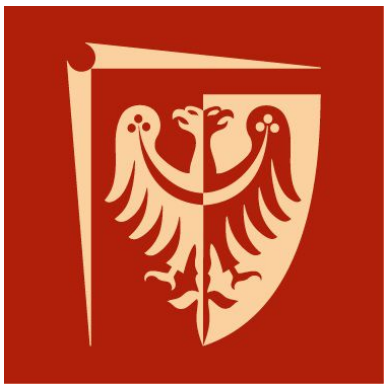


Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Informatyczne systemy automatyki

Sieci neuronowe - Fruits Classification

Autorzy:
Damian Filipowski id. 272555
Konrad Landzberg id. 272508

Przedmiot:
Sieci neuronowe - Projekt

18 stycznia 2025

Spis treści

Spis rysunków

1 Cel projektu:

Celem projektu było zaprojektowanie, implementacja oraz wytrenowanie sztucznej sieci neuronowej w celu rozwiązania problemu klasyfikacji wizualnej, polegającego na rozpoznawaniu różnych rodzajów owoców na podstawie obrazów. Projekt zakładał wykorzystanie istniejących frameworków do budowy i uczenia sieci neuronowych oraz zastosowanie odpowiednio złożonego zbioru danych, aby zademonstrować skuteczność opracowanego modelu.

2 Wykorzystane narzędzia:

- Język programowania: Python
- Biblioteka: PyTorch – użyta do budowy, trenowania i ewaluacji sztucznej sieci neuronowej.
- Model bazowy: TinyVGG – posłużył jako punkt wyjścia w projekcie. Na jego podstawie dokonano modyfikacji struktury modelu i parametrów, mających na celu poprawę efektów uczenia i zwiększenie skuteczności klasyfikacji owoców.

3 Plan realizacji projektu:

Aby zapewnić skuteczną realizację projektu, przygotowano szczegółowy plan działań. Plan ten obejmuje wszystkie kluczowe etapy, od przygotowania zbioru danych, poprzez budowę i trenowanie modelu, aż po testowanie i optymalizację wyników.

- Analiza problemu i dobór optymalnej architektury sieci neuronowej do rozwiązania zadania.
- Przygotowanie zbiorów danych na podstawie, których uczona będzie sieć neuronowa.
- Podział obrazów na odpowiednie zestawy: treningowe, walidacyjne, testowe.
- Przetworzenie obrazów do odpowiedniego formatu.
- Przygotowanie funkcji do uczenia, walidacji i testowania.
- Przygotowanie podstawowego modelu do klasyfikacji obrazów.
- Przeprowadzenie serii eksperymentów mających na celu ustalenie optymalnej struktury i konfiguracji sieci oraz wizualizacja wyników.
- Analiza otrzymanych wyników.

4 Analiza problemu i dobór architektury:

Do rozwiązania problemu klasyfikacji obrazów owoców zaprojektowano sieć neuronową opartą na lekkiej architekturze inspirowanej modelem TinyVGG. Celem było stworzenie efektywnego modelu zdolnego do przetwarzania obrazów RGB i przypisywania ich do określonych kategorii owoców. Wybrana architektura, składała się z bloków konwolucyjnych i modułu klasyfikującego. Zastosowanie warstw konwolucyjnych było kluczowe ze względu na ich zdolność do automatycznego wyodrębniania istotnych cech z obrazów, takich jak krawędzie, wzory i tekstury. Dodatkowo, warstwy konwolucyjne redukują liczbę parametrów w porównaniu z warstwami w pełni połączonymi, co sprawia, że są bardziej efektywne pod względem obliczeniowym.

5 Przygotowanie zbiorów danych:

W projekcie wykorzystano dwa różne zbiory danych, które pozwoliły na zróżnicowaną analizę skuteczności modelu:

1. Prosty zbiór danych – Zawierał zdjęcia owoców na jednolitym tle, co znacznie ułatwiało proces klasyfikacji. Zdjęcia były dobrze oświetlone, a obiekty jednoznaczne i wyraźnie odseparowane od tła. Poniżej przedstawiono przykładowe obrazy:

Rysunek 1: Przykłady obrazów z prostego zbioru danych.

2. Złożony zbiór danych – Zawierał zdjęcia owoców w różnorodnych kontekstach, np. jako część ciast. Zdjęcia charakteryzowały się dużym zróżnicowaniem oświetlenia, tła oraz obecnością innych obiektów, co znacząco utrudniało proces klasyfikacji. Główne problemy związane z tym zbiorem to:

   - Złożone tła: Tło zdjęcia często zawierało elementy o podobnym kolorze i fakturze co owoce, co mogło prowadzić do błędów w klasyfikacji.
   - Niepełne owoce: Część owoców była tylko częściowo widoczna, co mogło utrudniać ich rozpoznanie.
   - Zróżnicowane warunki oświetleniowe: Zmienne warunki oświetlenia mogły wpływać na kolor i widoczność obiektów.
   - Obecność innych obiektów: Inne elementy w kadrze mogły wprowadzać zakłócenia.

   Przykłady obrazów z tego zbioru przedstawiono poniżej:

Rysunek 2: Przykłady obrazów z złożonego zbioru danych.

6 Podział obrazów na zestawy:

Aby umożliwić efektywne trenowanie, walidację i testowanie modelu, dokonano podziału zbioru danych na trzy części: zbiór treningowy, walidacyjny oraz testowy. Podział danych został przeprowadzony z zachowaniem odpowiednich proporcji i przy użyciu zdefiniowanych wielkości dla każdego zbioru, aby zapewnić ich równomierne reprezentowanie w różnych kategoriach.

```
[6]  categories = ['Banana', 'Orange', 'Kiwi/Kiwi B']
     train_size = 800
     validation_size = 100
     test_size = 100
```

```
[12] for category in categories:
       category_path = os.path.join(path, category)
       train_category_path = os.path.join(output_path, 'train', category)
       validation_category_path = os.path.join(output_path, 'validation', category)
       test_category_path = os.path.join(output_path, 'test', category)

       os.makedirs(train_category_path, exist_ok=True)
       os.makedirs(validation_category_path, exist_ok=True)
       os.makedirs(test_category_path, exist_ok=True)

       images = os.listdir(category_path)
       train_images, remaining_images = train_test_split(images, train_size=train_size, random_state=42)
       test_images, validation_images = train_test_split(remaining_images, train_size=test_size, test_size=validation_size, random_state=42)

       for img in train_images:
           shutil.copy(os.path.join(category_path, img), os.path.join(train_category_path, img))

       for img in validation_images:
           shutil.copy(os.path.join(category_path, img), os.path.join(validation_category_path, img))

       for img in test_images:
           shutil.copy(os.path.join(category_path, img), os.path.join(test_category_path, img))
```

Rysunek 3: Fragment kodu z zaimplementowanym podziałem na poszczególne zbiory.

7 Przetworzenie obrazów:

W celu przygotowania danych do trenowania modelu, konieczne było przeprowadzenie procesu przetwarzania obrazów, obejmującego ich transformację oraz utworzenie odpowiednich struktur danych. Aby to osiągnąć zastosowano transformacje, takie jak skalowanie, normalizacja i augmentacja, które pozwalają poprawić jakość danych i zwiększyć zdolność modelu do generalizacji.

```
[ ]  import torchvision.transforms as transforms

     train_transforms = transforms.Compose([
         transforms.Resize((64, 64)),
         transforms.TrivialAugmentWide(num_magnitude_bins=31),
         transforms.ToTensor()
     ])


     test_transforms = transforms.Compose([
         transforms.Resize((64, 64)),
         transforms.ToTensor()
     ])
```

Rysunek 4: Inicjalizacja transformacji obrazów.

Przetworzone obrazy zostały następnie skonwertowane do formatu zbiorów danych (datasets) oraz załadowane przy użyciu DataLoaderów, co umożliwiło efektywne zarządzanie partiami danych podczas trenowania, walidacji i testowania modelu.

```
[ ]  import torchvision.datasets as datasets

     train_data = datasets.ImageFolder(root=output_path / "train",
                                       transform=train_transforms,
                                       target_transform=None)

     validation_data = datasets.ImageFolder(root=output_path / "validation",
                                            transform=test_transforms)

     test_data = datasets.ImageFolder(root=output_path / "test",
                                      transform=test_transforms)

     print(f"Train data:\n{train_data}\nTest data:\n{test_data}, {validation_data}")
```

Rysunek 5: Przekształcenie obrazów w datasety.

```
[ ]  from torch.utils.data import DataLoader
     BATCH_SIZE = 32
     train_dataloader = DataLoader(dataset=train_data,
                                   batch_size=BATCH_SIZE,
                                   num_workers=os.cpu_count(),
                                   shuffle=True)

     validation_dataloader = DataLoader(dataset=validation_data,
                                        batch_size=BATCH_SIZE,
                                        num_workers=os.cpu_count(),
                                        shuffle=False)

     test_dataloader = DataLoader(dataset=test_data,
                                  batch_size=BATCH_SIZE,
                                  num_workers=os.cpu_count(),
                                  shuffle=False)

     train_dataloader, validation_dataloader, test_dataloader
```

Rysunek 6: Przekształcenie obrazów w dataloadery.

8 Przygotowanie funkcji do uczenia, walidacji i testowania:

Aby umożliwić trenowanie modelu, zaimplementowano funkcję uczącą, która krok po kroku dostosowuje parametry modelu do danych treningowych. Funkcja ta obsługuje kluczowe etapy, takie jak obliczanie strat, aktualizowanie wag modelu oraz monitorowanie wyników. Została zaprojektowana tak, aby była uniwersalna i łatwa w użyciu dla różnych modeli i zbiorów danych.

```
[ ]  def train_step(model: torch.nn.Module,
                    dataloader: torch.utils.data.DataLoader,
                    loss_fn: torch.nn.Module,
                    optimizer: torch.optim.Optimizer):

         model.train()

         train_loss, train_acc = 0, 0

         for batch, (X, y) in enumerate(dataloader):
             X, y = X.to(device), y.to(device)

             y_pred = model(X)

             loss = loss_fn(y_pred, y)
             train_loss += loss.item()

             optimizer.zero_grad()

             loss.backward()

             optimizer.step()

             y_pred_class = torch.argmax(torch.softmax(y_pred, dim=1), dim=1)
             train_acc += (y_pred_class == y).sum().item()/len(y_pred)

         train_loss = train_loss / len(dataloader)
         train_acc = train_acc / len(dataloader)
         return train_loss, train_acc
```

Rysunek 7: Implementacja pętli treningowej.

W celu ocenienia skuteczności modelu, stworzono funkcję testującą, która pozwala na sprawdzanie jego wyników na danych walidacyjnych i testowych. Funkcja działa w trybie ewaluacji, obliczając straty i dokładność, co umożliwia ocenę zdolności modelu do generalizacji na dane, których nie widział podczas uczenia. Dzięki temu można monitorować jakość modelu i porównywać jego wyniki w różnych etapach projektu.

```python
def test_step(model: torch.nn.Module,
              dataloader: torch.utils.data.DataLoader,
              loss_fn: torch.nn.Module):

    model.eval()

    test_loss, test_acc = 0, 0

    with torch.inference_mode():
        for batch, (X, y) in enumerate(dataloader):
            X, y = X.to(device), y.to(device)

            test_pred_logits = model(X)

            loss = loss_fn(test_pred_logits, y)
            test_loss += loss.item()

            test_pred_labels = test_pred_logits.argmax(dim=1)
            test_acc += ((test_pred_labels == y).sum().item()/len(test_pred_labels))

    test_loss = test_loss / len(dataloader)
    test_acc = test_acc / len(dataloader)
    return test_loss, test_acc
```

Rysunek 8: Implementacja pętli testowej.

9 Podstawowy model do klasyfikacji obrazów:

W projekcie zaprojektowano architekturę sieci neuronowej odpowiednią do zadania klasyfikacji obrazów owoców.

9.1 Opis architektury modelu

Zaprojektowany model składa się z dwóch głównych bloków konwolucyjnych oraz modułu klasyfikacyjnego. Struktura sieci została szczegółowo opisana poniżej:

- Bloki konwolucyjne - Model zawiera dwa bloki konwolucyjne. Każdy blok składa się z dwóch warstw konwolucyjnych z funkcjami aktywacji ReLU oraz warstwą MaxPooling, która redukuje wymiar danych. Te bloki służą do wyodrębniania cech obrazu, takich jak krawędzie, faktury czy wzory.
- Moduł klasyfikujący - Obejmuje warstwę Flatten, która przekształca dane wielowymiarowe w wektor jednowymiarowy. Zawiera w pełni połączoną warstwę liniową (Linear), która mapuje wyodrębnione cechy na kategorie wyjściowe (klasy owoców).
- Funkcje aktywacji - W modelu zastosowano funkcję aktywacji ReLU, która wprowadza nieliniowość, pozwalając modelowi na lepsze odwzorowanie skomplikowanych zależności w danych.
- Parametry sieci - Liczba kanałów wejściowych wynosi 3 (dla obrazów RGB). Liczba neuronów w warstwach konwolucyjnych jest ustawiona na 10. Liczba neuronów wyjściowych jest równa liczbie klas w zbiorze danych.

