1 (numer do zmiany po dodaniu innych BPMNów). Na samym początku funkcja odczytuje obraz i zamienia jego kolory na odcienie szarości. Następnie odbywa się redukcja szumów oraz zwiększenie kontrastu fotografii. Kolejnym krokiem jest znalezienie krawędzi, a w następnej kolejności znalezienie konturów. Po tym rozpoznane krawędzie zostają posortowane od największej do najmniejszej. Następnie dla każdej kontury zostają ustalone krzywe wielokątne, po czym jest sprawdzana ich długość i jeśli rozmiar którejś z nich równa się cztery oraz jest ona największa ze wszystkich o tym wymiarze to jest uznawana za krawędź tablicy rejestracyjnej. W dalszej kolejności znaleziony kontur zostaje obrysowany na zdjęciu, po czym zostaje nałożona na niego maska, która składa się z samych zer, co daje obraz całkowicie czarny. Jedynie obszar obrysowany zostaje pominięty w procesie maskowania, dzięki czemu na rysunku widzimy tylko tablicę. Potem pola, które mają jedynie zera są wycinane i następuje progowanie oraz odwrócenie kolorów na otrzymanym fragmencie fotografii.

W ostatnim uzyskany obraz zostaje zwrócony do następnej fazy algorytmu. Kolejnym etapem jest przetworzenie wyodrębnionej tablicy rejestracyjnej w celu segmentacji znaków (Rys. 2.2). Po otrzymaniu fragmentu fotografii z tablicą funkcja odnajduje kontury oraz sortuje je od prawej do lewej. Następnie są one wyodrębniane i wybierany jest pierwszy z nich. Zostaje sprawdzona jego długość i szerokość, jeśli znajdują się one w określonych przedziałach to jest on dodawany do tablicy zawierającej znalezione numery rejestracyjne i sprawdza czy zostały jakieś kontury do sprawdzenia. Jeśli nie znajdują się, to algorytm od razu przechodzi do sprawdzenia czy zostały jakieś kontury. W przypadku gdy są, przechodzi on do następnego i proces opisany powyżej się powtarza, a jeśli nie to sprawdzane jest czy tablica ze znalezionymi znakami jest pusta, jeżeli jest to wyświetlany zostaje komunikat o nie znalezieniu znaków, w przeciwnym przypadku zostaje ona zwrócona do kolejnego etapu działania aplikacji.



Rys. 2.3. Diagram BPMN prezentujący wyodrębnienie tablicy rejestracyjnej ze zdjęcia

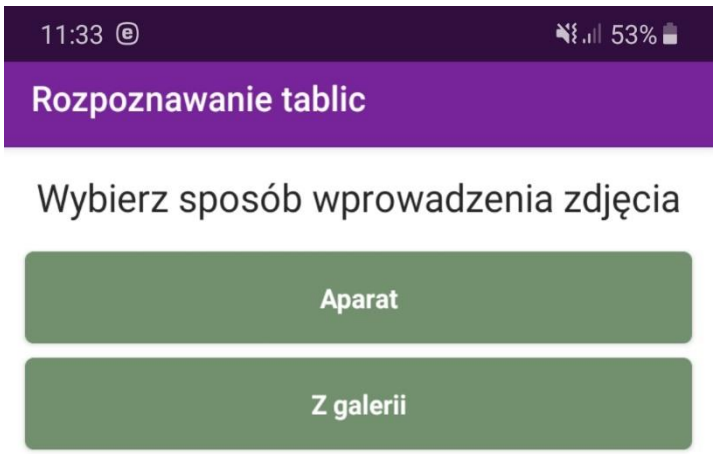


Rys. 2.2. Diagram BPMN prezentujący proces segmentacji

3. Opis działania aplikacji mobilnej

Aplikacja mobilna do rozpoznawania tablic rejestracyjnych ma za zadanie, w sposób prosty i szybki, rozpoznać sekwencję numerów rejestracyjnych ze zdjęcia pojazdu. Obraz jest pozyskiwany z wbudowanego aparatu fotograficznego lub spośród plików zapisanych w pamięci telefonu. Następnie program łącząc się z zewnętrznym serwerem przez Internet analizuje fotografię używając odpowiednich algorytmów i głębokiej sieci neuronowej.

Program jest łatwy w obsłudze i intuicyjny. Ekran główny prezentuje dwa duże, podpisane przyciski przekierowujące do odpowiednich funkcji (Rys.3.1).



Rys. 3.1. Ekran główny aplikacji

Za budowę interfejsu odpowiada kod napisany w języku HTML oraz arkusz stylów CSS (Listing.3.1 i Listing.3.2).

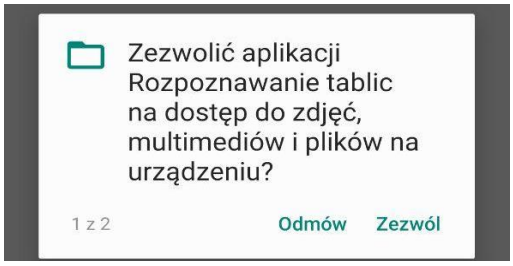
```
<ActionBar class="pasek-gorny" title="Rozpoznawanie tablic"></ActionBar>
<StackLayout *ngIf="wybranaSciezka == 'laduje' ? true : false">
  <ActivityIndicator busy="true"></ActivityIndicator>
</StackLayout>
<StackLayout>
  <Label class="etykieta" text="Wybierz sposób wprowadzenia zdjęcia"></Label>
  <Button class="przycisk" text="Aparat" (tap)="aparat()"></Button>
  <Button class="przycisk" text="Z galerii" (tap)="galeria()"></Button>
</StackLayout>
```

Listing 3.1. Kod opisujący elementy strony głównej aplikacji w języku HTML

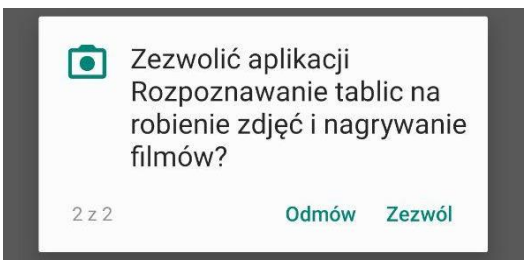
```css
.pasek-gorny {
  background-color: #772399;
  color: #fff;
}
.etykieta {
  font-size: 20;
  margin-bottom: 10;
  margin-top: 11;
  text-align: center;
}
.przycisk {
  background-color: #728f6e;
  border-radius: 5;
  color: #fff;
  font-weight: bold;
}
.zatwierdz {
  font-size: 21;
  margin-top: 25;
  padding-top: 5;
  padding-bottom: 5;
  text-align: center;
}
.tablica {
  font-size: 21;
  text-align: center;
  font-weight: bold;
  padding-top: 5;
}
```

Listing 3.2. Fragment arkusza CSS odpowiadającego za wygląd elementów aplikacji

Podczas pierwszego uruchomienia aplikacji, w zależności od wybranej funkcji, zostają wywołane zapytania dotyczące uprawnień do multimediów lub aparatu, w postaci okien dialogowych, takich jak na Rys.3.2 i Rys. 3.3.

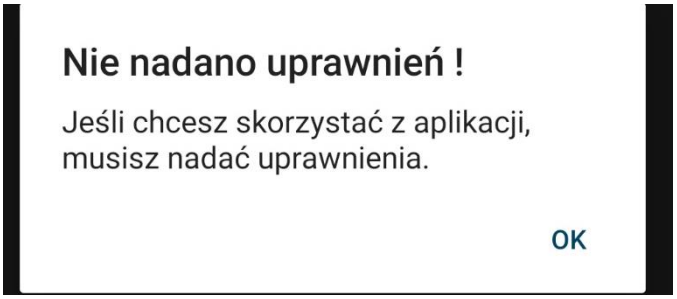


Rys. 3.2. Okno dialogowe dotyczące uprawnień do multimediów i plików



Rys. 3.3. Okno dialogowe dotyczące uprawnień do aparatu

Korzystanie z funkcji programu wymaga od użytkownika nadania uprawnień, inaczej nie jest możliwe prawidłowe działanie programu. Jeżeli zgoda na dostęp do multimediów, plików czy też aparatu nie zostanie udzielona, program powraca do ekranu głównego, jednocześnie wyświetlając okno dialogowe z komunikatem, takim jak na Rys.3.4.



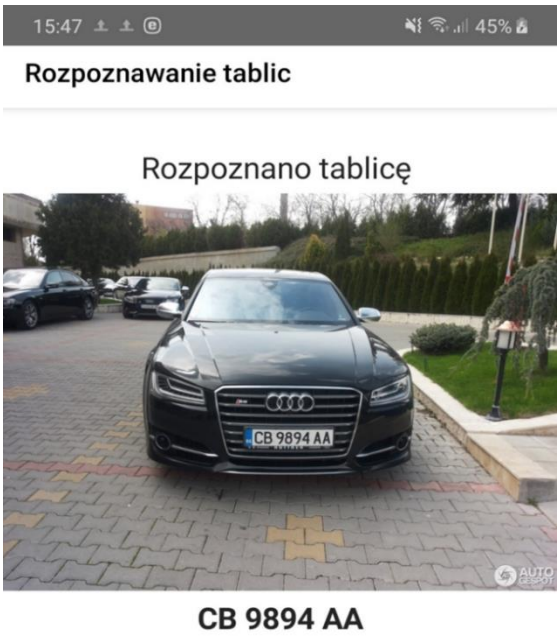
Rys. 3.4. Okno dialogowe dotyczące braku uprawnień

Wybranie opcji „Zdjęcie z aparatu" skutkuje przejściem do domyślnie ustawionej aplikacji obsługującej wbudowany aparat. Następuje wykonanie zdjęcia, po czym wywołany zostaje okno zatwierdzenia go.

Wybranie przycisku „Galeria" wywołuje domyślną w systemie „Galerię". Po wskazaniu jednego, odpowiedniego obrazu, tak jak w przypadku pierwszej opcji, zostaje wywołany ekran zatwierdzenia (Rys.3.5).

Rys. 3.5. Okno zatwierdzenia obrazu

Wybierając opcję „Zatwierdź” obraz zostaje wysłany do serwera, gdzie następnie zostaje poddany analizie poprzez sieć neuronową i odpowiednie algorytmy. Po wykonanej analizie zdjęcia, z serwera zostaje wysłany do aplikacji ciąg rozpoznanych znaków alfanumerycznych w postaci tekstu. Ostatni ekran programu zwraca wcześniej wskazany z galerii lub wykonany aparatem obraz oraz rezultat działań przeprowadzonych na serwerze (Rys.3.6 i Rys.3.7).



Rys. 3.7. Znaki znajdujące się na tablicy rejestracyjnej zostały rozpoznane

Rys. 3.6. Znaki na tablicy nie zostały rozpoznane

Podsumowanie

W niniejszym rozdziale zostały opisane funkcje oraz obsługa aplikacji mobilnej. We wstępie został przedstawiony ogólny jej zarys, a później sposób jej używania. Eksploatacja utworzonego programu nie jest trudna. Aplikacja ma czytelne komunikaty, duże przyciski, dzięki czemu nawet starsze osoby nie będą miały problemów z jej obsługą. Aby korzystać z tego oprogramowania wymagane jest nadanie uprawnień do odczytu i zapisu w pamięci urządzenia oraz do aparatu. Wszystko po to, by móc wykonać zdjęcie oraz odczytać je z galerii telefonu.

4. Wykorzystane technologie i narzędzia

Do utworzenia systemu rozpoznawania tablic rejestracyjnych zostały wykorzystane nowoczesne, sprawdzone i wydajne technologie. Dzięki narzędziom użytym do zaprogramowania wszystkich elementów projektu, kod jest czytelny i łatwy do korekcji.

4.1. Aplikacja mobilna

Do utworzenia aplikacji mobilnej zostały użyte jedne z nowoczesnych i popularnych technologii, dzięki którym w przyszłości będzie umożliwiony łatwy rozwój i jej utrzymanie. Biblioteka programistyczna, która została zastosowana w tym systemie pozwala na tworzenie aplikacji na główne platformy mobilne będące na rynku, czyli Android i iOS. Do zaprezentowania efektów działania systemu rozpoznawania znaków alfanumerycznych w tej pracy użyto systemu Android, jednakże aplikacja będzie w przyszłości rozwijana również pod platformę iOS.

4.1.1. NativeScript

Obecnie na rynku istnieje wiele technologii do tworzenia aplikacji multiplatformowych, jednakże duża część spośród nich, ze względu na krótki czas istnienia, jest w fazach rozwojowych, przez co nie obsługują wszystkich elementów systemów mobilnych. Jednymi z najpopularniejszych szkieletów programistycznych do tworzenia wieloplatformowych aplikacji mobilnych są: Xamarin, React Native oraz NativeScript .

Xamarin to szkielet programistyczny należący do firmy Microsoft. Wykorzystywane w nim technologie to język C# oraz biblioteka .Net. Używając Xamarin możliwe jest tworzenie aplikacji na systemach Windows oraz macOS. Biblioteka ta wykorzystuje jak najbardziej zbliżone nazwy elementów systemów iOS i Android, i jest w pełni z nimi kompatybilna [28].

React Native jest to biblioteka programistyczna języka Java Script. Wykorzystuje natywne elementy systemów mobilnych. Szkielet ten jest używany przy skomplikowanych modelach aplikacji, wykorzystujących duże ilości danych [43].

NativeScript to szkielet programistyczny, który wykorzystuje biblioteki Angular, Vue.js oraz języka JavaScript lub jego nadzbioru Typescript. Dzięki społeczności, która tworzy dodatki do tej biblioteki, łatwo można utworzyć programy korzystające z odpowiednich elementów systemów iOS i Android jednocześnie. W porównaniu do Xamarin, NativeScript sprawniej działa na urządzeniach mobilnych [39]. React Native

jest wydajniejszy w stosunku niniejszej biblioteki, jednakże najbardziej sprawdza się to w dużych aplikacjach o złożonej logice biznesowej. W niniejszej pracy zastosowano NativeScript, ze względu na łatwość w programowaniu i zastosowaniu oraz fakt, iż aplikacja nie jest skomplikowana.

