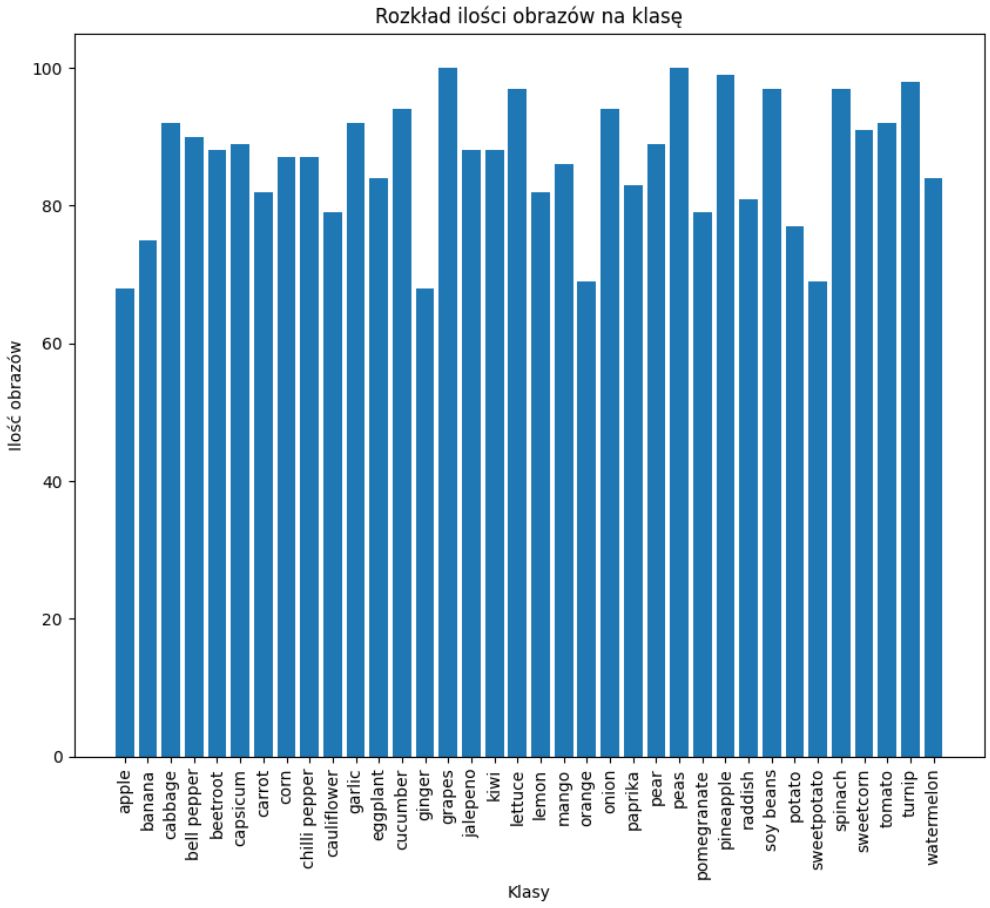
Projekt – sieci neuronowe	Data złożenia projektu: 02-02-2024
Numer grupy projektowej:	Imię i nazwisko I: Michał Rokita Imię i nazwisko II: Filip Sikora

