Łukasz Kielczyk

Nr indeksu 217611

Klasyfikacja gatunków warzyw przy użyciu Convolutional Neural Network i CNN + Random Forest

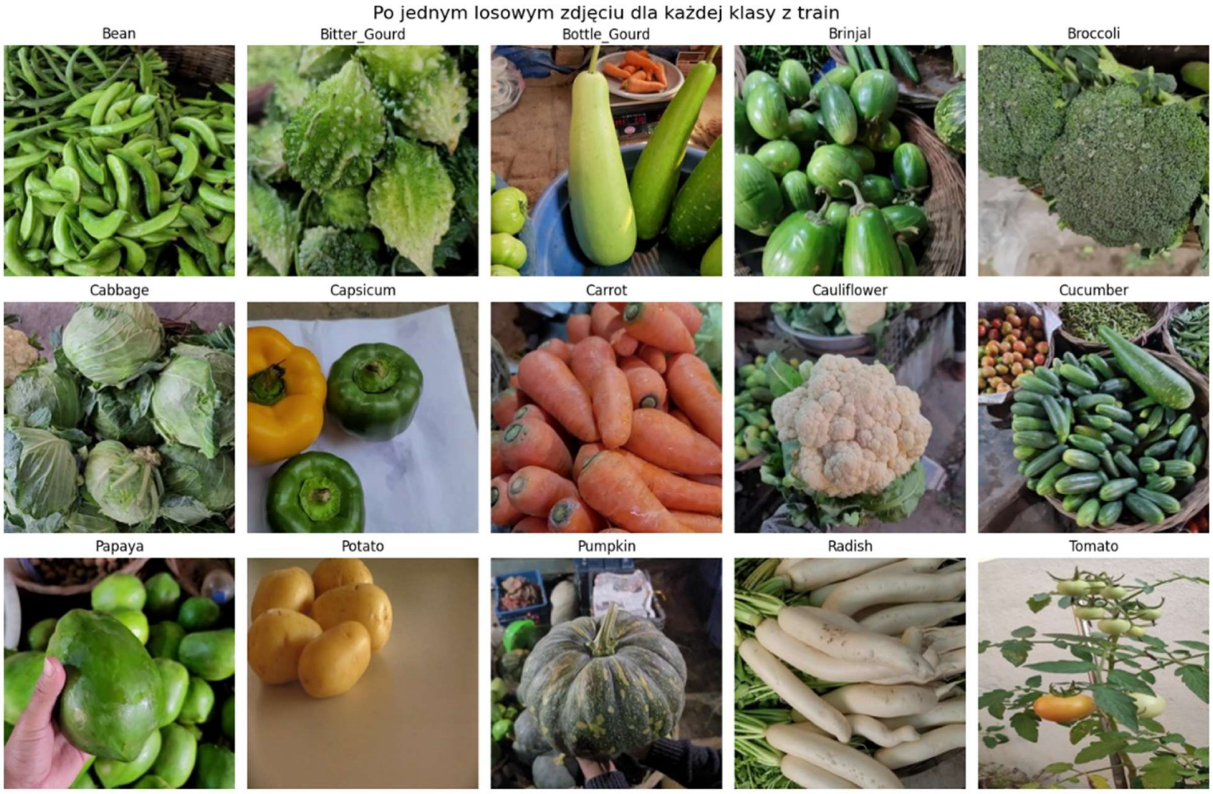
1. Charakterystyka zadania

Projekt dotyczy automatycznej klasyfikacji obrazów do jednej z 15 klas określających rodzaje warzyw. Do projektu wykorzystane zostały dane pobrane z Kaggle: https://www.kaggle.com/datasets/misrakahmed/vegetable-image-dataset

2. Przygotowanie Danych

Dataset zawiera obrazy o rozmiarach 224 x 224 w formacie jpg. Łącznie 21000 obrazow podzielonych na 15 klas, oraz 3 zbiory: ‘train’, ‘test’, ‘validation’.