4.1.2. Typescript

Jest to należący do firmy Microsoft nadzbiór języka JavaScript. W Typescript możliwe jest statyczne typowanie zmiennych oraz programowanie obiektowe oparte na klasach. Kod napisany w tym języku jest automatycznie kompilowany do JavaScript (Listing 4.1 i Listing 4.2).

```
1  function witacz(osoba:string){
2      return "Witaj, " + osoba;
3  }
4  let uzytkownik = "Jan Nowak";
5  document.body.textContent = witacz(uzytkownik);
```

Listing 4.1. Przykładowy kod w języku Typescript

```
1  function witacz(osoba) {
2      return "Witaj, " + osoba;
3  }
4  var uzytkownik = "Jan Nowak";
5  document.body.textContent = witacz(uzytkownik);
```

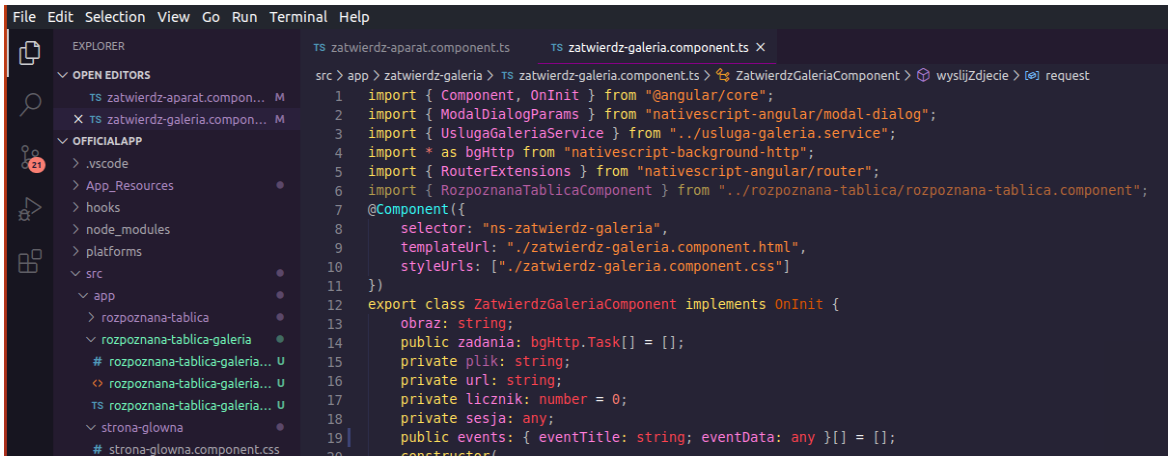
Listing 4.2. Powyższy kod przekompilowany do JavaScript

4.1.3. Angular

Biblioteka ta miała być drugą wersją słynnego szkieletu programistycznego AngularJS, lecz firma Google zmieniła decyzję i Angular rozwinięto jako osobny byt. Szkielet ten nie jest kompatybilny wstecz oraz nie pozwala na prostą aktualizację aplikacji z AngularJS do Angular. W tej platformie zastosowano takie usprawnienia jak m.in. renderowanie po stronie serwera, podział aplikacji na moduły i komponenty czy też lepsze wsparcie dla mobilnych wersji aplikacji internetowych. Dzięki zastosowaniu wątków roboczych (Web Workers) oraz renderowaniu przez serwer, aplikacje napisane w Angularze są znacznie szybsze i wydajniejsze. W Angularze ważnym elementem jest też separacja interfejsu użytkownika od logiki aplikacji [5], przez co każdą część można tworzyć osobno.

4.1.4. Visual Studio Code

Edytor ten został utworzony przez firmę Microsoft i dystrybuowany jest na licencji MIT. Z badań przeprowadzonych przez serwis StackOverflow w 2019 roku wynika, że Visual Studio Code jest najpopularniejszym edytorem kodu [30]. W programie tym możliwe jest pisanie w wielu językach programowania, zarządzanie wersjami kodu oraz jego debugowanie. Visual Studio Code posiada duże repozytorium dodatków, dzięki czemu program zyskuje wiele możliwości, takich jak integracja z nowymi językami programowania czy też używanie domyślnych skrótów klawiaturowych z innych edytorów. Interfejs programu jest prosty (Rys.4.1) i podobny do większości innych dostępnych na rynku.



Rys. 4.1. Okno programu Visual Studio Code

4.2. Serwer

Część serwerowa systemu została utworzona bazując na powszechnych i sprawdzonych technologiach. Dzięki temu wszelakie operacje będą wykonywane w sposób szybki i stabilny.

4.2.1. Apache

Serwer HTTP, który jest dostępny na wiele systemów operacyjnych. Projekt ten jest udostępniany na licencji Apache Licence i jest on w pełni otwarto źródłowy. Serwer jest skalowany, bezpieczny i wielowątkowy.

4.2.2. Język PHP

Jest to skryptowy język służący do tworzenia stron WWW oraz przetwarzania danych. Za wykonywanie instrukcji w PHP odpowiedzialny jest interpreter, który znajduje się po stronie serwera. W pracy język ten zastosowano w skrypcie obsługującym przekazywanie obrazu na serwer [12], wykonującym programy analizujące go w języku

Python oraz wysyłającym odpowiedź do użytkownika w postaci łańcucha znaków alfanumerycznych.

4.2.3. Serwer Nginx

Nginx jest wydajnym serwerem HTTP oraz serwerem pośredniczącym (proxy) rozwijanym przez firmę Nginx, Inc. na licencji BSD. Nginx pozwala na utworzenie komputera pośredniczącego, dzięki któremu klient może pobierać dane z wielu serwerów. Zastosowanie takiego rozwiązania pozwala też na zabezpieczenie głównego urządzenia obliczeniowego ukrywając jego adres IP.

4.3. Lokalizacja tablicy rejestracyjnej i rozpoznawanie znaków

4.3.1. Język Python

Jest to język programowania wysokiego poziomu, który posiada rozbudowany pakiet bibliotek standardowych. Jego ideą przewodnią jest czytelność i jasność kodu. Składnia charakteryzuje się zwięzłością i przejrzystością.

Został on opracowany na początku lat dziewięćdziesiątych dwudziestego wieku przez holenderskiego programistę Guido van Rossum'a w Centrum Matematyki i Informatyki w Amsterdamie (CWI). Jest on następcą języka programowania ABC. Wersja 1.2 języka Python była ostatnią wydaną przez CWI, od tego momentu był wydawany przez kilka różnych organizacji aż do wersji 2.1, od której jest on własnością Python Software Foundation, która jest organizacją niedochodową.

Python posiada w pełni dynamiczny system typów oraz automatyczne zarządzanie pamięcią, w czym jest podobny do języków takich jak Perl czy Ruby. Tak jak inne języki dynamiczne jest często używany jako język skryptowy. Jego standardową implementacją jest CPython napisany w C, ale istnieją też inne, np.: Jython - w Javie, CLPython - Common Lisp, IronPython - na platformę .NET i PyPy - w Pythonie. Możliwe jest w nim:

- programowanie obiektowe - jest to paradygmat programowania, w nim programy definiowane są za pomocą obiektów czyli elementów łączących dane (nazywane polami) i procedury (nazywane metodami),
- programowanie strukturalne - paradygmat programowania, który bazuje na podziale kodu źródłowego programu na procedury i bloki ułożone hierarchicznie z wykorzystaniem struktur kontrolnych mających postać instrukcji wyboru oraz pętli,

- programowanie funkcyjne - filozofia i metodyka programowania, która jest odmianą programowania deklaratywnego, gdzie funkcje należą do wartości podstawowych, a zwraca się szczególną uwagę na wartościowanie (w większości przypadków rekurencyjnych) funkcji, nie na wykonywanie instrukcji.

Python jest aktualnie jednym z najpopularniejszych języków programowania na świecie [3]. Dziedziny w których króluje to data science oraz uczenie maszynowe. W pracy tej został on wybrany z następujących powodów:

- łatwość kodowania – dzięki temu można się skupić na otrzymanych wynikach, ich analizie oraz poprawie,
- duża dostępność bibliotek przeznaczonych do obróbki zdjęć oraz uczenia maszynowego,
- darmowość – w porównaniu do MATLAB, który również specjalizuje się m.in.: w analizie danych i przetwarzaniu obrazów.

Innymi językami najczęściej stosowanymi w uczeniu maszynowym i obróbce zdjęć są:

- MATLAB,
- Java,
- C++.

Pierwszy z nich, jak już zostało wcześniej wspomniane, jest płatny, co jest ogromną przeszkodzą w tworzeniu rozwiązań darmowych. Drugi oraz trzeci są trudniejsze w kodowaniu, przez co większość czasu zostaje poświęcona na opracowanie kodu zamiast na opracowywanie i poprawianie otrzymanych wyników. Dodatkowo język C++ nie posiada dużej liczby bibliotek uczenia maszynowego, które są często potrzebne przy przetwarzaniu obrazów.

4.3.2. Jupyter Notebook

Jupyter to projekt open source utworzony w 2014 roku przez Fernando Péreza jako produkt poboczny IPythona. Jest on jest interfejsem, który tworzy notatniki w swoim formacie .ipynb, jednak kod utworzony w nim można ściągnąć jako pliki w różnych formatach, np.: HTML, plik Python, plik tekstowy, plik LaTeX. Można w nim także prowadzić obliczenia jak również wizualizować wyniki. Jądrem obliczeniowym tego projektu może być między innymi jeden z poniższych języków programowania:

- Python,
- R,
- Julia,
- JavaScript,
- Java,
- C#,
- Ruby,
- Kotlin,
- Haskell,
- Perl,
- Matlab,
- SQL.

Jąder tych jest dostępnych w sumie ponad 100, a wszystkie można sprawdzić podadresem [29].

Jupyter Notebook pracuje w układzie serwer - klient, gdzie serwerem jest wybrane jądro, a klientem przeglądarka www. Po jego uruchomieniu otworzona zostaje przeglądarka i następuje automatyczne przekierowanie na adres http://localhost:8888/tree.

4.3.3. Biblioteka OpenCV

OpenCV to biblioteka open source zawierająca funkcje stosowane podczas obróbki obrazu oraz jego rozpoznawania, co jest wykorzystywane w uczeniu maszynowym, np.: podczas wykrywania i rozpoznawania twarzy lub tablic rejestracyjnych. Projekt ten został oficjalnie zapoczątkowany w roku 1999 jako inicjatywa Intel Research. Biblioteka ta została napisana w języku C, jednak posiada ona nakładki, które umożliwiają korzystanie z niej między innymi w następujących językach:

- C++,
- Python,
- Java,
- MATLAB.

Wspiera też następujące struktury uczenia maszynowego:

- TensorFlow,
- Torch/PyTorch,
- Caffe.

OpenCV jest biblioteką wieloplatformową, która działa na następujących systemach operacyjnych:

- Windows,
- Linux,
- macOS,
- FreeBSD,
- NetBSD,
- OpenBSD.

Jest ona również dostępna na systemach mobilnych Android i iOS.

Biblioteka OpenCV posiada różne właściwości, między innymi:

- przetwarzanie danych obrazowych (konwersja, kopiowanie),
- podstawowe przetwarzanie obrazu (wykrywanie krawędzi, filtrowanie, zmiana kolorów, histogramy, interpolacja),
- analiza strukturalna (przetwarzanie konturów, dopasowanie wzorców, aproksymacja wielokątami, dopasowanie linii),
- rozpoznawanie obrazów,
- analiza ruchu,
- operacje na macierzach oraz wektorach,
- procedury algebry liniowej,
- ładowanie obrazów oraz wideo.

4.3.4. Biblioteka NumPy

NumPy jest biblioteką open source Pythona, która jest jednym z podstawowych modułów stosowanych do obliczeń naukowych. Dostarcza ona pomoc do przetwarzania dużych, wielowymiarowych tablic i macierzy oraz zapewnia użytkownikowi szereg funkcji matematycznych do stosowania na tych strukturach. Są one użyteczne między innymi w takich zagadnieniach jak:

- algebra liniowa,
- generowanie liczb losowych,
- transformacje Fouriera.

Została ona stworzona w roku 2005 przez Travis'a Oliphant'a poprzez włączenie funkcji z biblioteki Numarray z rozległymi modyfikacjami. NumPy została zaimplementowana w C i Fortranie dlatego jej wydajność jest bardzo dobra.

4.3.5. Biblioteka Tensorflow

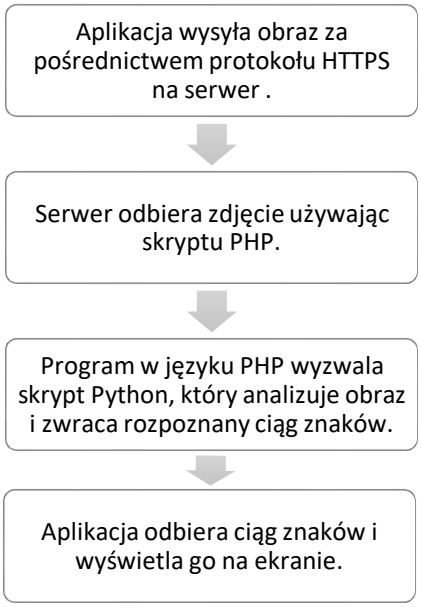
TensorFlow jest jedną z głównych otwartoźródłowych platform do uczenia maszynowego, tak jak między innymi PyTorch czy Theano. Stosowany jest w głównej mierze w strukturach głębokich sieci neuronowych.Oferuje on bardzo rozbudowany, wszechstronny i elastyczny w dopasowanu do własnych potrzeb zestaw narzędzi, będących prostymi do zaprojektowania i uruchomienia różnych architektur sieci neuronowych. Zawiera w sobie wysokopoziomowe API, umożliwiające pracę niezależnie od systemu operacyjnego czy dostępnego sprzętu - jest bardzo dobrze skalowalny. Tensorflow jest w stanie przeprowadzać obliczenia na rozmaite sposoby, wykorzystując do działania: procesory (CPU), karty graficzne (GPU), a także specjalny rodzaj mikroprocesorów przeznaczonych i zoptymalizowanych do obliczeń powiązanych z macierzami (TPU). Tensorflow może współpracować z różnymi językami programowania, takimi jak: Python, C/C++, Java, JavaScript, Go. Posiada on bardzo silną społeczność skupioną wokół niego, co skutkuje łatwością i skutecznością przekazywania wiedzy.

5. Serwer obliczeniowy

Do odbioru obrazu, jego analizy i przesłaniu rezultatu działania algorytmów i sieci neuronowej wykorzystano komputer wyposażony w wydajną kartę graficzną oraz zainstalowany system operacyjny Ubuntu Server 18.04.4 LTS. To głównie za pomocą GPU(Graphics Processor Unit – procesor graficzny) zdjęcie jest analizowane, a znaki znajdujące się na nim – rozpoznawane. Ubuntu Server jest lekką i stabilną dystrybucją, pozwalającą na sprawną konfigurację i zarządzanie komputerem.

5.1. Opis działania

W pracy został użyty serwer Apache połączony z odwróconym proxy Nginx. Połączenie aplikacji z zewnętrznym komputerem obliczeniowym odbywa się za pomocą protokołu HTTPS, ponieważ system Android od wersji 9 nie obsługuje nieszyfrowanej łączności HTTP. Poza tym, użyty protokół zapewnia bezpieczne i szyfrowane połączenie [50], dzięki czemu uniemożliwia to przechwycenie zdjęcia oraz pozostałych danych przez osoby postronne. Aplikacja poprzez sieć Internet wysyła obraz na serwer, który odbiera i zapisuje go lokalnie wykonując skrypt PHP. Później następuje uruchomienie programu analizującego zdjęcie oraz zwracającego znaki z tablicy rejestracyjnej. Aplikacja mobilna w sposób asynchroniczny oczekuje na odpowiedź z serwera i jeśli obraz zostanie przetworzony, następuje przekierowanie do ekranu z podglądem analizowanego zdjęcia oraz ciągiem znaków odczytanych z tablicy lub komunikatem o błędzie . Schemat komunikacji został przedstawiony na Rys.5.1.