9.2 Parametryzacja modelu

Bazowy model został zdefiniowany z możliwością łatwej modyfikacji kluczowych parametrów:

- Input shape: Ustalono liczbę kanałów wejściowych jako 3 (RGB).
- Hidden units: Liczba filtrów w każdej warstwie konwolucyjnej wynosi 10.
- Output shape: Liczba klas wyjściowych odpowiada liczbie kategorii owoców w zbiorze danych.

9.3 Dobór funkcji optymalizującej i strat

W naszym projekcie zdecydowano się na zastosowanie optymalizatora Adam oraz funkcji strat CrossEntropyLoss. Adam został wybrany ze względu na jego zdolność do dynamicznego dostosowywania tempa uczenia się na podstawie pierwszego i drugiego momentu gradientów, co czyni go skutecznym w zadaniach klasyfikacji obrazów. Funkcja strat CrossEntropyLoss jest idealna dla problemów wieloklasowej klasyfikacji, ponieważ mierzy różnicę między przewidywaniami modelu a rzeczywistymi etykietami, przyznając większą karę za większe błędy.

9.4 Podsumowanie

Przyjęta architektura oparta na blokach konwolucyjnych i module klasyfikacyjnym umożliwiła skuteczne przetwarzanie obrazów oraz klasyfikację. Zastosowanie warstw konwolucyjnych, ReLU i MaxPooling pozwoliło na wyodrębnienie kluczowych cech, natomiast moduł klasyfikacyjny zapewnił odpowiednie przypisanie danych wejściowych do kategorii wyjściowych.

```python
class FruitRecognitionModel(nn.Module):
  def __init__ (self, input_shape: int, hidden_units: int, output_shape: int) -> None:
      super().__init__()
      self.conv_block_1 = nn.Sequential(
          nn.Conv2d(in_channels=input_shape,
                    out_channels=hidden_units,
                    kernel_size=3,
                    stride=1,
                    padding=0,
                    ),
          nn.ReLU(),
          nn.Conv2d(in_channels=hidden_units,
                    out_channels=hidden_units,
                    kernel_size=3,
                    stride=1,
                    padding=0),
          nn.ReLU(),
          nn.MaxPool2d(kernel_size=2,
                       stride=2),
      )
      self.conv_block_2 = nn.Sequential(
          nn.Conv2d(hidden_units, hidden_units, kernel_size=3, padding=0),
          nn.ReLU(),
          nn.Conv2d(hidden_units, hidden_units, kernel_size=3, padding=0),
          nn.ReLU(),
          nn.MaxPool2d(2)
          )

      self.classifier = nn.Sequential(
          nn.Flatten(),
          nn.Linear(in_features=hidden_units*13*13,
                    out_features=output_shape)
          )

  def forward(self, x: torch.Tensor):
    x = self.conv_block_1(x)
    x = self.conv_block_2(x)
    x = self.classifier(x)
    return x

torch.manual_seed(42)
model_0 = FruitRecognitionModel(input_shape=3,
                  hidden_units=10,
                  output_shape=len(train_data.classes)).to(device)
model_0
```

Rysunek 9: Implementacja bazowego modelu na podstawie TinyVGG.

10 Badania:

Badania przeprowadzono z wykorzystaniem dwóch różnych zbiorów danych oraz różnych konfiguracji modeli, aby maksymalnie zoptymalizować wyniki i ocenić skuteczność podejścia w zróżnicowanych warunkach. Badania opierały się na uczeniu modelu przez określoną liczbe epok, jednakże w celu uniknięcia przeuczenia wprowadzono mechanizm, który przerywał proces uczenia w przypadku nastąpienia po sobie siedmiu epok bez poprawy skuteczności zapisując przy tym najlepszy model. Nauczony model ostatecznie sprawdzany był na danych testowych.

10.1 Badania na jednolitej bazie danych:

10.1.1 Badanie 1:

- Model z dwoma blokami konwolucyjnymi.
- Jedna warstwa liniowa.
- 10 neuronów w warstwach ukrytych.
- Prosty zbiór danych.
- Rozmiar obrazów 64 x 64.
- Brak augmentacji.
- Liczba epok 30 (uczenie zakończyło się po 19).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Banana	800	100	100
Orange	800	100	100
Kiwi	800	100	100

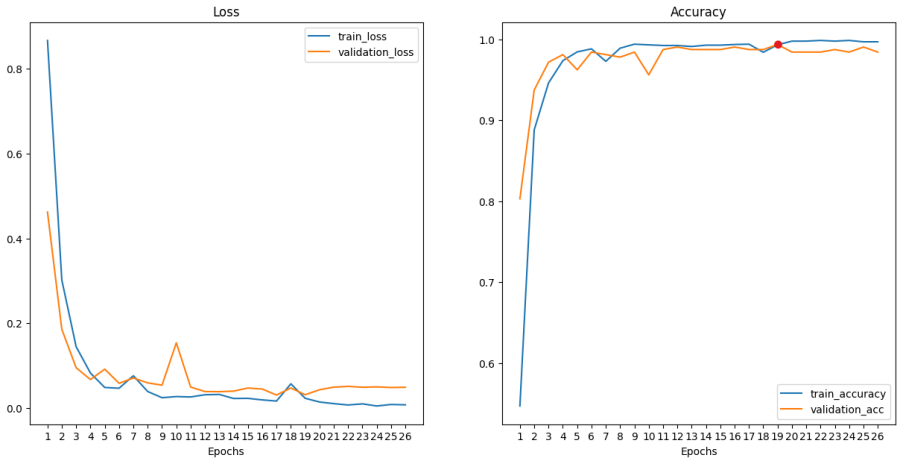
Tabela 1: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 13 | train_loss: 0.0319 | train_acc: 0.9912 | validation_loss: 0.0384 | validation_acc: 0.9875
0.9875
Epoch: 14 | train_loss: 0.0224 | train_acc: 0.9929 | validation_loss: 0.0397 | validation_acc: 0.9875
0.9875
Epoch: 15 | train_loss: 0.0227 | train_acc: 0.9929 | validation_loss: 0.0471 | validation_acc: 0.9875
0.9875
Epoch: 16 | train_loss: 0.0190 | train_acc: 0.9938 | validation_loss: 0.0447 | validation_acc: 0.9906
0.990625
Epoch: 17 | train_loss: 0.0163 | train_acc: 0.9942 | validation_loss: 0.0304 | validation_acc: 0.9875
0.9875
Epoch: 18 | train_loss: 0.0569 | train_acc: 0.9842 | validation_loss: 0.0471 | validation_acc: 0.9875
0.9875
Epoch: 19 | train_loss: 0.0228 | train_acc: 0.9933 | validation_loss: 0.0312 | validation_acc: 0.9938
0.99375
Best model updated at epoch 19 with test_acc: 0.9938
Epoch: 20 | train_loss: 0.0141 | train_acc: 0.9979 | validation_loss: 0.0429 | validation_acc: 0.9844
0.984375
Epoch: 21 | train_loss: 0.0102 | train_acc: 0.9979 | validation_loss: 0.0491 | validation_acc: 0.9844
0.984375
Epoch: 22 | train_loss: 0.0070 | train_acc: 0.9988 | validation_loss: 0.0509 | validation_acc: 0.9844
0.984375
Epoch: 23 | train_loss: 0.0097 | train_acc: 0.9979 | validation_loss: 0.0488 | validation_acc: 0.9875
0.9875
Epoch: 24 | train_loss: 0.0047 | train_acc: 0.9988 | validation_loss: 0.0497 | validation_acc: 0.9844
0.984375
Epoch: 25 | train_loss: 0.0082 | train_acc: 0.9971 | validation_loss: 0.0482 | validation_acc: 0.9906
0.990625
Epoch: 26 | train_loss: 0.0074 | train_acc: 0.9971 | validation_loss: 0.0488 | validation_acc: 0.9844
0.984375
Lack of any improvement
```

Rysunek 10: Badanie 1: Wyniki etapu uczenia.

```
Test loss: 0.006931615798370583 Test acc: 0.996875
```

Rysunek 11: Badanie 1: Wyniki etapu testowania.



Rysunek 12: Badanie 1: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 99,38% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 99,68%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.1.2 Badanie 2:

- Model z dwoma blokami konwolucyjnymi.
- Jedna warstwa liniowa.
- 10 neuronów w warstwach ukrytych.
- Prosty zbiór danych.
- Rozmiar obrazów 64 x 64.
- Brak augmentacji.
- Liczba epok 50 (uczenie zakończyło się po 22).
- Liczba owoców do rozpoznania: 8.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Banan	800	100	100
Peach	800	100	100
Orange	800	100	100
Pitaya	800	100	100
Plum	800	100	100
Tomatoes	800	100	100
Kiwi	800	100	100
Apple	800	100	100

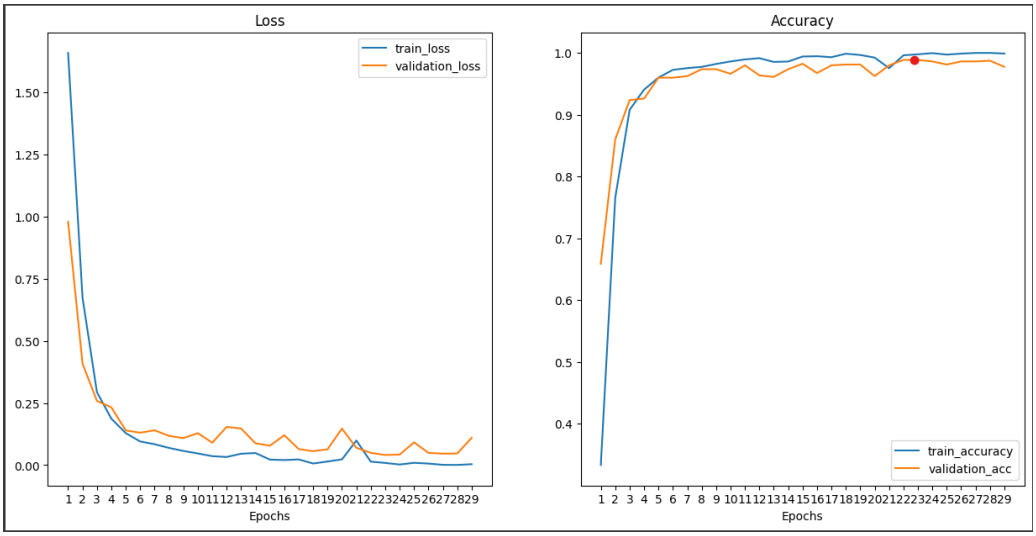
Tabela 2: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 16 | train_loss: 0.0203 | train_acc: 0.9947 | validation_loss: 0.1204 | validation_acc: 0.9675
0.9675
Epoch: 17 | train_loss: 0.0227 | train_acc: 0.9931 | validation_loss: 0.0644 | validation_acc: 0.9800
0.98
Epoch: 18 | train_loss: 0.0063 | train_acc: 0.9988 | validation_loss: 0.0559 | validation_acc: 0.9812
0.98125
Epoch: 19 | train_loss: 0.0144 | train_acc: 0.9967 | validation_loss: 0.0635 | validation_acc: 0.9812
0.98125
Epoch: 20 | train_loss: 0.0229 | train_acc: 0.9925 | validation_loss: 0.1469 | validation_acc: 0.9625
0.9625
Epoch: 21 | train_loss: 0.0088 | train_acc: 0.9753 | validation_loss: 0.0697 | validation_acc: 0.9800
0.98
Epoch: 22 | train_loss: 0.0138 | train_acc: 0.9962 | validation_loss: 0.0493 | validation_acc: 0.9888
0.98875
Best model updated at epoch 22 with test_acc: 0.9888
Epoch: 23 | train_loss: 0.0086 | train_acc: 0.9978 | validation_loss: 0.0406 | validation_acc: 0.9888
0.98875
Epoch: 24 | train_loss: 0.0022 | train_acc: 0.9997 | validation_loss: 0.0422 | validation_acc: 0.9862
0.98625
Epoch: 25 | train_loss: 0.0091 | train_acc: 0.9973 | validation_loss: 0.0916 | validation_acc: 0.9812
0.98125
Epoch: 26 | train_loss: 0.0061 | train_acc: 0.9989 | validation_loss: 0.0493 | validation_acc: 0.9862
0.98625
Epoch: 27 | train_loss: 0.0010 | train_acc: 1.0000 | validation_loss: 0.0460 | validation_acc: 0.9862
0.98625
Epoch: 28 | train_loss: 0.0006 | train_acc: 1.0000 | validation_loss: 0.0469 | validation_acc: 0.9875
0.9875
Epoch: 29 | train_loss: 0.0036 | train_acc: 0.9991 | validation_loss: 0.1099 | validation_acc: 0.9775
0.9775
Lack of any improvement
```

Rysunek 13: Badanie 2: Wyniki etapu uczenia.

```
Test loss: 0.1399441071263754 Test acc: 0.9625
```

Rysunek 14: Badanie 2: Wyniki etapu testowania.



Rysunek 15: Badanie 2: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 98,88% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 96,25%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

Ze względu na zaskakująco dobre wyniki osiągnięte przez podstawowy model, uznaliśmy za stosowne sprawdzenie, czy jego struktura nie jest zbyt złożona w kontekście tak jednolitego zbioru danych. W tym celu podjęliśmy decyzję o jego uproszczeniu.