Rozpoznawanie warzyw i owoców

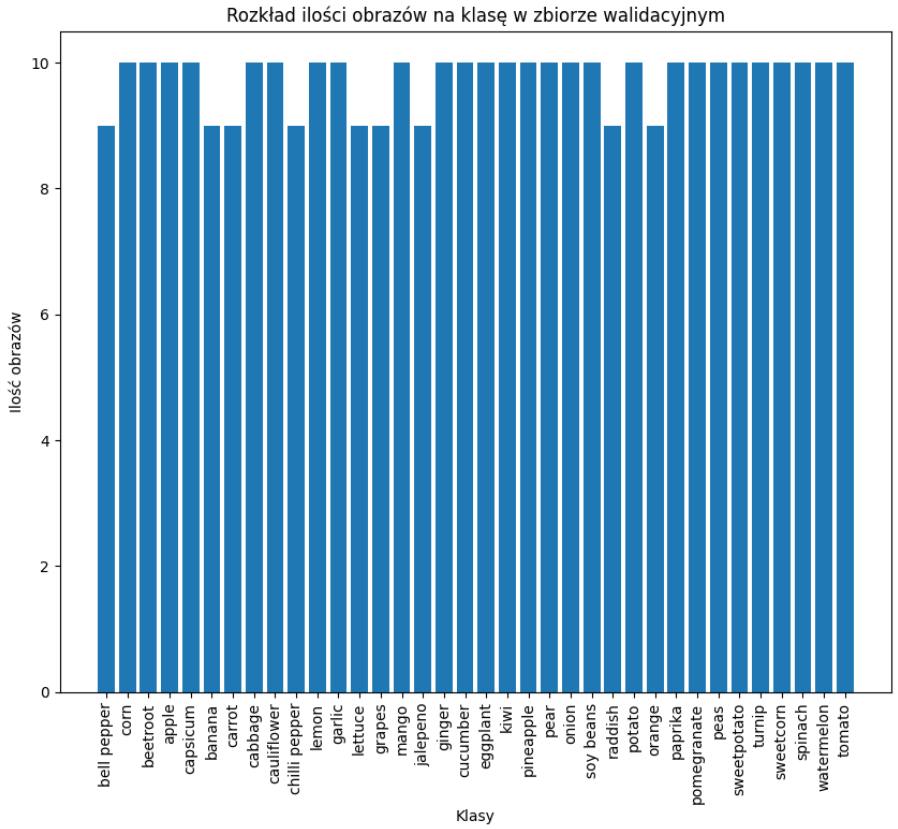
Opis problemu i danych

Projekt dotyczy dziedziny przetwarzania obrazów, konkretnie klasyfikacji obrazów owoców i warzyw. Jest to problem klasyfikacji wieloklasowej, gdzie celem jest przypisanie każdego obrazu do odpowiedniej klasy odpowiadającej rodzajowi owocu lub warzywa. Dane składają się z obrazów.

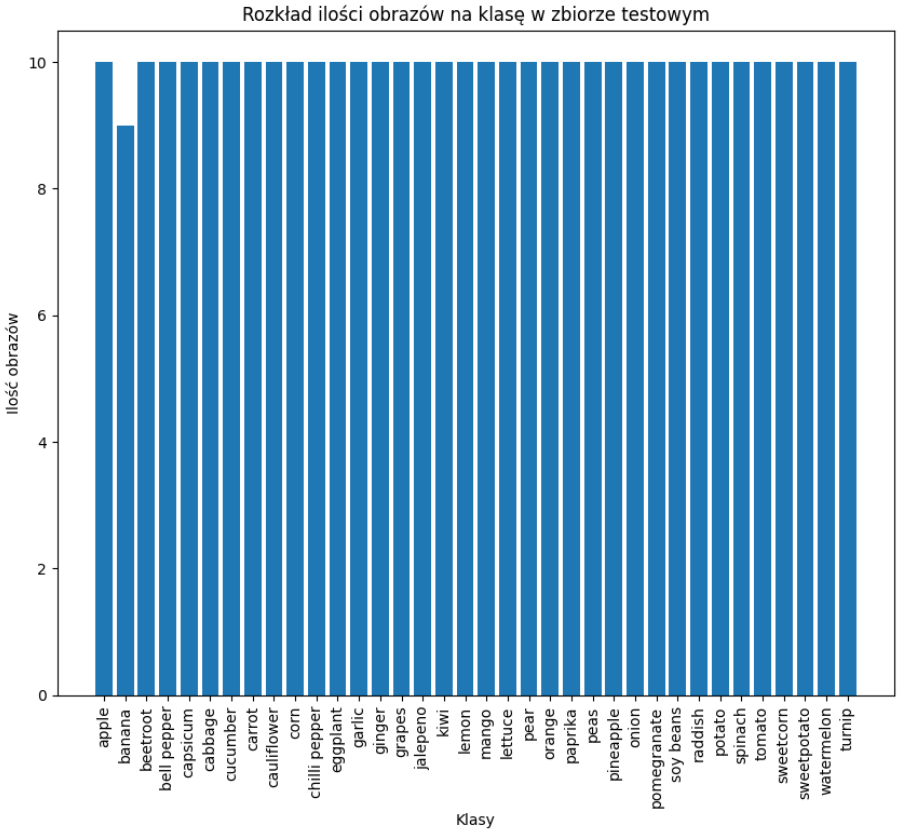
Baza treningowa obrazów liczy w sumie 36 folderów i 3115 obrazów:



Baza walidacyjna obrazów liczy w sumie 36 foldrów i 351 obrazów:



Baza testowa liczy w sumie 36 folderów i 359 obrazów:



Obróbka danych

Proces przygotowania danych: W projekcie klasyfikacji obrazów owoców i warzyw z wykorzystaniem sieci neuronowych CNN oraz ResNet50, kluczowym elementem jest skuteczna obróbka danych wejściowych. Proces ten skupia się na przygotowaniu obrazów w taki sposób, aby były one odpowiednio sformatowane i optymalne dla procesu uczenia sieci neuronowej.

