Projekt — Sieci neuronowe

Zastosowanie sieci neuronowej do oszacowania wieku osoby na podstawie zdjęcia dłoni prześwietlonej promieniami Röntgena

Skład grupy:

Michał Kordasz, 241289

Michał Bugno, 241357

Łukasz Siemański, 242416

Prowadzący:

Prof. dr hab. inż. Andrzej **Rusiecki**

2 lutego 2022

Spis treści

1 Opis problemu

Na podstawie zdjęcia dłoni prześwietlonej promieniami Röntgena możliwym jest (z pewnym przybliżeniem) oszacowanie wieku osoby, której dłoń została prześwietlona. W początkowej fazie rozwoju człowieka zarówno stawy między paliczkowe, jak i same kości palców są w dużej mierze tkanką chrząstną, która z upływem lat przekształca się w kość właściwą. Na podstawie oceny rozwoju kości i stawów (między-paliczkowych i śródręczno-paliczkowych) możliwym jest oszacowanie wieku dziecka. W zbiorze danych użytym przez nas mamy dostęp do zdjęć dłoni osób w wieku 0-19 lat [1].

2 Cel projektu

Celem projektu jest zaproponowanie koncepcji i stworzenie sieci neuronowej, która z możliwie najwyższą dokładnością oszacuje wiek dziecka na podstawie zdjęcia dłoni. Autorzy zestawu danych oświadczają, że nie udało im się oszacować wieku dziecka z dokładnością lepszą niż 4,2 miesiąca. Podrzędnym celem będzie próba dorównania z dokładnością do wyniku osiągniętego przez autorów, a być może i osiągniecie jeszcze lepszej dokładności.

3 Harmonogram

Proponuje się następujący harmonogram:

- 1. zajęcia - wstęp,
- 2. zajęcia - przedstawienie tematu,
- 4. zajęcia - preprocessing danych i wstępna koncepcja budowy sieci,
- 6. zajęcia - opracowany model sieci,
- 7. zajęcia - „ostatnie szlify” i poprawki,
- 8. zajęcia - Prezentacja wyników pracy wraz z jej omówieniem.

4 Dobór narzędzi

Całość zostanie przygotowana w języku Python, rozszerzonego o biblioteki:

- Keras [2],
- NumPy [3],
- Scikit-Learn [4].

Jako środowisko przewiduje się program PyCharm, wraz z repozytorium Github pod adresem `https://github.com/Sandalas98/SN-Projekt`. Dodatkowo do projektu przewiduje się oparcie w książce [6].

5 Realizacja

5.1 Preprocessing

Głównym problemem zdjęć jest etykieta z informacją, czy jest to lewa czy prawa ręka. Z punktu widzenia projektu jest to nie tyle zbędna rzecz, co niepożądana, ponieważ może negatywnie wpłynąć na wyniki uzyskiwane przez sieć. Każde zdjęcie posiada taką etykietę, z reguły jest ona umiejscowiona w rogu. Etykieta musi być poddana eliminacji.

Drugim problemem jest niedostateczna lub nadmierna jasność zdjęcia. Takie zdjęcia muszą być usunięte ze zbioru.

Preprocessing postępował według następującego algorytmu:

Przeskaluj zdjęcie do postaci czano-białej.
Przeskaluj do rozmiaru 512x512.
Oblicz średnią jasność pikseli (*mean_birghtness*).
Wyznacz jasność najjaśniejszego piksela (*max_birghtness*).
for each zdjęcia w zbiorze: **do**
  if *mean_birghtness* < 100 **AND** *mean_birghtness* > 60 **then** ▷ [1]
    for each piksela na zdjęciu **do**
      if *brightness* $x_{i,j} > 80\%$ *max_birghtness* **then** ▷ [2]
        funkcja_czyszczenia*($x_{i,j}$, mean_birghtness).*
      end if
    end for
  end if
end for
Zapisz obraz.