10.1.3 Badanie 3:

- Model z jednym blokiem konwolucyjnym.
- Jedna warstwa liniowa.
- 10 neuronów w warstwach ukrytych.
- Prosty zbiór danych.
- Rozmiar obrazów 64 x 64.
- Brak augmentacji.
- Liczba epok 30 (uczenie zakończyło się po 3).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Banana	800	100	100
Orange	800	100	100
Kiwi	800	100	100

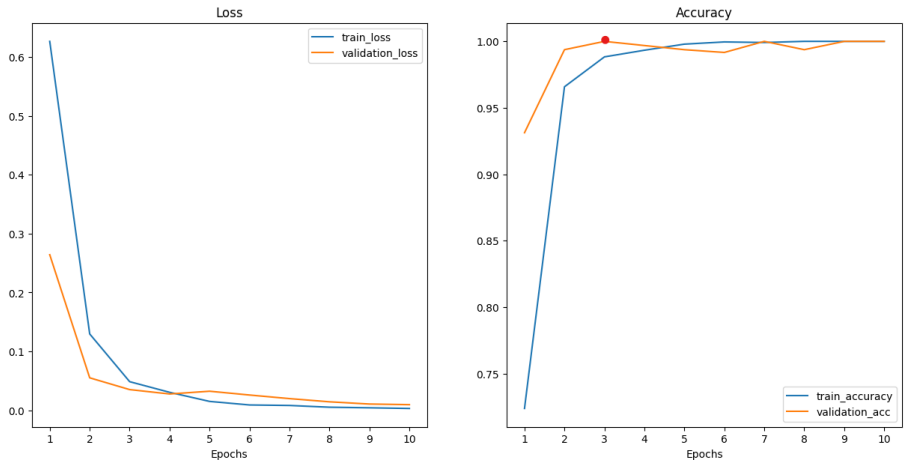
Tabela 3: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 1 | train_loss: 0.6266 | train_acc: 0.7238 | validation_loss: 0.2643 | validation_acc: 0.9313
0.93125
Best model updated at epoch 1 with test_acc: 0.9313
Epoch: 2 | train_loss: 0.1296 | train_acc: 0.9658 | validation_loss: 0.0551 | validation_acc: 0.9938
0.99375
Best model updated at epoch 2 with test_acc: 0.9938
Epoch: 3 | train_loss: 0.0485 | train_acc: 0.9883 | validation_loss: 0.0351 | validation_acc: 1.0000
1.0
Best model updated at epoch 3 with test_acc: 1.0000
Epoch: 4 | train_loss: 0.0305 | train_acc: 0.9933 | validation_loss: 0.0276 | validation_acc: 0.9969
0.996875
Epoch: 5 | train_loss: 0.0150 | train_acc: 0.9979 | validation_loss: 0.0323 | validation_acc: 0.9938
0.99375
Epoch: 6 | train_loss: 0.0090 | train_acc: 0.9996 | validation_loss: 0.0259 | validation_acc: 0.9917
0.9916666666666666
Epoch: 7 | train_loss: 0.0081 | train_acc: 0.9992 | validation_loss: 0.0198 | validation_acc: 1.0000
1.0
Epoch: 8 | train_loss: 0.0051 | train_acc: 1.0000 | validation_loss: 0.0143 | validation_acc: 0.9938
0.99375
Epoch: 9 | train_loss: 0.0042 | train_acc: 1.0000 | validation_loss: 0.0105 | validation_acc: 1.0000
1.0
Epoch: 10 | train_loss: 0.0030 | train_acc: 1.0000 | validation_loss: 0.0095 | validation_acc: 1.0000
1.0
Lack of any improvement
```

Rysunek 16: Badanie 3: Wyniki etapu uczenia.

```
Test loss: 0.00205632643410353 94 Test acc: 1.0
```

Rysunek 17: Badanie 3: Wyniki etapu testowania.



Rysunek 18: Badanie 3: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 100% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 100%.

10.1.4 Badanie 4:

- Model z jednym blokiem konwolucyjnym.
- Jedna warstwa liniowa.
- 10 neuronów w warstwach ukrytych.
- Prosty zbiór danych.
- Rozmiar obrazów 64 x 64.
- Brak augmentacji.
- Liczba epok 50 (uczenie zakończyło się po 12).
- Liczba owoców do rozpoznania: 8.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Banan	800	100	100
Peach	800	100	100
Orange	800	100	100
Pitaya	800	100	100
Plum	800	100	100
Tomatoes	800	100	100
Kiwi	800	100	100
Apple	800	100	100

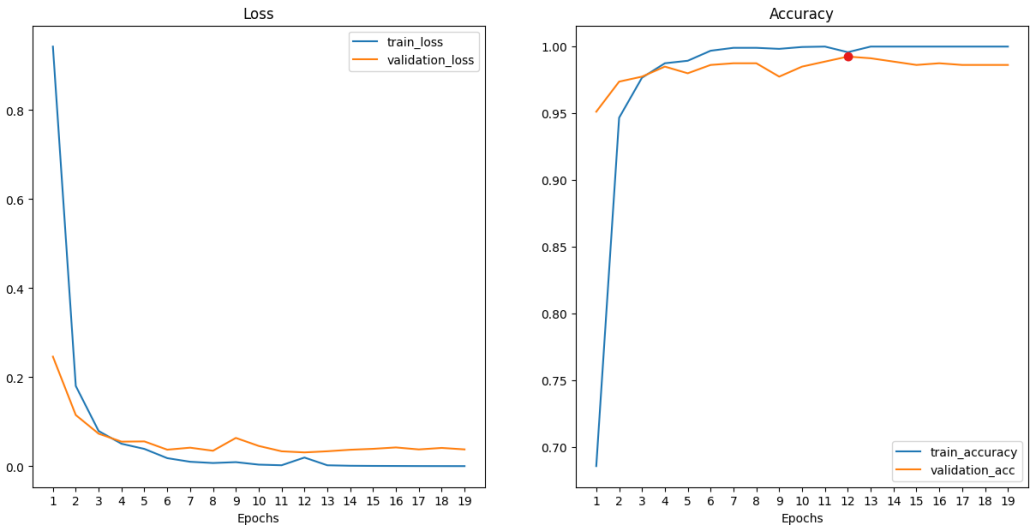
Tabela 4: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 5 | train_loss: 0.0390 | train_acc: 0.9894 | validation_loss: 0.0559 | validation_acc: 0.9800
0.98
Epoch: 6 | train_loss: 0.0183 | train_acc: 0.9969 | validation_loss: 0.0373 | validation_acc: 0.9862
0.98625
Best model updated at epoch 6 with test_acc: 0.9862
Epoch: 7 | train_loss: 0.0102 | train_acc: 0.9991 | validation_loss: 0.0418 | validation_acc: 0.9875
0.9875
Best model updated at epoch 7 with test_acc: 0.9875
Epoch: 8 | train_loss: 0.0074 | train_acc: 0.9991 | validation_loss: 0.0349 | validation_acc: 0.9875
0.9875
Epoch: 9 | train_loss: 0.0093 | train_acc: 0.9983 | validation_loss: 0.0637 | validation_acc: 0.9775
0.9775
Epoch: 10 | train_loss: 0.0038 | train_acc: 0.9997 | validation_loss: 0.0457 | validation_acc: 0.9850
0.985
Epoch: 11 | train_loss: 0.0023 | train_acc: 1.0000 | validation_loss: 0.0337 | validation_acc: 0.9888
0.98875
Best model updated at epoch 11 with test_acc: 0.9888
Epoch: 12 | train_loss: 0.0197 | train_acc: 0.9958 | validation_loss: 0.0312 | validation_acc: 0.9925
0.9925
Best model updated at epoch 12 with test_acc: 0.9925
Epoch: 13 | train_loss: 0.0023 | train_acc: 1.0000 | validation_loss: 0.0338 | validation_acc: 0.9912
0.99125
Epoch: 14 | train_loss: 0.0012 | train_acc: 1.0000 | validation_loss: 0.0371 | validation_acc: 0.9888
0.98875
Epoch: 15 | train_loss: 0.0009 | train_acc: 1.0000 | validation_loss: 0.0392 | validation_acc: 0.9862
0.98625
Epoch: 16 | train_loss: 0.0007 | train_acc: 1.0000 | validation_loss: 0.0424 | validation_acc: 0.9875
0.9875
Epoch: 17 | train_loss: 0.0005 | train_acc: 1.0000 | validation_loss: 0.0379 | validation_acc: 0.9862
0.98625
Epoch: 18 | train_loss: 0.0005 | train_acc: 1.0000 | validation_loss: 0.0413 | validation_acc: 0.9862
0.98625
Epoch: 19 | train_loss: 0.0004 | train_acc: 1.0000 | validation_loss: 0.0379 | validation_acc: 0.9862
0.98625
Lack of any improvement
```

Rysunek 19: Badanie 4: Wyniki etapu uczenia.

```
Test loss: 0.05057314498015333 Test acc: 0.9825
```

Rysunek 20: Badanie 4: Wyniki etapu testowania.



Rysunek 21: Badanie 4: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 99,25% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 98,25%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

Zredukowanie bloku konwolucyjnego przyniosło bardzo dobry efekt co sprawiło, że podjeliśmy decyzje o przeprowadzeniu dwóch badań dla zmniejszonej liczby neuronów.

10.1.5 Badanie 5:

- Model z jednym blokiem konwolucyjnymi.
- Jedna warstwa liniowa.
- 5 neuronów w warstwach ukrytych.
- Prosty zbiór danych.
- Rozmiar obrazów 64 x 64.
- Brak augmentacji.
- Liczba epok 100 (uczenie zakończyło się po 10).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100
Strawberry	800	100	100

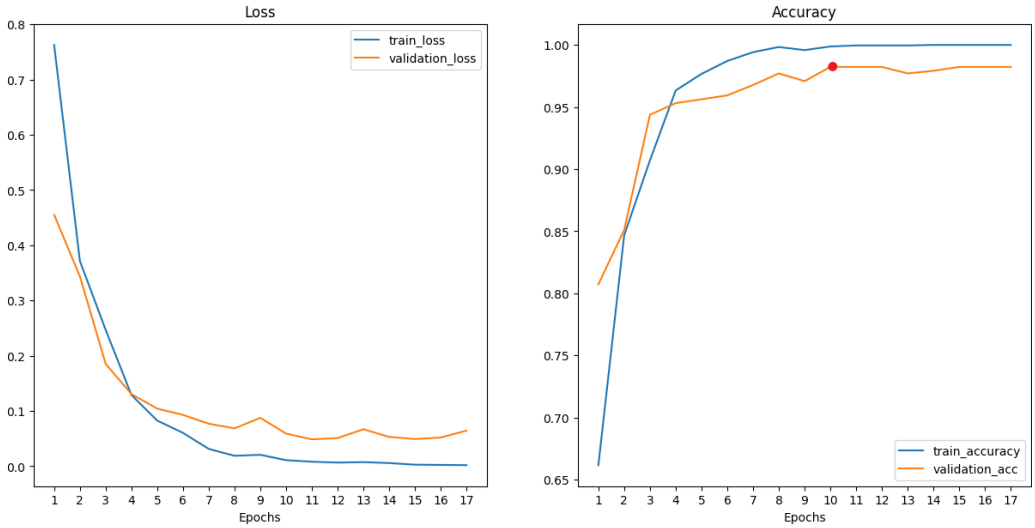
Tabela 5: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 5 | train_loss: 0.0827 | train_acc: 0.9767 | validation_loss: 0.1043 | validation_acc: 0.9563
0.95625
Best model updated at epoch 5 with test_acc: 0.9563
Epoch: 6 | train_loss: 0.0604 | train_acc: 0.9871 | validation_loss: 0.0929 | validation_acc: 0.9594
0.959375
Best model updated at epoch 6 with test_acc: 0.9594
Epoch: 7 | train_loss: 0.0313 | train_acc: 0.9942 | validation_loss: 0.0770 | validation_acc: 0.9677
0.9677083333333334
Best model updated at epoch 7 with test_acc: 0.9677
Epoch: 8 | train_loss: 0.0188 | train_acc: 0.9983 | validation_loss: 0.0685 | validation_acc: 0.9771
0.9770833333333334
Best model updated at epoch 8 with test_acc: 0.9771
Epoch: 9 | train_loss: 0.0207 | train_acc: 0.9958 | validation_loss: 0.0876 | validation_acc: 0.9708
0.9708333333333334
Epoch: 10 | train_loss: 0.0110 | train_acc: 0.9988 | validation_loss: 0.0591 | validation_acc: 0.9823
0.9822916666666666
Best model updated at epoch 10 with test_acc: 0.9823
Epoch: 11 | train_loss: 0.0082 | train_acc: 0.9996 | validation_loss: 0.0487 | validation_acc: 0.9823
0.9822916666666666
Epoch: 12 | train_loss: 0.0066 | train_acc: 0.9996 | validation_loss: 0.0510 | validation_acc: 0.9823
0.9822916666666666
Epoch: 13 | train_loss: 0.0074 | train_acc: 0.9996 | validation_loss: 0.0669 | validation_acc: 0.9771
0.9770833333333334
Epoch: 14 | train_loss: 0.0057 | train_acc: 1.0000 | validation_loss: 0.0531 | validation_acc: 0.9792
0.9791666666666666
Epoch: 15 | train_loss: 0.0028 | train_acc: 1.0000 | validation_loss: 0.0492 | validation_acc: 0.9823
0.9822916666666666
Epoch: 16 | train_loss: 0.0024 | train_acc: 1.0000 | validation_loss: 0.0520 | validation_acc: 0.9823
0.9822916666666666
Epoch: 17 | train_loss: 0.0019 | train_acc: 1.0000 | validation_loss: 0.0644 | validation_acc: 0.9823
0.9822916666666666
Lack of any improvement
```

Rysunek 22: Badanie 5: Wyniki etapu uczenia.