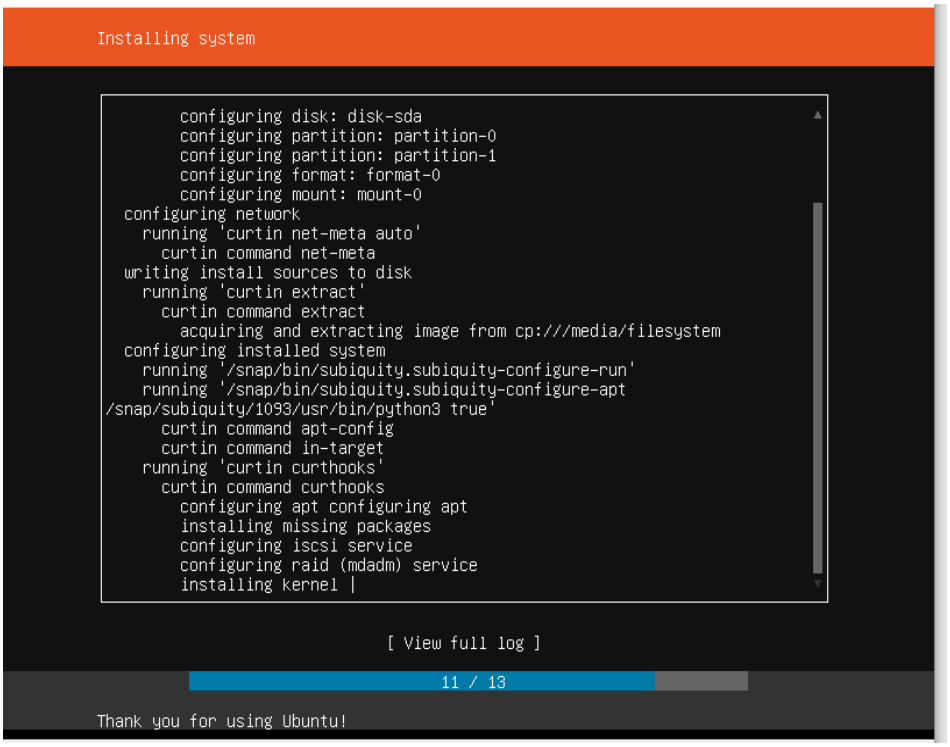


Rys. 5.1. Schemat komunikacji aplikacji z serwerem

5.2. Ustawienia serwera

Do prawidłowego i bezpiecznego połączenia między aplikacją a komputerem obliczeniowym wymagana jest odpowiednia konfiguracja. Pierwszym etapem jest instalacja systemu Ubuntu Serwer, która przebiega następująco:

- wybór języka instalacji oraz używanego później systemu,
- konfiguracja połączenia sieciowego,
- przydzielanie miejsca na dysku twardym,
- tworzenie kont administratora i użytkownika
- wybór dodatkowych modułów,
- oczekiwanie na instalację (Rys.5.2).

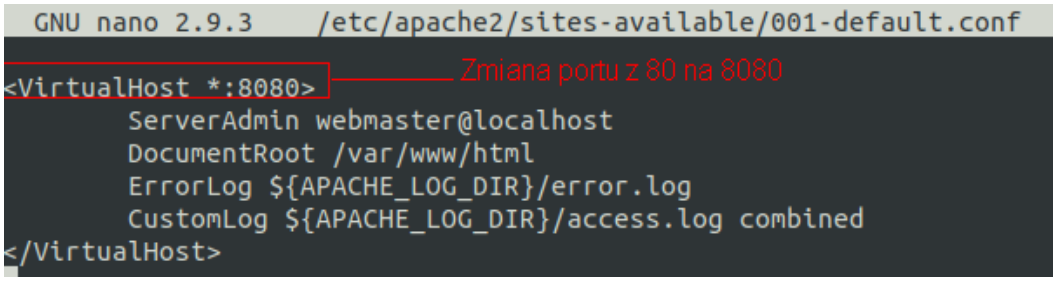


Rys. 5.2. Instalacja serwera

Kolejnym krokiem jest aktualizacja pakietów systemowych. Po wykonaniu podstawowych czynności związanych z pierwszym uruchomieniem systemu operacyjnego należy skonfigurować potrzebne moduły serwera [27].

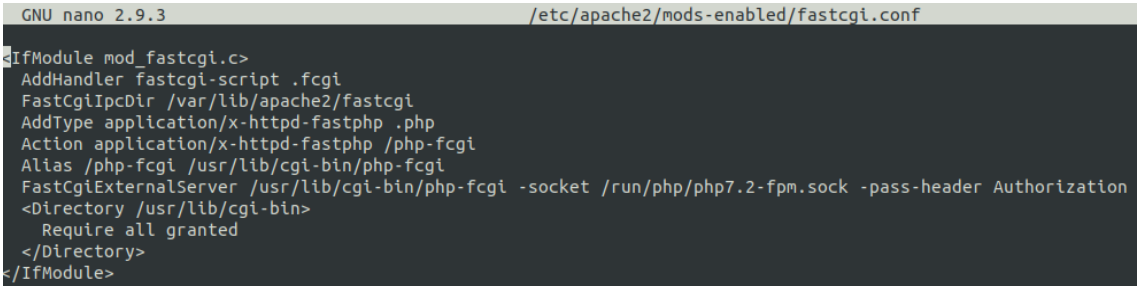
- Krok 1. Instalacja pakietów Apache i PHP-FPM – są to podstawowe pakiety serwera Apache.

- Krok 2. Ustawienia modułów Apache i PHP-FPM – w tym etapie następuje wyłączenie domyślnego wirtualnego hosta i utworzenie nowego. W pliku konfiguracyjnym przy zostaje zmieniony port nasłuchujący (Rys.5.3)



Rys. 5.3.Zmiana portu w pliku 001-default.conf

- Krok 3. Po zmianie portu, należy przeładować moduł Apache. Następnie wymagane jest utworzenie pliku *fastcgi.conf,* który zawiera ustawienia FastCGI, które przyspiesza wykonywanie zapytań wysyłanych do serwera (Rys.5.4).



Rys. 5.4. Plik *fastcgi.conf*

- Krok 4. W tym etapie jest tworzony katalog wirtualnego hosta dla domeny, którą się przypisuje do serwera. Oprócz tego tworzy się plik konfiguracyjny, gdzie podaje się adres domeny (Rys.5.5) .

```
        /etc/apache2/sites-available/zenithproject.waw.pl.conf

<VirtualHost *:8080>
        ServerName zenithproject.waw.pl
        ServerAlias www.zenithproject.waw.pl
        DocumentRoot /var/www/zenithproject.waw.pl
        <Directory /var/www/zenithproject.waw.pl>
                AllowOverride All
        </Directory>
</VirtualHost>
```

Rys. 5.5.Plik konfiguracyjny dla domeny

- Krok 5. Po konfiguracji domeny, poleceniem *sudo a2ensite nazwadomeny* uruchamiamy ją. Następnie restartujemy serwer Apache.
- Krok 6. Kolejnym elementem komputera obliczeniowego jest Nginx. Moduł ten jest instalowany w taki sam sposób jak Apache. Następnie jest usuwana jego domyślna konfiguracja.
- Krok 7. W tym etapie zostaje utworzony wirtualny host Nginx,, jego katalog i plik z ustawieniami (Rys.5.6). Oprócz tego za pomocą usługi Certbot i urzędu certyfikacji Let’s Enrypt połączenie z serwerem zostało zaszyfrowane.

```
  GNU nano 2.9.3                                  /etc/nginx/sites-available/apache

server {
    listen 80;
    server_name zenithproject.waw.pl www.zenithproject.waw.pl;
    root /var/www/zenithproject.waw.pl;
    index index.php index.htm index.html;

    location / {
        try_files $uri $uri/ /index.php;
    }

    location ~ \.php$ {
        proxy_pass http://80.211.240.44:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location ~ /\.ht {
        deny all;
    }

    listen 443 ssl;
    ssl_certificate /etc/letsencrypt/live/zenithproject.waw.pl/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/zenithproject.waw.pl/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
```

Rys. 5.6. Plik ustawień wirtualnego hosta

- Krok 8. Po skonfigurowaniu wirtualnego hosta Nginx zostaje on zrestartowany. Następnie instalowany jest pakiet mod_rpaf, który pozwala na ustawienie serwera pośredniczącego (Rys.5.7).

```
 GNU nano 2.9.3                                  /etc/apache2/mods-available/rpaf.conf

  <IfModule mod_rpaf.c>
      RPAF_Enable             On
      RPAF_Header             X-Real-Ip
      RPAF_ProxyIPs           80.211.240.44
      RPAF_SetHostName        On
      RPAF_SetHTTPS           On
      RPAF_SetPort            On
  </IfModule>
```

Rys. 5.7. Plik ustawień serwera pośredniczącego

5.3. Odbiór i analiza obrazu

Serwer obliczeniowy do odbioru zdjęć wykorzystuje język PHP, przez jego moduł znajdujący się w systemie musi być odpowiednio skonfigurowany. W pliku php.ini została więc włączona możliwość przejęcia plików z zewnątrz, a także maksymalna ich wielkość (Rys.5.8).

```
; Whether to allow HTTP file uploads.
; http://php.net/file-uploads
file_uploads = On

; Temporary directory for HTTP uploaded files (will use system default if not
; specified).
; http://php.net/upload-tmp-dir
;upload_tmp_dir =

; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
upload_max_filesize = 10M

; Maximum number of files that can be uploaded via a single request
max_file_uploads = 20
```

Rys. 5.8.Fragment pliku php.ini dotyczący odbioru plików

Zdjęcia odbierane z aplikacji mobilnej są przechowywane w katalogu *uploads*. Do wykonania operacji zapisu pliku jak i późniejszej jego analizy został utworzony skrypt w języku PHP (Rys.5.9).

```php
<?php
$uploaddir = 'uploads/';
if($_FILES['File-Name']['error'] == UPLOAD_ERR_OK){
    $new_name = $uploaddir.$_FILES['File-Name']['name'];
    $temp_name = $_FILES['File-Name']['tmp_name'];
    if(move_uploaded_file($temp_name, $new_name)){
        echo "Plik został załadowany.";
    }
    else{
        echo "Nie udało się załadować pliku.";
    }
}
else{
    echo "Wystąpił błąd: ";
    switch($_FILES['File-Name']['error']){
        case UPLOAD_ERR_INI_SIZE :
        case UPLOAD_ERR_FORM_SIZE :
            echo "Przekroczony maksymalny rozmiar pliku!";
        break;
        case UPLOAD_ERR_PARTIAL :
            echo "Odebrano tylko część pliku!";
        break;
        case UPLOAD_ERR_NO_FILE :
            echo "Plik nie został pobrany!";
        break;
        case UPLOAD_ERR_NO_TMP_DIR:
            echo "Brak dostępu do katalogu tymczasowego.";
        break;
        case UPLOAD_ERR_CANT_WRITE:
            echo "Nie udało się zapisać pliku na dysku serwera.";
        break;
        case UPLOAD_ERR_EXTENSION:
            echo "Ładowanie pliku przerwane przez rozszerzenie PHP.";
        break;
        default :
            echo "Nieznany typ błędu!";
}
}
$poleceniePython = escapeshellcmd('python3 rozpoznawanie_tablic.py');
$wyjscie = shell_exec($poleceniePython);
?>
```

Rys. 5.9. Skrypt wykonujący odbiór i analizę obrazu

Instrukcja *shell_exec()* pozwala na wykonywanie skryptów w innych językach niż PHP [42]. Rozpoznane znaki lub informacja o niepowodzeniu analizy obrazu są przekazywane ze skryptu *rozpoznawanie_tablic.py* i przesyłane do aplikacji.

Podsumowanie

W tym rozdziale zostały opisane konfiguracja i działanie serwera obliczeniowego wykorzystywanego w niniejszej pracy do odbioru i analizy obrazów. Na początku przedstawiony został sposób komunikacji aplikacji z serwerem, która jest bezpieczna i szyfrowana. Następnie opisano instalację systemu Ubuntu Server oraz konfigurację modułów serwera. Na końcu zaprezentowano sposób, w jaki komputer obliczeniowy odbiera pliki i wykonuje na nich odpowiednie działania.

6. Rozpoznawanie tablic rejestracyjnych

6.1. Wstęp teoretyczny

Automatyczne Rozpoznawanie Tablic Rejestracyjnych (ARTR), znane jest też pod angielską nazwą Automatic Number Plate Recognition (ANPR) lub Licence Plate Recognition (LPR). Jest to technika pozwalająca na rozpoznawanie tablic znajdujących się na zdjęciach. Analiza obrazu polega na jej zlokalizowaniu, a następnie podjęciu odpowiednich czynności, które doprowadzą do odczytania numerów znajdujących się na niej.

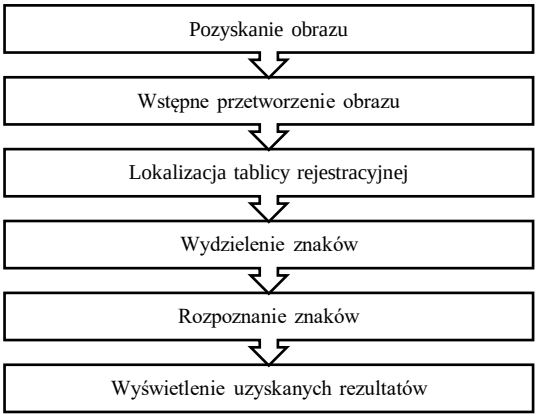
Według informacji zawartych w publikacji [18] istnieją dwa sposoby rozpoznawania tablic: rozpoznawanie statyczne (dotyczy samochodów nie poruszających się) oraz rozpoznawanie dynamiczne (auta poruszające się). Pierwszy z nich wymaga aby pojazd znajdował się w określonej pozycji do kamery i nie poruszał się. Drugi sposób nie ma takich ograniczeń, pozwala on na rozpoznawanie w czasie rzeczywistym. W tej pracy do badań został wybrany sposób pierwszy, jednak opracowany algorytm nie wymaga ustawienia pojazdu w konkretnej pozycji względem aparatu pod warunkiem, że tablica znajduje się w kadrze.

6.2. Etapy rozpoznawania

Istnieją trzy podstawowe etapy rozpoznawania tablic rejestracyjnych:

- lokalizacja tablicy rejestracyjnej,
- wydzielenie znaków,
- rozpoznanie znaków.

Biorąc pod uwagę cały system rozpoznawania podział ten można rozwinąć na etapy przedstawione na Rys.6.1.



Rys. 6.1. . Etapy działania system rozpoznawania numerów tablicy rejestracyjnej

6.3. Lokalizacja tablicy rejestracyjnej przy użyciu metody wykrywania krawędzi

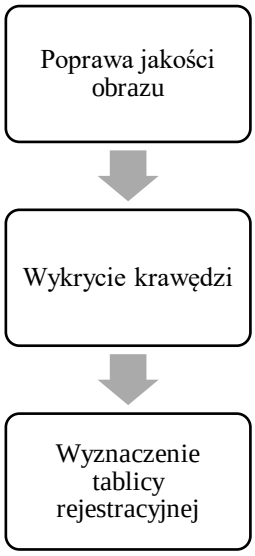
Zlokalizowanie tablicy rejestracyjnej na obrazie jest jedną z najtrudniejszych części. Z uwagi na to, że dalsze operacje w aplikacji są prowadzone na wyodrębnionej tablicy jest to również jeden z najważniejszych etapów działania całego algorytmu, ponieważ prawidłowe wyznaczenie obszaru w którym ona się znajduje decyduje o poprawnym działaniu całej aplikacji.

W dostępnych źródłach [17][19] można znaleźć wiele różnych metod lokalizacji tablic rejestracyjnych. Wśród nich można wyróżnić cztery podstawowe, najczęściej stosowane:

1) Wykrywanie krawędzi - identyfikuje punkty na obrazie gdzie jasność zmienia się gwałtownie - punkty te są grupowane w zbiór linii krzywych nazywanych krawędziami.
2) Wykrywanie obszaru za pomocą wzorców jasności - w tej metodzie system jest trenowany na dostarczonych zdjęciach z oznaczonymi tablicami, które mają różne stopnie jasności i na ich podstawie jest trenowany model. Gdy zostanie dostarczone zdjęcie z niezaznaczoną tablicą, system używa wytrenowanego modelu do wykrycia tablicy.
3) Wykrywanie z wykorzystaniem informacji koloru - ta metoda polega na nauczeniu systemu wykrywania regionów z danym kolorem, który jest kolorem tablicy rejestracyjnej, którą chcemy zlokalizować.
4) Radialna funkcja bazowa (RBF) - jest to funkcja rzeczywista, której wartość zależy wyłącznie od odległości od określonego punktu.