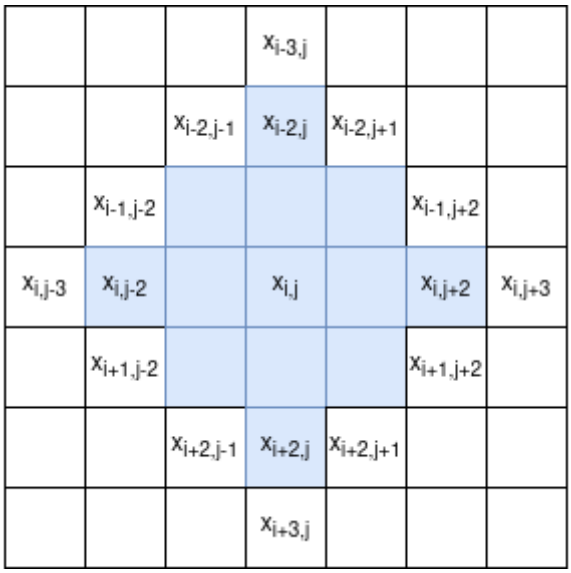
---

1. Sprawdzenie, czy zdjęcie odpowiednio jasne.

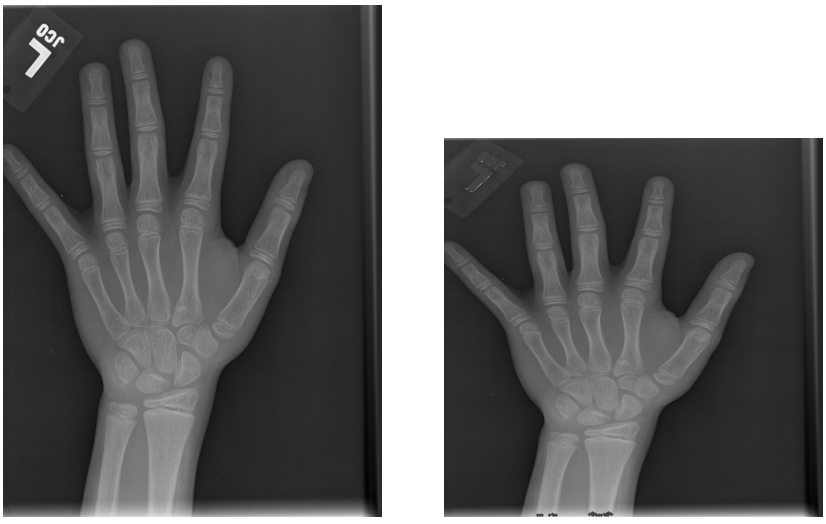
2. Sprawdzenie, czy bieżący piksel nie przekroczył progu jasności.

Głównym elementem preprocessingu jest: *funkcja_czyszczenia*, która odpowiada za zastąpienie piksela o dużej jasności oraz jego sąsiedztwa wartością średniej jasności. Gdy na zdjęciu (które jest macierzą wartości jasności) zostanie wykryty piksel, którego jasność

wynosiła co najmniej 80% jasności najjaśniejszego piksela (*max_birghtness*), to bieżącemu pikselowi i jego sąsiadom redukowano jasność. Proces ten dokładnie objaśnia poniższy obrazek (na niebiesko oznaczono piksele, których jasność ulegnie redukcji, $x_{i,j}$ — piksel, którego jasność przekroczyła próg jasności.



Poniżej przedstawiono rezultat działania preprocessingu:



Rysunek 1: „Przed” po lewej i „po” po prawej

W ten sposób przygotowano zdjęcia do nauki sieci.