Skalowanie rozmiaru obrazów: Pierwszym krokiem w obróbce danych jest zmiana rozmiaru obrazów do jednolitego formatu. W tym przypadku, wszystkie obrazy są skalowane do rozmiaru 64x64 pikseli. Jest to standardowa praktyka w przetwarzaniu obrazów dla sieci neuronowych, umożliwiająca modelowi przetwarzanie danych wejściowych w spójny sposób. Ta jednolitość jest niezbędna, ponieważ sieci neuronowe wymagają, aby wszystkie wejściowe obrazy miały ten sam rozmiar.

Konwersja i normalizacja danych: Po skalowaniu, obrazy są konwertowane do postaci tablic tensorów. Proces ten jest zautomatyzowany za pomocą funkcji tf.keras.utils.image_dataset_from_directory w TensorFlow, która zajmuje się ładowaniem obrazów, ich skalowaniem oraz konwersją do tablic tensorów. Kolejnym krokiem jest normalizacja danych. Wartości pikseli w obrazach RGB, które zwykle mieszczą się w zakresie od 0 do 255, są skalowane do zakresu 0-1. Ta normalizacja jest przeprowadzana przez podzielenie każdej wartości piksela przez 255. Normalizacja jest kluczowa dla ułatwienia procesu uczenia, ponieważ pomaga w utrzymaniu wartości wejściowych w niewielkim zakresie, co przyspiesza i stabilizuje trening sieci neuronowej.

Opis zastosowanych sieci neuronowych

Zastosowano dwie główne architektury sieci neuronowych: własną implementację sieci CNN oraz ResNet50 dostępną w TensorFlow.

CNN:

1. W projekcie zastosowano architektruę CNN, to typ sieci neuronowej szczególnie przydatny w przetwarzaniu obrazów, który automatycznie i adaptacyjnie uczy się hierarchii cech z obrazów. Własna architektura CNN została zaprojektowana do klasyfikacji obrazów owoców i warzyw na 36 różnych klas Warstwy MaxPooling do redukcji wymiarowości.

2. Struktura zastosowanego modelu CNN – model składa się z kilku kluczowych elementów, które wspólnie budują kompleksową architekturę do przetwarzania i klasyfikacji obrazów.

a. Warstwy konwolucyjne: Model zawiera kilka warstw konwolucyjnych (Conv2D), każda z filtrami o rozmiarze 3x3 i funkcją aktywacji ReLU (Rectified Linear Unit). Pierwsza z tych warstw przyjmuje obrazy wejściowe o wymiarach 64x64 pikseli z 3 kanałami kolorów (RGB). Warstwy konwolucyjne są kluczowe dla ekstrakcji cech z obrazów przez aplikowanie filtrów, które automatycznie identyfikują ważne cechy takie jak krawędzie, tekstury czy wzory.

b. Warstwy łączące (Pooling): Po warstwach konwolucyjnych stosowane są warstwy MaxPooling (MaxPool2D) z rozmiarem puli 2x2 i przesunięciem (stride) 2. Te warstwy redukują wymiarowość danych wyjściowych z poprzednich warstw, zachowując przy tym najważniejsze informacje. Działają one poprzez wybieranie maksymalnej wartości z każdego kwadratu 2x2 pikseli, co pomaga w zmniejszeniu liczby parametrów i obliczeń koniecznych dla kolejnych warstw sieci.

c. Dropout: Aby zapobiec nadmiernemu dopasowaniu (overfitting), w architekturze zastosowano warstwę Dropout z wartością 0.5, co oznacza, że losowo wyłącza ona połowę połączeń między neuronami podczas treningu, zwiększając zdolność generalizacji modelu.

d. Spłaszczenie danych (Flatten): Warstwa Flatten jest używana do przekształcenia wielowymiarowych danych wyjściowych z poprzednich warstw w jednowymiarowy wektor. Umożliwia to przekazanie danych do gęsto połączonych warstw sieci.

e. Gęsto połączone warstwy (Dense): Na końcu sieci znajdują się dwie gęsto połączone warstwy. Pierwsza z nich zawiera 128 neuronów i funkcję aktywacji ReLU, która umożliwia dalsze uczenie się złożonych wzorców z cech ekstrahowanych przez poprzednie warstwy. Ostatnia warstwa gęsto połączona zawiera 36 neuronów, po jednym dla każdej klasy, z funkcją aktywacji softmax, która przekształca wyniki sieci w rozkład prawdopodobieństwa przynależności do poszczególnych klas.