```
Test loss: 0.007734935058397241 Test acc: 1.0
```

Rysunek 23: Badanie 5: Wyniki etapu testowania.



Rysunek 24: Badanie 5: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 98,23% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 100%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.1.6 Badanie 6:

- Model z jednym blokiem konwolucyjnym.
- Jedna warstwa liniowa.
- 3 neuronów w warstwach ukrytych.
- Prosty zbiór danych.
- Rozmiar obrazów 64 x 64.
- Augmentacja.
- Liczba epok 100 (uczenie zakończyło się po 23).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100
Strawberry	800	100	100

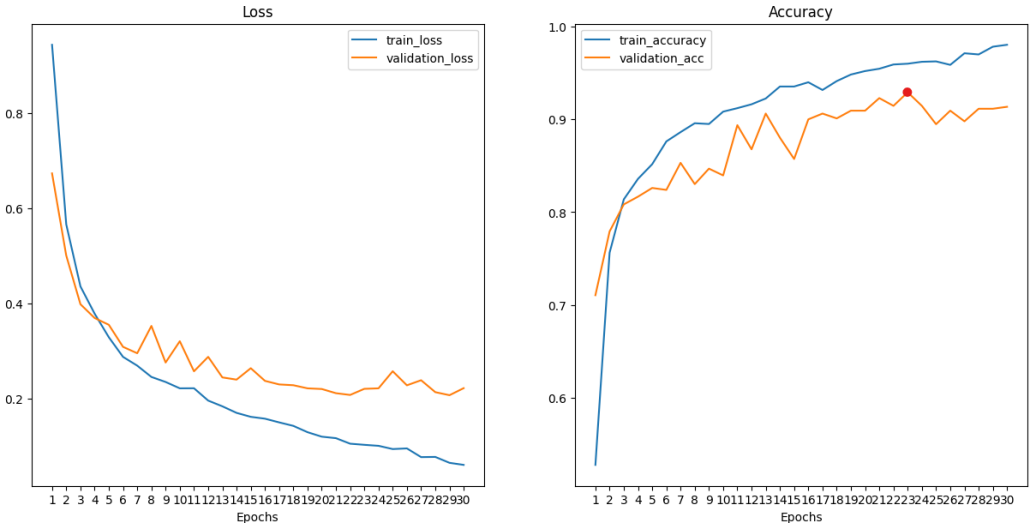
Tabela 6: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 18 | train_loss: 0.1424 | train_acc: 0.9413 | validation_loss: 0.2279 | validation_acc: 0.9010
0.9010416666666666
Epoch: 19 | train_loss: 0.1291 | train_acc: 0.9483 | validation_loss: 0.2213 | validation_acc: 0.9094
0.909375
Best model updated at epoch 19 with test_acc: 0.9094
Epoch: 20 | train_loss: 0.1198 | train_acc: 0.9521 | validation_loss: 0.2199 | validation_acc: 0.9094
0.909375
Epoch: 21 | train_loss: 0.1166 | train_acc: 0.9546 | validation_loss: 0.2111 | validation_acc: 0.9229
0.9229166666666666
Best model updated at epoch 21 with test_acc: 0.9229
Epoch: 22 | train_loss: 0.1050 | train_acc: 0.9592 | validation_loss: 0.2074 | validation_acc: 0.9146
0.9145833333333334
Epoch: 23 | train_loss: 0.1026 | train_acc: 0.9600 | validation_loss: 0.2203 | validation_acc: 0.9292
0.9291666666666666
Best model updated at epoch 23 with test_acc: 0.9292
Epoch: 24 | train_loss: 0.1006 | train_acc: 0.9621 | validation_loss: 0.2213 | validation_acc: 0.9146
0.9145833333333334
Epoch: 25 | train_loss: 0.0937 | train_acc: 0.9625 | validation_loss: 0.2572 | validation_acc: 0.8948
0.8947916666666667
Epoch: 26 | train_loss: 0.0951 | train_acc: 0.9587 | validation_loss: 0.2277 | validation_acc: 0.9094
0.909375
Epoch: 27 | train_loss: 0.0768 | train_acc: 0.9712 | validation_loss: 0.2383 | validation_acc: 0.8979
0.8979166666666666
Epoch: 28 | train_loss: 0.0772 | train_acc: 0.9700 | validation_loss: 0.2133 | validation_acc: 0.9115
0.9114583333333334
Epoch: 29 | train_loss: 0.0648 | train_acc: 0.9783 | validation_loss: 0.2069 | validation_acc: 0.9115
0.9114583333333334
Epoch: 30 | train_loss: 0.0605 | train_acc: 0.9804 | validation_loss: 0.2217 | validation_acc: 0.9135
0.9135416666666666
Lack of any improvement
```

Rysunek 25: Badanie 6: Wyniki etapu uczenia.

```
Test loss: 0.26698165529815016 Test acc: 0.9
```

Rysunek 26: Badanie 6: Wyniki etapu testowania.



Rysunek 27: Badanie 6: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 92,92% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 90%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.1.7 Badanie 7:

- Model z jednym uproszczonym blokiem konwolucyjnym.
- Jedna warstwa liniowa.
- 10 neuronów w warstwach ukrytych.
- Prosty zbiór danych.
- Rozmiar obrazów 64 x 64.
- Augmentacja.
- Liczba epok 100 (uczenie zakończyło się po 6).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100
Strawberry	800	100	100

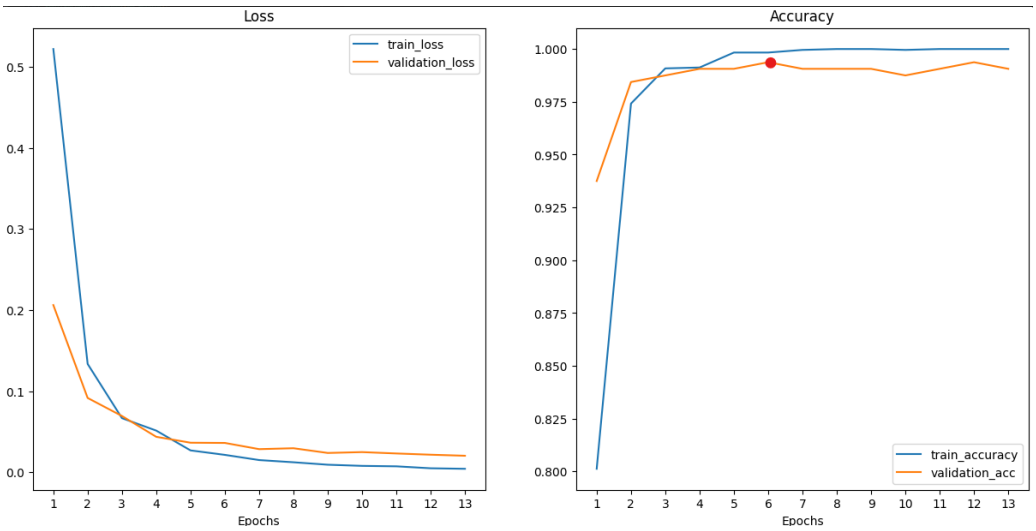
Tabela 7: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 1 | train_loss: 0.5218 | train_acc: 0.8013 | validation_loss: 0.2061 | validation_acc: 0.9375
0.9375
Best model updated at epoch 1 with test_acc: 0.9375
Epoch: 2 | train_loss: 0.1336 | train_acc: 0.9742 | validation_loss: 0.0916 | validation_acc: 0.9844
0.984375
Best model updated at epoch 2 with test_acc: 0.9844
Epoch: 3 | train_loss: 0.0667 | train_acc: 0.9908 | validation_loss: 0.0694 | validation_acc: 0.9875
0.9875
Best model updated at epoch 3 with test_acc: 0.9875
Epoch: 4 | train_loss: 0.0514 | train_acc: 0.9912 | validation_loss: 0.0437 | validation_acc: 0.9906
0.990625
Best model updated at epoch 4 with test_acc: 0.9906
Epoch: 5 | train_loss: 0.0270 | train_acc: 0.9983 | validation_loss: 0.0364 | validation_acc: 0.9906
0.990625
Epoch: 6 | train_loss: 0.0215 | train_acc: 0.9983 | validation_loss: 0.0362 | validation_acc: 0.9938
0.99375
Best model updated at epoch 6 with test_acc: 0.9938
Epoch: 7 | train_loss: 0.0151 | train_acc: 0.9996 | validation_loss: 0.0285 | validation_acc: 0.9906
0.990625
Epoch: 8 | train_loss: 0.0124 | train_acc: 1.0000 | validation_loss: 0.0296 | validation_acc: 0.9906
0.990625
Epoch: 9 | train_loss: 0.0093 | train_acc: 1.0000 | validation_loss: 0.0238 | validation_acc: 0.9906
0.990625
Epoch: 10 | train_loss: 0.0079 | train_acc: 0.9996 | validation_loss: 0.0249 | validation_acc: 0.9875
0.9875
Epoch: 11 | train_loss: 0.0073 | train_acc: 1.0000 | validation_loss: 0.0232 | validation_acc: 0.9906
0.990625
Epoch: 12 | train_loss: 0.0049 | train_acc: 1.0000 | validation_loss: 0.0217 | validation_acc: 0.9938
0.99375
Epoch: 13 | train_loss: 0.0043 | train_acc: 1.0000 | validation_loss: 0.0204 | validation_acc: 0.9906
0.990625
Lack of any improvement
```

Rysunek 28: Badanie 7: Wyniki etapu uczenia.