Ze względu na to, że w większości krajów tablice rejestracyjne składają się tylko z dwóch kontrastowych kolorów (w większości przypadków są to biały i czarny) naczęściej stosowaną metodą jest metoda wykrywania krawędzi. W krajach, których język zawiera litery mające dużo krzywych, np.: arabski, częściej wybierana jest metoda RBF [17]. Do opracowania tej metody została obrana metoda wykrywania krawędzi z funkcją Canny, ponieważ według porównania dokonanego w pracy [15] ma ona najwyższy stopień wykrywania tablic.

Poniżej przedstawiono i opisano kroki działania algorytmu opracowanego na potrzeby aplikacji (rys. 6.2).

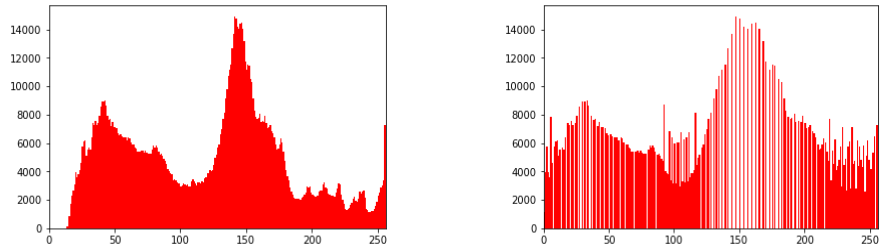


Rys. 6.2.Kroki działania opracowanego modelu

Krok 1. Poprawa jakości obrazu

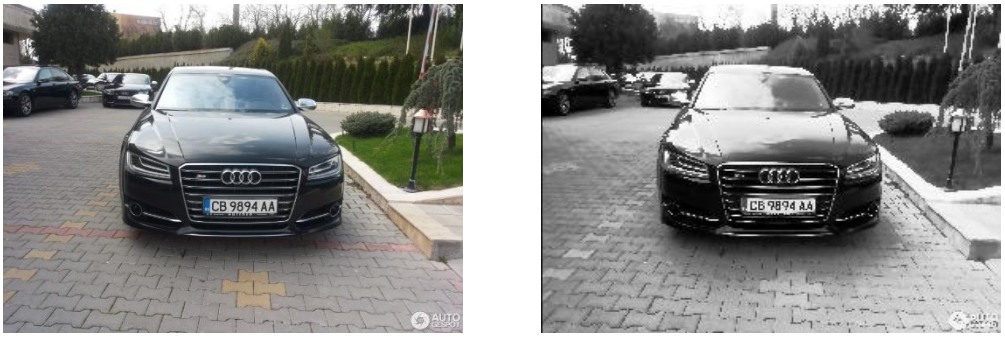
Na początku następuję zmiana koloru zdjęcia na odcienie szarości oraz jest dokonywana redukcja szumów za pomocą funkcji bilateralFilter, która znajduje się w bibliotece OpenCV (OCV). Jest ona bardzo skuteczna w usuwaniu szumów na zdjęciu z jednoczesnym zachowaniem ostrości krawędzi. Według informacji zamieszczonych w dokumentacji [25] metoda ta jest wolniejsza od innych filtrów, np.: filtru gaussowskiego, ale niestety one rozmazują również kontury co jest niechcianym efektem w danym przypadku. Kolejnym krokiem jest zwiększenie kontrastu obrazu, które jest dokonywane za pomocą funkcji equalizeHist, która pochodzi z biblioteki wymienionej wcześniej. Algorytm ten normalizuje jasność i zwiększa kontrast na zdjęciu.

Dla przedstawienia działania metody equalizeHist na rysunku 6.3a przedstawiono histogram obrazu po zastosowaniu funkcji bilateralFilter a na rysunku 6.3b histogram po użyciu metody normalizacji jasności i zwiększenia kontrastu.



a b

Rys. 6.3. Histogram zdjęcia (a) po zastosowaniu funkcji bilateralFilter, (b) po zastosowaniu funkcji equalizeHist



a b

Rys. 6.4. Zdjęcie (a) oryginalne, (b) po poprawieniu jakości

Zmiany wprowadzone przez funkcję z której skorzystano na rysunku 6.3b nie są łatwo dostrzegane na obrazie dlatego w tej pracy jej działanie postanowiono zaprezentować za pomocą histogramów.

Rys. 6.4a przedstawia oryginalne zdjęcie, natomiast Rys.6.4b przedstawia zdjęcie po poprawie jakości.

Krok 2. Wykrycie krawędzi

W tym etapie algorytm wyszukuje krawędzie znajdujące się na zdjęciu. W tym celu użyta została funkcja Canny z biblioteki OpenCV. Algorytm ten stosuje progowanie wyznaczające punkty, które są grupowane w linie krzywe tworzące krawędzie. Wszystkie fazy działania tego algorytmu zostały opisane na następującej stronie [41]. Canny w pierwszym etapie przetwarzania obrazu stosuje redukcje szumów za pomocą filtru gaussowskiego o wymiarze 5x5. Następnie znajduje intensywność gradientu danego zdjęcia. W tym celu korzysta z funkcji Sobel w kierunku pionowym (G_y) oraz poziomym (G_x) aby uzyskać pochodną pierwszego stopnia w obu orientacjach. Z powstałych obrazów G_x i G_y znajduje gradient krawędzi i kierunek dla każdego piksela. Przy obliczeniach korzysta ona z wzorów przedstawionych poniżej (wzór 6.1 i wzór 6.2):

$$Edge\_Gradient(G) = \sqrt{G_x^2 + G_y^2} \tag{6.1}$$

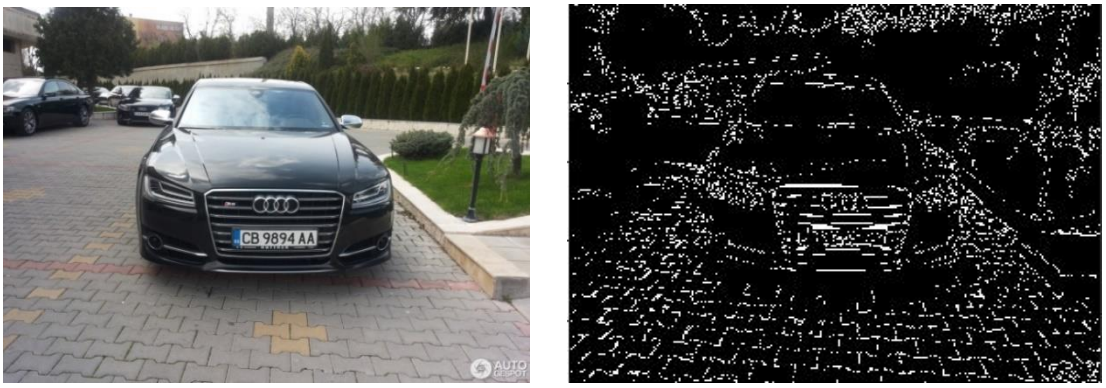
$$Angle(\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{6.2}$$

Wzór 6.1 przedstawia sposób w jaki jest obliczany gradient krawędzi a wzór 6.2 prezentuje sposób obliczania kierunku danego piksela.

Orientacja przejścia tonalnego odznacza się prostopadłością do krawędzi. Jest ona zaokrąglona do jednego z czterech krawędzi reprezentujących pionowe, poziome i dwa przekątne kierunki.

W dalszej kolejności funkcja Canny wykonuje tłumienie niemaksymalne (z angielskiego Non-maximum Suppression (NMS)). Jej działanie polega na tym, że po uzyskaniu wielkości i kierunku gradientu zostaje utworzony pełny skan obrazu w celu usunięcia niechcianych pikseli, które mogą nie tworzyć krawędzi. W tym celu w każdej grupie pikseli każdy jeden jest sprawdzany czy jest maksimum lokalnym w swoim sąsiedztwie, które jest zorientowane w kierunku gradientu. W rezultacie tego działanie otrzymuje się obraz binarny z „cienkimi krawędziami".

Ostatnią akcją jaką wykonuje ta metoda jest progowanie z histerezą. Ten etap decyduje o tym, które z krawędzi są naprawdę krawędziami a które nie. Do tego funkcja Canny potrzebuje dwóch wartości progowania, minimalnej i maksymalnej. Każdy kontur, którego intensywność gradientu jest większa niż wartość górnej granicy progu jest na pewno krawędzią, natomiast te znajdujące się poniżej dolnego pułapu bez wątpienia nimi nie są, więc zostają odrzucone. Kontury znajdujące się pomiędzy tymi dwoma wartościami są zakwalifikowane jako krawędzie albo nie w zależności ich połączenia. Jeśli są powiązane z pikselami, które są na pewno zarysem to są uznawane za część danego konturu. W przypadku gdy nie są połączone z takimi pikselami są one odrzucane. W rezultacie tych wszystkich etapów funkcji Canny otrzymuje się obraz taki jak przedstawiony na Rys.6.5.



a b

Rys. 6.5. Zdjęcie (a) oryginalne, (b) po zastosowaniu funkcji Canny

Rysunek 6.5a przedstawia obrazek oryginalny, natomiast rysunek 6.5b przedstawia rysunek 6.4b po zastosowaniu funkcji Canny.

Krok 3. Wyznaczenie tablicy rejestracyjnej

W ostatnim kroku algorytm wyznacza miejsce, w którym znajduje się tablica rejestracyjna na zdjęciu. Na samym początku tej części dokonuje się znalezienie konturów za pomocą funkcji findContours pochodzącej z biblioteki OpenCV. Metoda ta odnajduje krawędzie z obrazu binarnego za pomocą algorytmu opracowanego i opisanego przez Satoshi Suzuki i Keiichi Abe, który został opublikowany w roku 1985 w następującym czasopiśmie [1]. Model opracowany przez nich określa relacje otoczenia pomiędzy granicami obrazu binarnego. W pracy wymienionej powyżej prezentuje również drugi algorytm, który jest zmodyfikowaną wersją pierwszego lecz stosuje się go jedynie do krawędzi najbardziej oddalonych od środka zdjęcia.

Następnie znalezione kontury są sortowane od największej od najmniejszej. W celu wykrycia tej, która jest prawdopodobnie tablicą, algorytm ustala krzywe wielokątne dla każdej z nich. W tym celu wykorzystywana jest funkcja approxPolyDP znajdująca się w bibliotece wymienionej powyżej. Metoda ta używa wzorca Ramer-Douglas-Peucker'a znanego również pod nazwą iteracyjnego algorytmu dopasowania punktu końcowego. Model ten jest wzorem do redukowania liczby punktów w krzywej, która jest stworzona przez przybliżenie szeregów punktów. Polega to na tym, że metoda tworzy linie między punktem pierwszym i ostatnim w zbiorze, z którego składa się dana łamana. Następnie sprawdza, który jest najbardziej oddalony od utworzonej prostej. Jeśli jest on bliżej niż dany dystans, który nazywa się epsilonem, usuwane są wszystkie punkty znajdujące się pomiędzy tym punktem a prostą. W przypadku gdy ten „punkt odstający" jest dalej niż epsilon, krzywa zostaje podzielona na dwie części: pierwsza od punktu początkowego do odstającego (włącznie z nim), a druga składa się z punktów, które zostały oraz końcowego punktu poprzedniej części. Funkcja ta jest wywoływania rekurencyjnie na powstałych łamanych i powstałe zawężone formy są łączone w jedną, która daje zredukowaną krzywą pierwotną. Powyższy opis tej metody wraz z interaktywnymi przykładami można znaleźć na następującej stronie [8].

Jeśli długość krzywej wielokątnej danej kontury równa się cztery zakłada się, że jest to kontur tablicy rejestracyjnej. W kolejnym kroku jest on obrysowywany na czerwono za pomocą funkcji drawContours z biblioteki OCV.

Rys. 6.6. Zdjęcie oryginalne z wyznaczoną tablicą

W tym rozdziale został przedstawiony i opisany sposób znajdowania tablicy rejestracyjnej na zdjęciu. Na samym początku przedstawiono etapy przetwarzania obrazu w celu uzyskania rozpoznanych znaków tablicy rejestracyjnej. Następnie przedstawiono cztery metody lokalizacji i opisano w kolejności wykonywania kroki, które są wykonywane w aplikacji powstałej w ramach tej pracy, w celu wyodrębnienia tablicy ze zdjęcia przekazanego przez użytkownika za pomocą procedury wykrywania krawędzi. Pierwszym etapem jest zmiana koloru zdjęcia na odcienie szarości i poprawa jakości obrazu z wykorzystaniem funkcji bilateralFilter (redukcja szumów) oraz equalizeHist (zwiększenie kontrastu), które znajdują się w bibliotece OpenCV. Następny krok wykrywa krawędzie na fotografii za pomocą modułu Canny będącego częścią OCV. Etap kolejny wyznacza tablice. Wykorzystując funkcję findContours odnajduje krawędzie na obrazie binarnym następnie sortuje je od największej do najmniejszej, używając modułu approxPolyDP ustala krzywe wielokątne dla każdej z nich, po czym sprawdza czy długość danej łamanej jest równa cztery, jeśli tak to uznaje ją za kontur tablicy rejestracyjnej i obrysowuje na czerwoną za pomocą funkcji drawContours, która tak jak i pozostałe moduły należy do biblioteki OpenCV.

Powyższe sposób został przetestowany na 22 zdjęciach. Algorytm wykrył tablicę rejestracyjną poprawnie na wszystkich zdjęciach. Czas wykrycia był krótszy niż jedna sekunda.

7. Segmentacja znaków

7.1. Wstęp teoretyczny

Po poprawnym zlokalizowaniu tablicy rejestracyjnej następuje kolejny etap, którym jest wydzielenie znaków znajdujących się na wyznaczonym obszarze. Istnieje wiele różnych metod segmentacji, wśród których można wyróżnić m.in.:

- segmentację obszarową,
- segmentację krawędziową,
- segmentację poprzez analizę skupień,
- segmentację poprzez maskę R-CNN.

W tej pracy wybrano procedurę segmentacji krawędziowej, ze względu na to, że jest ona jedną z najbardziej popularnych metod używanych w analizie obrazów jak również z uwagi na to, że krawędzie są podstawową cechą każdego zdjęcia [14].

7.2. Segmentacja znaków za pomocą metody krawędziowej

Wykrywanie konturów zostało już opisane w rozdziale 6.3 w kroku 2 i 3 przy omawianiu sposobu klasyfikacji tablicy. W przypadku wykorzystania tej metody do wyseparowania obszarów zawierających znaki numeru rejestracyjnego danego pojazdu należy najpierw wyciąć obszar, w którym znajduje się tablica. W algorytmie jej wyodrębnienie następuje za pomocą nałożenia maski składającej się z prawie samych zer, co daje efekt czarnego obrazu, brak ich jedynie w miejscu, w którym znajduje się obszar wykryty i zaznaczony w etapie wcześniejszym (Rys.7.1).

Rys. 7.1. Nałożenie maski na zdjęcie

Następnie z obrazu zostają wycięte pola, które składają się z samych zer i zostaje jedynie tablica (Rys. 7.2).

Rys. 7.2. Wycięta tablica

W kolejnym kroku następuje progowanie zdjęcia tablicy i odwrócenie kolorów, co w rezultacie daje obraz widoczny na Rys. 7.3.