Klasa	Train	Valid	Test	Razem
Bean	1000	200	200	1400
Bitter_Gourd	1000	200	200	1400
Bottle_Gourd	1000	200	200	1400
Brinjal	1000	200	200	1400
Broccoli	1000	200	200	1400
Cabbage	1000	200	200	1400
Capsicum	1000	200	200	1400
Carrot	1000	200	200	1400
Cauliflower	1000	200	200	1400
Cucumber	1000	200	200	1400
Papaya	1000	200	200	1400
Potato	1000	200	200	1400
Pumpkin	1000	200	200	1400
Radish	1000	200	200	1400
Tomato	1000	200	200	1400
RAZEM	15000	3000	3000	21000



2.1. Preprocessing danych

Zastosowano następujące techniki przetwarzania obrazów:

```
transforms.Resize((64, 64)),
transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2),
transforms.RandomHorizontalFlip(),
transforms.RandomRotation(15),
transforms.ToTensor(),
transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
```

3. Metodologia

3.1 Architektura CNN

Warstwy konwolucyjne:

- Conv1: 3→48 kanałów, kernel 4×4, stride=2

- Conv2: 48→64 kanałów, kernel 2×2

- Conv3: 64→128 kanałów, kernel 3×3

- Conv4: 128→256 kanałów, kernel 2×2, stride=2

Warstwy normalizacyjne i regularyzacyjne:

- Batch Normalization po każdej warstwie konwolucyjnej

- MaxPooling 2×2 między warstwami

- Dropout (p=0.2) przed klasyfikatorem

Warstwy w pełni połączone:

- Flatten: spłaszczenie map cech (256×2×2 → 1024)

- FC1: 1024→64 neuronów z Batch Normalization

- FC2: 64→15 neuronów (liczba klas)

```
self.conv1 = nn.Conv2d(3, 48, 4, stride=2, padding=1)
self.bn1 = nn.BatchNorm2d(48)
self.conv2 = nn.Conv2d(48, 64, 2, padding=1)
self.bn2 = nn.BatchNorm2d(64)
self.conv3 = nn.Conv2d(64, 128, 3, padding=2)
self.bn3 = nn.BatchNorm2d(128)
self.conv4 = nn.Conv2d(128, 256, 2, stride=2, padding=0)
self.bn4 = nn.BatchNorm2d(256)
self.pool = nn.MaxPool2d(2, 2)
self.flatten = nn.Flatten()
self.dropout = nn.Dropout(0.2)
self.fc1 = nn.Linear(256 * 2 * 2, 64)
self.bn_fc1 = nn.BatchNorm1d(64)
self.fc2 = nn.Linear(64, num_classes, bias=False)
```

3.2 Trenowanie modelu CNN

Hiperparametry trenowania:

- Optimizer: AdamW z weight decay=1e-4
- Learning rate: 0.001 z ReduceLROnPlateau scheduler
- Loss function: CrossEntropyLoss
- Batch size: 64
- Early stopping: patience=5 epok

Techniki regularyzacji:

- Weight decay dla redukcji overfittingu
- Batch Normalization dla stabilizacji trenowania
- Dropout w warstwach w pełni połączonych
- Early stopping na podstawie validation loss

3.3 Klasyfikator Random Forest

Proces dwuetapowy:

1. Ekstrakcja cech: Wykorzystanie wytrenowanego CNN do ekstrakcji 64 wymiarowych wektorów cech z warstwy FC1
2. Klasyfikacja: Trenowanie Random Forest na wyekstrahowanych cechach

Parametry Random Forest:

- n_estimators: 100 drzew
- random_state: 42 (dla reprodukowalności)
- n_jobs: -1 (wykorzystanie wszystkich rdzeni CPU)