```
Test loss: 0.010249209415633231 Test acc: 1.0
```

Rysunek 29: Badanie 7: Wyniki etapu testowania.



Rysunek 30: Badanie 7: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 99,38% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 100%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.1.8 Wnioski do badań na prostym zbiorze danych:

- Wysoka skuteczność modeli - Na prostym zbiorze danych modele osiągały bardzo wysokie wyniki, dochodzące do 100% dokładności na danych testowych, co wskazuje, że zbiór jest mało wymagający i łatwy do klasyfikacji.
- Uproszczenie modelu - Redukcja liczby bloków konwolucyjnych w warstwach ukrytych nie wpłynęła negatywnie na skuteczność. Wskazuje to na możliwość stosowania prostszych modeli w przypadku jednolitych danych. Natomiast redukacja liczby neuronów spowodowała już pogorszenie w czasie i dokładności uczenia.
- Znaczenie augmentacji - Wprowadzenie augmentacji nie przyniosło zauważalnej poprawy jakości uczenia modelu. Może to wynikać z faktu, że prosty zbiór danych był na tyle nieskomplikowany, że model skutecznie nauczył się klasyfikacji nawet bez dodatkowych transformacji danych.
- Pomimo zwiększenia liczby klas owoców, model poradził sobie jedynie odrobinę gorzej, co wynika z prostoty zbioru danych. Nawet przy większej liczbie kategorii jakość uczenia nie uległa znacznemu pogorszeniu, co świadczy o skuteczności modelu w pracy z tego typu bazą danych.

10.2 Badania na złożonej bazie danych:

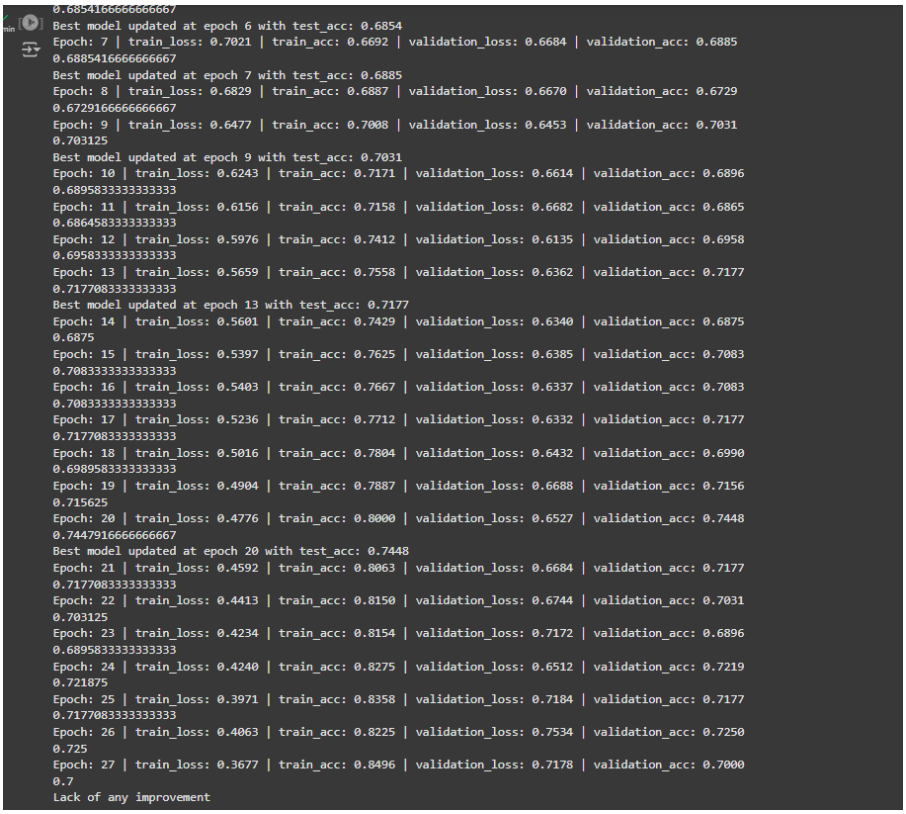
Proces uczenia modelu na bardziej złożonej bazie danych wymagałby większej liczby próbek w porównaniu z prostszą bazą, aby zapewnić lepsze zróżnicowanie danych i poprawę wyników. Jednak ze względu na ograniczony dostęp do dodatkowych obrazów, uczenie musiało zostać przeprowadzone na tej samej liczbie danych w obu przypadkach.

10.2.1 Badanie 8:

- Model z dwoma blokami konwolucyjnymi.
- Jedna warstwa liniowa.
- 10 neuronów w warstwach ukrytych.
- Złożony zbiór danych.
- Rozmiar obrazów 64 x 64.
- Brak augmentacji.
- Liczba epok 100 (uczenie zakończyło się po 20).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100
Strawberry	800	100	100

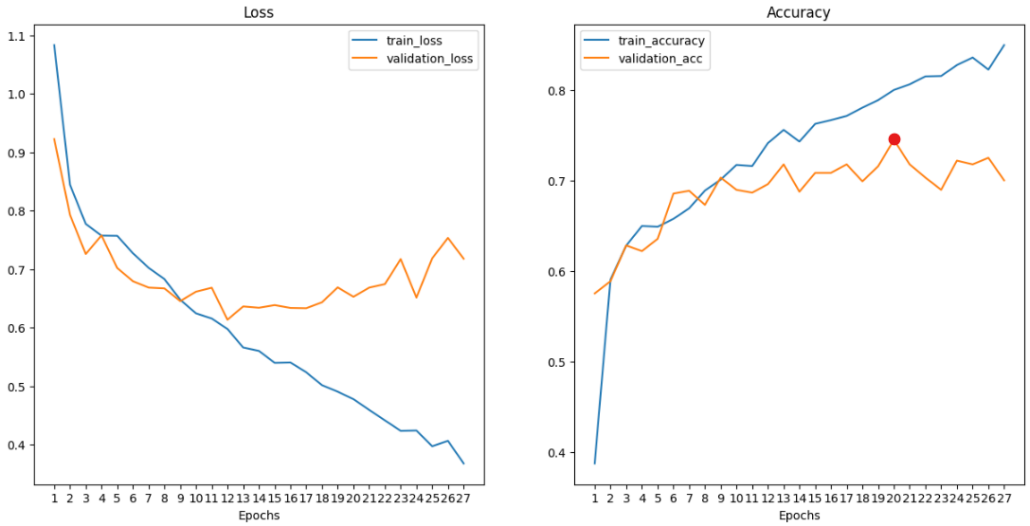
Tabela 8: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.



Rysunek 31: Badanie 8: Wyniki etapu uczenia.

```
Test loss: 0.7315747290849686 Test acc: 0.703125
```

Rysunek 32: Badanie 8: Wyniki etapu testowania.



Rysunek 33: Badanie 8: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 74,48% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 70,31%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.2.2 Badanie 9:

- Model z dwoma blokami konwolucyjnymi.
- Jedna warstwa liniowa.
- 10 neuronów w warstwach ukrytych.
- Złożony zbiór danych.
- Rozmiar obrazów 64 x 64.
- Augmentacja.
- Liczba epok 100 (uczenie zakończyło się po 8).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100
Strawberry	800	100	100

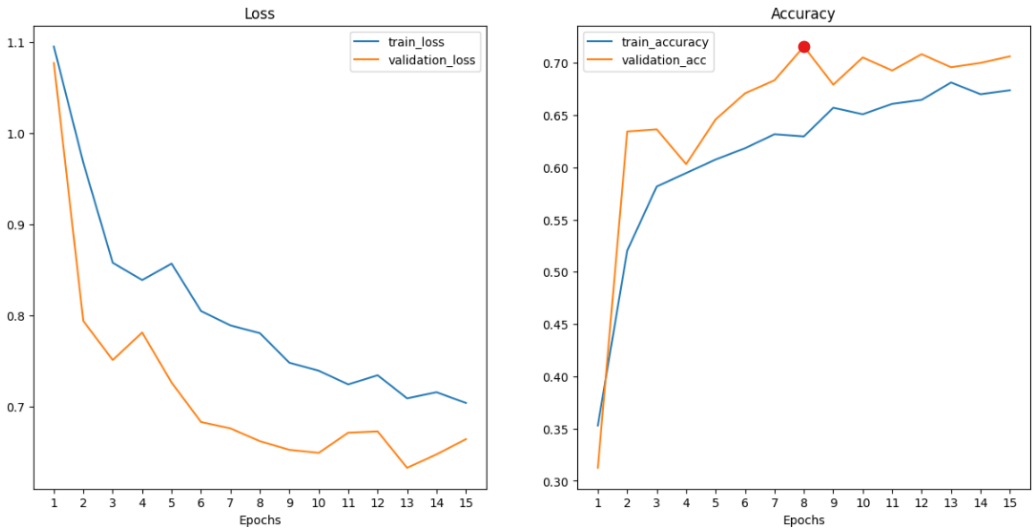
Tabela 9: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 1 | train_loss: 1.0950 | train_acc: 0.3529 | validation_loss: 1.0770 | validation_acc: 0.3125
0.3125
Best model updated at epoch 1 with test_acc: 0.3125
Epoch: 2 | train_loss: 0.9676 | train_acc: 0.5204 | validation_loss: 0.7940 | validation_acc: 0.6344
0.634375
Best model updated at epoch 2 with test_acc: 0.6344
Epoch: 3 | train_loss: 0.8578 | train_acc: 0.5817 | validation_loss: 0.7510 | validation_acc: 0.6365
0.6364583333333333
Best model updated at epoch 3 with test_acc: 0.6365
Epoch: 4 | train_loss: 0.8388 | train_acc: 0.5946 | validation_loss: 0.7812 | validation_acc: 0.6031
0.603125
Epoch: 5 | train_loss: 0.8568 | train_acc: 0.6075 | validation_loss: 0.7263 | validation_acc: 0.6458
0.6458333333333333
Best model updated at epoch 5 with test_acc: 0.6458
Epoch: 6 | train_loss: 0.8048 | train_acc: 0.6183 | validation_loss: 0.6830 | validation_acc: 0.6708
0.6708333333333333
Best model updated at epoch 6 with test_acc: 0.6708
Epoch: 7 | train_loss: 0.7890 | train_acc: 0.6317 | validation_loss: 0.6760 | validation_acc: 0.6833
0.6833333333333333
Best model updated at epoch 7 with test_acc: 0.6833
Epoch: 8 | train_loss: 0.7805 | train_acc: 0.6296 | validation_loss: 0.6620 | validation_acc: 0.7156
0.715625
Best model updated at epoch 8 with test_acc: 0.7156
Epoch: 9 | train_loss: 0.7481 | train_acc: 0.6571 | validation_loss: 0.6525 | validation_acc: 0.6792
0.6791666666666667
Epoch: 10 | train_loss: 0.7393 | train_acc: 0.6508 | validation_loss: 0.6492 | validation_acc: 0.7052
0.7052083333333333
Epoch: 11 | train_loss: 0.7243 | train_acc: 0.6608 | validation_loss: 0.6713 | validation_acc: 0.6927
0.6927083333333333
Epoch: 12 | train_loss: 0.7343 | train_acc: 0.6646 | validation_loss: 0.6727 | validation_acc: 0.7083
0.7083333333333333
Epoch: 13 | train_loss: 0.7090 | train_acc: 0.6813 | validation_loss: 0.6328 | validation_acc: 0.6958
0.6958333333333333
Epoch: 14 | train_loss: 0.7157 | train_acc: 0.6700 | validation_loss: 0.6475 | validation_acc: 0.7000
0.7
Epoch: 15 | train_loss: 0.7040 | train_acc: 0.6737 | validation_loss: 0.6642 | validation_acc: 0.7063
0.70625
Lack of any improvement
```

Rysunek 34: Badanie 9: Wyniki etapu uczenia.

```
Test loss: 0.6874947398900986 Test acc: 0.6927083333333333
```

Rysunek 35: Badanie 9: Wyniki etapu testowania.



Rysunek 36: Badanie 9: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 71,56% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 69,27%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.2.3 Badanie 10:

- Model z dwoma blokami konwolucyjnymi.
- Jedna warstwa liniowa.
- 15 neuronów w warstwach ukrytych.
- Złożony zbiór danych.
- Rozmiar obrazów 64 x 64.
- Augmentacja.
- Liczba epok 100 (uczenie zakończyło się po 24).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100
Strawberry	800	100	100

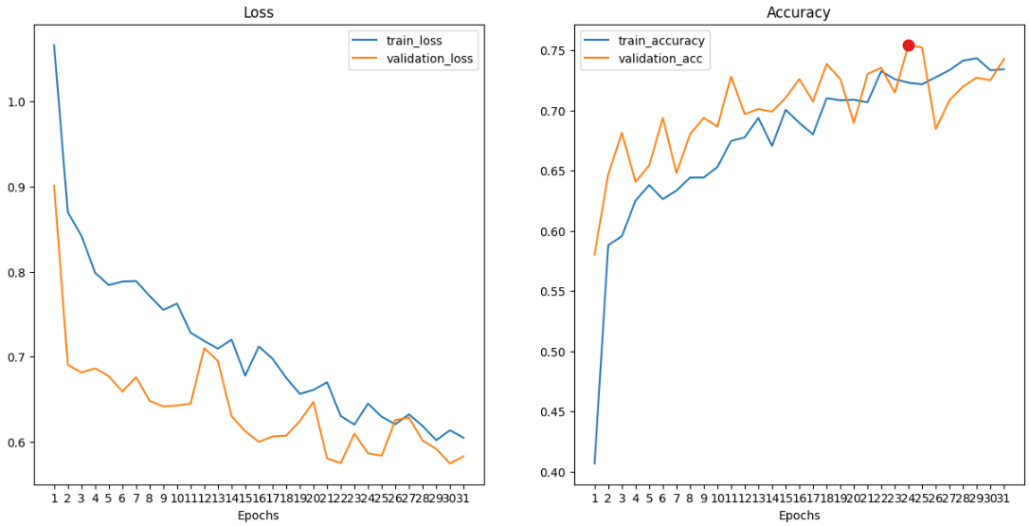
Tabela 10: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 1 | train_loss: 1.0663 | train_acc: 0.4067 | validation_loss: 0.9013 | validation_acc: 0.5802
Best model updated at epoch 1 with test_acc: 0.5802
Epoch: 2 | train_loss: 0.8699 | train_acc: 0.5879 | validation_loss: 0.6905 | validation_acc: 0.6469
Best model updated at epoch 2 with test_acc: 0.6469
Epoch: 3 | train_loss: 0.8422 | train_acc: 0.5954 | validation_loss: 0.6814 | validation_acc: 0.6813
Best model updated at epoch 3 with test_acc: 0.6813
Epoch: 4 | train_loss: 0.7987 | train_acc: 0.6250 | validation_loss: 0.6864 | validation_acc: 0.6406
Epoch: 5 | train_loss: 0.7843 | train_acc: 0.6379 | validation_loss: 0.6773 | validation_acc: 0.6542
Epoch: 6 | train_loss: 0.7884 | train_acc: 0.6262 | validation_loss: 0.6592 | validation_acc: 0.6937
Best model updated at epoch 6 with test_acc: 0.6937
Epoch: 7 | train_loss: 0.7890 | train_acc: 0.6333 | validation_loss: 0.6760 | validation_acc: 0.6479
Epoch: 8 | train_loss: 0.7714 | train_acc: 0.6442 | validation_loss: 0.6481 | validation_acc: 0.6802
Epoch: 9 | train_loss: 0.7551 | train_acc: 0.6442 | validation_loss: 0.6416 | validation_acc: 0.6937
Epoch: 10 | train_loss: 0.7625 | train_acc: 0.6529 | validation_loss: 0.6427 | validation_acc: 0.6865
Epoch: 11 | train_loss: 0.7283 | train_acc: 0.6746 | validation_loss: 0.6450 | validation_acc: 0.7281
Best model updated at epoch 11 with test_acc: 0.7281
Epoch: 12 | train_loss: 0.7186 | train_acc: 0.6775 | validation_loss: 0.7101 | validation_acc: 0.6969
Epoch: 13 | train_loss: 0.7095 | train_acc: 0.6937 | validation_loss: 0.6950 | validation_acc: 0.7010
Epoch: 14 | train_loss: 0.7201 | train_acc: 0.6704 | validation_loss: 0.6301 | validation_acc: 0.6990
Epoch: 15 | train_loss: 0.6778 | train_acc: 0.7004 | validation_loss: 0.6126 | validation_acc: 0.7104
Epoch: 16 | train_loss: 0.7120 | train_acc: 0.6896 | validation_loss: 0.5999 | validation_acc: 0.7260
Epoch: 17 | train_loss: 0.6977 | train_acc: 0.6800 | validation_loss: 0.6064 | validation_acc: 0.7073
Epoch: 18 | train_loss: 0.6753 | train_acc: 0.7100 | validation_loss: 0.6072 | validation_acc: 0.7385
Best model updated at epoch 18 with test_acc: 0.7385
Epoch: 19 | train_loss: 0.6564 | train_acc: 0.7083 | validation_loss: 0.6245 | validation_acc: 0.7260
Epoch: 20 | train_loss: 0.6610 | train_acc: 0.7087 | validation_loss: 0.6468 | validation_acc: 0.6896
Epoch: 21 | train_loss: 0.6700 | train_acc: 0.7067 | validation_loss: 0.5805 | validation_acc: 0.7302
Epoch: 22 | train_loss: 0.6303 | train_acc: 0.7325 | validation_loss: 0.5750 | validation_acc: 0.7354
Epoch: 23 | train_loss: 0.6203 | train_acc: 0.7258 | validation_loss: 0.6096 | validation_acc: 0.7146
Epoch: 24 | train_loss: 0.6450 | train_acc: 0.7229 | validation_loss: 0.5866 | validation_acc: 0.7542
Best model updated at epoch 24 with test_acc: 0.7542
Epoch: 25 | train_loss: 0.6295 | train_acc: 0.7217 | validation_loss: 0.5837 | validation_acc: 0.7521
Epoch: 26 | train_loss: 0.6206 | train_acc: 0.7275 | validation_loss: 0.6256 | validation_acc: 0.6844
Epoch: 27 | train_loss: 0.6324 | train_acc: 0.7333 | validation_loss: 0.6284 | validation_acc: 0.7083
Epoch: 28 | train_loss: 0.6186 | train_acc: 0.7412 | validation_loss: 0.6015 | validation_acc: 0.7198
Epoch: 29 | train_loss: 0.6019 | train_acc: 0.7433 | validation_loss: 0.5917 | validation_acc: 0.7271
Epoch: 30 | train_loss: 0.6137 | train_acc: 0.7333 | validation_loss: 0.5746 | validation_acc: 0.7250
Epoch: 31 | train_loss: 0.6048 | train_acc: 0.7342 | validation_loss: 0.5830 | validation_acc: 0.7427
Lack of any improvement
```

Rysunek 37: Badanie 10: Wyniki etapu uczenia.