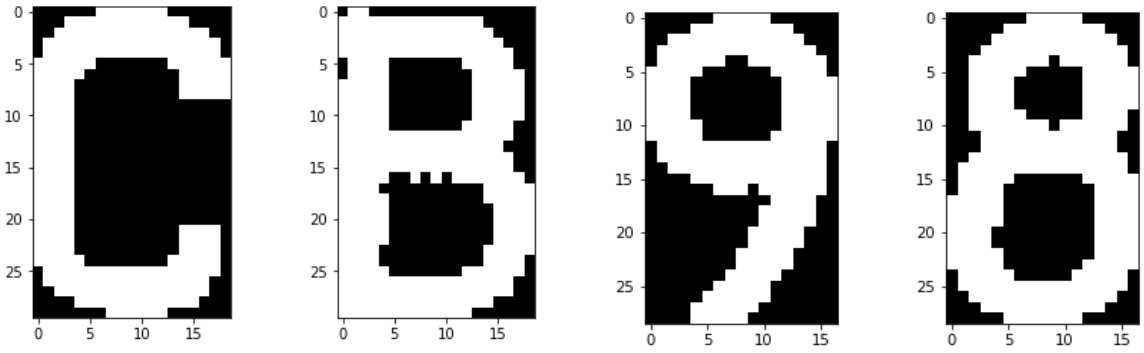
Rys. 7.3. Obraz tablicy po zastosowaniu progowania oraz odwrócenia kolorów

W kolejnym etapie wykrywane są kontury w podobny sposób jak w rozdziale 6.3 w kroku 3, jednak sortowanie nie jest przeprowadzane od największego do najmniejszego tylko od prawej do lewej. W następnym kroku znalezione kontury są wyodrębniane w podobny sposób co tablica rejestracyjna ze zdjęcia, po czym każda z nich zostaje sprawdzona czy posiada długość i szerokość znajdującą się w danym przedziale, jest to potrzebne aby wyeliminować znalezione kontury, które nie są znakami numeru rejestracyjnego, np.: kształty hologramów znajdujących się na niektórych tablicach.

Wymiary polskich tablicy rejestracyjnych, według norm określonych w [23], wynoszą 520 mm na 114 mm w przypadku tablic jednorzędowych, które są najczęściej spotykane w przypadku samochodów. Wysokość liter i cyfr znajdujących się na tablicy wynosi 80 mm (jest to w przybliżeniu 70% wysokości całej tablicy), a szerokość dla nich jest różna i wynosi kolejno dla liter – 54 mm (około 10,4% całości), cyfr – 43 mm (ok. 8,27% całości). W opracowanej funkcji przyjęto przedział ujednolicony dla obu typów znaków. Wynosi on dla długości od 45% do 90% wysokości całej tablicy, a dla szerokości od 1,1% do 50%. Wymiary te wyznaczono podczas testowania algorytmu na różnych zdjęciach. Jeżeli wartości minimalne były niższe wykrywane były odstępy między znakami lub inne elementy tablicy, jak np.: hologramy, natomiast w przypadku gdy były one wyższe to nie wykrywano nic. W przypadku zwiększenia górnych granic przedziału zwracano poza odnalezionymi

znakami całą tablicę lub jej fragment, kiedy zmniejszono te wartości niektóre cyfry lub litery nie były znajdywane.

Jeśli znaleziony kontur spełnia warunki wymienione powyżej to zostaje on dodany do tablicy, która zawiera wysegmentowane znaki i po wykryciu wszystkich kształtów jest przekazywana do dalszej części algorytmu, którą jest rozpoznawanie znaków alfanumerycznych.



a b c d

Rys. 7.4 (a-d) poszczególne rozpoznane znaki tablicy

Rysunki 7.4a – 7.4d przedstawiają cztery pierwsze wysegmentowane znaki tablicy rejestracyjnej.

Podsumowanie

W tym rozdziale został opisany sposób segmentacji znaków tablicy rejestracyjnej na zdjęciu. Na początku wymieniono cztery przykładowe metody wydzielenia liter i cyfr z obrazu. Następnie przedstawiono metodę wybraną w tej pracy i opisano w kolejności wykonywania czynności, które są realizowane w ramach tego algorytmu w celu identyfikacji znaków. W pierwszym kroku wycięty ze zdjęcia zostaje obszar w którym znajduje się tablica za pomocą nałożenia maski składającej się z prawie samych zer (daje to efekt czarnego obrazu), których brak jedynie w miejscu gdzie ona została wyznaczona. Następnie zostają wycięte wszystkie pola, które są czarne. Dzięki temu zostaje jedynie zdjęcie samej tablicy. Na pozyskanej fotografii zostaje przeprowadzone progowanie oraz odwrócenie kolorów. W kolejnym kroku następuje wykrywanie krawędzi oraz ich wyodrębnianie w podobny sposób co w rozdziale poprzednim, jednak zamiast sortowania od największego do najmniejszego konturu, przeprowadzane jest ono od prawej do lewej. Następnie długość

i szerokość tych konturów zostaje sprawdzona pod kątem przynależności do wyznaczonych przedziałów w celu wyeliminowania znaków lub kształtów, które nie są częścią numeru rejestracyjnego. Jeśli dany kształt spełnia te wymagania zostaje dodany do tablicy zawierającej wysegmentowane kontury. Gdy wszystkie znaki zostają wykryte lista ta zostaje przekazana do kolejnego etapu algorytmu, którym jest rozpoznanie znaków alfanumerycznych.

Powyższe zastosowanie zostało przetestowane na 22 zdjęciach. W każdym przypadku znaki zostały wydzielone poprawnie. Czas wykonania tego zadania był krótszy niż jedna sekunda.

8. Neurony i sposób ich działania

W celu zrozumienia zasad działania sieci neuronowych, niezbędne jest najpierw zrozumienie budowy i zasad działania podstawowego neuronu.

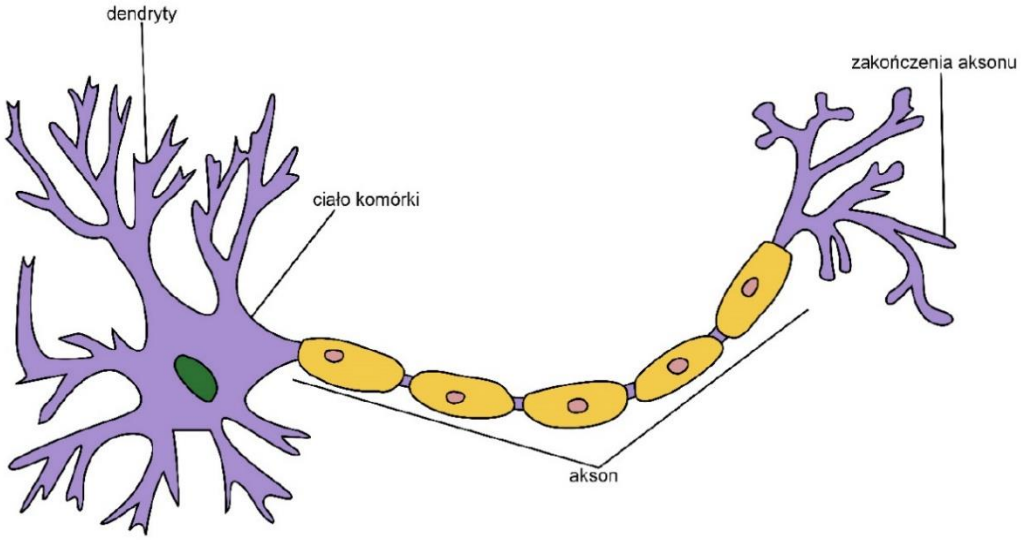
8.1. Neuron biologiczny

Neurony to podstawowe jednostki strukturalne i funkcjonalne układu nerwowego człowieka zdolne do szybkiego przekazywania impulsów elektrycznych. [2]

Mózg i system nerwowy zbudowane są z sieci około 100 miliardów (10^{11}) neuronów, z czego każdy z nich składa się z aksonu i dendrytów. Dendryty są to cienkie wypustki odchodzące od ciała komórki nerwowej. Odpowiadają za zbieranie sygnałów elektrycznych, które są przekazywane dalej. Akson, zwany również neurytem lub włóknem nerwowym, to pojedyncza długa wypustka nerwowa, której zadaniem jest przekazywanie zebranych przez dendryty sygnałów elektrycznych do innych neuronów lub narządu wykonawczego. Akson i dendryty kolejnego neuronu nie stykają się, dzieli je tzw. szczelina synaptyczna, poprzez którą przekazywane są sygnały. Transmisja polega na uwolnieniu odpowiedniej substancji chemicznej (neurotransmitera, mediatora) do neuronu odbiorczego. Substancje te wydzielane są przez kolbowato rozszerzone zakończenie aksonu, a w błonie sąsiadujących z nim dendrytów znajdują się receptory, które pasują przestrzennie do mediatorów. Gdy impuls elektryczny dotrze do kolbkowatego zakończenia aksonu, do wnętrza szczeliny synaptycznej wydzielane są neurotransmitery. Łączą się one z receptorami obecnymi w błonie komórki przyjmującej, co powoduje powstanie w niej impulsu elektrycznego przemieszczającego się dalej w kierunku jej aksonu [9].

Zdolność neuronów do komunikacji za pomocą impulsów elektrycznych, ale również działania w środowisku chemicznym, pozwala na określenie mózgu jako gęsto połączonej elektrycznej sieci przełączającej, której działanie jest uwarunkowane procesami biochemicznymi. Receptory przekazują sygnały wejściowe do sieci. Są to zarówno bodźce narządów sensorycznych jak i te z wnętrza ciała. Centralny system nerwowy w wyniku przetworzenia otrzymanych informacji przesyła dalej efektory, czyli sygnały sterujące, powodujące określoną reakcję organizmu czy działanie. Daje to obraz trzystopniowego systemu sterowniczego organizmu, który składa się z receptorów, sieci neuronowej i efektorów [9].

Na podstawie budowy biologicznego neuronu, jego sieci, a także całego systemu sterowania organizmem ludzkim wzorowane są sztuczne sieci neuronowe. Są one znacznie prostsze w budowie od swoich biologicznych odpowiedników.



Rys. 8.1. Uproszczona budowa biologicznego neuronu [51]

8.2. Neuron sztuczny (neuron McCullocha-Pittsa)

To, co nazywamy sztucznym neuronem, to uproszczone przedstawienie neuronu biologicznego za pomocą funkcji matematycznej.

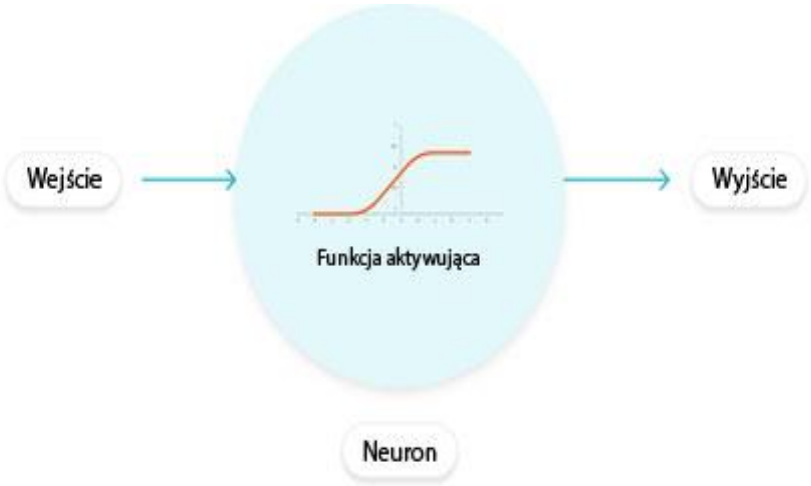
Jego nazwa pochodzi od Warrena McCullocha i Waltera Pittsa, którzy w 1943 roku zaproponowali dwie zamienne nazwy: Threshold Logic Unit (Progowa Jednostka Logiczna) lub Linear Threshold Unit (Liniowa Jednostka Progowa). Bazowo polegał on na dostarczeniu danych wejściowych, a także otrzymaniu danych wyjściowych, w formie binarnej. Dzięki czemu możliwa była implementacja funkcji logiki booleańskiej. Od razu zauważono, że za ich pomocą można zauważalnie przyspieszyć znaczną ilość obliczeń, łącząc sztuczne neurony w klastry obliczeniowe, tworząc w ten sposób sieci neuronowe. Jedną z głównych cech neuronu jest fakt, że posiada on więcej niż jedno wejście i tylko pojedyncze wyjście [21].

9. Funkcje aktywujące

W sieciach neuronowych, dane liczbowe, zwane „wejściami” (inputs), są kierowane do neuronów w warstwie wejściowej.

Wejście każdego neuronu w sieci ma określoną wagę. Wagi reprezentują istotność połączenia, na przykład waga oscylująca w okolicy 0, nie będzie miała znaczącego wpływu na wartości wyjściowe.

Funkcja aktywująca (aktywacji) jest matematycznym połączeniem pomiędzy wejściem zasilającym aktualny neuron a jego wyjściem (output), kierującym do następnej warstwy neuronów. Funkcje te mogą być tak proste jak zwyczajna funkcja przejścia, zmieniająca stan wyjścia neuronu na 0 lub 1, w zależności od ustalonych parametrów.



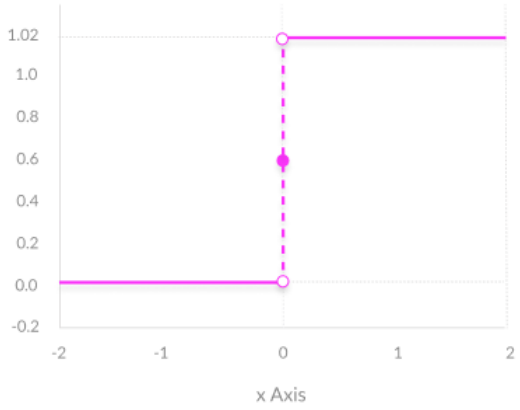
Rys. 9.1. Zasada działania funkcji aktywującej (opracowanie własne na podstawie [35])

W sieciach neuronowych jako funkcje aktywujące, stosowanych jest wiele różnych przekształceń matematycznych, między innymi:

9.1. Funkcja kroku binarnego

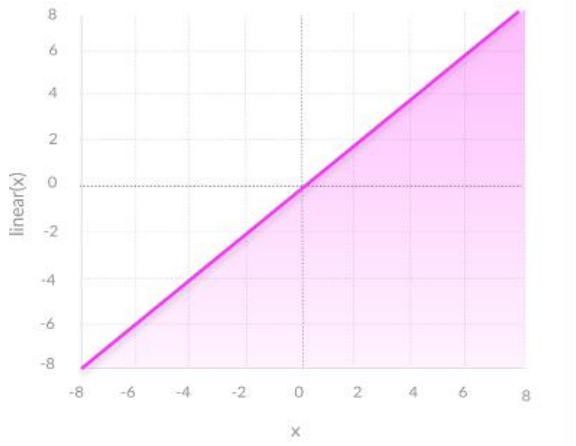
Funkcja kroku binarnego (ang. binary step function) zależna od ustalonego progu. Jej działanie polega na tym, że w zależności od przyjętego założenia, jeśli wartość wejściowa funkcji jest powyżej lub poniżej przyjętej wartości, neuron aktywuje się i przesyła dokładnie ten sam sygnał na wejście do następnej warstwy.

Jej największym ograniczeniem jest fakt, że nie dopuszcza ona wielowartościowych wyjść, więc na przykład nie nadaje się do przypisywania danych do więcej niż jednej kategorii [35].