4 Implementacja

4.1 Klasa SimpleCNN

```python
class SimpleCNN(nn.Module):
    def __init__(self, num_classes):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 48, 4, stride=2, padding=1)
        self.bn1 = nn.BatchNorm2d(48)
        self.conv2 = nn.Conv2d(48, 64, 2, padding=1)
        self.bn2 = nn.BatchNorm2d(64)
        self.conv3 = nn.Conv2d(64, 128, 3, padding=2)
        self.bn3 = nn.BatchNorm2d(128)
        self.conv4 = nn.Conv2d(128, 256, 2, stride=2, padding=0)
        self.bn4 = nn.BatchNorm2d(256)
        self.pool = nn.MaxPool2d(2, 2)
        self.flatten = nn.Flatten()
        self.dropout = nn.Dropout(0.2)
        self.fc1 = nn.Linear(256 * 2 * 2, 64)
        self.bn_fc1 = nn.BatchNorm1d(64)
        self.fc2 = nn.Linear(64, num_classes, bias=False)

    def get_features(self, x):
        x = F.relu(self.bn1(self.conv1(x)))
        x = self.pool(x)
        x = F.relu(self.bn2(self.conv2(x)))
        x = self.pool(x)
        x = F.relu(self.bn3(self.conv3(x)))
        x = self.pool(x)
        x = F.relu(self.bn4(self.conv4(x)))
        x = self.flatten(x)
        features = F.relu(self.fc1(x))
        return features

    def forward(self, x):
        features = self.get_features(x)
        x = self.fc2(features)
        return x
```

4.2 Pipeline trenowania

1. **Inicjalizacja modelu** na urządzeniu CUDA/CPU
2. **Pętla trenowania** z monitorowaniem loss i accuracy
3. **Walidacja** po każdej epoce
4. **Zapis najlepszego modelu** na podstawie validation loss
5. **Early stopping** w przypadku braku poprawy

4.3 Ekstrakcja cech i klasyfikacja RF

```
print("\nEkstraktowanie cech...")
X_train, y_train, _ = extract_features(model, train_loader)
X_test, y_test, p_test_cnn = extract_features(model, test_loader)
print(X_train.shape)
print("Trenowanie Random Forest...")
clf = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
clf.fit(X_train, y_train)
y_pred_rf = clf.predict(X_test)
y_pred_rf_proba = clf.predict_proba(X_test)
```

5 Generowanie wyników

```
CNN: ACC=0.9887, AUC=0.9999
RandomForest: ACC=0.9877, AUC=0.9999
```

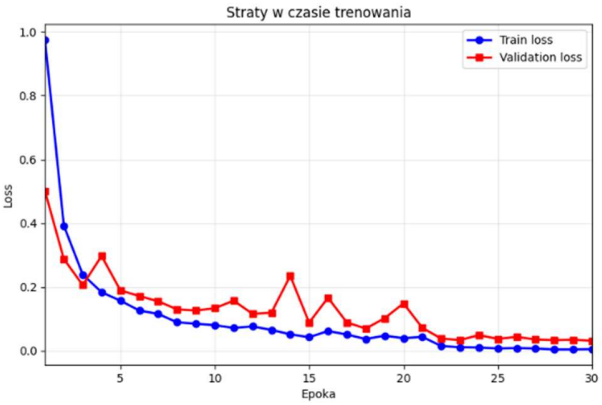
Raport klasyfikacji CNN:

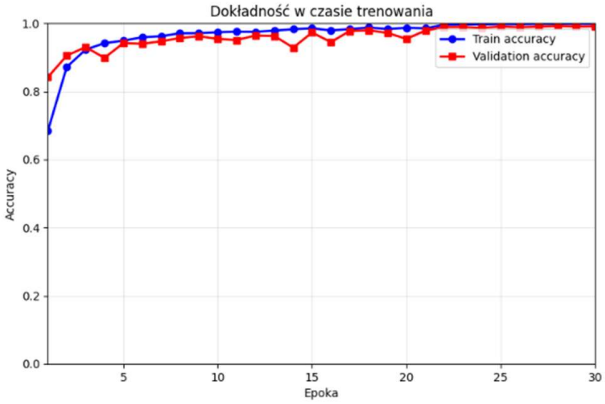
	precision	recall	f1-score	support
Bean	0.99	0.98	0.98	200
Bitter_Gourd	0.98	0.98	0.98	200
Bottle_Gourd	1.00	1.00	1.00	200
Brinjal	0.99	0.99	0.99	200
Broccoli	0.99	0.99	0.99	200
Cabbage	0.98	0.99	0.98	200
Capsicum	1.00	0.99	0.99	200
Carrot	1.00	1.00	1.00	200
Cauliflower	0.99	0.97	0.98	200
Cucumber	0.99	1.00	1.00	200
Papaya	0.98	1.00	0.99	200
Potato	1.00	1.00	1.00	200
Pumpkin	0.97	0.97	0.97	200
Radish	0.99	0.99	0.99	200
Tomato	0.99	0.95	0.97	200
accuracy			0.99	3000
macro avg	0.99	0.99	0.99	3000
weighted avg	0.99	0.99	0.99	3000