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(64, 64, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(36, activation='softmax')
])
```

ResNet50:

1. W projekcie zastosowano architekturę sieci neuronowej ResNet50, wykorzystującą mechanizm uczenia transferowego. ResNet, czyli Residual Network, to rodzaj głębokiej sieci neuronowej, która wykorzystuje tzw. połączenia rezydualne do przekazywania sygnałów z wejścia na wyjścia niektórych warstw. Model ResNet50, gdzie liczba 50 oznacza ilość warstw z wagami, został pierwotnie wytrenowany na ponad milionie obrazów z bazy ImageNet. Dzięki temu model jest w stanie rozpoznawać 1000 różnych kategorii obiektów, co czyni go wyjątkowo skutecznym jako punkt wyjścia dla wielu zadań związanych z przetwarzaniem obrazów.

2. Implementacja modelu ResNet50 dla klasyfikacji obrazów - w ramach projektu stworzono model klasyfikacyjny na bazie ResNet50, dostosowany do rozpoznawania 36 klas owoców i warzyw. Proces adaptacji modelu do zadania klasyfikacji wieloklasowej realizowany jest następująco:

    a. Wczytanie modelu ResNet50 bez górnych warstw przy pomocy tf.keras.applications.ResNet50(include_top=False, input_shape=(64, 64, 3), weights='imagenet') załadowano pre-trenowany model ResNet50, pomijając górną warstwę (ostatnią warstwę gęsto połączoną), co umożliwia dostosowanie modelu do nowego zadania. Parametr input_shape został ustalony na (64, 64, 3), co odpowiada wymiarom wejściowym obrazów, a weights='imagenet' wskazuje na wykorzystanie wag uzyskanych podczas treningu na zbiorze ImageNet.

    b. Dodanie warstw dostosowujących: Do modelu dodano kilka warstw, aby dostosować go do nowego zadania klasyfikacji:

        - GlobalAveragePooling2D: Ta warstwa redukuje wymiary przestrzenne wejścia (wysokość i szerokość), pozostawiając jedynie kanały głębi. Jest to efektywny sposób na zmniejszenie liczby parametrów i zapobieganie nadmiernemu dopasowaniu.
        - Dropout(0.5): Warstwa ta losowo wyłącza 50% neuronów podczas treningu, co pomaga w zapobieganiu nadmiernemu dopasowaniu przez zwiększenie zdolności generalizacji modelu.
        - Dense(128, activation='relu'): Gęsto połączona warstwa z 128 neuronami i funkcją aktywacji ReLU, służąca do nauki skomplikowanych wzorców z cech wydobytych przez ResNet50.
        - Dense(36, activation='softmax'): Ostatnia warstwa gęsto połączona z 36 neuronami, gdzie każdy neuron odpowiada jednej z klas. Funkcja aktywacji softmax jest używana do generowania rozkładu prawdopodobieństwa przynależności do poszczególnych klas.

Model ten łączy w sobie zaawansowane cechy ekstrakcyjne ResNet50 z możliwością nauki specyfik klasyfikacji owoców i warzyw, stanowiąc tym samym efektywne narzędzie do rozpoznawania wzorców w danych obrazowych.

```
def create_resnet_model(input_shape):
    base_model = tf.keras.applications.ResNet50(include_top=False,
input_shape=input_shape, weights='imagenet')
    model = tf.keras.models.Sequential()
    model.add(base_model)
    model.add(tf.keras.layers.GlobalAveragePooling2D())
    model.add(tf.keras.layers.Dropout(0.5))
    model.add(tf.keras.layers.Dense(units=128, activation='relu'))
    model.add(tf.keras.layers.Dense(units=36,
activation='softmax'))
    return model