Rys. 9.2. Wykres funkcji kroku binarnego [35]

9.2. Liniowa funkcja aktywująca



Rys. 9.3. Wykres liniowej funkcji aktywującej [35]

Działanie tej funkcji polega na pomnożeniu wejść każdego neuronu przez wagi i wyznaczeniu proporcjonalnego sygnału na wyjściu. Funkcja liniowa ma tą przewagę nad funkcją kroku binarnego, że zezwala na mnogość wyjść. Niestety, liniowa funkcja aktywacyjna ma dwie poważne wady [35]:

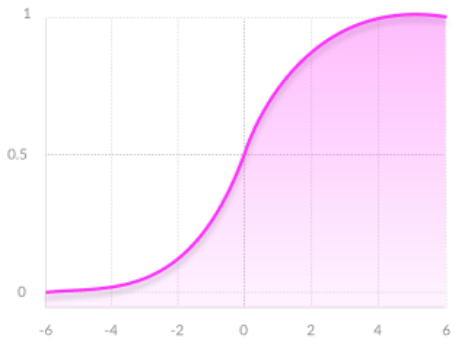
1. Uniemożliwia użycie rozpowszechniania wstecznego (backpropagation) [35][36] w celu wytrenowania modelu. Dzieje się tak ze względu na fakt, że pochodna tej funkcji jest stałą i nie ma odniesienia do wejścia. Nie jest więc możliwe cofnięcie się w celu zbadania możliwości poprawienia wag neuronów wejściowych tak, żeby umożliwić zwiększenie poprawności predykcji.

2. Spłaszcza wszystkie warstwy neuronów w jedną. Spowodowane jest to tym, że niezależnie od liczby warstw ukrytych, ostatnia warstwa będzie liniową funkcją pierwszej warstwy – spowodowane jest to własnością matematyczną mówiącą, że złożenie funkcji liniowych pozostaje funkcją liniową.

Podsumowując, liniowa funkcja aktywacyjna upraszcza sieć neuronową do modelu regresji liniowej. W ten sposób ograniczane są możliwości sieci w pracy ze zmiennymi danymi wejściowymi.

W przypadku bardziej zaawansowanych problemów, jak na przykład analiza obrazów, liniowa funkcja aktywacji okazuje się często niewystarczająca, ze względu na zbytnie upraszczanie wyników.

9.3. Funkcja sigmoidalna



Rys. 9.4.Wykres funkcji aktywującej Sigmoid [35].

Funkcja sigmoidalna osiąga wartości w zakresie pomiędzy 0 i 1, aczkolwiek nigdy nie osiągając wartości granicznych, pozwalając na znormalizowanie wyjść na każdym z neuronów. Jest dość „naturalnym” wyborem ze względu na porównanie:

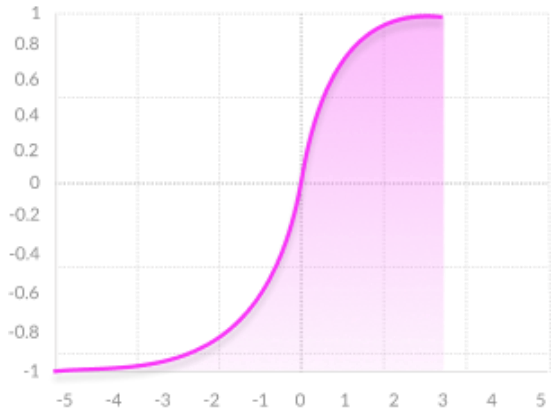
- 0 jako całkowity brak aktywności neuronu
- 1 jako maksymalna aktywność neuronu

W dzisiejszych czasach, funkcja sigmoidalna jest wybierana w coraz mniejszym stopniu, głównie ze względu na jej wady takie jak [35]:

- Dla wartości zbliżonych do 0 lub 1 predykcja zmienia się w bardzo niewielkim stopniu.
- Wartości wyjściowe nie koncentrują się w okolicy zera, więc średnie wartości danych przekazywanych do kolejnych warstw są coraz większe, co zaburza optymalizację modelu, a co za tym idzie – przedłuża ją.

- Ze względu na powyższe, funkcja sigmoidalna jest dodatkowo kosztowna obliczeniowo, co powoduje większe wymagania sprzętowe.

9.4. Tangens hiperboliczny

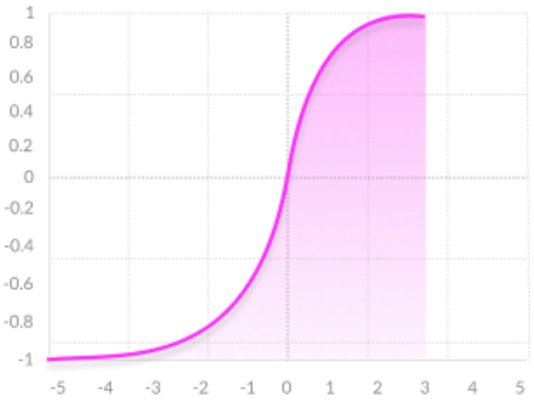


Rys. 9.5. Wykres funkcji aktywującej TanH [35]

Funkcja aktywująca tangensy hiperbolicznej jest wariantem funkcji sigmoidalnej, przeskalowanym w celu skoncentrowania wyjść neuronów w okolicy zera.

Stosowana jest w celu lepszego modelowania danych skrajnie ujemnych, skrajnie dodatnich lub neutralnych. Ze względu na podobieństwo do funkcji sigmoidalnej, dzieli również większość jej wad.

9.5. Funkcja Rectified Linear Unit



Rys. 9.6. Wykres funkcji aktywującej ReLU [35]

Funkcja aktywująca Rectified Linear Unit (ReLU) jest jedną z najbardziej wydajnych obliczeniowo – poprzez progowanie w zerze pozwala usprawnić implementację sieci a także obliczenia algorytmu [10].

Na pierwszy rzut oka mogło by się wydawać, że ReLU jest funkcją liniową, posiada ona jednak funkcje pochodne i można zastosować propagację wsteczną.

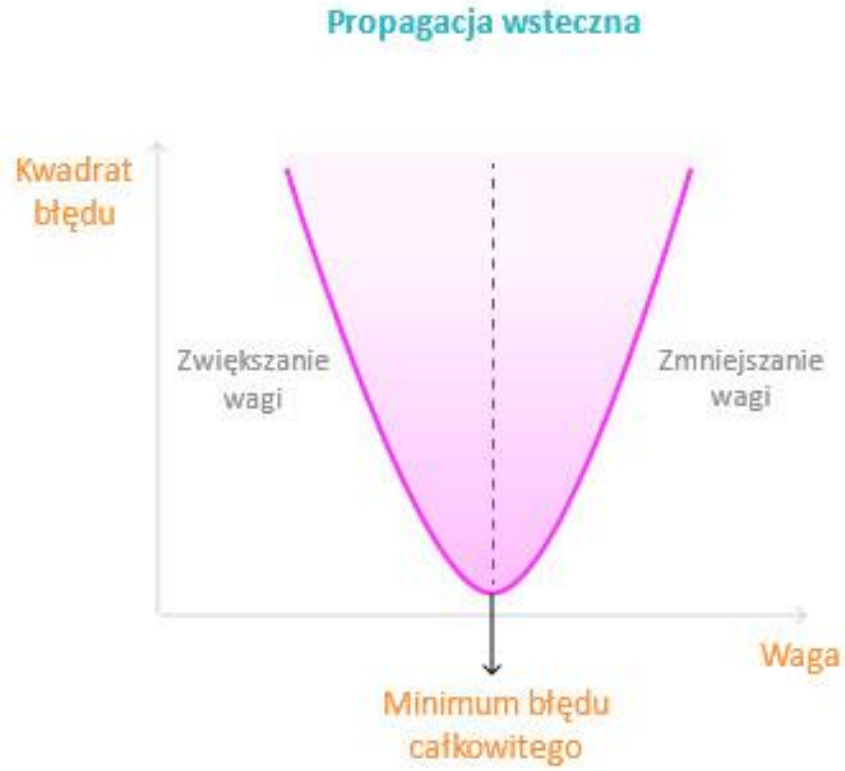
Problematyczny może okazać się przypadek, gdy wejścia są ujemne lub zerowe – sieć nie może wtedy wykonać propagacji wstecznej a w efekcie się uczyć. Można jednak zminimalizować to ryzyko ustalając niską wartość współczynnika nauki sieci.

9.6. Funkcja Softmax

Funkcja softmax to uogólnienie funkcji sigmoidalnej. Nie jest ona typową funkcją aktywującą. Jest stosowana przeważnie na ostatniej warstwie neuronów w zadaniach związanych z klasyfikacją.

Jej działanie polega na normalizacji wyjść dla każdej klasy będącej pomiędzy 0 a 1, a następnie podzieleniu przez ich sumę, w wyniku podając prawdopodobieństwo przynależności wartości wejściowych do danych klas.

9.7. Propagacja wsteczna



Rys. 9.7. Zasada działania algorytmu propagacji wstecznej (opracowanie własne na podstawie [35][36])

Rolą propagacji wstecznej (ang. backpropagation) jako algorytmu jest zmienianie wag poszczególnych neuronów w celu zminimalizowania błędów uczenia, za pomocą niewielkich nakładów mocy obliczeniowych. Algorytm propaguje błąd z ostatniej warstwy do warstw poprzedzających [19][33].

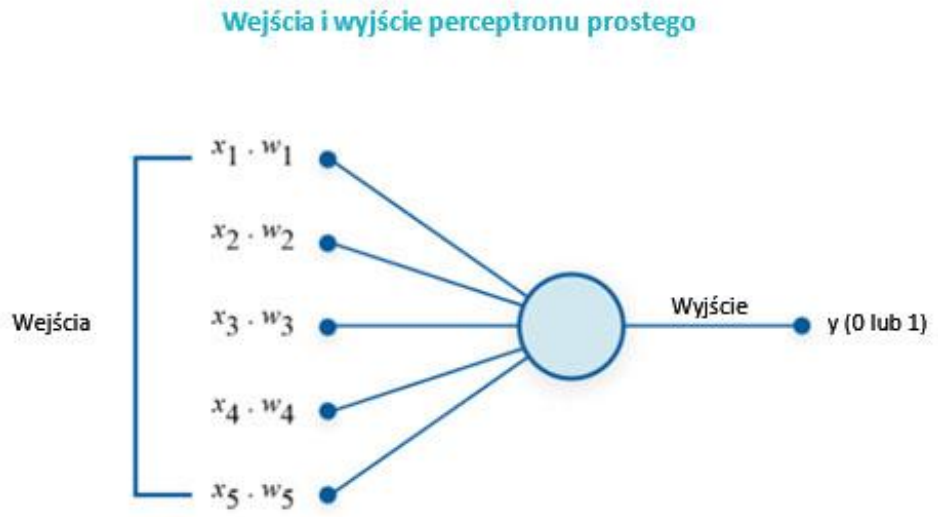
Na podstawie zasady uczenia delta [34] obliczana zostaje zmiana wag ostatniej warstwy, następnie oblicza się wartości warstwy przedostatniej, wpływające na błędne wagi warstwy ostatniej i w ten sposób aktualizowane są wagi dla warstwy przedostatniej. W analogiczny sposób oblicza się wagi i ich wpływy dla wcześniejszych warstw, aż do wyliczenia zmiany (poprawki) wag dla warstwy początkowej.

10. Sieci neuronowe

Wzorując się na mózgu, pojedyncze neurony łączą się w bardziej złożone struktury. Wyróżnia się dwa typy architektury:

- sieci rekurencyjne (ang. cyclic) – ich struktura ma charakter regularny, powtarzalny, są trudniejsze w przetrenowaniu,
- sieci skierowane (ang. feed-forward) – przepływ impulsów możliwy jest tylko w jednym kierunku, niedopuszczalna jest cykliczność.

10.1. Perceptron



Rys. 10.1. Budowa perceptronu (opracowanie własne na podstawie [37])

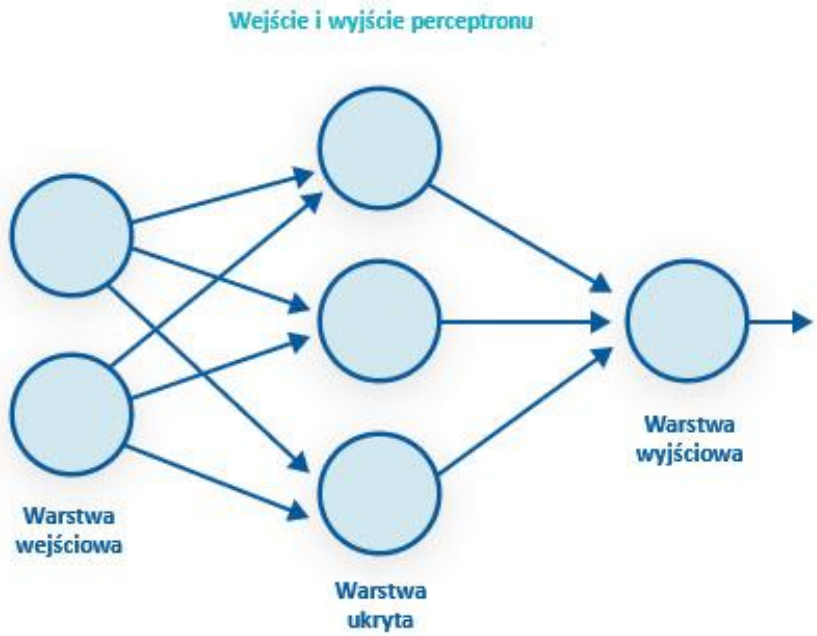
Jest on najprostszym modelem neuronu. Składa się on z określonej (N) liczby wejść i wag, jedna waga powiązana jest z jednym wejściem, a także funkcji aktywacji $f: R -> R$.

Przy danym kroku, na każdym wejściu znajdują się wartości – rzeczywiste lub całkowite. Pojedynczy neuron oblicza sumę ważoną na wszystkich wejściach, by zwrócić wartość funkcji aktywacji dla obliczonej wartości [4].

$$y(x) = \sum_{x=0}^{N} W_i X_i \tag{10.1}$$

Gdzie W_i – waga w zakresie (0, 1), a X_i to neuron, dla którego jest liczone prawdopodobieństwo.

10.2. Perceptrony wielowarstwowe



Rys. 10.2. Budowa perceptronu wielowarstwowego (opracowanie własne na podstawie [37])

Są one przykładem sieci skierowanej. Perceptrony grupuje się w warstwy. Dane podawane są do wszystkich jednostek neuronowych w pierwszej (najniższej) warstwie. Wyniki obliczeń z danej warstwy przekazywane są jako wejścia kolejnej z warstw, z kolei wyniki z ostatniej (najwyższej) warstwy, zwracane są jako wyniki dla całej sieci. Warstwy niebędące wyjściowymi nazywane są ukrytymi [36].

10.3. Sztuczne sieci neuronowe

Głównym założeniem implementacji sieci typu ANN (ang. artificial neural network) było umożliwienie działania komputera jako dokładnego odwzorowania ludzkiego mózgu, szybko jednak zorientowano się w ograniczeniach tego typu infrastruktury, postanowiono więc trenować je w określonych specjalizacjach, takich jak na przykład rozpoznawanie obrazu lub mowy.

Sztuczne sieci neuronowe zbudowane są z pewnej liczby jednostek neuronowych, które wzorowane są na biologicznych neuronach. Jednostki te łączą się ze sobą wzajemnie, przesyłając sygnały, czerpiąc z wzorca, jakim są synapsy w organicznym mózgu.

W większości przypadków neurony grupowane są w warstwy, każde z nich mogą, aczkolwiek nie muszą, przeprowadzać inne przekształcenia wejściowe.