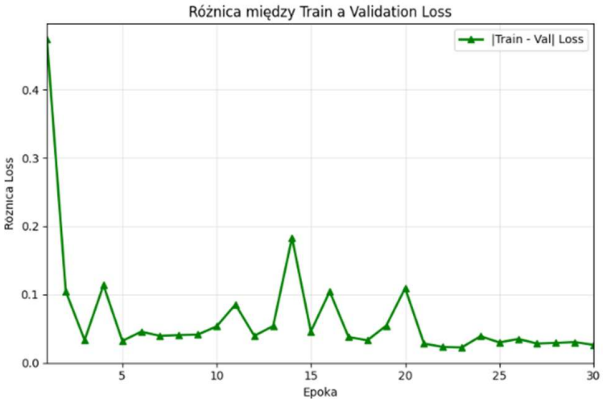
Raport klasyfikacji Random Forest:

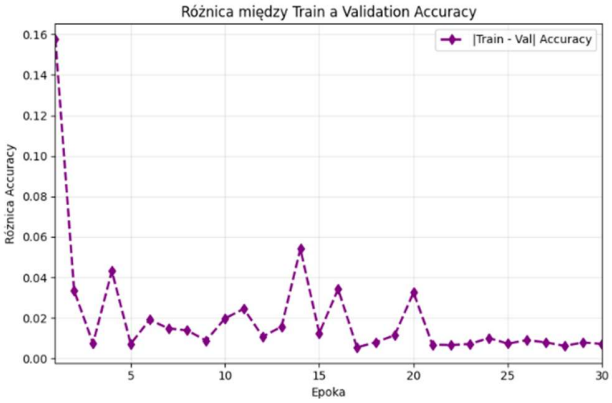
	precision	recall	f1-score	support
Bean	0.98	0.98	0.98	200
Bitter_Gourd	0.98	0.98	0.98	200
Bottle_Gourd	0.99	0.99	0.99	200
Brinjal	0.98	0.98	0.98	200
Broccoli	0.99	0.98	0.99	200
Cabbage	0.98	0.99	0.98	200
Capsicum	1.00	0.99	1.00	200
Carrot	1.00	1.00	1.00	200
Cauliflower	0.98	0.97	0.98	200
Cucumber	0.98	0.99	0.99	200
Papaya	0.99	1.00	1.00	200
Potato	1.00	1.00	1.00	200
Pumpkin	0.98	0.97	0.97	200
Radish	0.99	0.99	0.99	200
Tomato	0.99	0.97	0.98	200
accuracy			0.99	3000
macro avg	0.99	0.99	0.99	3000
weighted avg	0.99	0.99	0.99	3000