resnet_model = create_resnet_model((64, 64, 3))
```

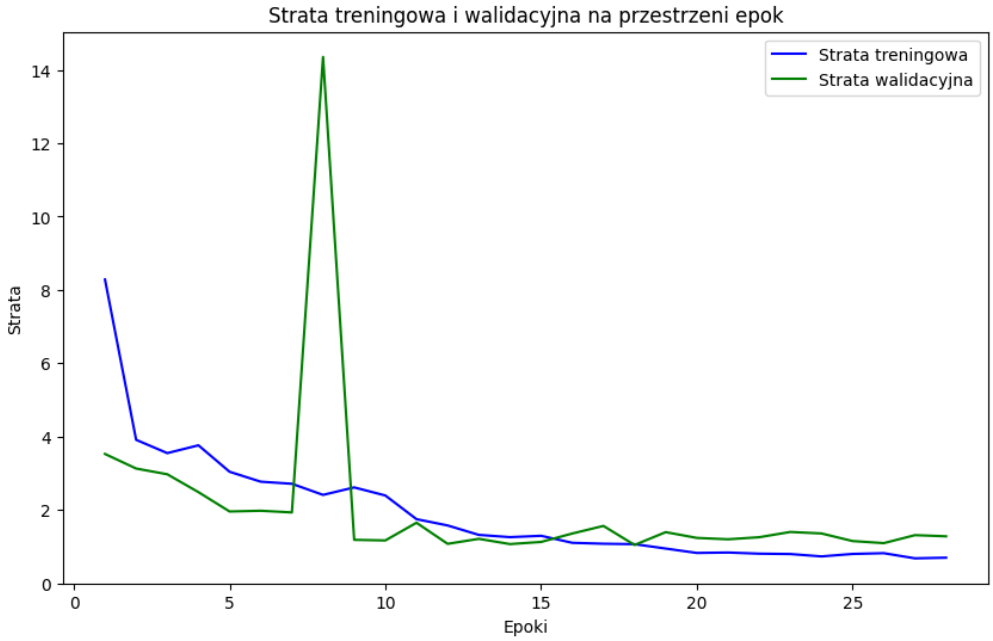
Trening obu modeli odbył się przez 50 epok, z użyciem optymalizatora 'rmsprop' i funkcji straty 'categorical_crossentropy'. Dla każdego modelu zastosowano mechanizm przerwania treningu, jeżeli w ciągu kolejnych 20 epok nauki nie został zanotowany progres. Przerwanie nastąpiło dla modelu opartego o sieć CNN.

Wnioski:

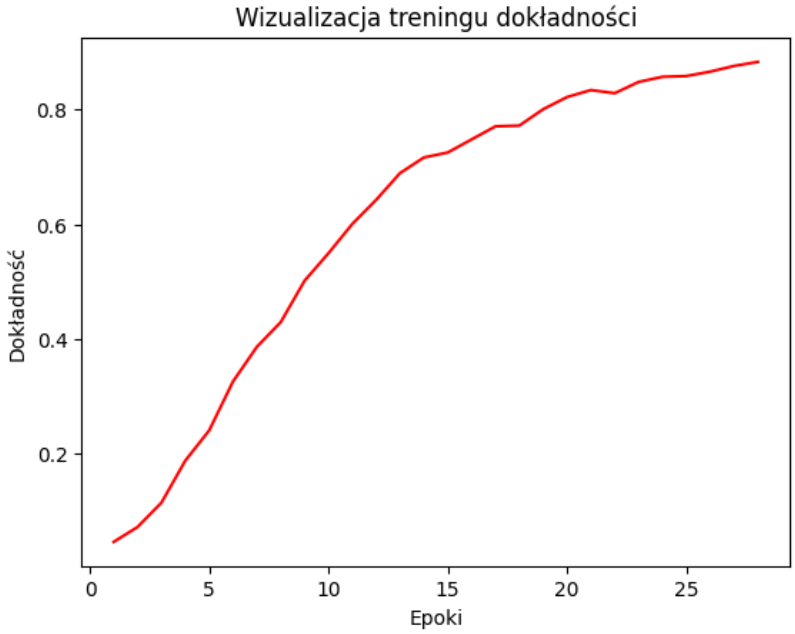
CNN:

Analizując wyniki dla sieci CNN, obserwujemy następujące trendy:

Zmniejszanie straty: Strata treningowa maleje znacząco z początkowej wartości około 8.28 do około 0.70 w ostatnich epokach, co wskazuje na skuteczne uczenie się i dopasowywanie modelu do danych treningowych.



Poprawa dokładności: Dokładność treningowa wzrasta z około 4.6% na początku do około 88.3% w ostatniej epoce. Jest to znak, że model staje się coraz lepszy w klasyfikacji obrazów.



Wyniki na zbiorze walidacyjnym: Dokładność na zbiorze walidacyjnym także poprawia się, choć z pewnymi fluktuacjami, osiągając wartość około 92%. Strata walidacyjna również jest zmienna, z kilkoma wyraźnymi wzrostami, które mogą wskazywać na problemy z generalizacją modelu.

Wnioski dla CNN:

Ewolucja modelu: Model wykazuje poprawę w uczeniu się na przestrzeni epok, co widać zarówno w spadającej stracie, jak i rosnącej dokładności. Jest to pozytywny znak, że architektura modelu i proces uczenia są odpowiednie dla zadania.