```
Test loss: 0.5798727914690971 Test acc: 0.7395833333333333
```

Rysunek 38: Badanie 10: Wyniki etapu testowania.



Rysunek 39: Badanie 10: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 75,42% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 73,96%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.2.4 Badanie 11:

- Model z jednym blokiem konwolucyjnym.
- Jedna warstwa liniowa.
- 10 neuronów w warstwach ukrytych.
- Złożony zbiór danych.
- Rozmiar obrazów 64 x 64.
- Brak augmentacji.
- Liczba epok 100 (uczenie zakończyło się po 8).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100
Strawberry	800	100	100

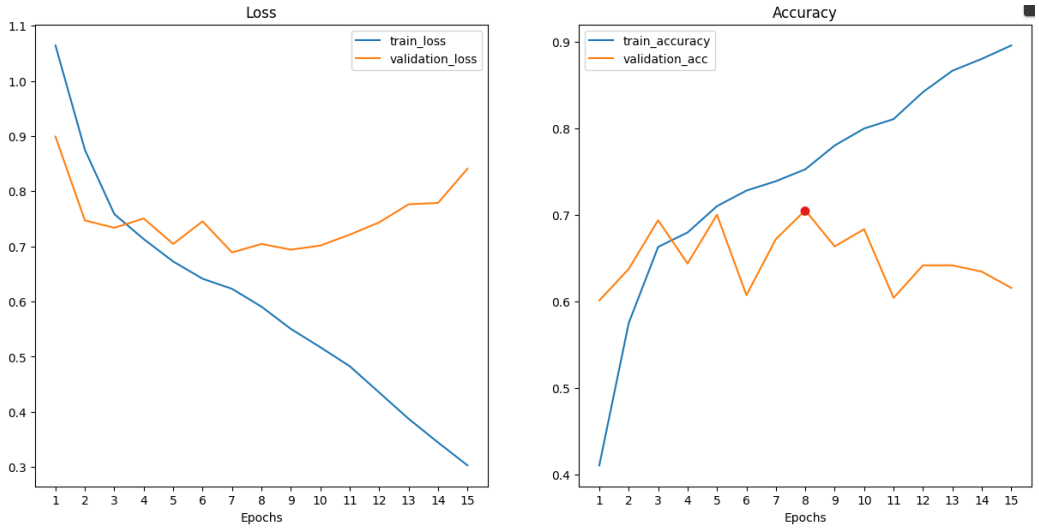
Tabela 11: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 1 | train_loss: 1.0644 | train_acc: 0.4108 | validation_loss: 0.8991 | validation_acc: 0.6010
Best model updated at epoch 1 with test_acc: 0.6010
Epoch: 2 | train_loss: 0.8751 | train_acc: 0.5750 | validation_loss: 0.7470 | validation_acc: 0.6375
Best model updated at epoch 2 with test_acc: 0.6375
Epoch: 3 | train_loss: 0.7585 | train_acc: 0.6629 | validation_loss: 0.7338 | validation_acc: 0.6937
Best model updated at epoch 3 with test_acc: 0.6937
Epoch: 4 | train_loss: 0.7132 | train_acc: 0.6796 | validation_loss: 0.7507 | validation_acc: 0.6438
Epoch: 5 | train_loss: 0.6726 | train_acc: 0.7100 | validation_loss: 0.7044 | validation_acc: 0.7000
Best model updated at epoch 5 with test_acc: 0.7000
Epoch: 6 | train_loss: 0.6414 | train_acc: 0.7279 | validation_loss: 0.7455 | validation_acc: 0.6073
Epoch: 7 | train_loss: 0.6231 | train_acc: 0.7388 | validation_loss: 0.6890 | validation_acc: 0.6719
Epoch: 8 | train_loss: 0.5907 | train_acc: 0.7525 | validation_loss: 0.7045 | validation_acc: 0.7052
Best model updated at epoch 8 with test_acc: 0.7052
Epoch: 9 | train_loss: 0.5504 | train_acc: 0.7800 | validation_loss: 0.6941 | validation_acc: 0.6635
Epoch: 10 | train_loss: 0.5172 | train_acc: 0.7996 | validation_loss: 0.7015 | validation_acc: 0.6833
Epoch: 11 | train_loss: 0.4827 | train_acc: 0.8104 | validation_loss: 0.7213 | validation_acc: 0.6042
Epoch: 12 | train_loss: 0.4351 | train_acc: 0.8417 | validation_loss: 0.7436 | validation_acc: 0.6417
Epoch: 13 | train_loss: 0.3874 | train_acc: 0.8662 | validation_loss: 0.7764 | validation_acc: 0.6417
Epoch: 14 | train_loss: 0.3445 | train_acc: 0.8800 | validation_loss: 0.7789 | validation_acc: 0.6344
Epoch: 15 | train_loss: 0.3030 | train_acc: 0.8954 | validation_loss: 0.8408 | validation_acc: 0.6156
Lack of any improvement
```

Rysunek 40: Badanie 11: Wyniki etapu uczenia.

```
Test loss: 0.7109641915559768 Test acc: 0.67125
```

Rysunek 41: Badanie 11: Wyniki etapu testowania.



Rysunek 42: Badanie 11: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 70,52% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 67,31%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.2.5 Badanie 12:

- Model z jednym blokiem konwolucyjnym.
- Trzy warstwy liniowe.
- 15 neuronów w warstwach ukrytych.
- Złożony zbiór danych.
- Rozmiar obrazów 64 x 64.
- Augmentacja.
- Liczba epok 100 (uczenie zakończyło się po 13).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100
Strawberry	800	100	100

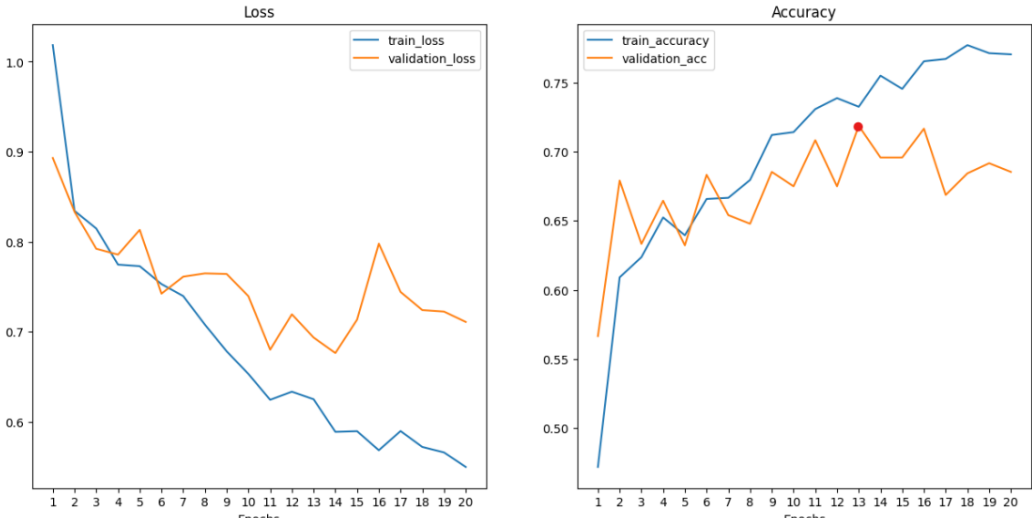
Tabela 12: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 1 | train_loss: 1.0184 | train_acc: 0.4721 | validation_loss: 0.8930 | validation_acc: 0.5667
Best model updated at epoch 1 with test_acc: 0.5667
Epoch: 2 | train_loss: 0.8343 | train_acc: 0.6092 | validation_loss: 0.8333 | validation_acc: 0.6792
Best model updated at epoch 2 with test_acc: 0.6792
Epoch: 3 | train_loss: 0.8147 | train_acc: 0.6238 | validation_loss: 0.7922 | validation_acc: 0.6333
Epoch: 4 | train_loss: 0.7746 | train_acc: 0.6525 | validation_loss: 0.7857 | validation_acc: 0.6646
Epoch: 5 | train_loss: 0.7729 | train_acc: 0.6396 | validation_loss: 0.8132 | validation_acc: 0.6323
Epoch: 6 | train_loss: 0.7530 | train_acc: 0.6658 | validation_loss: 0.7423 | validation_acc: 0.6833
Best model updated at epoch 6 with test_acc: 0.6833
Epoch: 7 | train_loss: 0.7397 | train_acc: 0.6667 | validation_loss: 0.7612 | validation_acc: 0.6542
Epoch: 8 | train_loss: 0.7079 | train_acc: 0.6796 | validation_loss: 0.7648 | validation_acc: 0.6479
Epoch: 9 | train_loss: 0.6783 | train_acc: 0.7121 | validation_loss: 0.7642 | validation_acc: 0.6854
Best model updated at epoch 9 with test_acc: 0.6854
Epoch: 10 | train_loss: 0.6532 | train_acc: 0.7142 | validation_loss: 0.7395 | validation_acc: 0.6750
Epoch: 11 | train_loss: 0.6245 | train_acc: 0.7308 | validation_loss: 0.6801 | validation_acc: 0.7083
Best model updated at epoch 11 with test_acc: 0.7083
Epoch: 12 | train_loss: 0.6335 | train_acc: 0.7388 | validation_loss: 0.7194 | validation_acc: 0.6750
Epoch: 13 | train_loss: 0.6251 | train_acc: 0.7325 | validation_loss: 0.6936 | validation_acc: 0.7188
Best model updated at epoch 13 with test_acc: 0.7188
Epoch: 14 | train_loss: 0.5889 | train_acc: 0.7550 | validation_loss: 0.6764 | validation_acc: 0.6958
Epoch: 15 | train_loss: 0.5896 | train_acc: 0.7454 | validation_loss: 0.7135 | validation_acc: 0.6958
Epoch: 16 | train_loss: 0.5684 | train_acc: 0.7654 | validation_loss: 0.7979 | validation_acc: 0.7167
Epoch: 17 | train_loss: 0.5898 | train_acc: 0.7671 | validation_loss: 0.7442 | validation_acc: 0.6687
Epoch: 18 | train_loss: 0.5721 | train_acc: 0.7771 | validation_loss: 0.7241 | validation_acc: 0.6844
Epoch: 19 | train_loss: 0.5660 | train_acc: 0.7712 | validation_loss: 0.7224 | validation_acc: 0.6917
Epoch: 20 | train_loss: 0.5499 | train_acc: 0.7704 | validation_loss: 0.7109 | validation_acc: 0.6854
Lack of any improvement
```

Rysunek 43: Badanie 12: Wyniki etapu uczenia.