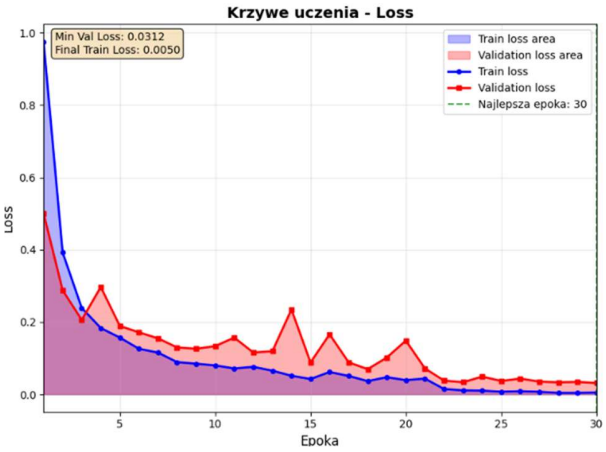
Wykresy

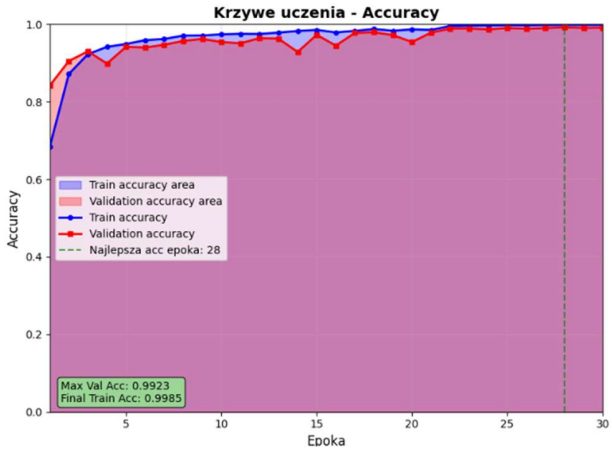


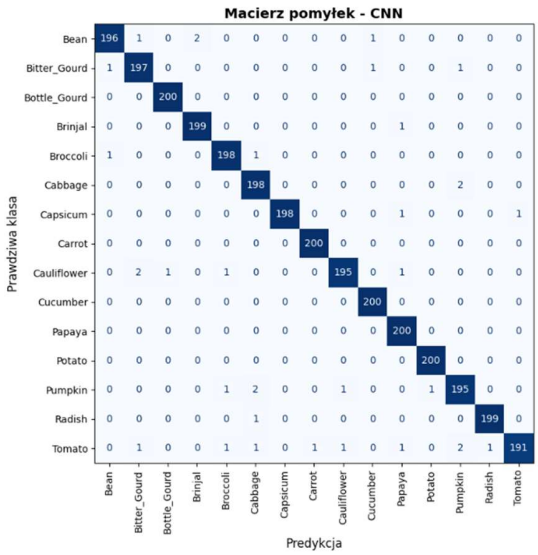


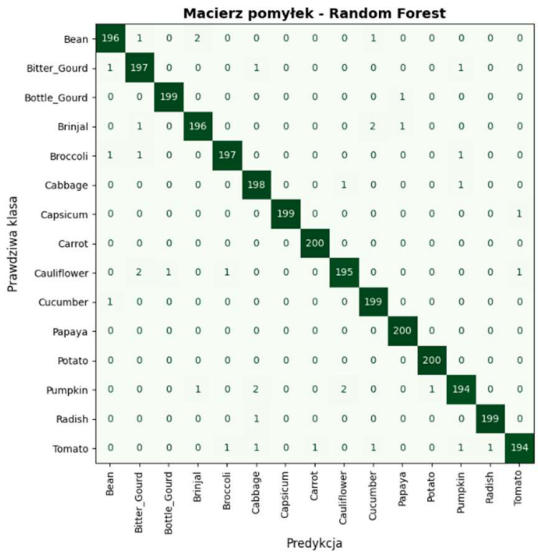

Różnica między Train a Validation Loss
|Train - Val| Loss
Roznica Loss
Epoka

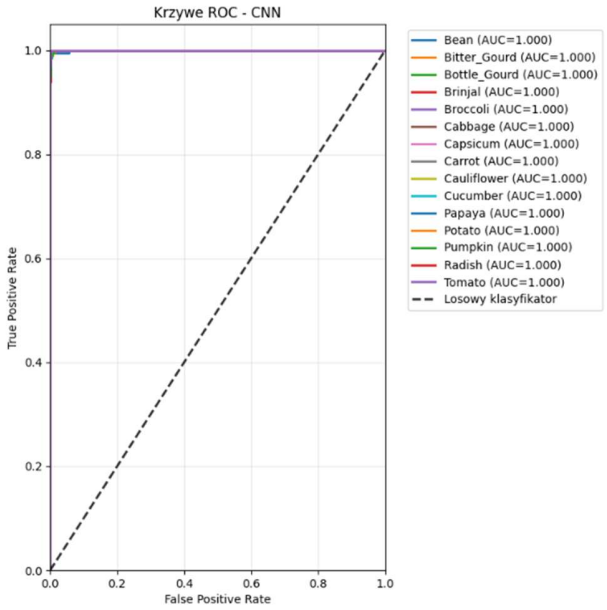

Różnica między Train a Validation Accuracy
|Train - Val| Accuracy
Różnica Accuracy
Epoka

