POLITECHNIKA WARSZAWSKA

CYFROWE PRZETWARZANIE OBARZÓW

SPRAWOZDANIE

PROJEKT

Wykonał: Mikołaj Zbaraski 305231

Prowadzący: dr inż. Maria Cywińska

1. Wstęp teoretyczny

Rozpoznawanie gestów rąk za pomocą sieci neuronowych znajduje szerokie zastosowanie w interakcji człowiek – komputer, m.in. w systemach sterowania ruchem, rzeczywistości rozszerzonej czy komunikacji bezdotykowej. Kluczową rolę odgrywają tu konwolucyjne sieci neuronowe (CNN), które skutecznie analizują obrazy dłoni i identyfikują konkretne gesty. Aby zwiększyć efektywność uczenia i ograniczyć potrzebę dużych zbiorów danych, często stosuje się transfer learning. Polega on na wykorzystaniu wcześniej wytrenowanego modelu (np. na zbiorze ImageNet) i dostosowaniu go do nowego zadania. Dzięki temu proces treningu jest szybszy, a model osiąga lepsze wyniki nawet przy ograniczonych danych. Metoda ta znajduje zastosowanie w wielu nowoczesnych systemach rozpoznawania gestów.

2. Ścieżka przetwarzania

2.1. Środowisko

Środowisko *CPO_env* zostało przygotowane z myślą o projektach opartych na uczeniu maszynowym i przetwarzaniu obrazów. Zawiera m.in. biblioteki *fastai, torch, torchvision* oraz *mediapipe*, które umożliwiają budowę i trenowanie modeli głębokiego uczenia, w tym do rozpoznawania gestów. Wersja *Pythona 3.11* oraz obsługa GPU (*CUDA 12.6*) zapewniają nowoczesne i wydajne środowisko treningowe. Dodatkowo uwzględniono pakiety takie jak *opencv, pandas* i *kaggle*, co ułatwia pracę z obrazami, danymi tabelarycznymi oraz danymi z platformy Kaggle.

2.2. Dataset

W projekcie wykorzystano wcześniej zasugerowany zbiór danych Hand Gestures Dataset dostępny na platformie Kaggle: https://www.kaggle.com/datasets/ritikagiridhar/2000-hand-gestures. Zawiera on ponad 2000 zdjęć dłoni przypisanych do 8 różnych kategorii gestów.

Zdjęcia zostały zebrane w różnych warunkach oświetleniowych i z udziałem różnych osób, co zwiększa różnorodność zbioru i wspiera zdolność modelu do generalizacji.

Jednak na cele tego projektu został on zmodyfikowany z kilku powodów. Po pierwsze niektóre bardzo podobne gesty znajdowały się w dwóch różnych folderach np.:

fingerSymbols

fingerCircle

czy:

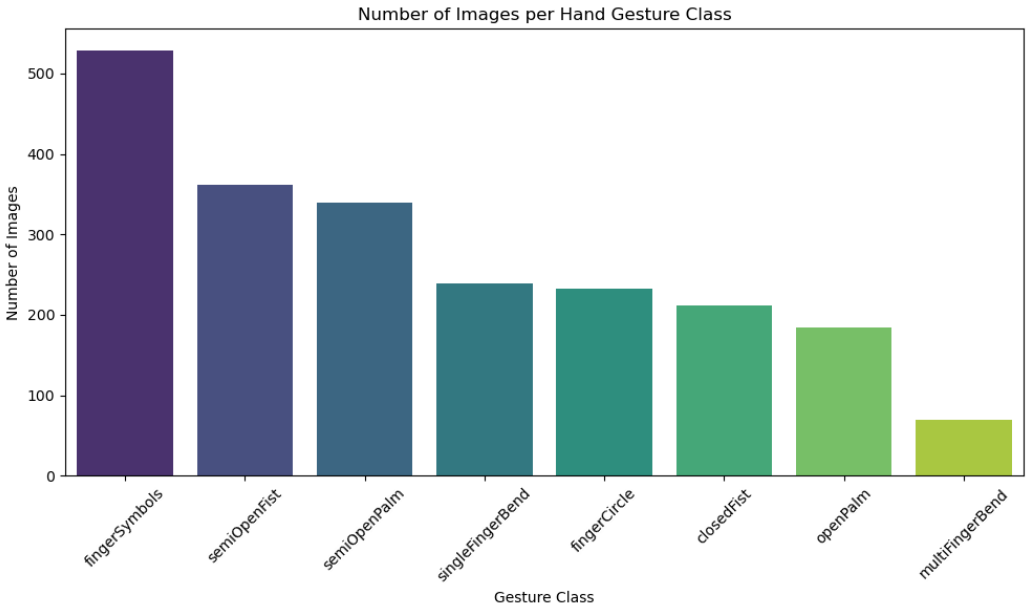
fingerSymbols

semiOpenFist

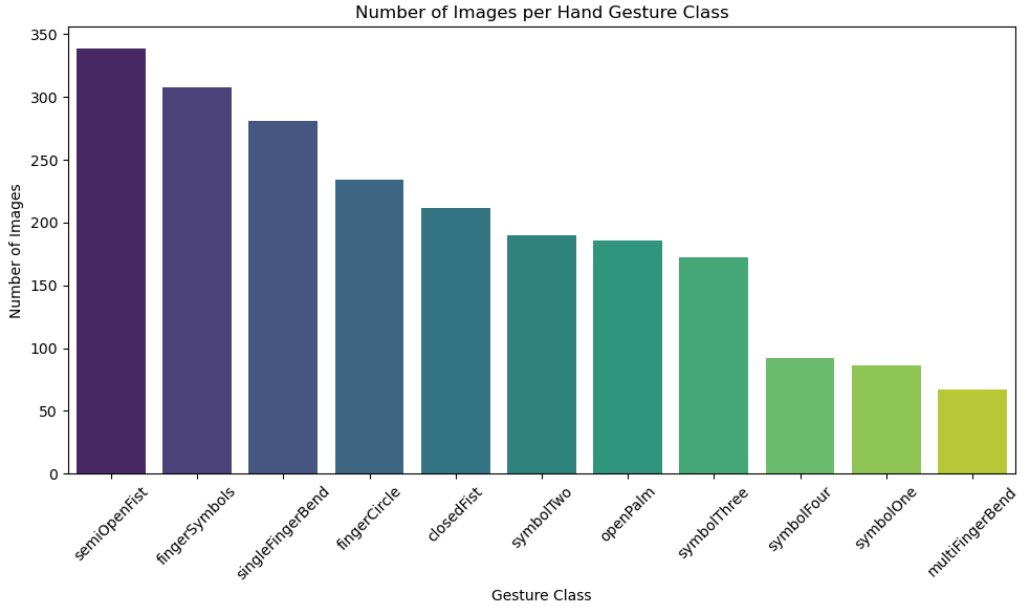
Kategorię semiOpenPalm zamieniono na cztery kategorie symbolOne, symbolTwo, symbolThree, symbolFour przedstawiające liczby 1-4 utworzone z palców poza kciukiem w kolejności od wskazującego do małego.

Wszystkie zdjęcia zostały na nowo ręcznie zaklasyfikowane do 11 kategorii w celu pozbycia się niejednoznacznie zaklasyfikowanych symboli w oryginalnym datasecie.

Pomogło to również w wyważeniu datasetu – najliczniejsza kategoria fingerSymbols zmniejszyła się z 529 zdjęć do 308 zdjęć, a najmniej liczna multiFingerBend pozostała prawie bez zmian. Niestety dataset nadal nie jest dobrze zrównoważony.



Oryginalny Dataset



Zmodyfikowany Dataset

2.3. Podział na dane treningowe i walidacyjne

Całość datasetu została losowo podzielona w stosunku 80:20 na dane treningowe i walidacyjne. Każda z kategorii została podzielona niezależnie w tym stosunku, aby uniknąć sytuacji gdzie niektóre kategorie są gorzej reprezentowane w jednym ze zbiorów.

Kategoria	Liczebność całej kategorii	Liczebność zbioru testowego	Liczebność zbioru walidacyjnego
closedFist	212	169	43
fingerCircle	234	187	47
fingerSymbols	308	246	62
multiFingerBend	67	53	14
openPalm	186	148	38
semiOpenFist	339	271	68
singleFingerBend	281	224	57
symbolFour	92	73	19
symbolOne	86	68	18
symbolThree	172	137	35
symbolTwo	190	152	38

2.4. Model

W celu znalezienia najlepszego modelu do użytego datasetu sprawdzono 4 modele: *ResNet18*, *ResNet34*, *ResNet50* oraz *DenseNet121*. Dla każdego z modeli przeprowadzono Fine Tuning z zamrożoną bazą przez 5 epok, a następnie One Cycle Policy z odmrożoną bazą przez kolejne 5 epok. Na koniec porównano wyniki dla badanych modeli.

Model	Macro F1 score	Błędne predykcje
ResNet18	0.6678	137
ResNet34	0.7100	118
ResNet50	0.5996	164
DenseNet121	0.7244	116

Na podstawie powyższych wyników podjęto decyzje o wykorzystaniu modelu **DenseNet121**.

DenseNet121 często sprawdza się lepiej niż ResNet18, 34 czy 50 przy zbiorach danych liczących około 2000 zdjęć, ponieważ jego architektura umożliwia efektywniejsze wykorzystanie informacji dzięki gęstym połączeniom między warstwami. Dzięki temu model uczy się bogatszych reprezentacji przy mniejszej liczbie parametrów, co zmniejsza ryzyko przeuczenia na ograniczonym zbiorze danych. Ponadto, DenseNet zapewnia bardziej stabilny i efektywny trening, co przekłada się na lepszą generalizację. Niestety skutkuje to dłuższym czasem uczenia, lecz w tym przypadku nie jest to duży problem – uczenie jednej warstwy zajmowało ok. 45s.