```
Test loss: 0.6725665831565857 Test acc: 0.69625
```

Rysunek 44: Badanie 12: Wyniki etapu testowania.



Rysunek 45: Badanie 12: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 71,88% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 65,62%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.2.6 Badanie 13:

- Model z trzema blokami konwolucyjnymi.
- Trzy warstwy liniowe.
- 15 neuronów w warstwach ukrytych.
- Złożony zbiór danych.
- Rozmiar obrazów 64 x 64.
- Augmentacja.
- Liczba epok 100 (uczenie zakończyło się po 51).
- Liczba owoców do rozpoznania: 3.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100
Strawberry	800	100	100

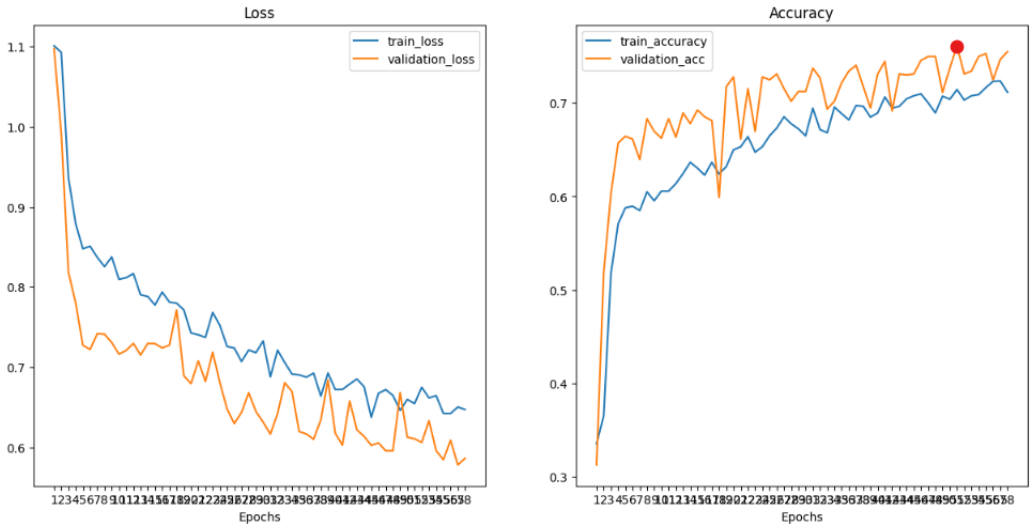
Tabela 13: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 15 | train_loss: 0.7776 | train_acc: 0.6304 | validation_loss: 0.7295 | validation_acc: 0.6927
Best model updated at epoch 15 with test_acc: 0.6927
Epoch: 16 | train_loss: 0.7938 | train_acc: 0.6229 | validation_loss: 0.7241 | validation_acc: 0.6854
Epoch: 17 | train_loss: 0.7813 | train_acc: 0.6367 | validation_loss: 0.7275 | validation_acc: 0.6813
Epoch: 18 | train_loss: 0.7799 | train_acc: 0.6242 | validation_loss: 0.7714 | validation_acc: 0.5990
Epoch: 19 | train_loss: 0.7715 | train_acc: 0.6321 | validation_loss: 0.6892 | validation_acc: 0.7177
Best model updated at epoch 19 with test_acc: 0.7177
Epoch: 20 | train_loss: 0.7426 | train_acc: 0.6500 | validation_loss: 0.6795 | validation_acc: 0.7281
Best model updated at epoch 20 with test_acc: 0.7281
Epoch: 21 | train_loss: 0.7405 | train_acc: 0.6533 | validation_loss: 0.7078 | validation_acc: 0.6615
Epoch: 22 | train_loss: 0.7372 | train_acc: 0.6642 | validation_loss: 0.6822 | validation_acc: 0.7156
Epoch: 23 | train_loss: 0.7684 | train_acc: 0.6475 | validation_loss: 0.7187 | validation_acc: 0.6698
Epoch: 24 | train_loss: 0.7522 | train_acc: 0.6533 | validation_loss: 0.6809 | validation_acc: 0.7281
Epoch: 25 | train_loss: 0.7258 | train_acc: 0.6650 | validation_loss: 0.6477 | validation_acc: 0.7250
Epoch: 26 | train_loss: 0.7239 | train_acc: 0.6733 | validation_loss: 0.6296 | validation_acc: 0.7312
Best model updated at epoch 26 with test_acc: 0.7312
Epoch: 27 | train_loss: 0.7072 | train_acc: 0.6854 | validation_loss: 0.6435 | validation_acc: 0.7156
Epoch: 28 | train_loss: 0.7214 | train_acc: 0.6779 | validation_loss: 0.6681 | validation_acc: 0.7021
Epoch: 29 | train_loss: 0.7181 | train_acc: 0.6725 | validation_loss: 0.6447 | validation_acc: 0.7125
Epoch: 30 | train_loss: 0.7328 | train_acc: 0.6650 | validation_loss: 0.6314 | validation_acc: 0.7125
Epoch: 31 | train_loss: 0.6880 | train_acc: 0.6946 | validation_loss: 0.6165 | validation_acc: 0.7375
Best model updated at epoch 31 with test_acc: 0.7375
Epoch: 32 | train_loss: 0.7213 | train_acc: 0.6717 | validation_loss: 0.6422 | validation_acc: 0.7271
Epoch: 33 | train_loss: 0.7058 | train_acc: 0.6683 | validation_loss: 0.6806 | validation_acc: 0.6937
Epoch: 34 | train_loss: 0.6915 | train_acc: 0.6958 | validation_loss: 0.6695 | validation_acc: 0.7021
Epoch: 35 | train_loss: 0.6904 | train_acc: 0.6887 | validation_loss: 0.6197 | validation_acc: 0.7219
Epoch: 36 | train_loss: 0.6875 | train_acc: 0.6821 | validation_loss: 0.6167 | validation_acc: 0.7344
Epoch: 37 | train_loss: 0.6926 | train_acc: 0.6975 | validation_loss: 0.6100 | validation_acc: 0.7406
Best model updated at epoch 37 with test_acc: 0.7406
Epoch: 38 | train_loss: 0.6639 | train_acc: 0.6967 | validation_loss: 0.6343 | validation_acc: 0.7177
Epoch: 39 | train_loss: 0.6927 | train_acc: 0.6850 | validation_loss: 0.6840 | validation_acc: 0.6948
Epoch: 40 | train_loss: 0.6721 | train_acc: 0.6896 | validation_loss: 0.6176 | validation_acc: 0.7312
Epoch: 41 | train_loss: 0.6722 | train_acc: 0.7067 | validation_loss: 0.6027 | validation_acc: 0.7448
Best model updated at epoch 41 with test_acc: 0.7448
Epoch: 42 | train_loss: 0.6790 | train_acc: 0.6946 | validation_loss: 0.6579 | validation_acc: 0.6917
Epoch: 43 | train_loss: 0.6852 | train_acc: 0.6967 | validation_loss: 0.6219 | validation_acc: 0.7312
Epoch: 44 | train_loss: 0.6757 | train_acc: 0.7046 | validation_loss: 0.6136 | validation_acc: 0.7302
Epoch: 45 | train_loss: 0.6375 | train_acc: 0.7079 | validation_loss: 0.6025 | validation_acc: 0.7312
Epoch: 46 | train_loss: 0.6674 | train_acc: 0.7100 | validation_loss: 0.6054 | validation_acc: 0.7458
Best model updated at epoch 46 with test_acc: 0.7458
Epoch: 47 | train_loss: 0.6720 | train_acc: 0.7004 | validation_loss: 0.5958 | validation_acc: 0.7500
Best model updated at epoch 47 with test_acc: 0.7500
Epoch: 48 | train_loss: 0.6648 | train_acc: 0.6896 | validation_loss: 0.5958 | validation_acc: 0.7500
Epoch: 49 | train_loss: 0.6461 | train_acc: 0.7075 | validation_loss: 0.6683 | validation_acc: 0.7115
Epoch: 50 | train_loss: 0.6598 | train_acc: 0.7042 | validation_loss: 0.6125 | validation_acc: 0.7375
Epoch: 51 | train_loss: 0.6546 | train_acc: 0.7146 | validation_loss: 0.6107 | validation_acc: 0.7615
Best model updated at epoch 51 with test_acc: 0.7615
Epoch: 52 | train_loss: 0.6749 | train_acc: 0.7033 | validation_loss: 0.6060 | validation_acc: 0.7312
Epoch: 53 | train_loss: 0.6617 | train_acc: 0.7079 | validation_loss: 0.6332 | validation_acc: 0.7344
Epoch: 54 | train_loss: 0.6645 | train_acc: 0.7092 | validation_loss: 0.5959 | validation_acc: 0.7500
Epoch: 55 | train_loss: 0.6423 | train_acc: 0.7167 | validation_loss: 0.5845 | validation_acc: 0.7531
Epoch: 56 | train_loss: 0.6422 | train_acc: 0.7233 | validation_loss: 0.6087 | validation_acc: 0.7250
Epoch: 57 | train_loss: 0.6503 | train_acc: 0.7238 | validation_loss: 0.5780 | validation_acc: 0.7469
Epoch: 58 | train_loss: 0.6472 | train_acc: 0.7117 | validation_loss: 0.5862 | validation_acc: 0.7552
Lack of any improvement
```

Rysunek 46: Badanie 13: Wyniki etapu uczenia.

```
Test loss: 0.5981987044215202 Test acc: 0.7416666666666667
```

Rysunek 47: Badanie 13: Wyniki etapu testowania.



Rysunek 48: Badanie 13: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 74,16% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 76,15%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.2.7 Badanie 14:

- Model z jednym blokamiem konwolucyjnymi.
- Trzy warstwy liniowe.
- 10 neuronów w warstwach ukrytych.
- Złożony zbiór danych.
- Rozmiar obrazów 64 x 64.
- Augmentacja.
- Liczba epok 100 (uczenie zakończyło się po 9).
- Liczba owoców do rozpoznania: 2.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100

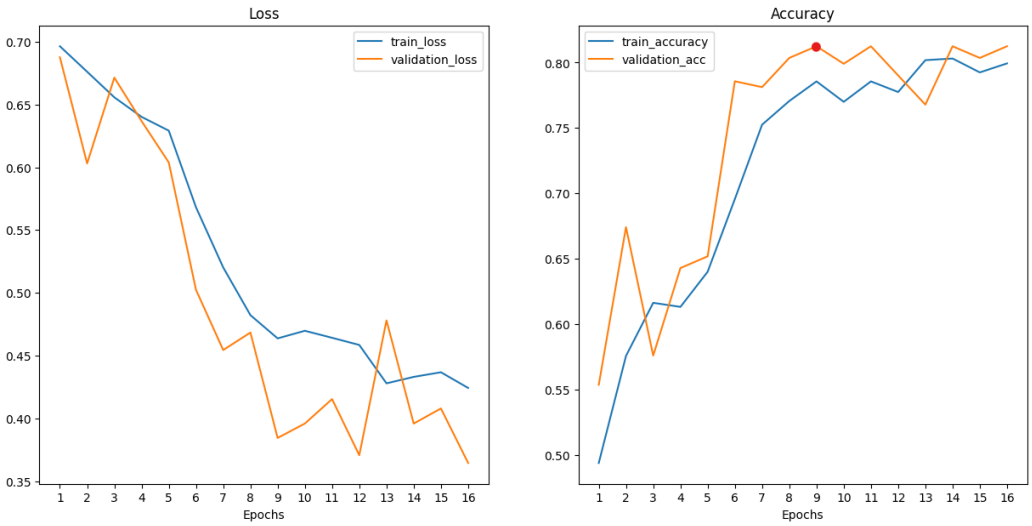
Tabela 14: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 1 | train_loss: 0.6964 | train_acc: 0.4938 | validation_loss: 0.6876 | validation_acc: 0.5536
Best model updated at epoch 1 with test_acc: 0.5536
Epoch: 2 | train_loss: 0.6761 | train_acc: 0.5756 | validation_loss: 0.6031 | validation_acc: 0.6741
Best model updated at epoch 2 with test_acc: 0.6741
Epoch: 3 | train_loss: 0.6557 | train_acc: 0.6162 | validation_loss: 0.6714 | validation_acc: 0.5759
Epoch: 4 | train_loss: 0.6401 | train_acc: 0.6131 | validation_loss: 0.6368 | validation_acc: 0.6429
Epoch: 5 | train_loss: 0.6292 | train_acc: 0.6400 | validation_loss: 0.6039 | validation_acc: 0.6518
Epoch: 6 | train_loss: 0.5682 | train_acc: 0.6956 | validation_loss: 0.5025 | validation_acc: 0.7857
Best model updated at epoch 6 with test_acc: 0.7857
Epoch: 7 | train_loss: 0.5203 | train_acc: 0.7525 | validation_loss: 0.4545 | validation_acc: 0.7812
Epoch: 8 | train_loss: 0.4823 | train_acc: 0.7706 | validation_loss: 0.4684 | validation_acc: 0.8036
Best model updated at epoch 8 with test_acc: 0.8036
Epoch: 9 | train_loss: 0.4637 | train_acc: 0.7856 | validation_loss: 0.3845 | validation_acc: 0.8125
Best model updated at epoch 9 with test_acc: 0.8125
Epoch: 10 | train_loss: 0.4698 | train_acc: 0.7700 | validation_loss: 0.3959 | validation_acc: 0.7991
Epoch: 11 | train_loss: 0.4642 | train_acc: 0.7856 | validation_loss: 0.4154 | validation_acc: 0.8125
Epoch: 12 | train_loss: 0.4585 | train_acc: 0.7775 | validation_loss: 0.3707 | validation_acc: 0.7902
Epoch: 13 | train_loss: 0.4280 | train_acc: 0.8019 | validation_loss: 0.4781 | validation_acc: 0.7679
Epoch: 14 | train_loss: 0.4331 | train_acc: 0.8031 | validation_loss: 0.3959 | validation_acc: 0.8125
Epoch: 15 | train_loss: 0.4368 | train_acc: 0.7925 | validation_loss: 0.4079 | validation_acc: 0.8036
Epoch: 16 | train_loss: 0.4243 | train_acc: 0.7994 | validation_loss: 0.3645 | validation_acc: 0.8125
Lack of any improvement
```

Rysunek 49: Badanie 14: Wyniki etapu uczenia.