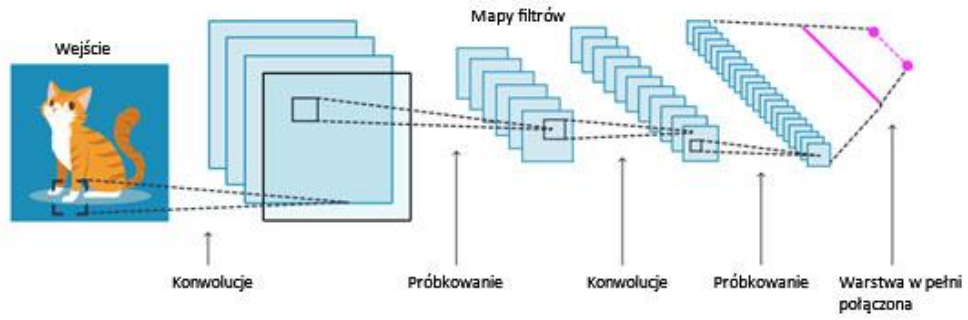
W uproszczeniu, sztuczny neuron odbiera sygnał od poprzednika lub poprzedników, przetwarza w odpowiedni dla niego sposób i przesyła go dalej do następnego odbiorcy.

10.4. Konwolucyjne sieci neuronowe (splotowe)

Rodzaj sztucznych sieci neuronowych, stosowany tam, gdzie proste perceptrony nie wystarczają z powodu złożoności obliczeniowej problemu [48][49]. Ich sposób budowy jest znacznie bardziej skomplikowany. Jest to spowodowane faktem, że zawierają w sobie co najmniej jedną, a najczęściej wiele warstw ukrytych [16][31][33].

Ich głównym zastosowaniem są wszelakie obliczenia związanie z obrazem. Wśród potencjalnych zastosowań można znaleźć:

- klasyfikacja obrazów do kategorii [45],
- wykrywanie obiektów na zdjęciach,
- analiza zdjęć satelitarnych.



Rys. 10.3. Budowa przykładowej sieci konwolucyjnej (opracowanie własne na podstawie [38]).

Konwolucyjne sieci neuronowe zbudowane są z pewnej liczby neuronów pogrupowanych ze sobą w warstwy. Warstwy te można podzielić na kilka głównych kategorii:

- konwolucyjna,
- aktywująca,
- łącząca,
- w pełni połączona.

Warstwa konwolucyjna polega na przeskanowaniu całego danego obrazu wejściowego za pomocą filtru [24] o stałym rozmiarze (na przykład 4x4 piksele) w celu wyodrębnienia cech potencjalnie ważnych przy klasyfikowaniu. Taki filtr jest też zwany jądrem splotu

(convolution kernel) [24]. Jądro zawiera także w sobie wagi neuronów, które są na bieżąco dopasowywane w celu zoptymalizowania predykcji (m. in. za pomocą propagacji wstecznej).

W jądrze o założonym rozmiarze - tutaj 4x4 piksele - dla każdego obszaru 4x4 model oblicza iloczyn skalarny pomiędzy wartościami pikseli a ich wagami zapisanymi w filtrze [40].

Po przepracowaniu pełnej konwolucji, czyli przeskanowaniu za pomocą filtra całego obrazu wejściowego, powstaje macierz nazywana „mapą cech". Każda pełna konwolucja dostarcza inną mapę cech dla tego samego obrazu wejściowego. Liczba warstw konwolucyjnych definiowana jest na początku projektowania modelu, jeszcze przed procesem uczenia. Im większa liczba filtrów, tym więcej cech zostanie wyszczególnionych, a w efekcie zwiększy się sprawność sieci w rozpoznawaniu wzorców w przypadku obrazów widzianych po raz pierwszy [20][38].

Warstwa łącząca (pooling) opiera swoje działanie na zmniejszeniu wymiarowości każdej mapy cech, aczkolwiek zachowując najbardziej istotne dane zawierające się w niej.

Najpopularniejsze warianty warstw łączących to: max (największy element), min (najmniejszy element), average (średnia), sum (suma elementów). Na przykładzie max pooling, który został wykazany jako najskuteczniejszy, definiuje się mniejsze filtry (np. 2x2) i wybierany jest największy element w każdym z tych filtrów zaktywowanych funkcją ReLU.

Zadaniem warstw łączących jest stopniowe redukowanie rozmiaru przestrzennego obrazu wejściowego, a dokładniej:

- zmniejszenie rozmiarów próbek wejściowych, co umożliwia lepszą kontrolę nad nimi,
- obniża liczbę koniecznych obliczeń i zmiennych, umożliwiając lepszą kontrolę nad przeuczaniem sieci (przekłamywaniem wyników),
- uodparnia sieć na zakłócenia i przekłamania w obrazie wejściowym,
- pozwala na skalowalność rozwiązania – umożliwia zlokalizowanie szukanego elementu niezależnie od jego umiejscowienia na obrazie.

Warstwa w pełni połączona jest to klasyczny perceptron wielowarstwowy, używający funkcji aktywującej softmax w warstwie wyjściowej. Sformułowanie „w pełni połączona"

zakłada, że każdy neuron z warstwy poprzedzającej jest połączony z każdym neuronem w kolejnej warstwie.

Warstwa konwolucyjna w połączeniu z warstwą łączącą zwracają na wyjściu najważniejsze cechy obrazu wejściowego. Założeniem warstwy w pełni połączonej jest zastosowanie tychże cech do sklasyfikowania obrazu wejściowego do jednej z klas (kategorii) w odniesieniu do treningowego zbioru danych. Dodatkowo, użycie warstwy w pełni połączonej często pozwala niskim kosztem obliczeniowym nauczyć model nieliniowej kombinacji pozyskanych cech, co może poprawić skuteczność klasyfikacji.

Funkcja softmax jest używana jako aktywator, gdyż zbierając dane z poprzedzającej warstwy, będące różnymi liczbami rzeczywistymi, grupuje je w wektor wartości bezwzględnych rzeczywistych i przekształca w wektor wartości pozostających w zakresie od 0 do 1, sumując wartości prawdopodobieństw do 1.

Całokształt procesu uczenia sieci konwolucyjnych można podsumować w kilku krokach:

1. Inicjalizacja wszystkich wag i filtrów za pomocą losowych wartości,
2. Wprowadzenie na wejście sieci obrazu treningowego, wykonanie wszystkich kroków w przód i przypisanie z pewnym prawdopodobieństwem do kategorii wynikowej – zakładając, że wagi posiadają losowe wartości - prawdopodobieństwa wyjściowe także,
3. Obliczenie błędu całkowitego na warstwie wyjściowej:

$$Błąd\ całkowity = \sum \frac{1}{2}(Pd - Pw)^2 \tag{10.2}$$

gdzie Pd – oznacza prawdopodobieństwo docelowe, a Pw – prawdopodobieństwo wyjściowe.

4. Użycie propagacji wstecznej do obliczenia gradientów błędów w odniesieniu do wszystkich wag w sieci i zastosowanie opadania gradientu (gradient descent) [34][46] do zaktualizowania wartości wag / filtrów i parametrów w celu redukcji błędu wyjściowego [6]:
    - wagi są dostosowywane w zależności od istotności ich wpływu na błąd całkowity,

- jeśli obraz wejściowy się powtórzy, prawdopodobieństwa wyjściowe będą się różniły od poprzednich dla tego samego obrazu, ale mogą być bliższe docelowym,
- powyższa sytuacja oznaczałaby, że sieć nauczyła się poprawnie klasyfikować obraz poprzez dostosowywanie wag / filtrów tak, żeby zmniejszyć błąd wyjściowy,
- najważniejsze parametry sieci, takie jak: architektura sieci, liczba filtrów, ich rozmiary itp., zostały ustalone jeszcze przed krokiem pierwszym i są niezmienne podczas procesu treningowego. Dostosowywane są – w zależności od potrzeb - tylko wartości filtrów i/lub wag.

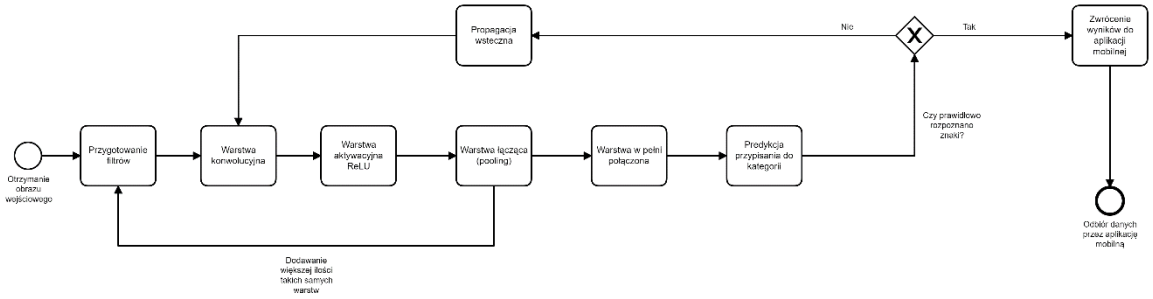
5. Powtórzenie kroków 2-4 dla wszystkich obrazów zbioru treningowego.

Kiedy obraz jest dostarczany na wejście sieci po raz pierwszy, przechodzi kolejne warstwy i zwraca prawdopodobieństwo przynależności obrazu do każdej z klas – dla nowych obrazów prawdopodobieństwa obliczane są z użyciem wag zoptymalizowanych w celu poprawnego sklasyfikowania wszystkich wcześniejszych danych treningowych [11]. W zależności od rozmiaru zbioru treningowego – jeśli jest wystarczająco duży – sieć powinna być w stanie poprawnie uogólnić nowe obrazy i poprawnie przypisać je do odpowiednich kategorii.

11. Sposoby uczenia sieci neuronowych

Zbiory danych wejściowych dostarczonych na wejście sieci neuronowej powinny być podzielone na podzbiory treningowe, testowe, a także w wielu wypadkach dodatkowo walidacyjne.

11.1. Uczenie nadzorowane



Rys. 11.1. Diagram BPMN przedstawiający proces uczenia nadzorowanego

Ten wariant uczenia sieci neuronowych polega na dostarczeniu do sieci przygotowanych, opisanych i oznaczonych danych. Dla każdej próbki wejściowej, jest zestaw wartości wyjściowych i prawdopodobieństwo przynależności tej próbki do tych wartości [32]. Schemat uczenia nadzorowanego został przedstawiony na Rys. 11.1.

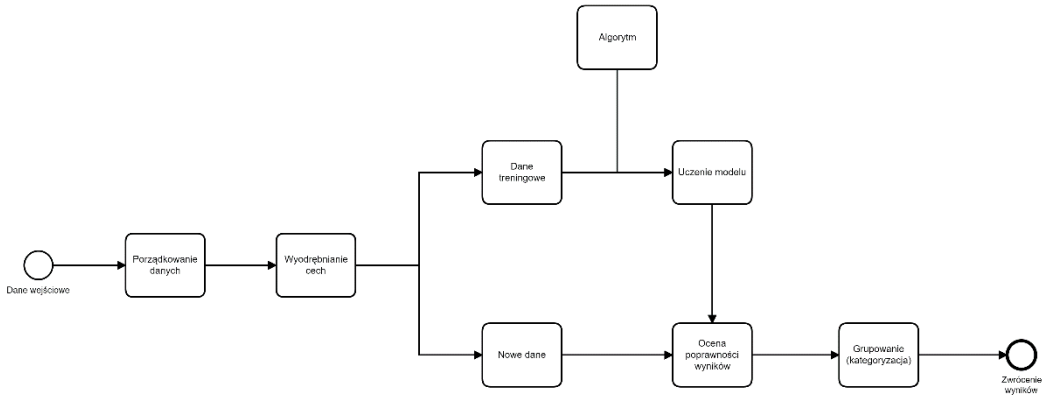
Głównym celem tego typu uczenia jest generalizacja danych i zredukowanie błędów dopasowania. W wypadku rozpoznawania obrazów, uczenie nadzorowane będzie jednym z najlepszych, jeśli nie najlepszym wyborem.

Najważniejsze wykorzystywane algorytmy do uczenia nadzorowanego to:

- regresja liniowa,
- regresja logistyczna,
- drzewa decyzyjne,
- maszyna wektorów nośnych,
- k najbliższych sąsiadów,
- liniowa analiza dyskryminacyjna.

11.2. Uczenie nienadzorowane

Różni się od uczenia nadzorowanego tym, że dane wejściowe nie posiadają opisów.) Schemat uczenia nienadzorowanego został przedstawiony na Rys. 11.2. Miarą sukcesu często jest ocena czy sieć w stanie zmienić (zmniejszyć lub zwiększyć) koszt obliczeniowy [32].



Rys. 11.2. Diagram BPMN przedstawiający proces uczenia nienadzorowanego

Dwie główne metody uczenia bez nadzoru to analiza skupień (ang. cluster analysis) [47] i analiza głównych składowych (ang. principal component analysis) [44].

Analiza skupień polega na pogrupowaniu danych nie posiadających kategorii, opisów lub klasyfikacji. Algorytm ten doszukuje się podobieństw i różnic pomiędzy poszczególnymi danymi i w ten sposób przyporządkowuje je do pewnych grup.

Analiza głównych składowych, jest z kolei procesem statystycznym, umożliwiającym przekształcenie zestawu potencjalnie powiązanych cech lub zmiennych w zestaw nie połączonych ze sobą w sposób liniowy wartości nazywanych głównymi składowymi [7].

Najważniejszym aktualnie zastosowaniem dla tego typu uczenia jest estymacja gęstości prawdopodobieństwa w statystyce. [44].

Najczęściej używane algorytmy związane z uczeniem nienadzorowanym to:

- grupowanie hierarchiczne,
- algorytm centroidów,
- DBSCAN,
- Algorytm OPTICS.

12. Testy

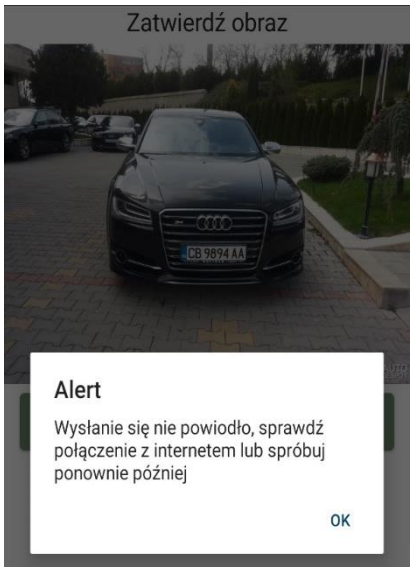
Jednym z istotnych etapów tworzenia oprogramowania jest przetestowanie działania utworzonego kodu. Fachowym podejściem do utworzenia aplikacji jest przetestowanie każdej funkcjonalności przed tym jak zostanie ona dostarczona użytkownikom. Dzięki temu unika się większości nieprzewidzianych zachowań aplikacji, jak również eliminuje się błędy programistów. W doskonałej sytuacji testy pokrywają działającą aplikację i każdą jej funkcję. W celu zweryfikowania działania systemu do rozpoznawania numerów tablicy rejestracyjnej przetestowane zostały następujące części projektu: rozpoznawanie tablicy rejestracyjnej na zdjęciu, wyodrębnienie znaków z tablicy oraz działanie aplikacji mobilnej. Wszystkie te elementy należało przetestować w celu sprawdzenia czy system został dobrze zaprojektowany oraz zaimplementowany. W tym rozdziale zostaną przedstawione jedynie testy, które są kluczowe dla działania opracowanego programu.