5.2 Budowa sieci neuronowej i eksperymenty

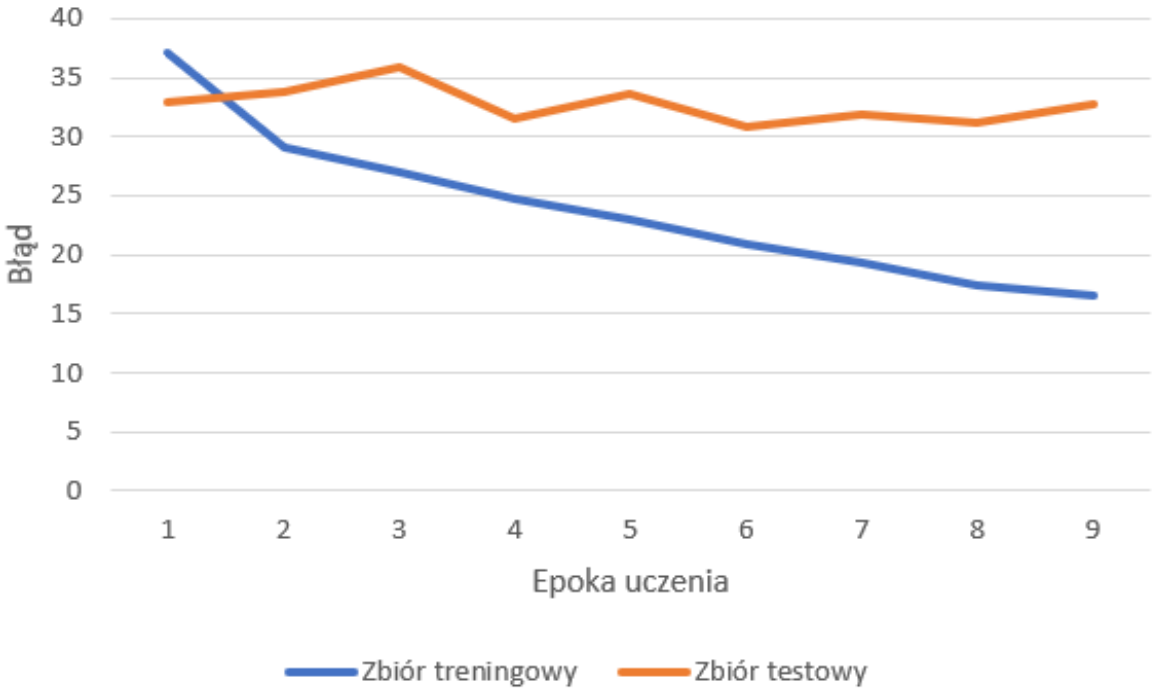
Przy budowie konwolucyjnej sieci neuronowej zastosowano różne strategie, sprawdzając eksperymentalnie, jakie wyniki dają proponowane architektury dla zbioru testowego. Nie prowadzono przeglądu zupełnego rozwiązań, ponieważ byłoby to bardzo czasochłonne. Zamiast tego kierowano się wynikami dla testowanej architektury i próbowano wprowadzić do niej usprawnienia. Eksperymenty zatrzymywano ręcznie, gdy zauważano brak poprawy. Jako funkcję straty wykorzystano średni błąd absolutny. Zbiór testowy składał się z 200 obrazów, zbiór treningowy z 10000. Każdy z obrazów był wcześniej przeskalowany i poddany procedurze preprocessingu. Wszystkie testowane architektury rozwiązań nie zostaną przedstawione, a jedynie te, które są warte odnotowania, a ich działanie i wyniki mogą prowadzić do ciekawych wniosków.

Ogólnie testowane architektury składały się z:

- warstwy wejściowej - dostosowanej do rozmiaru obrazu na wejściu
- warstwy konwolucji - z różnym rozmiarem filtra oraz różną liczbą stosowanych filtrów, które są wykorzystywane przez Tensorflow do uczenia sieci
- warstwy max pooling - z różnymi rozmiarami
- warstwy spłaszczającej
- warstwy ukrytej - z różną liczbą neuronów oraz funkcją aktywacji ReLU, próbowano również zastosować więcej warstw ukrytych
- warstwy wynikowej, składającej się z jednego neuronu

Najlepsza architektura sieci pozwoliła na uzyskanie wyniku 30.81 średniego błędu absolutnego w 5 epoce uczenia. Architektura ta składała się z:

- warstwy wejściowej dla obrazu o rozmiarze 512 x 512 pikseli
- jednej warstwy konwolucyjnej z 32 filtrami o rozmiarach 3x3
- warstwy max pooling o rozmiarze 3x3
- warstwy spłaszczającej
- warstwy ukrytej ze 128 neuronami i funkcją aktywacji ReLU
- warstwy wynikowej