2.5. Preprocesing

W projekcie nie stosuje się rozbudowanego preprocesingu obrazów. Jedynym etapem wstępnego przetwarzania jest weryfikacja poprawności plików graficznych – sprawdzane jest, czy obrazy dają się poprawnie otworzyć oraz czy ich reprezentacje tensorowe zawierają wyłącznie skończone wartości (brak *NaN* i *inf*). Dzięki temu z danych usuwane są pliki uszkodzone lub potencjalnie problematyczne, co minimalizuje ryzyko błędów podczas trenowania modelu.

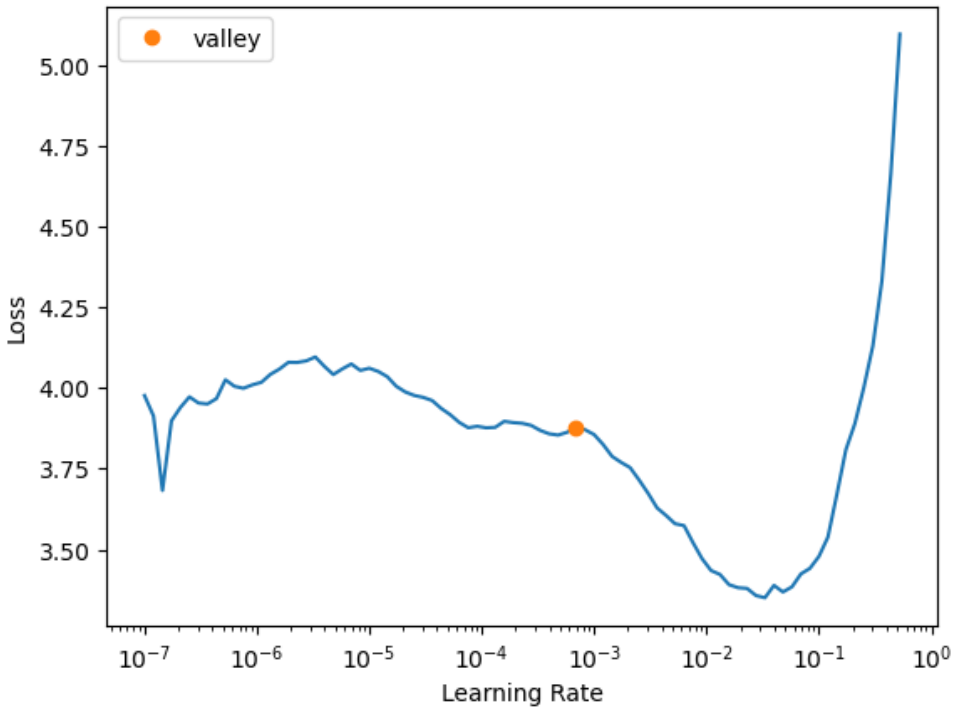
2.6. Augmentacje

W celu zwiększenia różnorodności danych treningowych i poprawy uogólnienia modelu zastosowano augmentacje czyli losowe przekształcenia nakładane na obrazy. augmentacje pomagają zapobiegać przeuczeniu i sprawiają, że model lepiej generalizuje do nowych, nieznanych danych — szczególnie ważne, gdy zbiór treningowy jest niewielki. Wykorzystana została funkcja *aug_transforms* z biblioteki *fastai*. Wykorzystane parametry z wyjaśnieniami:

- *mult=1.0* – Skala intensywności augmentacji. Domyślnie 1.0 oznacza pełną siłę zadanych transformacji.
- *do_flip=True* – Włącza losowe poziome odbicie lustrzane obrazów (symetria w poziomie). Przydatne w rozpoznawaniu obiektów, które mogą występować w różnych kierunkach.
- *flip_vert=False* – Wyłącza pionowe odbicie lustrzane.
- *max_rotate=20.0* – Maksymalny kąt obrotu obrazów (±20 stopni). Pomaga modelowi radzić sobie z lekkim obrotem danych wejściowych.
- *max_zoom=1.2* – Pozwala na losowe przybliżenie obrazów do 120% oryginalnego rozmiaru. Pomaga nauczyć model rozpoznawania obiektów w różnych skalach.
- *max_lighting=0.2* – Kontroluje zmiany jasności i kontrastu. Zwiększa odporność modelu na różne warunki oświetleniowe.
- *max_warp=0.2* – Stosuje niewielkie zniekształcenia perspektywy (ang. *warping*), które uczą model odporności na deformacje obrazu.
- *p_affine=0.75* – Prawdopodobieństwo zastosowania transformacji geometrycznych (obrót, zoom, warp itp.) – 75% szans, że będą zastosowane dla danego obrazu.
- *p_lighting=0.75* – Prawdopodobieństwo zastosowania zmian jasności i kontrastu – również 75%.

2.7. Uczenie

Przed rozpoczęciem pierwszego etapu uczenia generowany jest wykres zależności straty od prędkości uczenia w celu znalezienia najbardziej optymalnej wartości prędkości uczenia.

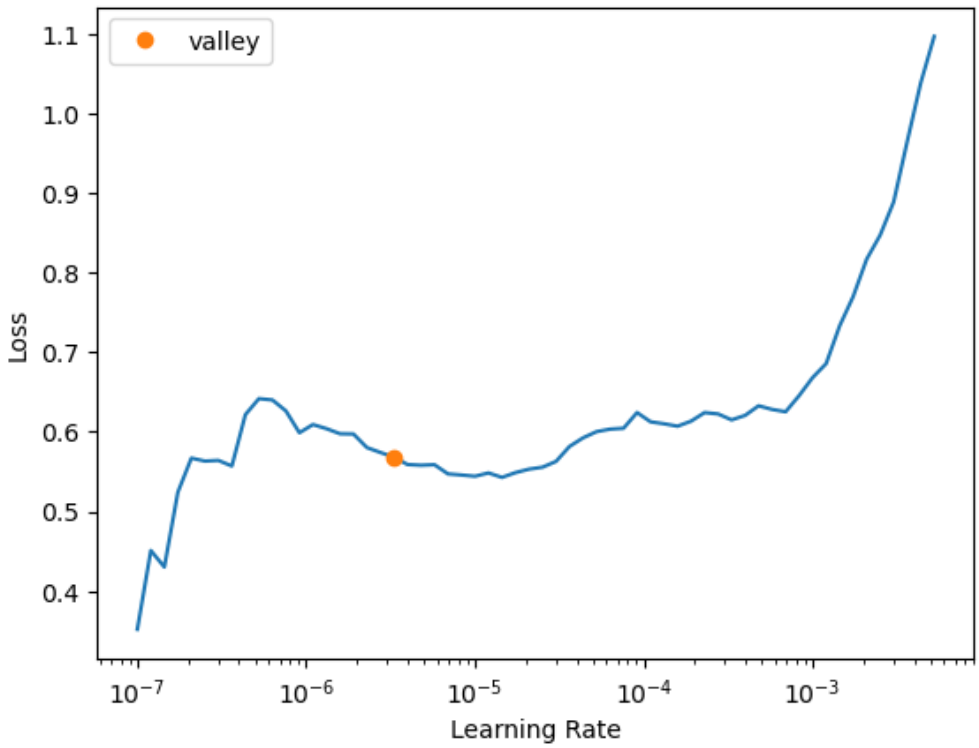


Na podstawie kilku prób ustalono prędkość uczenia na 10^{-3}.

W pierwszym etapie sieć uczona jest przez 8 epok z zamrożoną bazą. Zapobiega to przeuczaniu przy tak małym datasecie. Trenowane są tylko ostatnie warstwy (głowa modelu), które zostały dodane, aby dopasować model do nowego zadania.

Podczas treningu, po każdej epoce model jest zapisywany tylko wtedy, gdy osiągnie lepszy wynik (niższy *error rate*) niż we wcześniejszych epokach. Dzięki temu w kolejnych etapach można wykorzystać najlepszą wersję modelu uzyskaną podczas całego treningu. Mechanizm ten zabezpiecza również przed przeuczeniem – nawet jeśli model zacznie się pogarszać w kolejnych epokach, zachowana zostanie jego najbardziej efektywna wersja.

W drugim etapie baza zostaje odmrożona i ponownie generowany jest wykres zależności straty od prędkości uczenia.



Na tym etapie zastosowano strategię **One Cycle Policy** – technikę, która dynamicznie zmienia wartość współczynnika uczenia (*learning rate*) w trakcie treningu, co sprzyja szybszej i bardziej stabilnej konwergencji modelu. Wartość współczynnika uczenia była zmieniana w zakresie od 10^{-6} do 10^{-4}.

Model jest ponownie zapisywany za każdym razem, gdy osiąga lepszy wynik niż w poprzednich epokach. Proces uczenia trwa kolejnych 8 epok.

W obu etapach treningu zdecydowano się na 8 epok, ponieważ na podstawie wcześniejszych prób była to wartość zapewniająca optymalny balans między niskim *error rate* a ryzykiem przeuczenia. Dalsze zwiększanie liczby epok nie przynosiło istotnej poprawy, a mogło prowadzić do overfittingu.