Nadmierna adaptacja do danych treningowych: Fluktuacje w stracie walidacyjnej i jej okresowe wzrosty mogą wskazywać na nadmierne dopasowanie modelu do danych treningowych. Może to oznaczać, że model nie radzi sobie równie dobrze z nowymi, niewidzianymi wcześniej danymi.

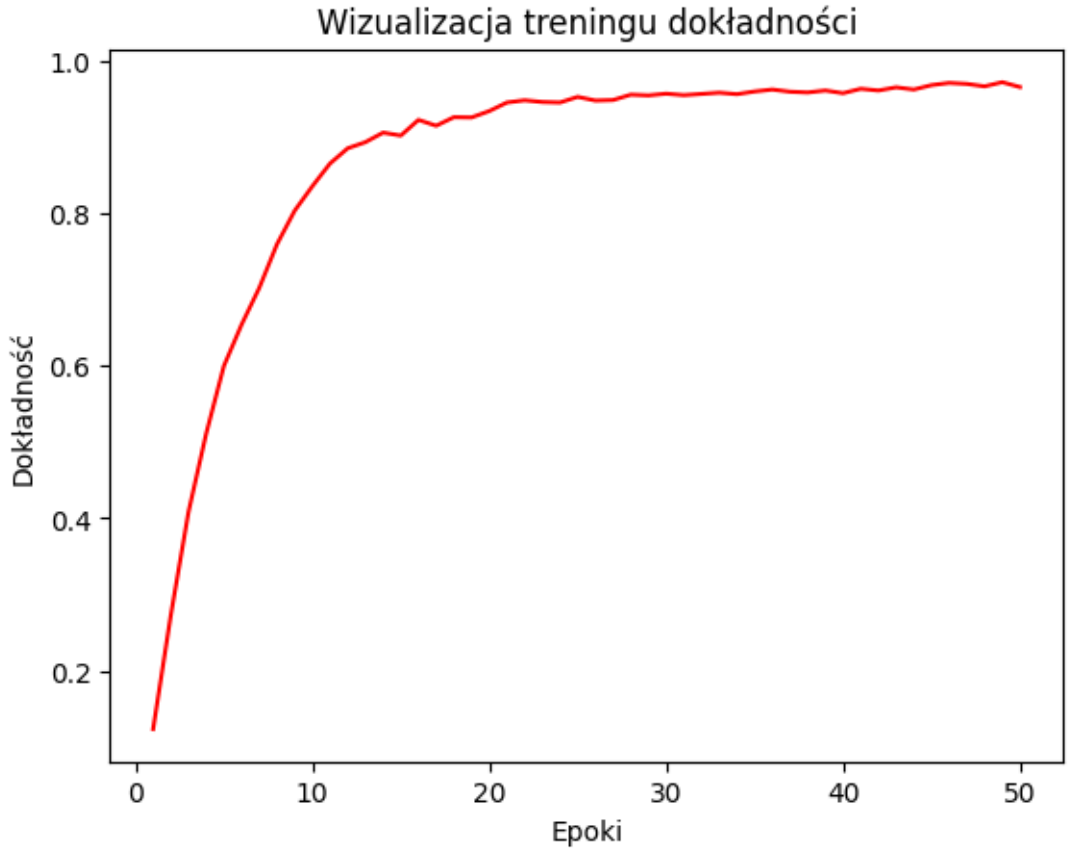
Generalizacja modelu: Chociaż dokładność walidacyjna jest wysoka, zmienność straty walidacyjnej sugeruje, że model może mieć trudności z generalizacją na danych, których nie widział podczas treningu.

Podsumowując, model CNN pokazuje obiecujące wyniki, ale istnieje przestrzeń do poprawy w kontekście stabilności i generalizacji na zbiorze walidacyjnym. Przeprowadzenie dalszych eksperymentów i optymalizacji może pomóc w rozwiązaniu tych problemów.

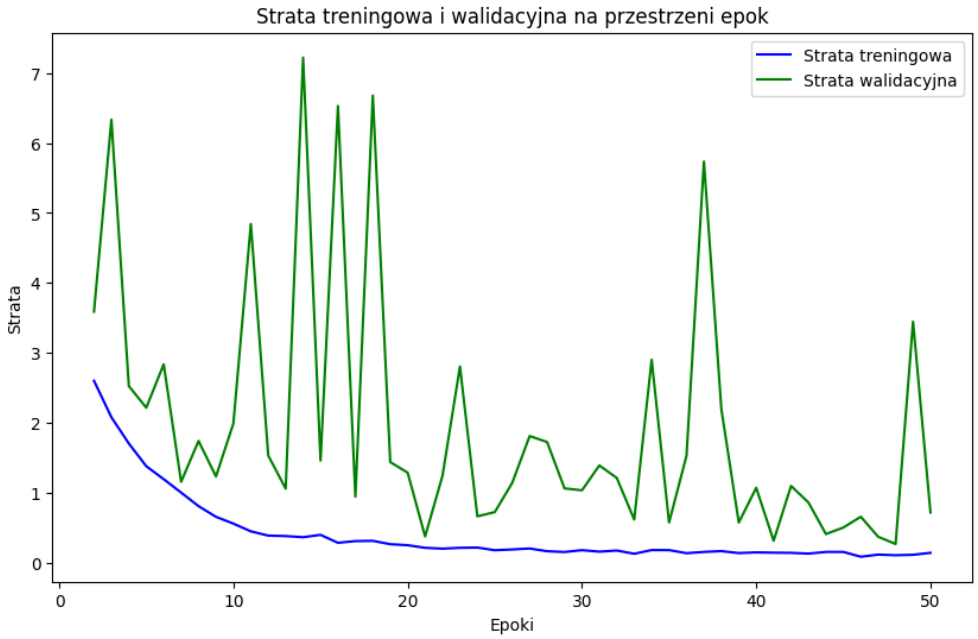
ResNet50:

Na podstawie dostarczonych danych dla modelu trenowanego z użyciem architektury ResNet50, można zaobserwować następujące tendencje:

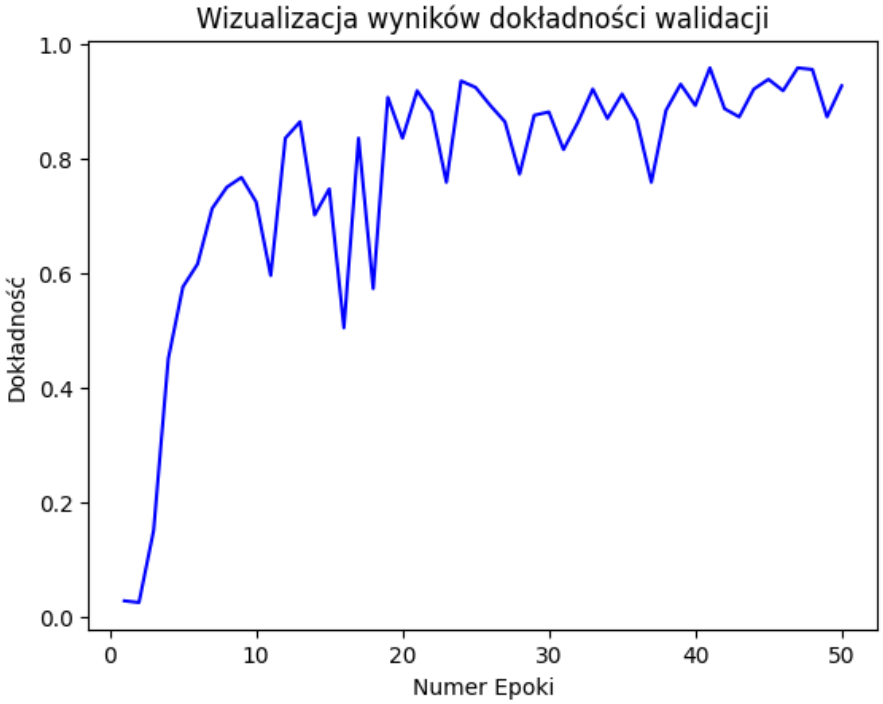
Poprawa dokładności: Dokładność treningowa wzrasta progresywnie z około 12.3% do około 96.5%, co świadczy o skutecznym uczeniu się modelu w rozróżnianiu między różnymi klasami.



Zmniejszanie straty: Strata treningowa systematycznie maleje z wartości początkowej około 3.47 do około 0.14 w ostatniej epoce, co wskazuje na efektywne uczenie się modelu i dobieranie się do charakterystyki danych treningowych (pierwszy pomiar został odrzucony).