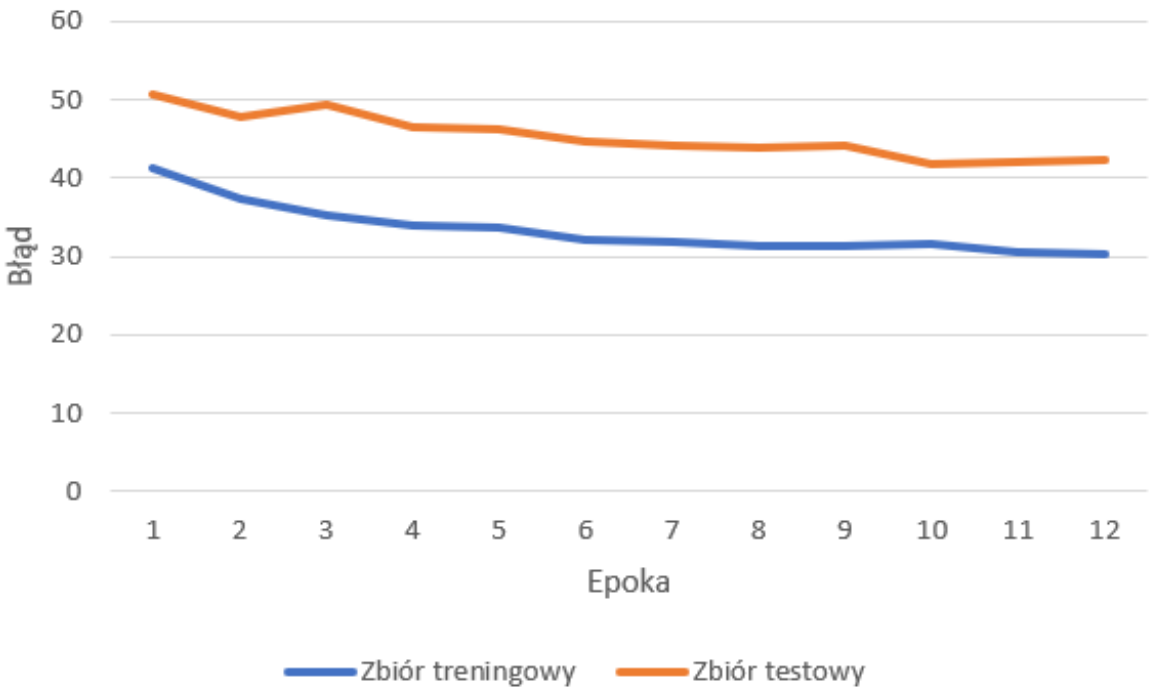
Rysunek 2: Błąd w kolejnych epokach uczenia dla najlepszej architektury

Rysunek 2 przedstawia błąd w kolejnych epokach uczenia dla zaproponowanej architektury. Widoczne jest, że błąd na zbiorze testowym nie zmienia się za bardzo w czasie i zawiera się w przedziale od 30 do 35. Błąd na zbiorze treningowym zmniejsza się w każdej

kolejnej epoce, co może prowadzić do wniosku, że następuje zjawisko przeuczenia sieci i zbytniego dostosowania jej do danych treningowych.

Kolejną ciekawą architekturą, którą warto przeanalizować, jest przypadek, gdy sieć nie posiada warstwy konwolucyjnej ani, co za tym idzie, max pooling. Architektura składała się z:

- warstwy wejściowej dla obrazu 512x512
- warstwy spłaszczającej
- warstwy ukrytej z 256 neuronami i funkcją aktywacji ReLU
- warstwy ukrytej z 100 neuronami i funkcją aktywacji ReLU
- warstwy wynikowej