12.1. Testy aplikacji mobilnej

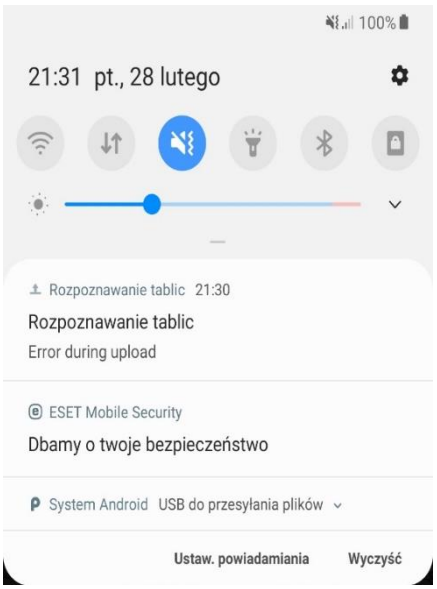
Aplikacja została wypróbowana na telefonie z systemem Android 9.0 (Pie).

Komunikacja z serwerem

Testy przebiegły poprawnie. Jeśli urządzenie nie ma dostępu do Internetu lub występują problemy z serwerem, aplikacja wyświetla informację zwrotną w postaci alertu oraz powiadomienia (Rys.12.1 i Rys.12.2).

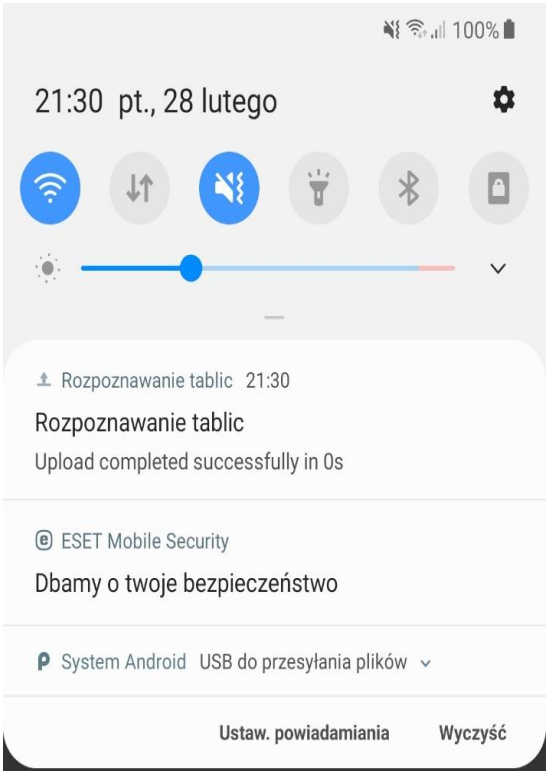


Rys. 12.1. Komunikat o problemach z połączeniem



Rys. 12.2. Powiadomienie o niepowodzeniu przesłania obrazu

Prawidłowe przesłanie obrazu na zewnętrzny komputer skutkuje wyświetleniem komunikatu w pasku powiadomień (Rys. 12.3).



Rys. 12.3.Powiadomienie o powodzeniu przesłania zdjęcia

Wnioski

Komunikacja aplikacji z serwerem obliczeniowym odbywa się prawidłowo. Użytkownik otrzymuje odpowiednie komunikaty w zależności od sytuacji.

12.2. Rozpoznawanie tablicy rejestracyjnej na zdjęciu

System rozpoznawania tablic rejestracyjnych na zdjęciu został przetestowany na 22 zdjęciach samochodów zebranych samodzielnie. Zostały one zrobione w różnych warunkach. Na rysunku 12.4a zostało przedstawione zdjęcie samochodu zrobione w idealnych warunkach: tablica jest bardzo dobrze oświetlona i widoczna, na samochodzie nie ma dużo odbić, zdjęcie nie jest za jasne bądź za ciemnie. Jak widzimy na rysunku 12.4b tablica została rozpoznana i zaznaczona na zdjęciu poprawnie.

a b

Rys. 12.4. Testowane zdjęcie (a) oryginalne, (b) z oznaczonym obszarem zawierającym tablicę rejestracyjną

Na rysunku 12.5a widzimy zdjęcie samochodu słabej jakości: zostało ono zrobione przez szybę co powoduje zwiększenie liczby odbić na zdjęciu oraz jest ono dość ciemne. Jak zauważamy na rysunku 12.5b mimo nienajlepszej jakości obrazu tablica została poprawnie rozpoznana.

a b

Rys. 12.5. Testowane zdjęcie (a) oryginalne, (b) z oznaczonym obszarem zawierającym tablicę rejestracyjną

Na rysunku 12.6a widzimy zdjęcie dobrej jakości: tablica jest widoczna, nie jest ono za jasne ani za ciemne, jednak problemem w tym wypadku jest duża liczba odbić oraz kształtów podobnych do tablicy rejestracyjnej, np.: część fasady budynku w tle. Na rysunku 12.6b możemy zauważyć, że mimo bardzo dobrej widoczności tablicy nie została ona rozpoznana z powodów opisanych powyżej.

a b

Rys. 12.6.Testowane zdjęcie (a) oryginalne, (b) z oznaczonym obszarem, który nie zawiera tablicy rejestracyjnej

Wnioski

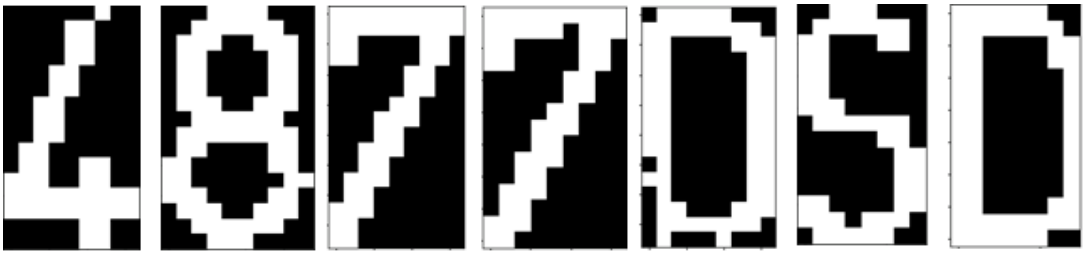
Z testów przeprowadzonych powyżej możemy wywnioskować, że wykrycie tablicy rejestracyjnej zależy od jakości zdjęcia. Jeśli zdjęcie jest słabej jakości, znajduje się na nim dużo obiektów kształtem przypominających tablicę lub na samochodzie znajduje się dużo odbić, jej poprawne wykrycie może być utrudnione lub nawet niemożliwe.

12.3. Wyodrębnienie znaków z tablicy rejestracyjnej

Segmentacja znaków została przetestowana na tym samym zbiorze zdjęć co w rozdziale 12.2. Na rysunku 12.7 widzimy wyodrębnione numery tablicy ze zdjęcia 12.4a. Jak możemy zauważyć wszystkie znaki zostały wyodrębnione poprawnie i żaden z nich nie został pominięty.

Rys. 12.7. Wyodrębnione znaki z tablicy ze zdjęcia Rys.12.4(a)

Rysunek 5a-g przedstawia znaki wyodrębnione z tablicy znajdującej się na rysunku 12.5a.



Rys. 12.8.Wyodrębnione znaki z tablicy z Rys.12.5(a)

Tak jak w poprzednim przypadku zostały one wysegmentowane popranie mimo nie za dobrej jakości zdjęcia.

Rysunek 12.9 przedstawia samochód z arabską tablicą rejestracyjną, która została wykryta jednak algorytm nie wykrył żadnego znaku, ponieważ jak wspomniano w rozdziale 6.3. do wykrycia znaków, np.: arabskich jest potrzebne stworzenie odpowiedniego algorytmu.

Rys. 12.9.Testowane zdjęcie samochodu z oznaczonym obszarem zawierającym arabską tablicę rejestracyjną

Wnioski

Z testów przeprowadzonych powyżej widzimy, że opracowana funkcja segmentacji działa poprawnie, nawet jeśli wyodrębniona tablica ze zdjęcia nie jest najlepszej jakości. Wykrycie nie następuje jedynie w momencie jeśli są one znakami posiadającymi dużo

krzywych jak język arabski, którego przykład tablicy rejestracyjnej został zaprezentowany w tym rozdziale.

13. Wnioski

Celem niniejszej pracy było opracowanie projektu i implementacji systemu rozpoznawania znaków alfanumerycznych opartego na głębokich sieciach neuronowych. Na potrzeby tego rozwiązania stworzona została aplikacja mobilna na platformę Android. Wszystkie wykorzystane technologie zostały opisane oraz uzasadniono ich wybór. Zaprezentowane zostało działanie aplikacji jak i również utworzonego algorytmu. W utworzonym systemie uzyskano wszystkie zaplanowane założenia.

W trakcie implementacji funkcji rozpoznawania oraz aplikacji mobilnej poszerzona została wiedza autorów w zakresie wykorzystanych narzędzi, technologii oraz opracowywanego tematu. Niniejsza praca dyplomowa pokazuje, że wykorzystane technologie bardzo dobrze nadają się do tworzenia rozwiązań nauczania maszynowego oraz rozwiązań mobilnych. Czas, który został wykorzystany na utworzenie algorytmu i programu oraz napisanie pracy inżynierskiej spowodował zwiększeniem wiadomości na temat użytych technologii, poprawą umiejętności programistycznych jak również rozwojem umiejętności pracy zespołowej.

Mimo, że utworzona aplikacja spełnia swoje założenia to posiada ona wiele możliwości rozwoju. Opracowany system można w przyszłości rozbudować m.in. o rozpoznawanie tablic ze znakami np.: arabskimi, chińskimi, koreańskimi, japońskimi, przechowywanie historii wykrytych numerów rejestracyjnych. Oprócz tego aplikację mobilną, dzięki zastosowaniu wieloplatformowej struktury programistycznej, można uzupełnić o wsparcie dla systemu iOS.

Bibliografia

[1] Abe K., Suzuki S.: Topological Structural Analysis of Digitized Binary Images by Border Following. Computer Vision, Graphics, and Image Processing, April 1985,Volume 30, Issue 1, Strony 32-46

[2] Bochenek, A., & Reicher, M. Anatomia Człowieka, tom IV (Układ nerwowy ośrodkowy). PZWL, Warszawa, 2000.

[3] Chan R.: The 10 most popular programming languages, according to the Microsoft-owned GitHub, https://www.businessinsider.com/most-popular-programming-languages-github-2019-11?IR=T [dostęp: grudzień 2019]

[4] Czoków, M., & Piersa, J., Wstęp do sieci neuronowych, wykład 01 Neuron biologiczny. Model perceptronu prostego, [dostęp: listopad 2019], 2011.

[5] Fain Y., Moiseev A. Angular. Programowanie z użyciem języka TypeScript. Wydanie II, Helion 2020.

[6] Girshick, R. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448), 2015 [dostęp: listopad 2019].

[7] Hearty J.: Zaawansowane uczenie maszynowe z językiem Python. Helion, 2017.

[8] Karthaus M.: Javascript implementation of the Ramer Douglas Peucker Algorithm, https://karthaus.nl/rdp/ [dostęp: grudzień 2019]

[9] Kozubski, W., & Liberski, P.: Neurologia. Podręcznik dla studentów medycyny. PZWL, Warszawa, 282-286, 2006.

[10] Krizhevsky, A., Sutskever, I., & Hinton, G. E., Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105), 2012.

[11] Kunysz A.: Jak działa rozpoznawanie obrazu?. Programista, 6/2019 (85).

[12] Lis M. PHP7. Praktyczny kurs. Helion 2017.

[13] Lockhart J. PHP. Nowe możliwości, najlepsze praktyki. Helion 2015.

[14] Maduria V.B., Vydehi S.: Edge Detection Techniques using Character Segmentation and Object Recognition, https://www.ijsr.net/archive/v2i2/IJSRON2013307.pdf [dostęp: styczeń 2020]

[15] Musoromy Z., Ramalingam S., Bekooy N.: Edge Detection Comparison for License Plate Detection,https://ieeexplore.ieee.org/document/5707935 [dostęp: grudzień 2019]

[16] O'Shea, K., & Nash, R., An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458, 2015.

[17] Parikh D.: License Plate Region Detection for Automatic License Plate Recognition Systems, https://medium.com/datadriveninvestor/license-plate-region-detection-3c97fbb3a29 [dostęp: listopad 2019]

[18] Shan B.: License Plate Character Segmentation and Recognition Based on RBF Neural Network, https://ieeexplore.ieee.org/abstract/document/5459969 [dostęp: styczeń 2020]

[19] Sharma P.: Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques (Part 1), https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/ [dostęp: listopad 2019]

[20] Widrow, B., & Lehr, M. A., 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. Proceedings of the IEEE, 78(9), 1415-1442, 1990. [dostęp: listopad 2019],

[21] Zeiler, M. D., & Fergus, R., Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham, [dostęp: listopad 2019], 2014.

[22] Żurada, J., Barski, M., Jędruch, W.: Sztuczne sieci neuronowe: podstawy teorii i zastosowania. Wydawnictwo Naukowe PWN, 1996.

[23] Obwieszczenie Ministra Infrastruktury i Rozwoju z dnia 11 lipca 2014 r. (Dz.U. 2014 poz. 1522), załącznik nr. 8

[24] Learning Filter Basis for Convolutional Neural Network Compression, https://arxiv.org/pdf/1908.08932.pdf [dostęp: grudzień 2019]

[25] OpenCV Documentation, https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html [dostęp: grudzień 2019]

[26] https://bdl.stat.gov.pl/BDL/dane/podgrup/tablica [dostęp: luty 2020]

[27] https://www.digitalocean.com/community/tutorials/how-to-configure-nginx-as-a-web-server-and-reverse-proxy-for-apache-on-one-ubuntu-18-04-server [dostęp: luty 2020]

[28] https://docs.microsoft.com/pl-pl/xamarin/essentials/?WT.mc_id=docs-dotnet-xamarin [dostęp: marzec 2020]

[29] https://github.com/jupyter/jupyter/wiki/Jupyter-kernels [dostęp: styczeń 2020]

[30] https://insights.stackoverflow.com/survey/2019/#technology-_-most-popular-development-environments [dostęp styczęń 2020]

[31] https://karpathy.github.io/ [dostęp: grudzień 2019].

[32] https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ [dostęp: grudzień 2019].

[33] https://medium.com/@karpathy [dostęp: grudzień 2019].

[34] https://medium.com/@neuralnets/delta-learning-rule-gradient-descent-neural-networks-f880c168a804 [dostęp: listopad 2019].

[35] https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/ [dostęp: grudzień 2019].

[36] https://missinglink.ai/guides/neural-network-concepts/backpropagation-neural-networks-process-examples-code-minus-math/ [dostęp: grudzień 2019].

[37] https://missinglink.ai/guides/neural-network-concepts/perceptrons-and-multi-layer-perceptrons-the-artificial-neuron-at-the-core-of-deep-learning/ [dostęp: grudzień 2019].

[38] https://missinglink.ai/guides/tensorflow/tensorflow-conv2d-layers-practical-guide/ [dostęp: styczeń 2020].

[39] https://www.nativescript.org/blog/nativescript-and-xamarin [dostęp: luty 2020]

[40] http://neuralnetworksanddeeplearning.com/index.html [dostęp: listopad 2019]

[41] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html [dostęp: luty 2020]

[42] https://www.php.net/manual/en/function.shell-exec.php [dostęp: luty 2020]

[43] https://reactjs.org/ [dostęp: luty 2020]

[44] https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c [dostęp: styczeń 2020]

[45] https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d [dostęp: grudzień 2019]

[46] https://towardsdatascience.com/machine-learning-101-an-intuitive-introduction-to-gradient-descent-366b77b52645 [dostęp: styczeń 2020]

[47] https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68 [dostęp: grudzień 2019]

[48] http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/ [dostęp: grudzień 2019]

[49] https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ [dostęp: listopad 2019]

[50] https://www.varonis.com/blog/what-is-a-proxy-server/ [dostęp: luty 2020]

[51] https://www.zabezpieczenia.com.pl/ [dostęp: listopad 2019]