Wyniki na zbiorze walidacyjnym: Dokładność na zbiorze walidacyjnym generalnie poprawia się, osiągając maksymalną wartość około 95.7%. Jednak strata walidacyjna jest bardziej zmienna i nie zawsze odzwierciedla tendencje obserwowane w danych treningowych. Mimo że osiągamy niskie wartości straty walidacyjnej w niektórych epokach, w innych obserwujemy znaczne skoki.



Wnioski dla ResNet50:

Skuteczność modelu ResNet50: Model ResNet50 wykazuje dobrą zdolność do nauki i klasyfikacji obrazów. Osiągnięta wysoka dokładność treningowa oraz zadowalająca dokładność walidacyjna wskazują na ogólną skuteczność wybranej architektury w rozwiązywanym zadaniu.

Problem nadmiernego dopasowania: Zmienna strata walidacyjna wskazuje na potencjalne problemy z nadmiernym dopasowaniem modelu do danych treningowych. Momentami wysoka strata walidacyjna przy niskiej stracie treningowej sugeruje, że model może być przesadnie dopasowany do danych treningowych i gorzej radzić sobie z nowymi, nieznajomymi danymi.

Podsumowując, model ResNet50 osiąga obiecujące wyniki w zadaniu klasyfikacji obrazów, ale istnieje przestrzeń do poprawy, szczególnie w zakresie unikania nadmiernego dopasowania i zwiększenia stabilności wyników na zbiorze walidacyjnym.

Porównanie CNN i ResNet50:

Skuteczność i generalizacja: ResNet50 wydaje się być bardziej skuteczny w klasyfikacji obrazów owoców i warzyw, osiągając wyższą dokładność i niższą stratę zarówno na danych treningowych, jak i walidacyjnych. Sugeruje to, że architektura ResNet50 może lepiej się uczyć na nowych danych.