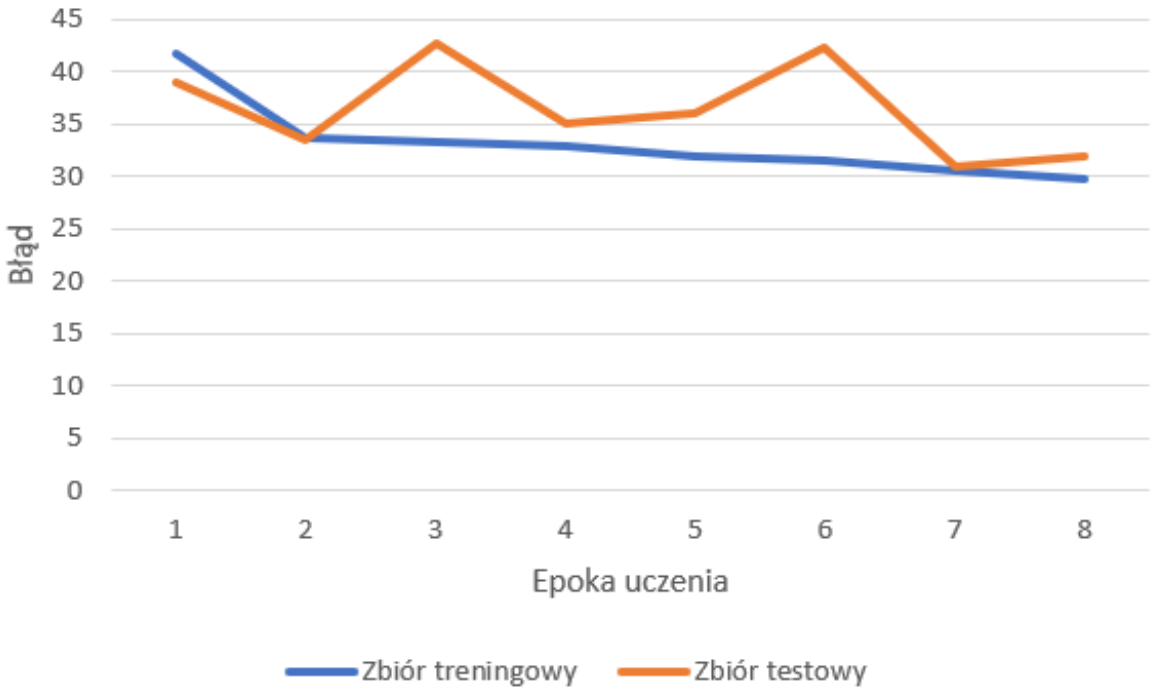
Rysunek 3: Błąd w kolejnych epokach uczenia dla architektury bez warstwy konwolucyjnej

Rysunek 3 przedstawia błąd w kolejnych epokach uczenia dla zaproponowanej architektury. Widać tutaj, że wynik jest znacznie słabszy w porównaniu do architektury z

warstwą konwolucyjną i max pooling. Świadczy to o tym, że te warstwy są konieczne w analizowanym problemie. Ekstrakcja cech, dokonywana przez warstwy konwolucyjną i max pooling, jest bardzo ważnym elementem dla sieci, której zadaniem jest przetwarzanie obrazu. Jednak i bez niej sieć jest w stanie się uczyć i uzyskać nie najgorsze wyniki.

Kolejnym ciekawym przypadkiem jest zastosowanie dwóch warstw konwolucyjnych o nieco większych filtrach oraz max pooling. Architektura składała się z:

- warstwy wejściowej dla obrazu 512x512
- warstwy konwolucyjnej z 4 filtrami o rozmiarach 8x8
- warstwy max pooling
- warstwy konwolucyjnej z 4 filtrami o rozmiarach 8x8
- warstwy max pooling
- warstwy spłaszczającej
- warstwy ukrytej z 100 neuronami i funkcją aktywacji ReLU
- warstwy wynikowej



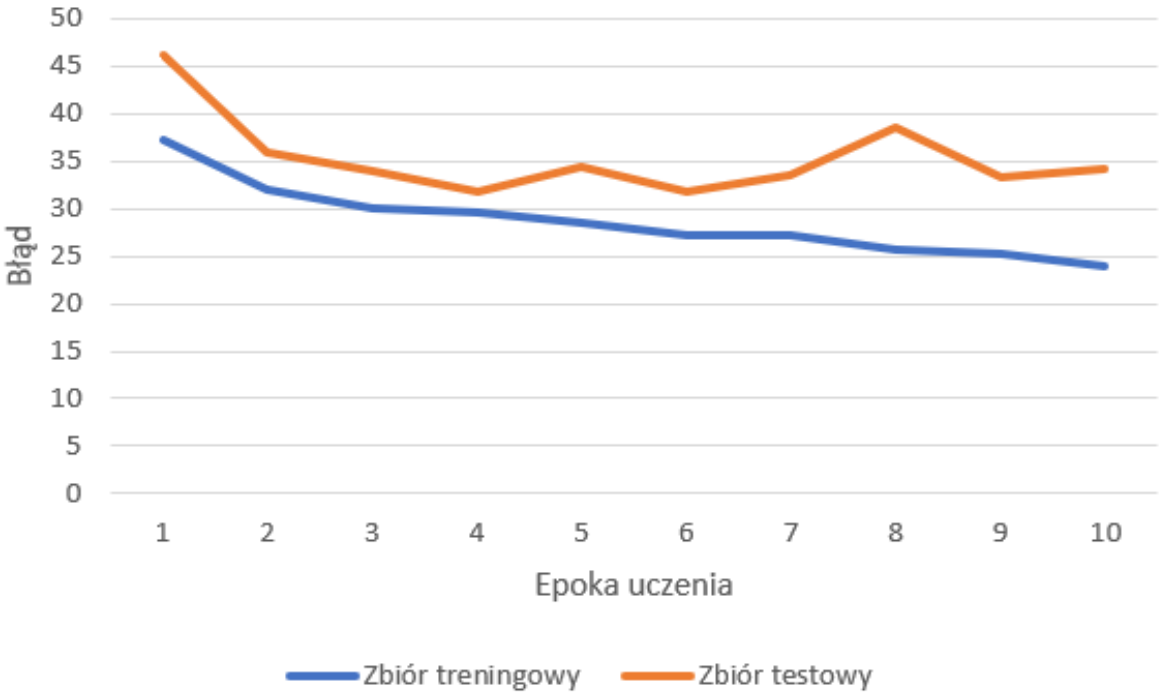
Rysunek 4: Błąd w kolejnych epokach uczenia dla architektury z 2 warstwami konwolucyjnymi