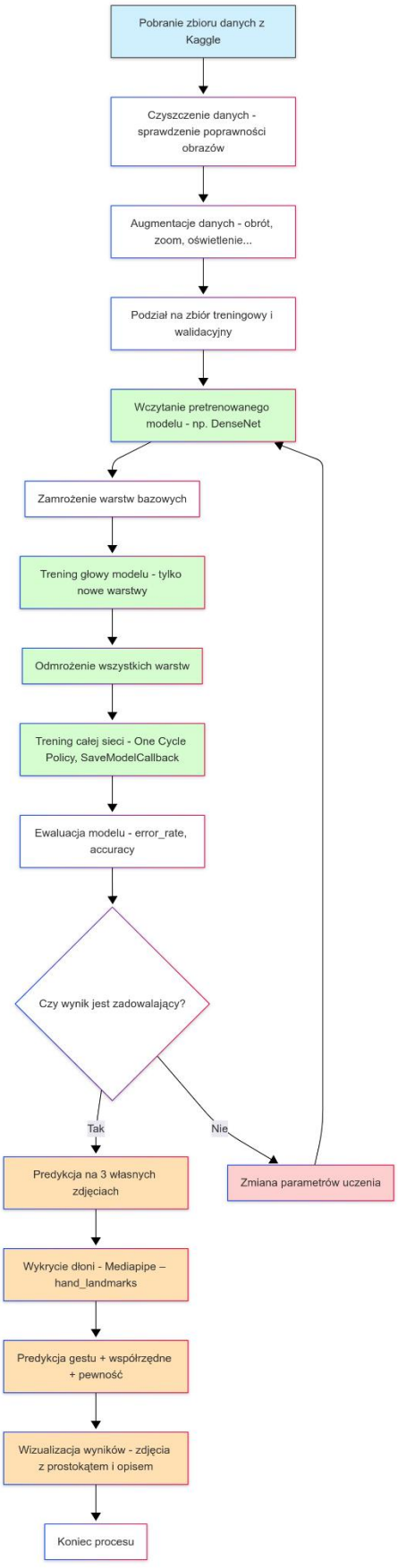
2.8. Przedstawienie wyników

Po wyszkoleniu sieci neuronowej przystąpiono do etapu predykcji, w którym wykorzystano wcześniej zarejestrowane trzy obrazy przedstawiające własne gesty użytkownika. Do klasyfikacji użyto najlepszego modelu zapisanego podczas treningu, czyli tego, który osiągnął najniższy *error rate*.

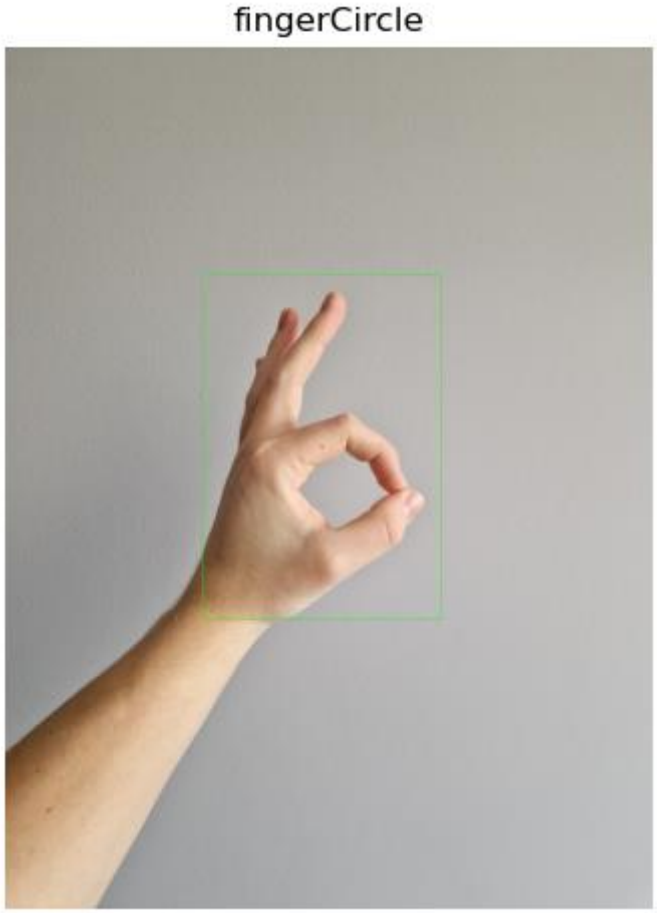
Program wyświetla te trzy obrazy wraz z przewidywanym gestem w formie podpisu, zaznaczonym prostokątem wokół wykrytej dłoni oraz środkiem dłoni oznaczonym na obrazie. Dodatkowo, pod każdym zdjęciem znajdują się współrzędne środka dłoni oraz wartość pewności modelu co do przewidzianej klasy gestu.

Do wykrywania położenia dłoni na obrazach wykorzystano bibliotekę *mediapipe*, a konkretnie moduł *hand_landmarks*, który umożliwia precyzyjne lokalizowanie charakterystycznych punktów dłoni.

3. Schemat blokowy



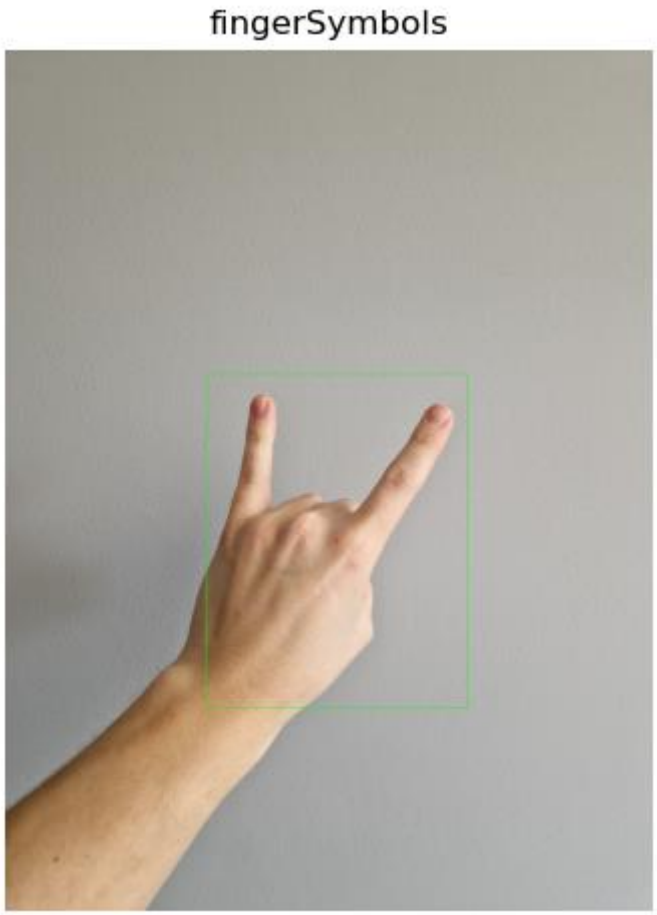
4. Wyniki



Center of hand: (1467, 1854)

Confidence: 0.9906

Gest został poprawnie zaklasyfikowany jako *fingerCircle*. Jest to jeden z prostszych do rozpoznania gestów, ponieważ wyróżnia się unikalną i łatwo zauważalną cechą – tworzeniem zamkniętego kółka za pomocą palców, co nie występuje w innych klasach gestów w zbiorze danych. Ta charakterystyczna forma ułatwia modelowi jednoznaczną identyfikację i rozróżnienie tego gestu od pozostałych. Dzięki temu model osiągnął bardzo wysoki poziom pewności predykcji, co świadczy o skuteczności jego rozpoznawania i dobrze wytrenowanej reprezentacji tej klasy w zbiorze treningowym. Ponadto, niski poziom wariancji w przykładach treningowych dla tego gestu dodatkowo zwiększa stabilność i dokładność klasyfikacji.

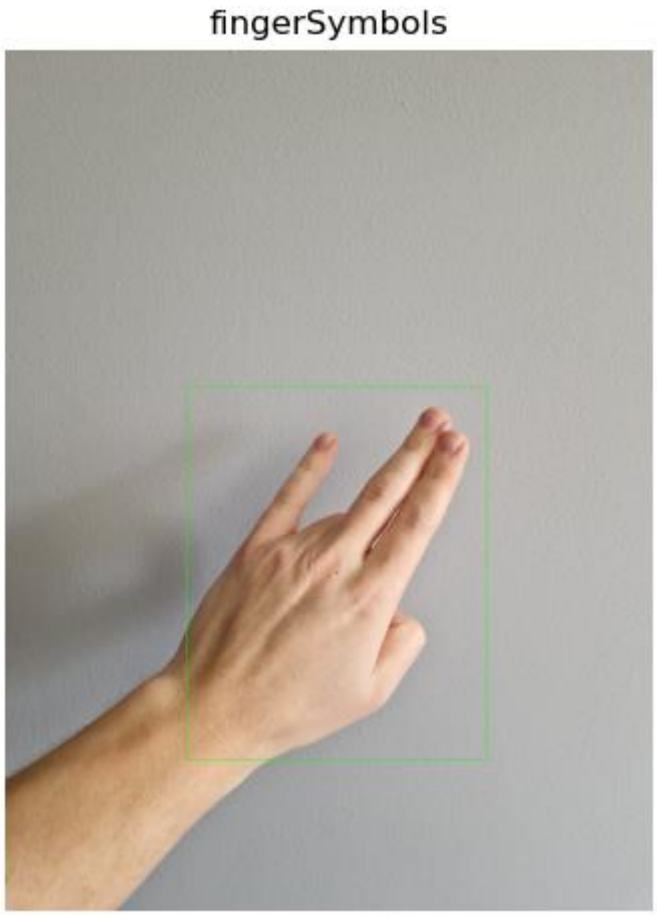


Center of hand: (1537, 2280)

Confidence: 0.7075

Gest został prawidłowo rozpoznany jako *fingerSymbols* z umiarkowanie wysokim poziomem pewności. Oznacza to, że model uchwycił kluczowe cechy tego gestu, choć nie całkowicie jednoznacznie – co może wynikać z różnic w ułożeniu palców lub jakości obrazów, które wpływają na niepewność predykcji. Mimo to, klasyfikacja jest trafna i świadczy o dobrej zdolności modelu do rozróżniania podobnych gestów.

Częste błędne przypisywanie tego gestu do klasy *semiOpenFist* wskazuje na wizualne podobieństwa między nimi, zwłaszcza częściowo zaciśnięte palce, co utrudnia jednoznaczną identyfikację. Problemy te mogą mieć związek z ograniczoną liczbą przykładów lub niewystarczającą różnorodnością danych treningowych. W celu poprawy rozróżniania warto zwiększyć ilość i jakość danych oraz zastosować bardziej zaawansowane techniki augmentacji czy ekstrakcji cech.



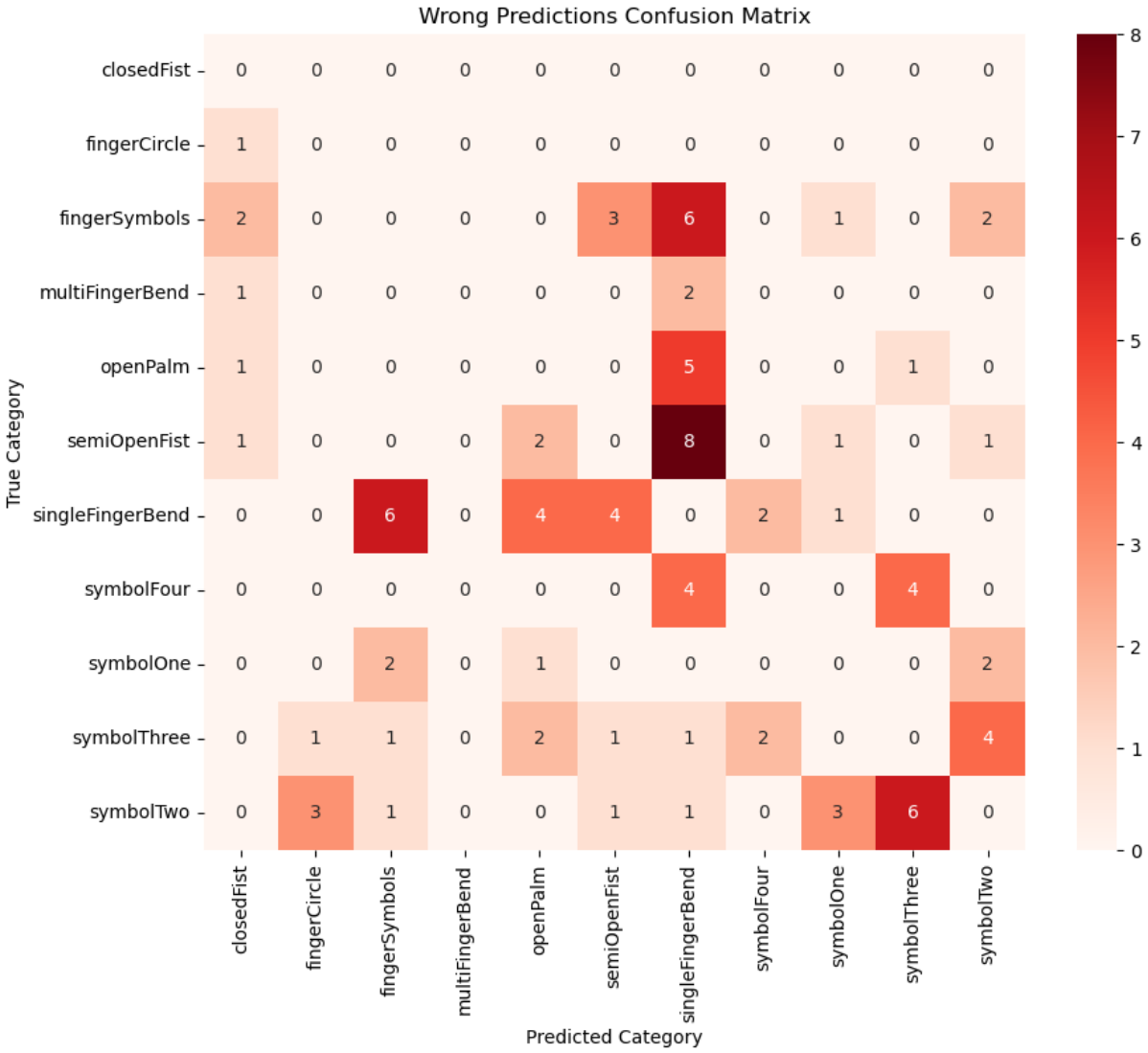
Center of hand: (1536, 2429)

Confidence: 0.7757

Na pierwszy rzut oka może się wydawać, że gest został błędnie sklasyfikowany jako *fingerSymbols*, podczas gdy powinien należeć do kategorii *singleFingerBend*. Jednakże, na zdjęciu widać, że kciuk jest zgięty, podczas gdy w definicji klasy *singleFingerBend* dokładnie jeden palec może być zgięty. W związku z tym obraz nie spełnia kryteriów tej kategorii. Oznacza to, że model prawidłowo wykluczył przypisanie gestu do *singleFingerBend* i zaklasyfikował go do klasy o cechach najbardziej zbliżonych do przedstawionego gestu.

5. Macierz pomyłek

Aby dokładnie przeanalizować skuteczność działania modelu oraz ocenić jakość wykorzystanego zbioru danych, stworzono macierz pomyłek (*confusion matrix*). Macierz ta przedstawia porównanie przewidywanych przez model kategorii z rzeczywistymi etykietami obrazów ze zbioru walidacyjnego. Dzięki temu możliwe jest zidentyfikowanie, które klasy są poprawnie rozpoznawane, a w których model popełnia najwięcej błędów. Taka analiza pozwala również ocenić, czy problemy z klasyfikacją wynikają z ograniczeń modelu, czy może z niedoskonałości lub nierównomiernego rozkładu danych w zbiorze treningowym. W efekcie macierz pomyłek stanowi cenne narzędzie do dalszej optymalizacji modelu oraz selekcji i ulepszania danych treningowych.



Macro F1 Score: 0.7737

Total wrong predictions: 95

Łatwo zauważyć, że model napotyka znaczne trudności z poprawną klasyfikacją gestów z kategorii *singleFingerBend*. Błędy związane z tą klasą stanowią aż 46% wszystkich pomyłek predykcyjnych, co wskazuje na istotne wyzwania w rozróżnianiu tych gestów. Przyczyną mogą być subtelne różnice w ułożeniu palców, które są trudne do uchwycenia przez model, zwłaszcza przy ograniczonej liczbie przykładów treningowych i niewystarczającej jakości danych.

Dodatkowo, warto zwrócić uwagę na częste błędne klasyfikacje gestów *symbolThree* i *symbolTwo*. Częściowo wynika to z faktu, że niektóre zdjęcia tych gestów zostały wykonane z boku dłoni, a nie z jej frontu, co znacząco utrudnia zarówno automatyczne rozpoznanie, jak i ocenę przez człowieka. W takich ujęciach palce często są częściowo zasłonięte lub tylko fragmentarycznie widoczne, co ogranicza dostępność kluczowych cech potrzebnych do jednoznacznej identyfikacji gestu. Ta sytuacja podkreśla potrzebę starannego doboru i standaryzacji danych treningowych, a także ewentualnego zastosowania bardziej zaawansowanych technik przetwarzania obrazu, które mogłyby pomóc w poprawie skuteczności klasyfikacji w trudnych warunkach.

Zdjęcia błędnie sklasyfikowane zostały skopiowane do folderu *wrong_predicts* i oznaczone nazwami o strukturze: *sklasyfikowanaKategoria_poprawnaKategoria_nazwa*. Takie rozwiązanie ułatwiło szczegółową analizę popełnionych błędów. Z obserwacji wynika, że największe problemy pojawiały się przy zdjęciach o niejednoznacznym kształcie, które mogły być interpretowane różnie nawet przez ludzi, co dodatkowo utrudnia ich jednoznaczną klasyfikację przez model.

6. Wnioski końcowe

Na podstawie przeprowadzonej analizy można wyciągnąć kilka kluczowych wniosków. Po pierwsze, wykorzystanie modelu *DenseNet121* okazało się trafnym wyborem dla zbioru danych liczącego około 2000 zdjęć, ze względu na jego zdolność do efektywnego wykorzystywania informacji oraz lepszą generalizację na niewielkich zbiorach. Model osiągnął dobre wyniki, jednak zauważalne były trudności z klasyfikacją niektórych gestów, zwłaszcza kategorii *singleFingerBend*, która odpowiadała za prawie połowę wszystkich błędów predykcji. Częste pomyłki między gestami o podobnych cechach, takimi jak *fingerSymbols* i *semiOpenFist*, wskazują na konieczność dalszego doskonalenia modelu oraz rozbudowy i zróżnicowania zbioru treningowego.

Dodatkowo, jakość i sposób wykonania zdjęć (np. gesty fotografowane z boku dłoni) miały istotny wpływ na skuteczność klasyfikacji, podkreślając potrzebę standaryzacji danych oraz ewentualnego zastosowania bardziej zaawansowanych technik przetwarzania obrazu. W przyszłości warto skoncentrować się na wzbogaceniu zbioru danych, precyzyjniejszym oznaczaniu gestów oraz optymalizacji procesu treningowego, co pozwoli zwiększyć dokładność i pewność predykcji modelu.