Złożoność modelu: ResNet50 jest zdecydowanie bardziej złożonym modelem w porównaniu do stosunkowo prostszej architektury CNN, co może przyczyniać się do jego wyższej efektywności. Jednak większa złożoność wiąże się również z większymi wymaganiami obliczeniowymi.

Potencjalne zastosowania: Obie sieci wykazały wysoką dokładność, co czyni je użytecznymi w praktycznych aplikacjach klasyfikacji obrazów. Wybór pomiędzy CNN a ResNet50 może zależeć od specyficznych wymagań projektu, takich jak ograniczenia sprzętowe, wymagania czasowe czy potrzeba maksymalizacji dokładności.

Dalszy rozwój projektu:

Rozwój projektu klasyfikacji obrazów owoców i warzyw może zostać wzbogacony o nowy, innowacyjny wymiar, wykorzystując zdobyte do tej pory doświadczenie i wiedzę. Jedną z propozycji jest rozszerzenie funkcjonalności modelu do analizy obrazu video w czasie rzeczywistym korzystając z algorytmu You Only Look Once lub Single Shot MultiBox Detector, umożliwiając rozpoznawanie i klasyfikację owoców i warzyw umieszczonych w koszyku zakupowym. Taka funkcjonalność może znaleźć zastosowanie w różnych scenariuszach, od ułatwienia zakupów w supermarketach po edukacyjne aplikacje promujące zdrowe odżywianie.

Linki do kodu źródłowego:

Model CNN: https://colab.research.google.com/drive/1oq4Hp6cCo1R07SGuFkYxItIM5H0TguVk?usp=sharing

Model ResNet50: https://colab.research.google.com/drive/1CnukMywTroD3zt42FrvOIg4uDUO-Fi1W?usp=sharing

Testowanie modelu: https://colab.research.google.com/drive/1dd2QGegg_dtxvqTjN7OFKL89dmd7HUGu?usp=sharing

Link do wykorzystanych obrazów:
https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition