

---

# Rozpoznawanie w czasie rzeczywistym prostego zestawu gestów przy użyciu kamery webowej

## RAPORT

---

**Daniel Popek**  
228030

**Marcin Wątroba**  
228163

### Abstrakt

Dokument ten jest raportem podsumowującym projekt z Analizy Obrazów i Wideo. Celem projektu było opracowanie rozwiązania umożliwiającego rozpoznawanie zestawu gestów w strumieniu wideo pobieranym z kamerki internetowej. W projekcie ograniczono się do gestów statycznych, zaimplementowano rozwiązanie w oparciu o klasyfikator neuronowy oraz o metodę wykrywania koniuszków palców. Przebadano również szereg metod segmentacji dloni. Efektem końcowym jest aplikacja desktopowa umożliwiająca przetestowanie w czasie rzeczywistym opracowanej metody segmentacji oraz klasyfikatora. W raporcie zawarto wyniki prowadzonych badań.

## 1 Wstęp

### 1.1 Wprowadzenie do problemu

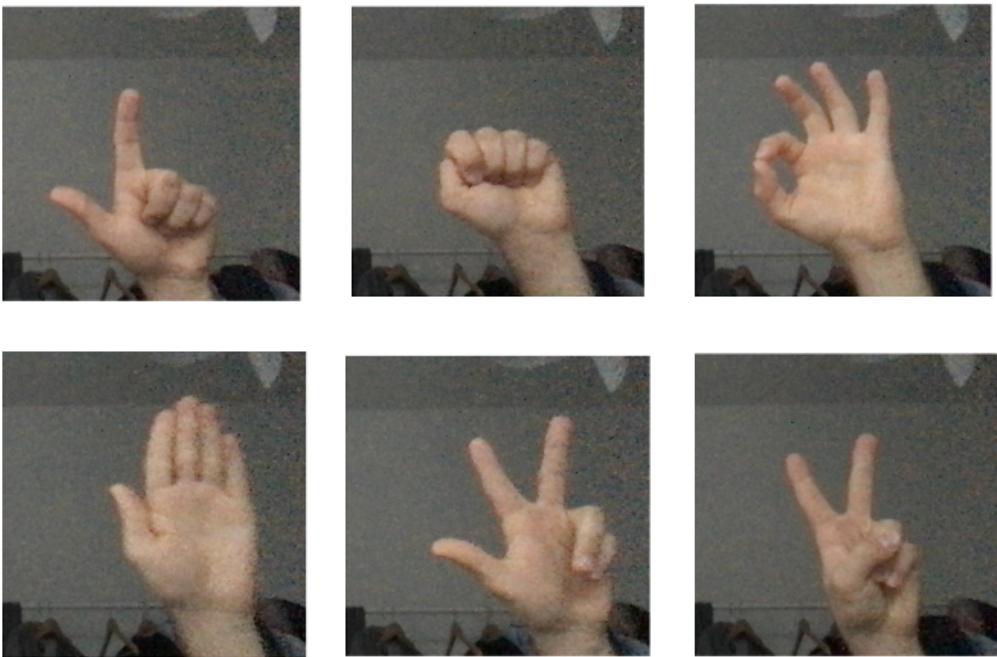
Celem projektu było opracowanie podejścia służącego do rozpoznawania prostego zestawu gestów przy użyciu kamery webowej. Ograniczono się do gestów statycznych widocznych na Rysunku 1. Typowymi podzadaniami dotyczącymi problemu rozpoznawania gestów na obrazie są: śledzenie dloni (ang. hand tracking), segmentacja dloni (ang. hand segmentation) oraz klasyfikacja gestów (prostych lub sekwencyjnych).

### 1.2 Zastosowania

Istnieje szereg dziedzin, w których coraz częściej rozwiązań podobnej kategorii znajdują zastosowanie. Wykrywanie gestów wykorzystuje się najczęściej jako sposób realizacji interfejsu człowiek-komputer, np. do sterowania interaktywnymi ekranami, tablicami, urządzeniami mobilnymi lub konsolami do gier. W branży IoT, wykorzystuje się je w rozwiązaniach typu Smart Home (m.in. do sterowania oświetleniem, muzyką, zaciemnianiem okien), jak również do sterowania robotami (w Human Robot Interaction). Coraz częściej wdraża się rozpoznawanie gestów w przemyśle samochodowym - gestami można obierać połączenia, zmieniać kanały w radio i wykonywać inne komendy tego pokroju. Szereg rozwiązań badanych w literaturze testowany był również pod kątem przydatności w rehabilitacji, w przypadkach kiedy pacjent próbuje odzyskać władzę w dloniach po uszkodzeniach mechanicznych. Najbardziej oczywistą kategorią zastosowań jest natomiast automatyczne rozpoznawanie języka migowego. Z perspektywy tego zadania bardziej istotne jest rozpoznawanie gestów sekwencyjnych.

### 1.3 Przyjęte założenia i ograniczenia

Ze względu na złożony charakter zadania postanowiono przyjąć następujące ograniczenia. Przed wszystkim ograniczono się do gestów statycznych (niesekwencyjnych). Kolejnym uproszczeniem



Rysunek 1: Zestaw gestów, na którym pracowano. W kolejności od lewej do prawej, od góry do dołu: arrow (strzałka), fist (pięść), okey, palm (palma), three (trzy) oraz victoria.

było zrezygnowanie z podzadania dotyczącego śledzenia dloni. Postawiono ustawić na stałe w prawym górnym rogu obrazu zwracanego przez kamerę internetową tzw. obszar zainteresowania - ROI (ang. region of interest). Tym samym, przyjęto uproszczenie, że system będzie bardziej przyjazny dla osób praworęcznych. W niektórych metodach binaryzacji w oparciu o kolor skóry rozważano jedynie przedział wartości parametrów dla osób o jasnej karnacji.

#### 1.4 Założony zakres rozwiązania

Skupiono się na opracowaniu dobrze działającego klasyfikatora gestów. Jako podstawę przyjęto klasyfikator neuronowy, który starano się sukcesywnie ulepszać, tak aby osiągnąć możliwie najlepszą wartość predykcji. Kolejnym postawionym celem było przetestowanie szeregu metod segmentacji dloni. Ostatnim - zaimplementowanie rozwiązania bazującego na wykrywaniu koniuszków palców i sprawdzenie jego przydatności pod kątem wykrywania gestów. Modele wyuczano w oparciu o własne dane, których zbieranie również stanowiło istotny nakład pracy. Rozwiązania testowano we własnej aplikacji przechwytyjącej strumień z kamerki internetowej. Istotnym aspektem stało się więc to, aby zaimplementowane rozwiązania można było wykorzystać w czasie rzeczywistym. Zrezygnowano natomiast z początkowo rozważanego trybu definiowania własnych gestów w czasie działania aplikacji.

## 2 Przegląd literaturowy

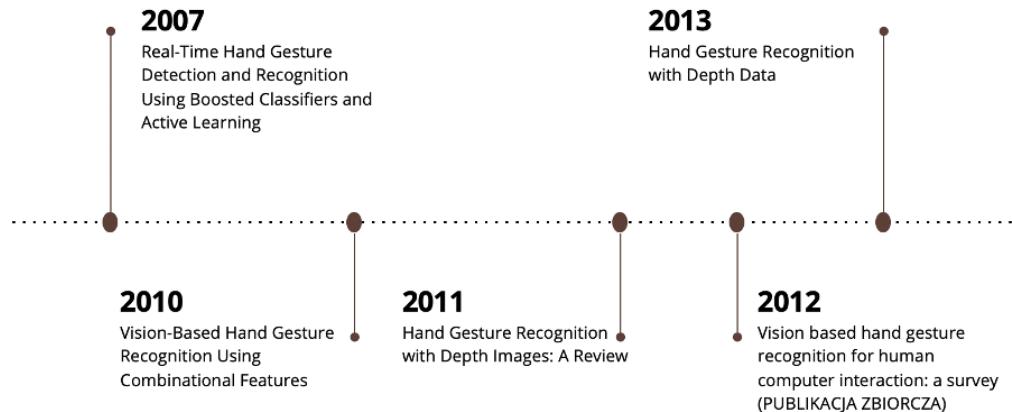
### 2.1 Przykładowe publikacje

W ramach rozpoznania problemu znalezione zostały następujące publikacje. Część z nich odnosi się do gestów sekwencyjnych, część stanowi prace przeglądowe, część skupia się na wybranych podzadaniach. Stanowiły one źródło inspiracji. Chronologia publikacji została przedstawiona na Rysunku 2.

- Real-Time Hand Gesture Detection and Recognition Using Boosted Classifiers and Active Learning (1) - Artykuł przedstawia metodę opierającą się na wykorzystaniu Boostingu

do wykrywania rąk i rozpoznawania gestów, a także na zastosowaniu segmentacji skóry i procedur śledzenia dloni. Główną nowością proponowanego podejścia jest zastosowanie innowacyjnych technik uczenia - active learningu i bootstrapingu.

- Vision-Based Hand Gesture Recognition Using Combinational Features (2) - W tym artykule przedstawiono metodę ekstrakcji cech dla gestów ręki opartą na wielowarstwowych sieciach neuronowych. Wykorzystuje się kolor skóry dloni w przestrzeni kolorów YCbCr do jej rozpoznawania.
- Hand Gesture Recognition with Depth Images: A Review (3) - Jest to praca przeglądowa na temat wykorzystania aspektu głębokości do śledzenia dloni i rozpoznawania gestów. W badaniu przeanalizowano 37 artykułów opisujących oparte na głębokości systemy rozpoznawania gestów.
- Vision based hand gesture recognition for human computer interaction: a survey (4) - Artykuł przeglądowy zawierający analizę badań porównawczych przeprowadzonych w obszarze rozpoznawania gestów. Skupia się na taksonomiach gestów, ich reprezentacjach i technikami rozpoznawania oraz platformach programowania. Koncentruje się na trzech głównych fazach detekcji gestów dloni, tj. wykrywaniu, śledzeniu i rozpoznawaniu.
- Hand Gesture Recognition with Depth Data (5) - W tym artykule przedstawiono nowatorski schemat rozpoznawania gestów rąk oparty na danych o głębokości. Ręka jest najpierw wydzielona z uzyskanych map głębokości oraz informacji o kolorze skóry. Następnie dłoń jest dzielona na obszary dloni i palców, wyodrębniane się dwa różne zestawy deskryptorów cech, oparte na odległościach palców od środka dloni oraz krzywiznie konturu dloni. Wreszcie, do rozpoznania wykonywanych gestów wykorzystywany jest wieloklasowy klasyfikator SVM. Proponowany schemat działa w czasie rzeczywistym i jest w stanie osiągnąć bardzo wysoką dokładność na danych pozyskanych za pomocą Kinecta.
- A Hidden Markov Model based Dynamic Hand Gesture Recognition System using OpenCV (6) - Podejście polegające na rozszerzeniu standardowej metody rozpoznawania gestów z wykorzystaniem ukrytego modelu Markowa o globalną zmienność parametryczną prawdopodobieństw wyjściowych stanów HMM. Stosując liniowy model zależności, formułuje się metodę Expectation Maximization (EM) do wyuczenia parametrycznego HMM.



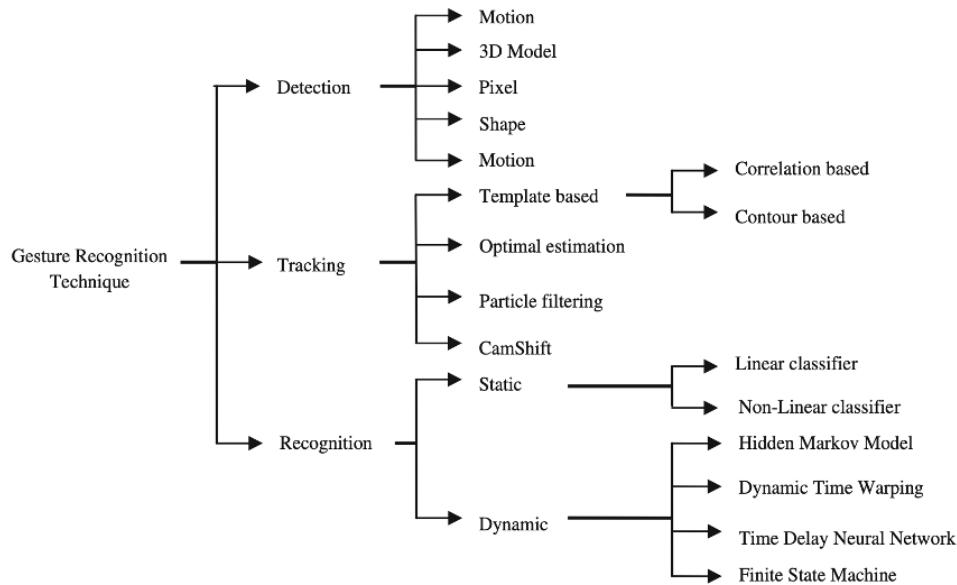
Rysunek 2: Podobne tematyką publikacje znalezione w sieci

## 2.2 Ocena przydatności publikacji

Zapoznanie się z publikacjami opisanymi w poprzednim podpunkcie dało nam pogląd na temat typowych rozwiązań dotyczących zadania wykrywania gestów. Z perspektywy projektu istotne szczególnie były opisy potoku przetwarzania oraz konkretne podejścia do segmentacji i klasyfikacji dloni. Oczywiście większość publikacji odnosi się do konkretnych przypadków rozmijających się z

niniejszym projektem. Dotyczy to modeli opartych o badanie głębi (np. przy zastosowaniu Kinecta) oraz modeli do klasyfikacji gestów sekwencyjnych.

### 2.3 Typowe metody



Rysunek 3: Grafika przedstawia typowe metody i algorytmy służące do realizacji podzadań w ramach systemów do wykrywania gestów. Źródło: (5)

Algorytmy detekcji dloni (segmentacji) opieraj± siê najczêstziej o wykrywanie cech dotycz±cych kszta³tu lub wartoœci pikseli w obranej przestrzeni barwnej. Czêœc metod bazuje natomiast na wykrywaniu elementów ruchomych na obrazie. Jeœli chodzi o tracking, najczêstiej powtarzaj±cym siê w publikacjach algorytmem by³ CamShift. W niniejszym projekcie nie skupiano siê jednak na sledzeniu dloni. Do klasyfikacji gestów statycznych wykorzystuje siê najczêstiej klasyfikatory liniowe (np. regresje logistyczna) lub nieliniowe (głęboki sieci neuronowe). Klasyfikacja gestów sekwencyjnych opiera siê na sieciach rekurencyjnych, maszynach stanowych lub ukrytych łañcuchach Markowa.

### 2.4 Wyniki podobnych rozwi±zañ z publikacji

W czêœci prac naukowych nie podawano dok³adnych wyników, ograniczono siê tylko do tego jak dobrze metoda rozpoznaje gesty. Prezentowane wyniki iloœciowe dotyczyły własnych zbiorów testowych, co zawsze nasuwa pytanie o ich rzetelnoœć. Nie istnieje żaden popularny zbiór benchmarkowy, na którym mo¿na by porównywaæ predykcje przygotowanych własnych modeli oraz zestawić jà z rozwi±zaniemi z publikacji. Odnosząc siê do wyników z publikacji, nale¿y podejœæ do liczb z rezerw¹ i mieœ na uwadze to, że nie uwzglêdniaj¹ one jednolitych warunków testowych. Uda³o nam siê znaleœæ wyniki dla nastêpuj±cych prac.

#### 2.4.1 Wyniki pozyskane z publikacji Real-Time Hand Gesture Detection and Recognition Using Boosted Classifiers and Active Learning

W tej publikacji zastosowano wykrywanie regionu z rek¹ (nie by³o stosowane ROI). Nastêpnie wykorzystywana by³a sieœ konwolucyjna do rozpoznawania gestu. widaœ na rys. 4, œe wyniki utrzymuj¹ siê na poziomie 60-85%, jest to stosunkowo dobry wynik jak na wykorzystanie takiej metody, jednak¿e slaby jeœeli zbiór testowy by³ podobny do treningowego.

Class\Predicted	Fist	Palm	Pointing	Five	Unknown	<i>Detection and recognition rates [%]</i>
Fist	1533	2	870	9	15	63.1
Palm	39	1196	10	659	15	62.3
Pointing	436	36	1503	27	86	72.0
Five	103	32	6	1446	127	84.3

Rysunek 4: Wyniki uzyskane z publikacji Real-Time Hand Gesture Detection and Recognition Using Boosted Classifiers and Active Learning

#### 2.4.2 Wyniki pozyskane z publikacji Hand Gesture Recognition with Depth Data

Publikacja ta dotyczy modelu z klasyfikatorem SVM, który rozpoznaje 10 gestów z dobrą skutecznością. Skuteczność (accuracy) dla poszczególnych gestów waha się w przedziale 85-100%

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G1	1	0	0	0	0	0	0	0	0	0
G2	0	1	0	0	0	0	0	0	0	0
G3	0	0.05	0.95	0	0	0	0	0	0	0
G4	0	0	0	0.8	0.05	0.15	0	0	0	0
G5	0	0	0	0	1	0	0	0	0	0
G6	0	0	0	0	0.05	0.95	0	0	0	0
G7	0	0	0.05	0	0	0	0.85	0	0.1	0
G8	0	0	0	0	0	0	0	1	0	0
G9	0	0	0.05	0	0	0	0	0.1	0.85	0
G10	0	0	0	0.05	0.1	0	0	0	0	0.85

Rysunek 5: Macierz konfuzji z publikacji Hand Gesture Recognition with Depth Data

#### 2.5 Wyniki z sieci

Postanowiono również odwołać się do artykułów popularno naukowych odnalezionych w Internecie, w których pojawiły się opisy metod, wraz z kodami źródłowymi do programów.

##### 2.5.1 Artykuł – Tutorial: Using Deep Learning and CNNs to make a Hand Gesture recognition model

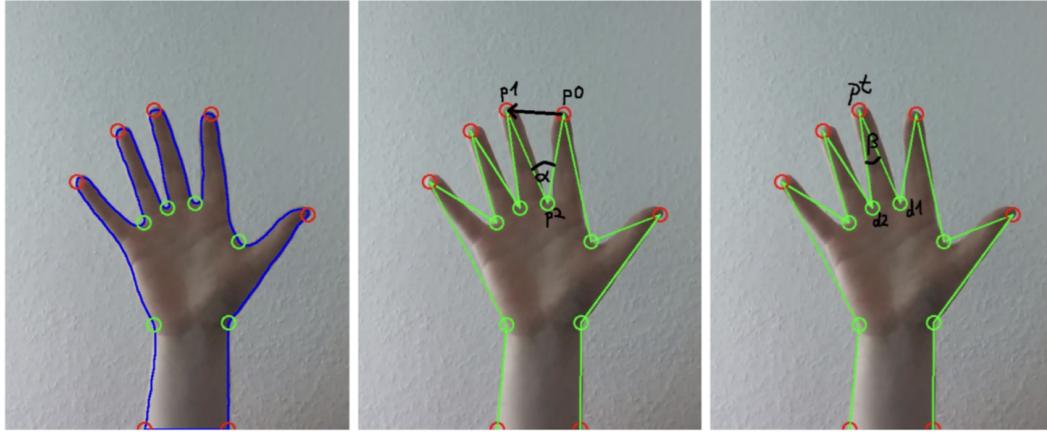
Artykuł przedstawia prosty model oparty o CNN, który na wejściu ma obraz z bardzo dobrze wydzieloną dłonią pokazującą gest. W macierzy pomyłek na Rysunku 6. widać, że klasyfikator doskonale spełnił swoje zadanie. Niestety w praktyce obrazów z tak dobrze wydzieloną dłonią nie można otrzymać w warunkach testowych - zastosowanie tego modelu do rzeczywistych zdjęć nie będzie prawdopodobnie przynosiło dobrych rezultatów.

	Predicted Thumb Down	Predicted Palm (H)	Predicted L	Predicted Fist (H)	Predicted Fist (V)
Actual Thumb Down	604	0	0	0	0
Actual Palm (H)	0	617	0	1	0
Actual L	0	0	621	0	0
Actual Fist (H)	0	0	0	605	0
Actual Fist (V)	0	0	0	0	596

Rysunek 6: Wyniki uzyskane z artykułu Tutorial: Using Deep Learning and CNNs to make a Hand Gesture recognition model

### 2.5.2 Artykuł – Simple Hand Gesture Recognition using OpenCV and JavaScript

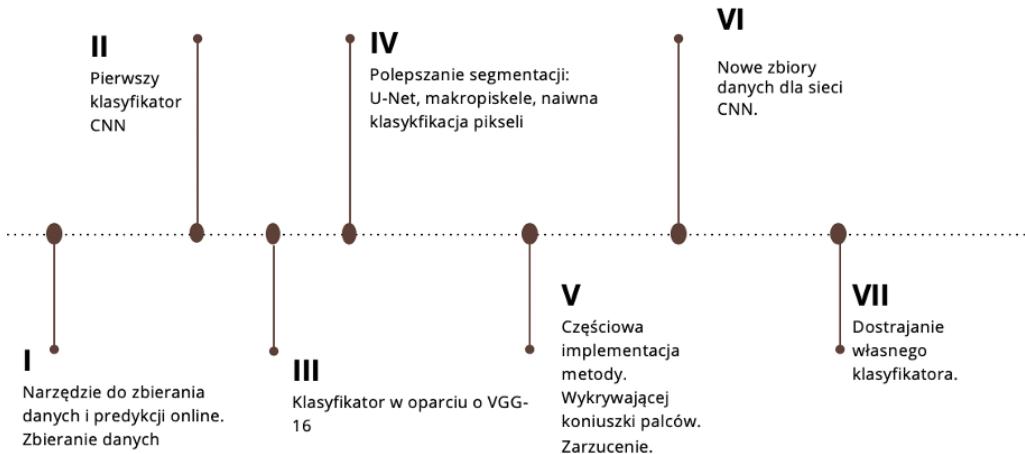
Ten artykuł nie porusza tematu samego rozpoznawania gestów lecz palców. Podstawą dla tej metody jest bardzo dobre oddzielenie ręki od tła. Sieć konwolucyjna może wybaczyć mały błąd segmentacji, natomiast w prezentowanej metodzie zachodzi szczególna analiza konturów dłoni, która może być zaburzona wraz z małym błędem segmentacji. Model nie został przebadany przez autora ilościowo, jednak w prezentowanym materiale wideo model bezbłędnie rozpoznawał palce u ręki. Do rozwiązania należy ponownie podejść z rezerwą.



Rysunek 7: Przykładowe wyniki wykrywania palców pozyskane z artykułu Simple Hand Gesture Recognition using OpenCV and JavaScript

## 3 Własne rozwiązanie

### 3.1 Rozwiązywane problemy krok po kroku



Rysunek 8: Kolejne etapy rozwoju projektu

W pierwszej kolejności zaimplementowano narzędzie umożliwiające nakładanie na strumień video pożądanych efektów wizualnych w czasie rzeczywistym oraz pozwalające na zapisywanie zrzutów z tego strumienia, czyli wzorców uczących dla później tworzonych modeli.

Następnie zaimplementowano własny klasyfikator działający w oparciu o sieć konwolucyjną oraz klasyfikator nadbudowujący sieć VGG-16 o warstwy w pełni połączone. Zaobserwowano, że sieć wyuczana obrazami kolorowymi sprawdza się słabo w rzeczywistych warunkach testowych, pomimo dobrego wyuczenia (rzędu 90%). Prawdopodobna przyczyną jest konieczność odpowiednio dużego zróżnicowania tła w próbkach uczących. Postanowiono więc skupić się na klasyfikatorach uczonych obrazami binarnymi – z wyciętym tłem, które pozbywają się tego problemu.

W kolejnej fazie skupiono się więc na ulepszaniu techniki segmentacji dloni ( i tym samym binaryzacji obrazu na dwa obszary – z dłonią i bez) Pierwsza wersja binaryzacja polegała na wykrywaniu pikseli należących do skóry, bazując na charakterystycznym dla niej przedziale wartości parametrów w przestrzeni barwnej HSV. W kolejnym kroku badano metodę hierarchiczną bazującą na makrosiskelach. Próbowano również wyuczyć zwykły klasyfikator bayesowski klasyfikujący pojedyncze piksele, jak również bardziej zaawansowaną segmentacyjną głęboką sieć konwolucyjną U-Net.

W międzyczasie dopracowano własny klasyfikator neuronowy, tak aby osiągał jak najlepszą skuteczność klasyfikacji oraz przetestowano metodę wykrywania koniuszków palców na obrazie, korzystając z otoczki wypukłej (ang. convex hull). Ostatecznie zarzucono to rozwiązanie, ze względu na uzależnienie jego skuteczności od segmentacji dłoni. Nie wprowadzało ono żadnej wartości dodanej ponad klasyfikator neuronowy.

W ostatniej fazie projektu stworzono zbiory treningowe i testowe, tak aby wprowadzić do danych uczących zaszumienie, a zbiór testowy przybliżyć do rzeczywistych warunków użytkowych.

### 3.2 Opis ostatecznego rozwiązania

Ostatecznie postanowiono oprzeć pierwszą fazę binaryzacji o pobieranie maski tła otoczenia. Rozwiązanie to pozwoliło uzyskać bardzo dobre efekty segmentacji dloni. Jego wadą jest jednak podatność na zmienne oświetlenie oraz zmiany w otoczeniu. Postanowiono więc dodać możliwość rekalibracji tła. Przy dobrych warunkach oświetleniowych( brak zmiennego zacienienia, zmiennego źródła światła lub refleksów) efekty takiego rozwiązania oceniamy jako bardzo dobre.



Rysunek 9: Potok przetwarzania w ostatecznym rozwiązaniu

W momencie przed pojawiением się gestu pobierana jest maska tła. Służy ona jako podstawa do wycięcia pojawiającej się przed nią dłoni. Obraz zamieniany jest do skali szarości oraz nakładane jest na niego rozmycie Gaussowskie. Ostatnim etapem jest binaryzacja w oparciu o progowanie i metodę Otsu. Zaletą rozwiązania jest to, że działa bardzo dobrze w czasie rzeczywistym. Wadą natomiast konieczność redefinicji raz na jakiś czas tła.



Rysunek 10: Kolejne etapy segmentacji dłoni

Kolejnym ważnym elementem systemu jest klasyfikator, który działa w oparciu o konwolucyjną sieć neuronową o następującej architekturze:

- 2 bloki konwolucyjne
- BatchNormalization i połączenia rezydualne
- 64 filtry konwolucyjne w bloku
- 2 warstwy FullyConnected z ReLU z dropoutem
- Warstwa końcowa FullyConnected z Softmaxem
- Kernel 3x3
- Average pooling 3x3, stride 2x2
- ADAM jako optymalizator

Zastosowanie połączeń rezydualnych oraz normalizacji Batch Normalization wynikało z próby poprawienia osiągów klasyfikatora. Obydwa rozwiązania przyniosły poprawę, co przedstawiono w sekcji dotyczącej badań.

### 3.3 Wykorzystane komponenty obce i ich użycia funkcjonalność

Podczas projektu zostały użyte następujące biblioteki dla języka Python:

- opencv-python – główna biblioteka do obróbki zdjęć. Była wykorzystywana do zmiany rozmiarów zdjęć, binaryzacji, maski tła oraz pozyskiwania obrazu z kamery internetowej
- tensorflow-gpu – z pomocą tej biblioteki były uczone wszelkie modele predykcji, zapisywanie modeli oraz data augmentation; dodatkowo było wykorzystywane wsparcie dla GPU co korzystnie przełożyło się na krótkie czasy uczenia
- seaborn – główna biblioteka do przygotowywania wykresów
- sklearn – z jej wykorzystaniem obliczane były różnorodne metryki takie jak accuracy, F1-score czy confusion matrix
- skimage – biblioteka zawierająca różne funkcje do klasteryzacji obrazów

### 3.4 Architektura i opis zaimplementowanego programu

W ramach projektu została przygotowana aplikacja, która pozwalała na bieżąco pokazywać wykrywane gesty i zbierać dane. W prawym górnym rogu umieszczony jest obszar ROI (Region Of Interest) – to właśnie tam powinna pojawić się ręka z gestem. Pozostały obszar z kamery jest pomijany przy predykcji. Aplikacja ma dwa główne tryby:

- Tryb zbierania danych – pozwala wykonywać użytkownikowi sesje, które pozwolą pozyskać zdjęcia do późniejszej analizy i wyuczania modeli. Główną myślą jest pozyskiwanie oetykietowanych zdjęć, co zostało osiągnięte poprzez klawisze funkcyjne zapisujące zdjęcia gestów takich jak:
  - Strzałka – 'a' (arrow)
  - Ok – 'o'

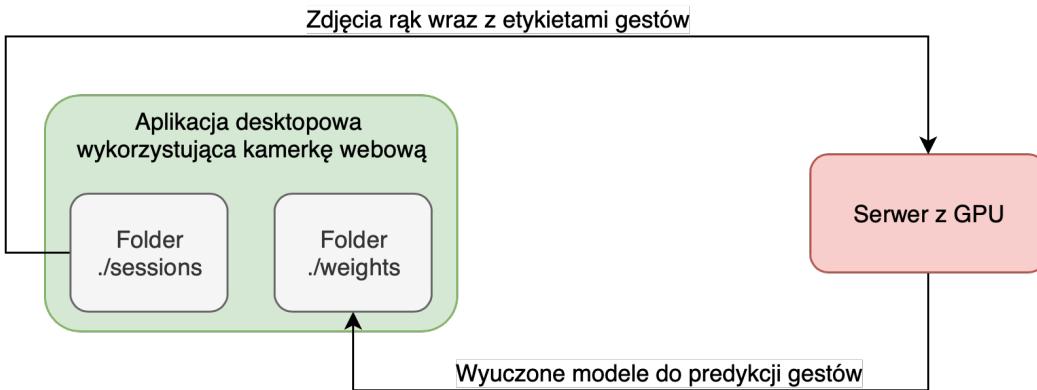
- Victoria – 'v'
- Otwarta dłoń – 'p' (palm)
- Trzy – 't'
- Pięć – 'f' (fist)

Ponadto aplikacja może nakładać na strumień binaryzacje z nałożoną maską tła. Aby wykonywać zdjęcia w trybie klasycznym należy wcisnąć klawisz '1'; aby wykonywać zdjęcia z maską - należy wcisnąć klawisz '8'.

- Tryb predykcji gestów – pod aplikację podpiętych jest kilka modeli predykcji. Podane poniżej cyfry to klawisze, które należy wcisnąć aby przejść do predykcji z wykorzystaniem danego modelu:
  - Sieć VGG z obrazem kolorowym na wejściu – '2'
  - Sieć VGG z obrazem binarnym na wejściu (naiwna binaryzacja opisana w punkcie 4.2) – '3'
  - Sieć CNN z obrazem kolorowym na wejściu – '4'
  - Sieć CNN z obrazem binarnym na wejściu (naiwna binaryzacja opisana w punkcie 4.2) – '5'
  - Sieć VGG z obrazem binarnym na wejściu pozyskiwanym z maski tła – '6'
  - **Ostateczna sieć CNN z obrazem binarnym na wejściu pozyskiwanym z maski tła – '7'**

W lewym górnym rogu aplikacji pokazywana jest etykieta obecnie rozpoznanego gestu.

W lewym dolnym rogu znajduje się informacja o obecnym trybie w jakim działa aplikacja. Część desktopowa nie była jedynym komponentem wykorzystywanym projekcie.



Rysunek 11: Architektura komponentów biorących udział w projekcie

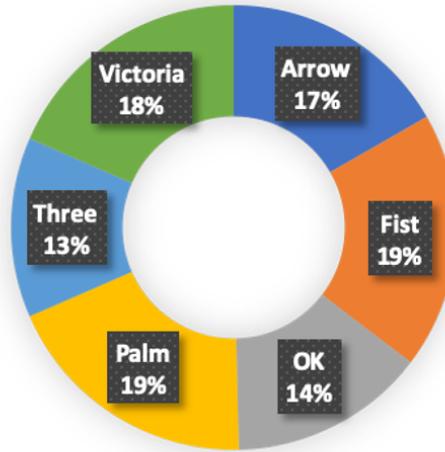
Na Rysunku 11. widać, że poza aplikacją czynnie wykorzystywany był serwer obliczeniowy wyposażony w jednostkę GPU. Pozwoliła ona na wykonanie ciężkich obliczeń potrzebnych do przygotowania modelu predykującego. Model ten powracał do aplikacji desktopowej, tak aby mogła ona funkcjonować jako detektor gestów.

## 4 Rozwiązania pośrednie

### 4.1 Klasyfikatory obrazów kolorowych (w oparciu o VGG-16)

Klasyfikator obrazów kolorowych zbudowano w oparciu o sieć VGG-16 z wagami sieci ImageNet. Jako metodę poolingu zastosowano AveragePooling. Ponad nią nadbudowano warstwę we pełni połączoną z 6 ma wyjściami oraz softmaxem jako funkcja aktywacji. Błędem uczenia była entropia krzyżowa. Optymalizatorem - ADAM. Zdjęcia kolorowe były klasyfikowane na 3 kanałach, w przypadku próby klasifikacji zdjęć binarnych powielano jeden kanał na 3. Tak zbudowana sieć pozwoliła

na bardzo dobre wyuczenie modelu. Zastosowano zbiór 18,280 własnych, kolorowych zdjęć, który podzielono w stosunku 70:30 na zbiór uczący i testowy. Na Rysunku 12. widać zbalansowanie zbioru.



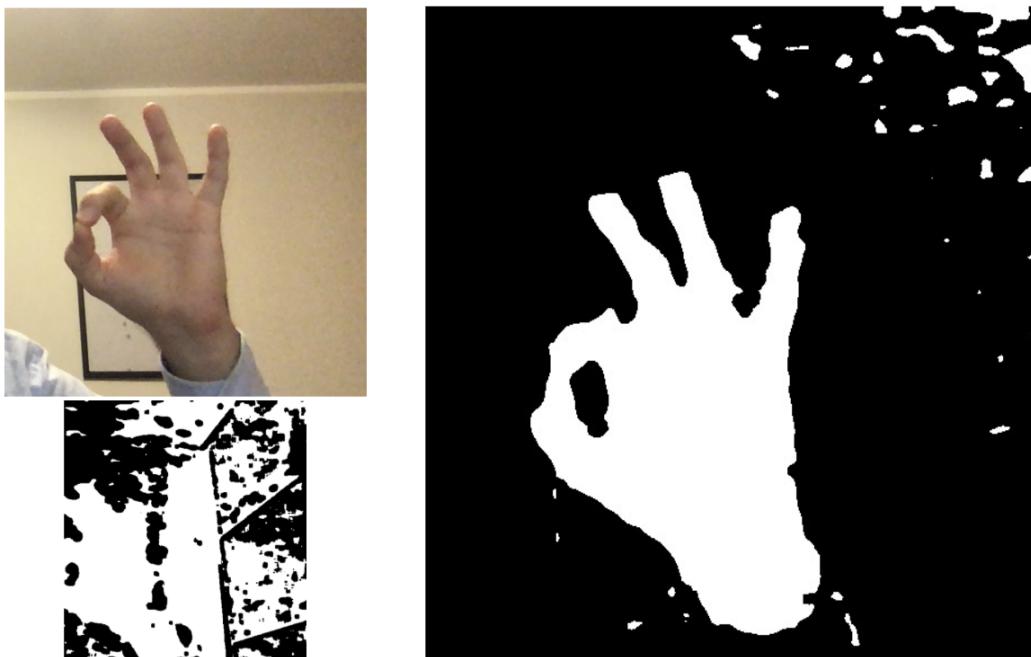
Rysunek 12: Zbalansowanie zbioru danych ze zdjęciami kolorowymi wykorzystywanego w początkowej fazie projektu

Jak widać w tabeli na Rysunku 13. metoda pozwoliła osiągnąć bardzo dobre wyniki klasyfikacji na zbiorze testowym, który miał taki sam charakter jak zbiór treningowy. Jednak w rzeczywistości (w warunkach testowych) model zachowywał się niemalże losowo. Można to wy tłumaczyć mało urozmaiconym tłem w zbiorze danych. Dane były zbierane jedynie w kilku miejscach. Próba urozmaicenia tła otoczenia nie przyniosła również pożądanych efektów. Można wysnuć wniosek, że przy odpowiednio dobrym zróżnicowaniu tła we wzorach uczących, sieć taką będzie mogła wdrożyć do systemów rozpoznawania gestów. W ramach niniejszego projektu nie udało się uzyskać odpowiedniego zróżnicowania danych, dlatego postanowiono skupić się na metodach z binaryzacją.

EPOCHS	OPTIM	BATCH	ACC_TRAIN	ACC_TEST	ERROR_TRAIN	ERROR_TEST
179	20	ADAM,0.001	16	95.496321	94.995707	0.185518
178	19	ADAM,0.001	16	95.202208	94.759452	0.191012
175	16	ADAM,0.001	16	94.751841	94.716495	0.210431
176	17	ADAM,0.001	16	94.880515	94.630587	0.202984
199	20	ADAM,0.001	32	94.880515	94.504309	0.208389
174	15	ADAM,0.001	16	94.549632	94.480240	0.218215
						0.220872

Rysunek 13: Osiągi klasyfikatora obrazów kolorowych działającego w oparciu o sieć VGG. Klasyfikator wyuczany był na ok. 12700 wzorach i testowany na ok 5400 obrazach

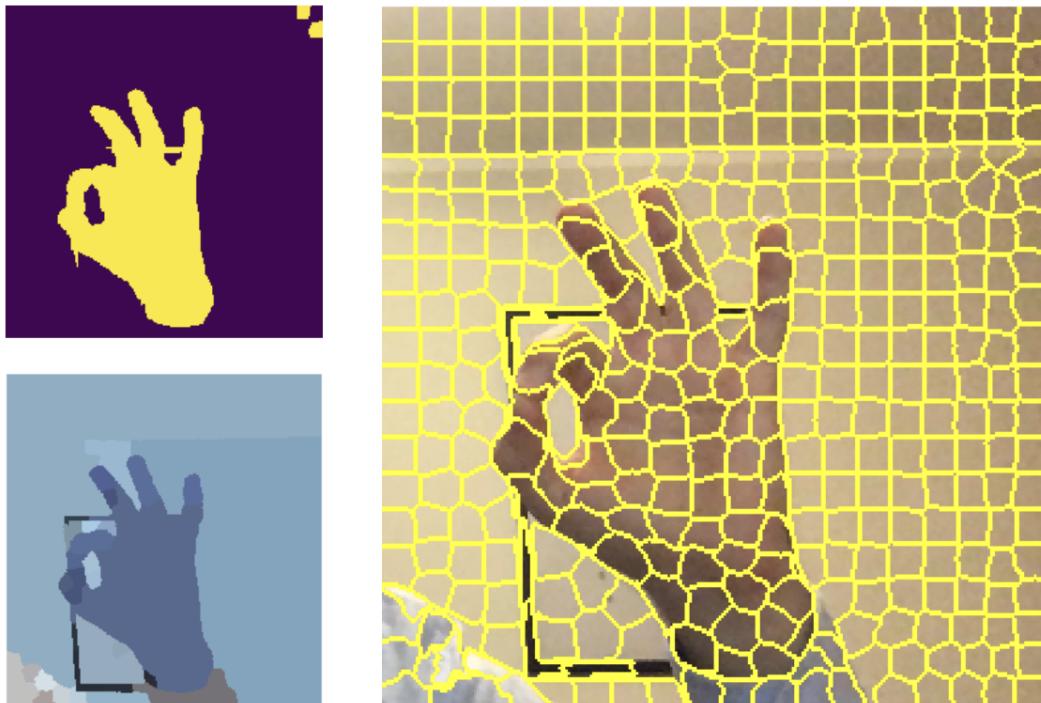
#### 4.2 Pierwsze klasyfikatory binarne. Opis pierwszej metody binaryzacji



Rysunek 14: Przedstawienie działania naiwnej binaryzacji w przestrzeni kolorów HSV.

Metoda opiera się na charakterystycznej dla dłoni barwie. W przestrzeni kolorów HSV można zauważyc, że często skóra znajduje się w przedziale pomiędzy [H: 0, S: 48, V: 80] i [H: 20, S: 255, V: 255]. Metoda ta była zastosowana w początkowych - najbardziej naiwnych wersjach modelu. Niestety – w większości przypadków binaryzacja spisywała się bardzo słabo. Nierazko zdarzało się, że tło zostało zaklasyfikowane jako ręka – to bardzo częste zjawisko w pokoju oświetlonym sztucznym światłem mającym białe lub żółte ściany.

#### 4.3 Segmentacja w oparciu o makropisekle

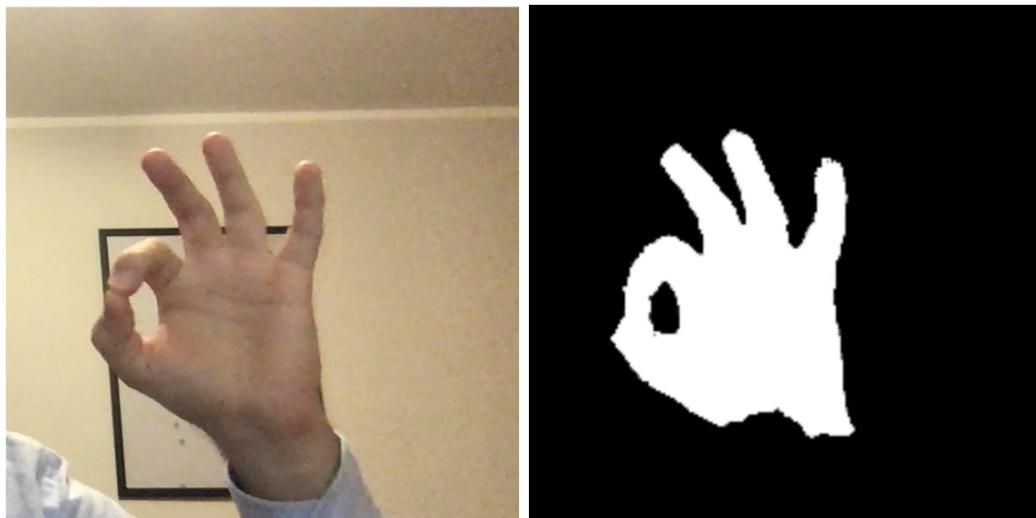


Rysunek 15: Przedstawienie działania segmentacji pikseli w kontekście segmentacji dloni.

Postawiono sprawdzić jak w zadaniu segmentacji dloni radzą sobie metody oparte o grupowanie pikseli. Niewielkim nakładem obliczeniowym wykonano nadsegmentację. Otrzymano wtedy rękę podzieloną na wiele mniejszych elementów (makropikseli). Następnie te elementy traktowano jako grupę i klasyfikowano ją w kategorii dloni – np. w oparciu o przestrzeń kolorów HSV (biorąc średni kolor dla segmentu). W rezultacie otrzymano model niewiele różniący się działaniem od klasycznego model HSV bez segmentacji. Przy dobrych warunkach oświetleniowych radził sobie bardzo dobrze, w warunkach gorszych - słabo.

Spróbowano też grupować segmenty hierarchicznie – jednak ta metoda nie pozwalała nigdy otrzymać ręki w jednym segmencie, a granice pomiędzy segmentami psuły obraz dając wyraźny kontur, który mylił sieć klasyfikującą.

#### 4.4 Segmentacja w oparciu o sieć U-Net



Rysunek 16: Przedstawienie działania naiwnej binaryzacji w przestrzeni kolorów HSV.

Zdecydowanie bardziej zaawansowaną metodą wydzielania dloni jest zastosowanie konwolucyjnej sieci do segmentacji o architekturze U-NET. Model ten otrzymuje dane do nauki: zdjęcia przedstawiające ręce oraz maski, czyli binarne obrazy, które pokazują w którym miejscu znajduje się ręka. Czas wyuczenia takiego modelu na GPU (GeForce RTX 2080) jest stosunkowo długi – zajmuje kilka godzin. Na wyjściu otrzymujemy model, do którego podajemy zdjęcia, a w odpowiedzi otrzymujemy maskę obrazującą, gdzie jest ręka. Rezultaty tego rozwiązania okazały się być bardzo dobre, jednak czas przetwarzania jednego zdjęcia wynosi ok 2 sekundy z wykorzystaniem GPU, więc to zdecydowanie za dugo.

W architekturze tej dostrzegamy duży potencjał. Jedyny problem stanowi wdrożenie jej w czasie rzeczywistym.

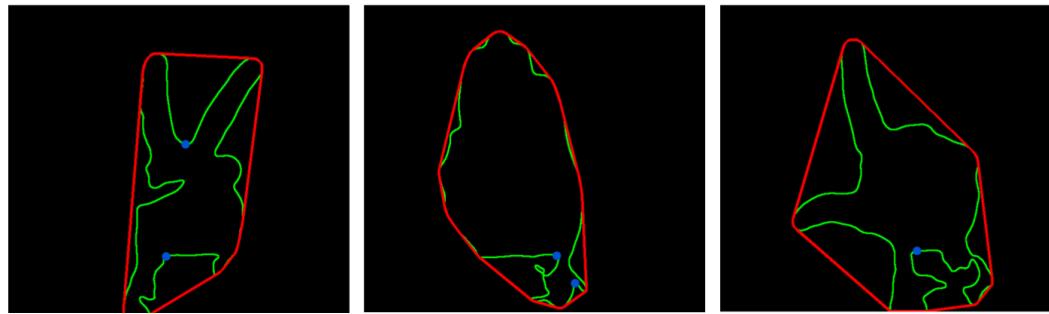
#### 4.5 Segmentacja w oparciu o klasyfikację pikseli



Rysunek 17: Przedstawienie działania klasyfikatora poszczególnych pikseli jako modelu wydzielającego dłoń

Klasyfikator opracowano z wykorzystaniem znalezioneego w sieci zbioru danych zawierającego informacje o pikselach wraz z przyporządkowaną binarną etykietą pokazującą, czy dany piksel ma być zaklasyfikowany jako kolor ręki. W zadaniu tym przetestowano prosty klasyfikator bayesowski (Naive Bayes) oraz sieć MLP, a następnie sprawdzono rezultaty. Okazało się, że metoda jest równie zadowodna jak metoda oparta o przedział dla przestrzeni barw HSV.

#### 4.6 Opis metody wykrywania koniuszków palców (Convex hull)



Rysunek 18: Prezentacja metody opartej o wykrywanie palców.

Metoda stosuje zupełnie inne podejście i nie służy do samego rozpoznawania gestów, lecz do wykrywania palców na zdjęciu. Jej działanie opiera się na rozpoznawaniu obszarów na zbinaryzowanym zdjęciu, które można zaklasyfikować jako palec.

Aby skorzystać z metody i otrzymać rezultaty jak na rys. 18 należy wykonać następujące operacje:

1. Wykonać maskę tła.
2. Na kolorowy obraz ręki nanieść lekkie rozmycie w celu złagodzenia krawędzi.

3. Wykonać binaryzację obrazu z wykorzystaniem maski tła.
4. Znaleźć wszystkie kontury obrazu i z pośród wszystkich wybrać jeden, który zakreśla największe pole – przy zastosowaniu maski tła to powinien być docelowy kontur ręki.
5. Zastosować metodę Convex Hull dla wybranego konturu. Jako rezultat zostanie zwrócony obszar otaczający rękę. Na rys. 18 taki obszar jest zaznaczony kolorem czerwonym.
6. Następnie z konturu ręki wyznaczyć punkty najbardziej oddalone od konturu Convex Hull. Dla każdego punktu obliczyć kąt, którego jest on wierzchołkiem a jego ramiona łączą się z punktami które przecinają kontur ręki z konturem Convex Hull. Jeśli otrzymana wartość nie przekracza zadanej wartości progowej to zakłada się, że łuk określa szczelinę pomiędzy palcami.
7. Na podstawie liczby szczelin pomiędzy palcami można określić całkowitą liczbę palców.

Niestety, mimo stosowania najlepszej metody binaryzacji – jaką w tym przypadku jest maska tła model miał wiele problemów z rozpoznawaniem palców. W tym przypadku głównym kluczem do sukcesu jest dobra binaryzacja ręki – metoda jest najbardziej czuła na źle zbinaryzowaną rękę, dzieje się tak, ponieważ wykonywane są tutaj operacje na konturach.

## 5 Opis wkładu własnego

Jako oryginalny wkład własny w zadaniu klasyfikacji gestów w strumieniu video upatruje się:

- Przebadanie szeregu metod segmentacji dloni ( w tym metod nowoczesnych - sieci U-Net)
- Wdrożenie efektywnej metody segmentacji dloni wraz z rozpoznaniem jej wag i konceptualnym opracowaniem jej ulepszeń.
- Stworzenie aplikacji umożliwiającej szybkie zbieranie danych i testowanie modeli w czasie rzeczywistym
- Opracowanie klasyfikatora neuronowego w oparciu o sieć CNN, którego architektura pozwala na osiągnięcie niemalże stuprocentowych wartości miary accuracy na zbiorze testowym (o podobnym charakterze do zbioru treningowego)
- Zebranie zasumionych zbiorów danych przybliżających rzeczywiste warunki testowe.

## 6 Badania rozwiązania

### 6.1 Cel badań. Co badano?

Przedstawione poniżej badania dotyczą modelu ostatecznego. Badano wpływ parametrów uczenia klasyfikatora oraz jakość działania przy wyuczaniu i testowaniu na różnych zbiorach. Poprzez zbiory te próbowało osiągnąć warunki testowe przybliżone do rzeczywistych. Opis zbiorów znajduje się w kolejnym podpunkcie. Jako miary jakości klasyfikacji obrano miarę ACCURACY i F-SCORE.

Brane pod uwagę parametry uczenia sieci:

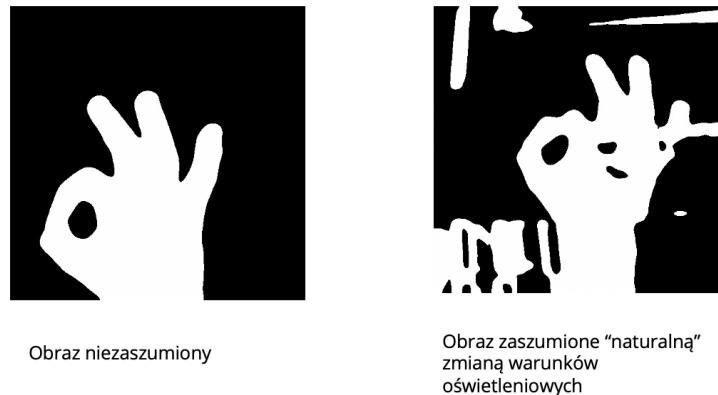
- liczba epok uczenia
- współczynnik uczenia, weight decay
- wpływ połączeń rezydualnych
- wpływ BatchNormalization
- wpływ dropoutu stosowanego w warstwach FullyConnected

### 6.2 Opis zbiorów uczących i testowych

Wszystkie badania prowadzono na własnych zbiorach uczących, które otrzymywano poprzez zastosowanie opisanej wyżej, własnej metody segmentacji/binaryzacji w aplikacji desktopowej działającej w trybie zbierania danych. Część z nich traktowano jako zbiory treningowe, część jako testowe, a część występowała w podwójnej roli. Kombinacja D1 i D2 (zbior treningowy, zbiór testowy) sprawdzała zdolność wyuczania klasyfikatora. Kombinacja D4 i D5 uznana została za najbardziej wiarygodną do odtwarzania warunków rzeczywistych. Opracowano następujące zbiory, którym nadano oznaczenia:

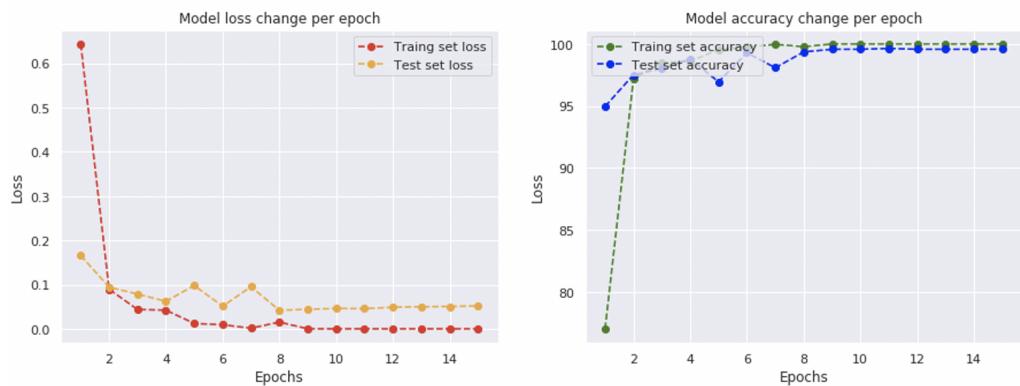
- ZBIÓR D1 -> niezaszumiony, ok. 3600 próbek
- ZBIÓR D2 -> niezaszumiony, ok. 1490 próbek
- ZBIÓR D3 -> zaszumiony, ok. 1250 próbek
- ZBIÓR D4 -> niezaszumiony i zaszumiony (w proporcji 50:50), ok. 2500 danych
- ZBIÓR D5 -> zaszumiony z sali zajęciowej, ok. 360 danych

Zaszumienie danych otrzymywano poprzez naturalnie występujące zmiany oświetlenia i zmiany tła (np. poprzez przesuwanie komputera). W ten sposób starano odtworzyć relistyczne warunki użytkowe. Przykład obrazu zaszumionego i niezaszumionego przedstawiony został na Rysunku 19.

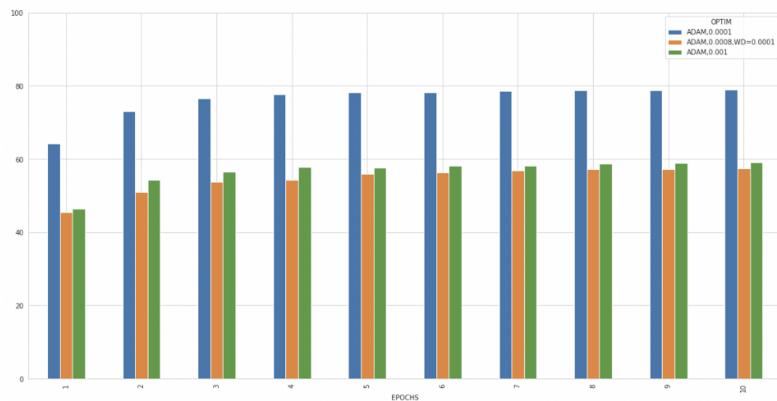


Rysunek 19: Przykłady obrazów zaszumionych i niezaszumionych wykorzytywanych w zbiorach uczących.

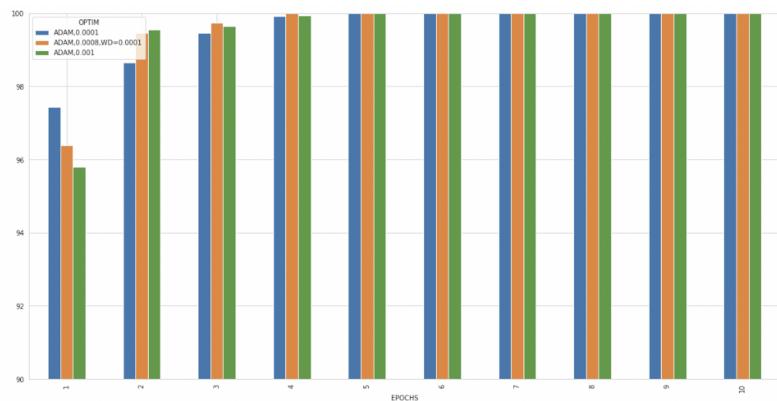
### 6.3 Badania wpływu parametrów klasyfikatora



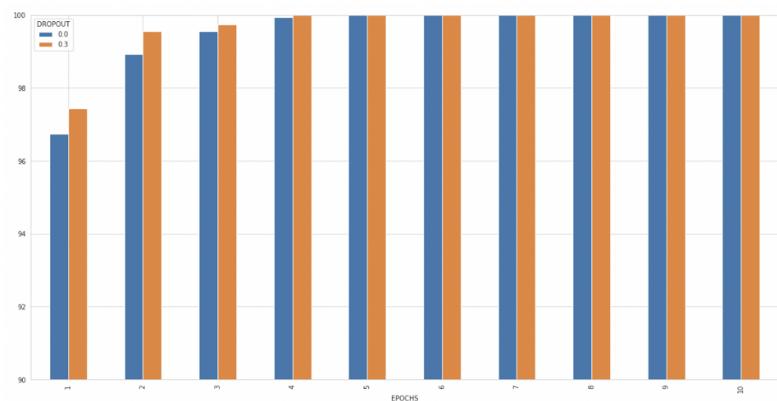
Rysunek 20: Proces uczenia modelu przy kombinacji (D1,D2). Po lewej wykres błędu, po prawej skuteczność predykcji na zbiorze treningowym i testowym.



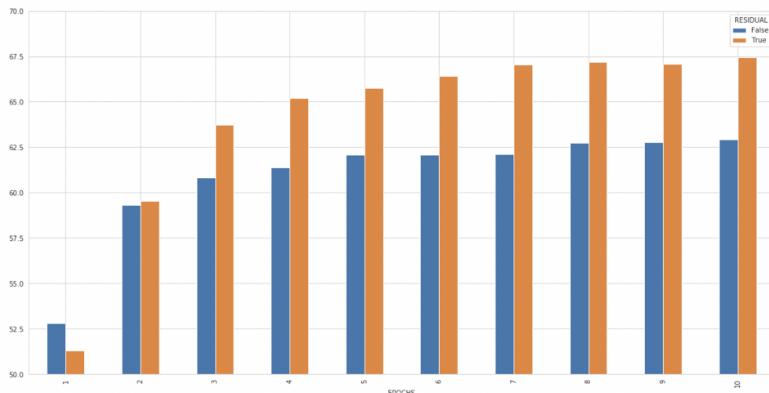
Rysunek 21: Wpływ optymalizatora (średnie wartości ACCURACY)



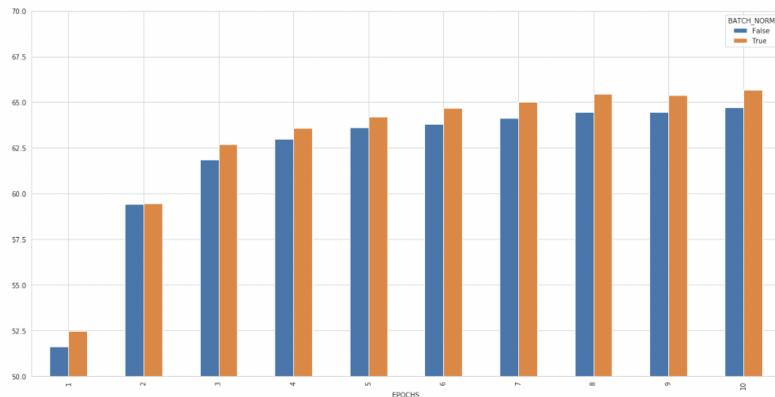
Rysunek 22: Wpływ optymalizatora (maksymalne wartości ACCURACY)



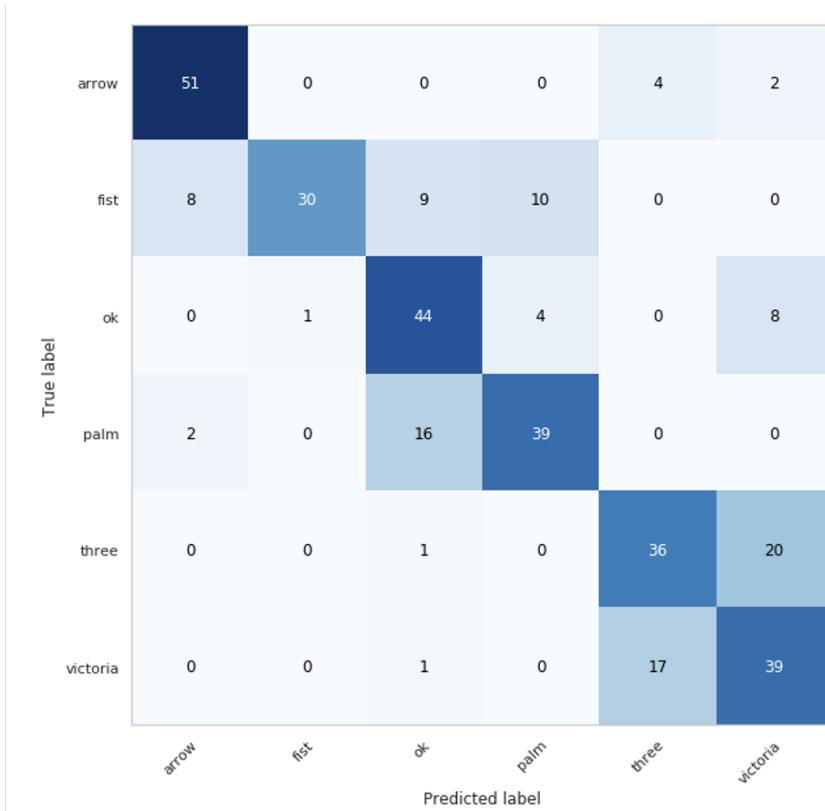
Rysunek 23: Wpływ dropoutu (maksymalne wartości ACCURACY)



Rysunek 24: Wpływ połączeń rezydualnych (średnie wartości ACCURACY)



Rysunek 25: Wpływ BatchNormalization (średnie wartości ACCURACY)



Rysunek 26: Macierz pomyłek dla zbiorów D4, D5

#### 6.4 Wyniki wyuczania na różnych zbiorach

ACCURACY				F-SCORE			
train/test	D2	D3	D5	train/test	D2	D3	D5
<b>D1</b>	99.502844	70.089287	74.374998	<b>D1</b>	0.984590	0.679030	0.741292
<b>D3</b>	74.573863	100.000000	75.937498	<b>D3</b>	0.684682	0.982276	0.674824
<b>D4</b>	100.000000	100.000000	84.375000	<b>D4</b>	0.988498	0.993930	0.779596

Rysunek 27: Najlepsze wyniki ACCURACY i F-SCORE przy różnych kombinacjach zbiorów treningowych i testowych.

#### 6.5 Komentarz do wyników

Model jest w stanie wyuczyć się bardzo dobrze na danych treningowych - walidowany na zbiorach o podobnym charakterze pozwala na uzyskanie wyników rzędu 99,5% ACCURACY i 98,5 % F-SCORE. Model gorzej radzi sobie natomiast z walidacją na zbiorach odzwierciedlających warunki rzeczywiste. Zakłada się, że najbliższe rzeczywistości warunki spefnią zbiór testowy D5. Osiągnięto na nim skuteczność rzędu 84,4% ACCURACY i 77,9% F-SCORE.

Najlepsze średnie wyniki osiągnięto dla optymalizatora ADAM bez Weight Decay, natomiast globalnie najlepsze - przy zastosowaniu tej techniki.

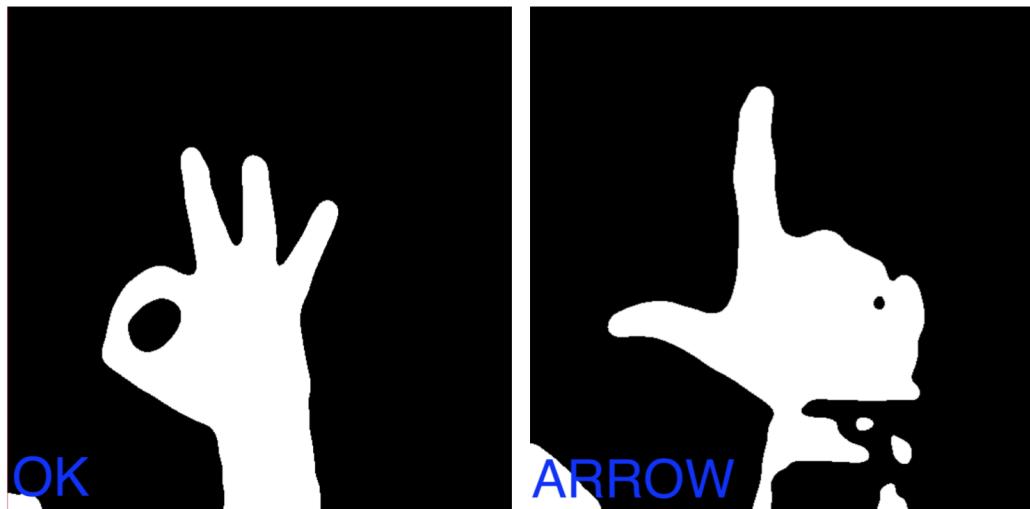
Rozwiązań takie jak dropout, połączenia rezydualne oraz Batch Normalization przyniosły poprawę jakości klasyfikacji.

W macierzy konfuzji widać, że mylone są ze sobą gesty najbardziej podobne do siebie.