```
Test loss: 0.4430443729673113 Test acc: 0.8080357142857143
```

Rysunek 50: Badanie 14: Wyniki etapu testowania.



Rysunek 51: Badanie 14: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 81,25% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 80,80%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.2.8 Badanie 15:

- Model z trzema blokami konwolucyjnymi.
- Trzy warstwy liniowe.
- 10 neuronów w warstwach ukrytych.
- Złożony zbiór danych.
- Rozmiar obrazów 64 x 64.
- Augmentacja.
- Liczba epok 100 (uczenie zakończyło się po 43).
- Liczba owoców do rozpoznania: 2.

Kategoria	Zbiór treningowy	Zbiór walidacyjny	Zbiór testowy
Apple	800	100	100
Bannana	800	100	100

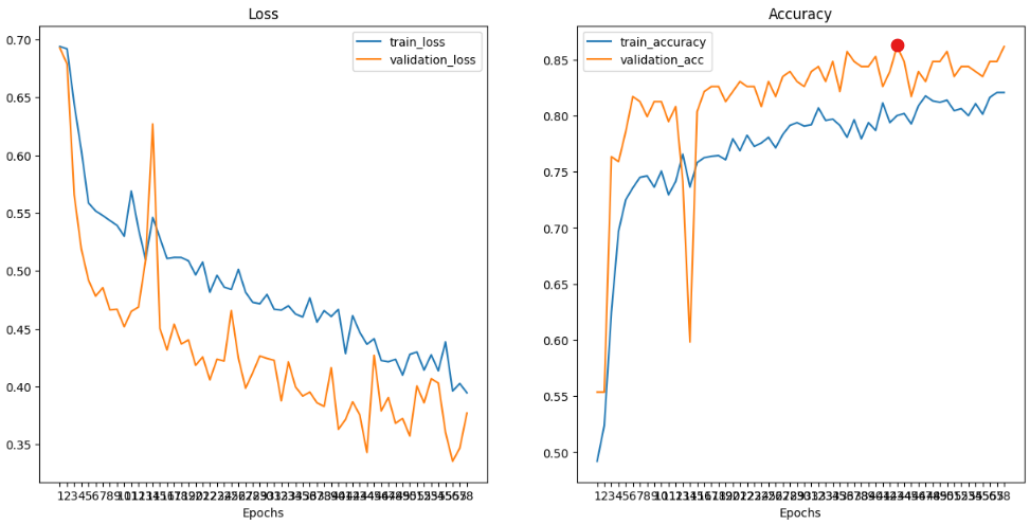
Tabela 15: Spis owoców wraz z liczbą obrazów w poszczególnych zbiorach danych.

```
Epoch: 6 | train_loss: 0.5517 | train_acc: 0.7356 | validation_loss: 0.4781 | validation_acc: 0.8170
Best model updated at epoch 6 with test_acc: 0.8170
Epoch: 7 | train_loss: 0.5478 | train_acc: 0.7450 | validation_loss: 0.4854 | validation_acc: 0.8125
Epoch: 8 | train_loss: 0.5435 | train_acc: 0.7462 | validation_loss: 0.4662 | validation_acc: 0.7991
Epoch: 9 | train_loss: 0.5393 | train_acc: 0.7362 | validation_loss: 0.4668 | validation_acc: 0.8125
Epoch: 10 | train_loss: 0.5300 | train_acc: 0.7506 | validation_loss: 0.4517 | validation_acc: 0.8125
Epoch: 11 | train_loss: 0.5690 | train_acc: 0.7294 | validation_loss: 0.4650 | validation_acc: 0.7946
Epoch: 12 | train_loss: 0.5363 | train_acc: 0.7412 | validation_loss: 0.4687 | validation_acc: 0.8080
Epoch: 13 | train_loss: 0.5098 | train_acc: 0.7656 | validation_loss: 0.5093 | validation_acc: 0.7411
Epoch: 14 | train_loss: 0.5461 | train_acc: 0.7362 | validation_loss: 0.6269 | validation_acc: 0.5982
Epoch: 15 | train_loss: 0.5284 | train_acc: 0.7581 | validation_loss: 0.4502 | validation_acc: 0.8036
Epoch: 16 | train_loss: 0.5107 | train_acc: 0.7625 | validation_loss: 0.4317 | validation_acc: 0.8214
Best model updated at epoch 16 with test_acc: 0.8214
Epoch: 17 | train_loss: 0.5116 | train_acc: 0.7638 | validation_loss: 0.4539 | validation_acc: 0.8259
Best model updated at epoch 17 with test_acc: 0.8259
Epoch: 18 | train_loss: 0.5116 | train_acc: 0.7644 | validation_loss: 0.4368 | validation_acc: 0.8259
Epoch: 19 | train_loss: 0.5086 | train_acc: 0.7606 | validation_loss: 0.4403 | validation_acc: 0.8125
Epoch: 20 | train_loss: 0.4965 | train_acc: 0.7794 | validation_loss: 0.4184 | validation_acc: 0.8214
Epoch: 21 | train_loss: 0.5077 | train_acc: 0.7688 | validation_loss: 0.4256 | validation_acc: 0.8304
Best model updated at epoch 21 with test_acc: 0.8304
Epoch: 22 | train_loss: 0.4815 | train_acc: 0.7825 | validation_loss: 0.4058 | validation_acc: 0.8259
Epoch: 23 | train_loss: 0.4961 | train_acc: 0.7725 | validation_loss: 0.4236 | validation_acc: 0.8259
Epoch: 24 | train_loss: 0.4860 | train_acc: 0.7756 | validation_loss: 0.4220 | validation_acc: 0.8080
Epoch: 25 | train_loss: 0.4839 | train_acc: 0.7806 | validation_loss: 0.4657 | validation_acc: 0.8304
Epoch: 26 | train_loss: 0.5012 | train_acc: 0.7712 | validation_loss: 0.4246 | validation_acc: 0.8170
Epoch: 27 | train_loss: 0.4816 | train_acc: 0.7831 | validation_loss: 0.3986 | validation_acc: 0.8348
Best model updated at epoch 27 with test_acc: 0.8348
Epoch: 28 | train_loss: 0.4728 | train_acc: 0.7913 | validation_loss: 0.4119 | validation_acc: 0.8393
Best model updated at epoch 28 with test_acc: 0.8393
Epoch: 29 | train_loss: 0.4715 | train_acc: 0.7937 | validation_loss: 0.4264 | validation_acc: 0.8304
Epoch: 30 | train_loss: 0.4796 | train_acc: 0.7906 | validation_loss: 0.4243 | validation_acc: 0.8259
Epoch: 31 | train_loss: 0.4668 | train_acc: 0.7919 | validation_loss: 0.4227 | validation_acc: 0.8393
Epoch: 32 | train_loss: 0.4661 | train_acc: 0.8069 | validation_loss: 0.3878 | validation_acc: 0.8438
Best model updated at epoch 32 with test_acc: 0.8438
Epoch: 33 | train_loss: 0.4698 | train_acc: 0.7956 | validation_loss: 0.4213 | validation_acc: 0.8304
Epoch: 34 | train_loss: 0.4626 | train_acc: 0.7969 | validation_loss: 0.3997 | validation_acc: 0.8482
Best model updated at epoch 34 with test_acc: 0.8482
Epoch: 35 | train_loss: 0.4602 | train_acc: 0.7913 | validation_loss: 0.3918 | validation_acc: 0.8214
Epoch: 36 | train_loss: 0.4766 | train_acc: 0.7806 | validation_loss: 0.3952 | validation_acc: 0.8571
Best model updated at epoch 36 with test_acc: 0.8571
Epoch: 37 | train_loss: 0.4556 | train_acc: 0.7963 | validation_loss: 0.3860 | validation_acc: 0.8482
Epoch: 38 | train_loss: 0.4656 | train_acc: 0.7794 | validation_loss: 0.3829 | validation_acc: 0.8438
Epoch: 39 | train_loss: 0.4606 | train_acc: 0.7937 | validation_loss: 0.4164 | validation_acc: 0.8438
Epoch: 40 | train_loss: 0.4667 | train_acc: 0.7869 | validation_loss: 0.3630 | validation_acc: 0.8527
Epoch: 41 | train_loss: 0.4285 | train_acc: 0.8113 | validation_loss: 0.3716 | validation_acc: 0.8259
Epoch: 42 | train_loss: 0.4612 | train_acc: 0.7937 | validation_loss: 0.3868 | validation_acc: 0.8393
Epoch: 43 | train_loss: 0.4469 | train_acc: 0.8000 | validation_loss: 0.3755 | validation_acc: 0.8616
Best model updated at epoch 43 with test_acc: 0.8616
```

Rysunek 52: Badanie 15: Wyniki etapu uczenia.

```
Test loss: 0.4163668139491763 Test acc: 0.8233928571428571
```

Rysunek 53: Badanie 15: Wyniki etapu testowania.



Rysunek 54: Badanie 15: Wizualizacja przebiegu procesu uczenia.

Podczas uczenia model osiągnał 86,16% skuteczności na danych walidacyjnych, natomiast podczas ostatecznego testu 82,33%. Czerowna kropka na wykresie pokazuje moment, od którego nastąpiło przeuczanie się modelu.

10.2.9 Wnioski do badań na złożonym zbiorze danych:

- Wyższe wymagania dla modeli - Złożoność danych (złożone tła, różnorodne oświetlenie, niepełne owoce) obniżyła skuteczność modeli, co wskazuje na potrzebę bardziej zaawansowanych architektur.
- Zastosowanie augmentacji - Augmentacja danych przyczyniła się do poprawy wyników modeli, co sugeruje, że w przypadku trudnych danych jest to kluczowy element w procesie przygotowania danych.
- Lepsze wyniki z bardziej złożonymi modelami - Modele z większą liczbą bloków konwolucyjnych oraz neuronów w warstwach ukrytych osiągały lepsze wyniki na złożonym zbiorze danych, co potwierdza konieczność dostosowania architektury do trudności problemu. Powodowały też one jednak większy czas osiągania dobrego train_acc w procesie uczenia.
- Dla złożonej bazy danych najlepszym rozwiązaniem okazało się ograniczenie klasyfikacji do dwóch rodzajów owoców oraz wprowadzenie trzech bloków konwolucyjnych i trzech warstw liniowych. Taka konfiguracja spowodowała wzrost skuteczności procesu uczenia do 86%, co oznacza poprawę aż o około 10% w porównaniu do wcześniejszych podejść. Dodatkowo, zastosowanie trzech bloków konwolucyjnych przyniosło poprawę wyników o 2% w porównaniu do architektury z jednym blokiem, co pokazuje znaczenie bardziej złożonej architektury w przypadku trudnych danych.

11 Wnioski:

- Znaczenie złożoności modelu - Wyniki badań pokazują, że bardziej złożone architektury, takie jak modele z większą liczbą bloków konwolucyjnych i warstw liniowych, są lepiej przystosowane do pracy z trudniejszymi zbiorami danych. Proste modele natomiast świetnie radzą sobie z jednolitymi zbiorami, co pozwala na optymalizację zasobów obliczeniowych, a co za tym idzie skrócenie czasu uczenia.
- Znaczenie odpowiedniego doboru parametrów - Optymalna liczba epok i zastosowanie mechanizmów wczesnego zatrzymania zapobiegły przeuczeniu modelu i pozwoliły osiągnąć najlepsze wyniki.
- Dla obu baz danych przeprowadzone badania z różnymi wartościami współczynnika uczenia (lr) optymalizatora Adam wykazały, że jedynie domyślna wartość $lr = 0.001$ była skuteczna. Zastosowanie innych wartości nie tylko nie poprawiało jakości uczenia, ale w niektórych przypadkach prowadziło do znacznego pogorszenia wyników.
- Wpływ augmentacji - Augmentacja miała ograniczony wpływ na poprawę jakości uczenia dla prostego zbioru danych, co wynikało z jego prostoty. Natomiast w przypadku bardziej złożonego zbioru mogła przyczynić się do lepszego dostosowania modelu do zmiennych warunków.

Kod: https://github.com/Bergu1/fruits-classification