Rysunek 4 przedstawia błąd w kolejnych epokach uczenia dla zaproponowanej architektury. Widoczne są znacznie większe wahania w wynikach dla zbioru testowego - może to świadczyć o tym, że sieć nie jest w tym przypadku przeuczona, a jednocześnie podatna na wszelkie zmiany w parametrach, występujące w kolejnych epokach. Same wyniki są nieco niższe od rezultatów dla najlepszej architektury.

Sprawdzono też, co stanie się, gdy obraz wejściowy zostanie zmniejszony do rozmiaru 128x128 pikseli i podany do odpowiednio przygotowanej sieci, o następującej architekturze:

- warstwa wejściowa dla obrazu 128x128
- warstwa konwolucyjna z 32 filtrami o rozmiarach 3x3
- warstwa max pooling
- warstwa spłaszczająca

- warstwa ukryta z 128 neuronami i funkcją aktywacji ReLU
- warstwa wynikowa



Rysunek 5: Błąd w kolejnych epokach uczenia dla architektury z 2 warstwami konwolucyjnymi

Rysunek 5 przedstawia błąd w kolejnych epokach uczenia dla zaproponowanej architektury. Widać duże wahania w wynikach na zbiorze testowym, ale niektóre z nich są porównywalne z najlepszą architekturą (w 4 epoce uczenia - 31,84). Jest to zaskoczenie, ponieważ spodziewano się utracić część informacji po przeskalowaniu obrazu wejściowego do rozmiaru 128x128, tymczasem sieć poradziła sobie z obrazami w bardzo dobry sposób. Należy tutaj też zauważyć, jak takie zmniejszenie obrazu wpływa na mniejszy czas uczenia się sieci. To, że sieć uczy się szybciej i jednocześnie nie traci przy tym skuteczności, w porównaniu do obrazów o większym rozmiarze, jest ważnym wnioskiem płynącym z eksperymentów.

6 Podsumowanie i wnioski

Główny cel projektu, jakim było zaproponowanie koncepcji i stworzenie sieci neuronowej, która z możliwie najwyższą dokładnością oszacuje wiek dziecka na podstawie zdjęcia dłoni, został zrealizowany. Najlepsza architektura sieci uzyskała średni błąd absolutny 30,81 na zbiorze testowym. Z realizacji projektu można wyciągnąć następujące wnioski:

- Zbiory treningowy oraz testowy posiadały zdjęcia bardzo różnej jakości, w różnych rozmiarach, przedstawiające zarówno lewą, jak i prawą dłoń. Z całą pewnością utrudniło to proces uczenia oraz wpłynęło negatywnie na wyniki uzyskiwane przez sieć, zgodnie z zasadą Garbage In Garbage Out. Z całą pewnością, gdyby analizowane zdjęcia były w jakiś sposób ustandaryzowane, możliwe byłoby uzyskanie lepszych wyników, co dałoby również realizowanemu projektowi większą wartość w jego zastosowaniu praktycznym.

- Przed użyciem zdjęć ze zbiorów treningowego oraz testowego, konieczne było poddanie ich procesowi preprocessingu oraz odrzucenie części z nich. Zaproponowany algorytm pozwolił na usunięcie zbędnych elementów ze zdjęć, takich jak oznaczenia, która dłoń znajduje się na zdjęciu, a także na uwypuklenie ważnych cech samej dłoni.

- Największym problemem ze zbiorem danych mogły być różne rozmiary zdjęć. Konieczne było, żeby wszystkie zdjęcia podawane na wejście sieci były tego samego rozmiaru, co wymagało jego dostosowania. Mogło to wpłynąć na utratę pewnych informacji, które okazywały się ważne w uczeniu sieci i rozpoznawaniu wieku na podstawie zdjęcia. Również jakość niektórych zdjęć miała duży wpływ na wyniki, co było widoczne dla zbioru testowego, gdzie dla zdjęć "dobrze wykonanych" uzyskiwano znacznie lepsze wyniki, niż dla zdjęć, które miały różne odchylenia, jak zbyt duża jasność.

- Możliwe, że zastosowany sposób preprocessingu nie był odpowiedni dla rozwiązywanego problemu. Jego głównym celem było usunięcie ze zdjęć tabliczek z informacją o tym, czy jest to prawa czy lewa ręka i na tym się skupiono. Dodatkowe poddanie zdjęć innym procesom obróbki, choćby przez nałożenie prostych filtrów, dostępnych w każdym programie graficznym, mogłoby pomóc w uwypukleniu istotnych

cech dla przetwarzanych zdjęć. Ciekawym przykładem jest tutaj praca [5], w której analizowano jak zastosowanie różnych metod graficznych wpłynie na wyniki sieci, klasyfikującej uszkodzone ziemniaki.

- Najlepszy wynik uzyskała sieć o architekturze warstwy wejściowej dla obrazu 512x512 pikseli, warstwy konwolucyjnej z 32 filtrami o rozmiarach 3x3, warstwy max pooling o rozmiarze 3x3, warstwy spłaszczającej, warstwy ukrytej ze 128 neuronami i funkcją aktywacji ReLU oraz warstwy wynikowej. Jest to standardowa architektura dla sieci konwolucyjnej i nie było tutaj zaskoczeń.

- Sieć o podobnej architekturze, wykorzystująca obrazy przeskalowane do rozmiaru 128x128 pikseli uzyskała niewiele gorsze wyniki od analogicznej sieci przetwarzającej zdjęcia w rozmiarze 512x512 pikseli, jednocześnie znacznie przyspieszając czas potrzebny na uczenie. Widoczna jest jednak utrata informacji, wynikająca z przeskalowania obrazu. Wybór lepszej z tych dwóch architektur zależy od tego, czy gorszy wynik może zostać zrekompensowany przez to, że czas uczenia sieci jest niższy. W przypadku tego projektu ważniejsze było uzyskanie lepszego wyniku dla zbioru testowego.

- Architektura sieci, w której nie zastosowano warstwy konwolucyjnej oraz max pooling uzyskała wyraźnie gorsze wyniki od sieci, które wymienione warstwy posiadały. Ekstrakcja cech, która jest przez nie dokonywana, jest konieczna dla uzyskania przez sieć lepszych wyników.

- Architektura z większą liczbą warstw konwolucyjnych również może być skuteczna, jednak wymaga ona większej liczby parametrów, a co za tym idzie - zwiększenia czasu uczenia sieci. Zasadność użycia takiej sieci może być również zależna od problemu, który jest rozwiązywany.

7 Projekt GitHub

Cały projekt znajduje się na platformie GitHub, pod adresem:
www.github.com/Sandalas98/SN-Projekt

8 Bibliografia

Literatura

[1] Baza danych. `https://www.kaggle.com/kmader/rsna-bone-age`.

[2] Keras - strona projektu. `https://keras.io/guides/`.

[3] Numpy - strona projektu. `https://numpy.org/`.

[4] Scikit-learn - strona projektu. `https://scikit-learn.org/stable/`.

[5] Stanislav Suvorov Aleksey Osipov, Andrey Filimonov. Applying machine learning techniques to identify damaged potatoes. 2021. [Dostęp: 02-02-2022].

[6] A. Geron. *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow.* 2020.