## 7 Ocena końcowa

### 7.1 Przypadki dobrze klasyfikowane

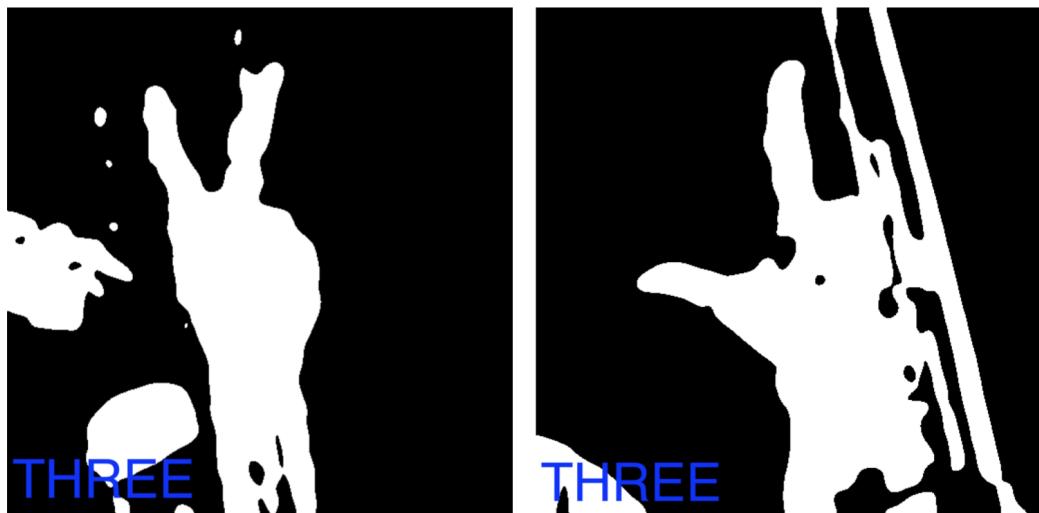
Jak widać na Rysunku 28, dla idealnie zbinaryzowanej ręki predykcja jest bezbłędna.



Rysunek 28: Przypadki dobrze sklasyfikowane.

### 7.2 Przypadki źle sklasyfikowane

Rysunek 29. przedstawia źle sklasyfikowane gesty. Główną przyczyną tego błędu jest zła binaryzacja ręki - powoduje ona to, że część otoczenia jest brana pod uwagę przy predykcji przykładowo model "widzi trzeci palec" w geście ARROW.

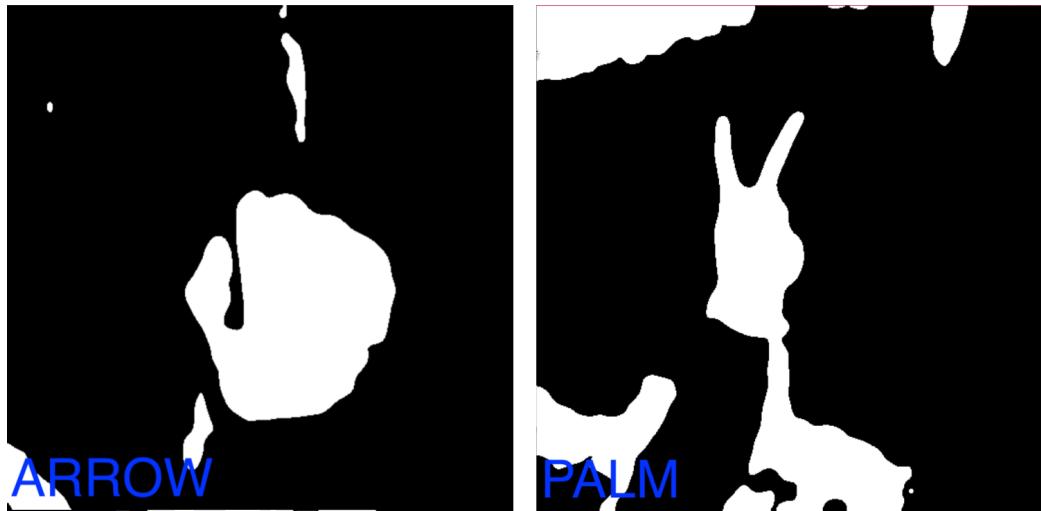


Rysunek 29: Przypadki źle sklasyfikowane – predykcja gestu jako "Trzy".



Rysunek 30: Przypadki źle sklasyfikowane – predykcja gestu jako "Trzy".

Podczas testów live w przypadku błędnej klasyfikacji najczęściej była zwracana etykieta dla gestu "Trzy". Gest ten ma najwięcej rozszczepionych palców. Możliwe, że szum pochodzący ze złej binaryzacji układa się w taki sposób że formuje się kolejny "sztuczny" palec który myli klasyfikator. Oczywiście nie tylko ten gest jest myloną – inne pomyłki przedstawiono na rys. 31.



Rysunek 31: Przypadki źle sklasyfikowane.

### 7.3 Źródła błędów

Wyuczony klasyfikator z ogromną skutecznością radzi sobie z predykcją, jeżeli gest dotyczy poprawnie zbinaryzowanego zdjęcia. Problem pojawia się w momencie kiedy binaryzacja zwraca szum, bądź defekty wynikające ze zmieniających się obszarów tła np. na skutek poruszenia kamerą lub komputerem. Binaryzacja to pierwsza transformacja jaką przechodzi obraz, aby odczytany został z niego gest – jeśli zakończy się ona słabym rezultatem, to trudno też spodziewać się bezbłędnej predykcji.

### 7.4 Pomysły na usprawnienia

Skoro sam model działa bardzo dobrze, to główne skupienie należy poświęcić wyodrębnieniu ręki. Przygotowano kilka pomysłów, jakie mogą zostać wykorzystane podczas dalszego rozwoju projektu:

- Przyspieszenie binaryzacji w oparciu o U-Net
- Śledzenie tła ręki, aby mogło być dynamiczne – rozwiąże to część problemów z maską tła, która zaczyna dawać złe efekty kiedy dowolny element tła się przemieści
- Przygotowanie bardziej zbalansowanego zbioru kolorowych zdjęć tak, aby wyuczone klasyfikatory zdjęć kolorowych uczyły się tylko gestów a nie skupiały na tle

### 7.5 Ocena konkurencyjności

Opracowany model wykazuje dużą skuteczność w niezmiennych warunkach oświetleniowych i przy statycznym ułożeniu kamerki webowej. Istnieje szereg rozwiązań, w których takie rozwiązanie znalazłyby komercyjną rację bytu. Prawdopodobnym zastosowaniem wydaje się wdrożenie modelu w samochodach osobowych, w których wnętrzu zmienność tła jest mała. Takie wykorzystanie wymagałoby jednak zaimplementowania mechanizmu automatycznego rekalibrowania tła (prawdopodobnie opracowania algorytmu), który wykrywałby jego zmianę (np. związaną z okresowymi zmianami oświetlenia) oraz zmieniałby automatycznie maskę służącą do segmentacji. Model musiałby charakteryzować się mniejszą podatnością na drgania i drobne ruchy. Ocenia się jednak, że wdrożenie takich mechanizmów jest możliwe, a sam model może być przez to konkurencyjny w branży motoryzacyjnej.

## 8 Opis załączników

Do raportu dołączono 2 płyty DVD. Na płycie DVD-1 znajdują się napisane programy, notebooki Jupyterowe z kodem uczenia modeli oraz część zbioru danych. Hierarchia folderów wygląda następująco:

- gesture\_detection\_app – główna aplikacja zbierająca dane oraz predykująca pokazywane gesty; aby uruchomić program należy zainstalować zależności z pliku requirements.txt a następnie uruchomić za pomocą pythona skrypt app.py
- fingers\_detection – program obrazujący rozpoznawanie dloni na podstawie koniuszków palców
- datasets – część zbioru danych, zawiera folder raw\_data gdzie znajdują się surowe dane, zawiera notebook data\_preparation.ipynb, który należy wykonać mając połączone wszystkie części datasetu – tworzy on zbiór danych treningowych i testowych
- notebooks – tutaj znajdują się notebooki, gdzie były przeprowadzane badania sprawdzające jakość modeli predykujących gest ręki
- single\_pixel\_as\_hand\_classifier – tutaj znajduje się notebook wraz z danymi uczącymi klasyfikator binarny określający czy piksel o danym kolorze ma być zaklasyfikowany jako skóra
- segmentation – notebook wraz z niezbędnymi danymi, który pokazuje działanie segmentacji w zadaniu binaryzacji ręki
- raport.pdf – raport z zajęć
- prezentacja.pdf – ostatnia prezentacja z zajęć

Na załączonej płycie DVD-2 znajduje się druga część danych, które są potrzebne do uruchomienia notebooków wykonujących testy. Jego zawartość należy wgrać do datasets/raw\_data, pochodzącym z płyty DVD-1.

## Literatura

- [1] H. Francke, J. Ruiz-del-Solar, R. Verschae, Real-Time Hand Gesture Detection and Recognition Using Boosted Classifiers and Active Learning, Springer, 2007
- [2] C Yu, X Wang, H Huang, J Shen, Vision-Based Hand Gesture Recognition Using Combinational Features, 2010,  
<https://ieeexplore.ieee.org/abstract/document/5635592> (styczeń 2020)
- [3] J Suarez, RR Murphy, Hand Gesture Recognition with Depth Images: A Review , 2011,  
<https://ieeexplore.ieee.org/document/6343787> (grudzień 2019)
- [4] SS Rautaray, A Agrawal, Vision based hand gesture recognition for human computer interaction: a survey , 2012  
<https://ieeexplore.ieee.org/document/6516326> (grudzień 2019)
- [5] F Dominio, M Donadeo, G Marin, P Zanuttigh , Hand Gesture Recognition with Depth Data, Springer, 2013
- [6] R Shrivastava, A Hidden Markov Model based Dynamic Hand Gesture Recognition System using OpenCV, 2013,  
<https://ieeexplore.ieee.org/document/6514354> (grudzień 2019)
- [7] F. Borbra, Tutorial: Using Deep Learning and CNNs to make a Hand Gesture recognition model 2019,  
<https://towardsdatascience.com/tutorial-using-deep-learning-and-cnns-to-make-a-hand-gesture-recognition-model-371770b63a51> (grudzień 2019)
- [8] V. Mühler, Simple Hand Gesture Recognition using OpenCV and JavaScript. 2017,  
<https://medium.com/@muehler.v/simple-hand-gesture-recognition-using-opencv-and-javascript-eb3d6ced28a0> (grudzień 2019)