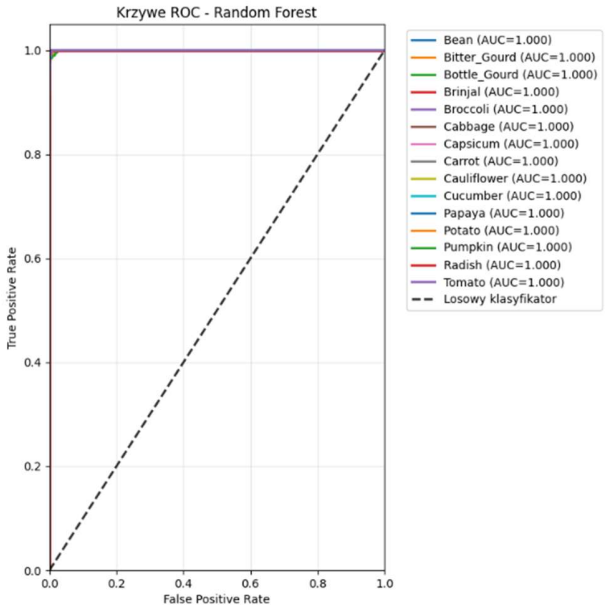

Krzywe uczenia - Loss
Min Val Loss: 0.0312
Final Train Loss: 0.0050
Train loss area
Validation loss area
Train loss
Validation loss
Najlepsza epoka: 30
Loss
Epoka

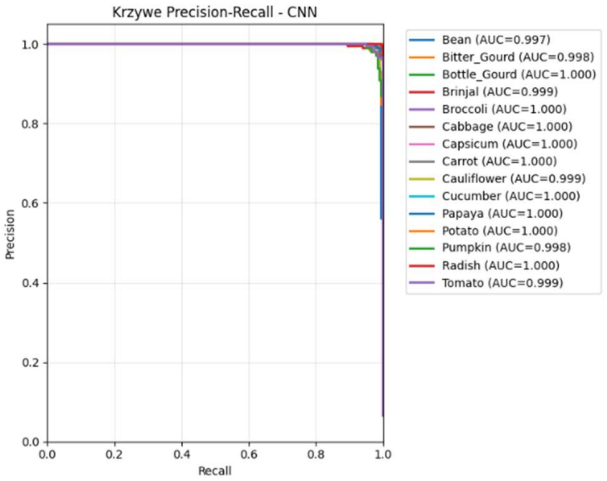

Krzywe uczenia - Accuracy
Train accuracy area
Validation accuracy area
Train accuracy
Validation accuracy
Najlepsza acc epoka: 28
Max Val Acc: 0.9923
Final Train Acc: 0.9985
Accuracy
Epoka

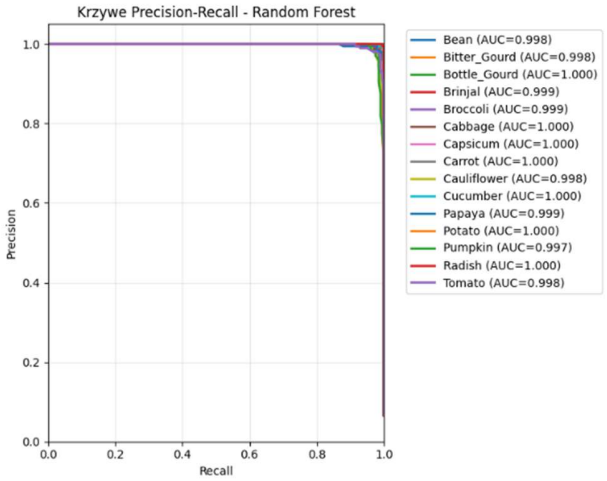

Macierz pomyłek - CNN
Prawdziwa klasa
Predykcja
Bean
Bitter_Gourd
Bottle_Gourd
Brinjal
Broccoli
Cabbage
Capsicum
Carrot
Cauliflower
Cucumber
Papaya
Potato
Pumpkin
Radish
Tomato


Macierz pomyłek - Random Forest
Prawdziwa klasa
Predykcja
Bean
Bitter_Gourd
Bottle_Gourd
Brinjal
Broccoli
Cabbage
Capsicum
Carrot
Cauliflower
Cucumber
Papaya
Potato
Pumpkin
Radish
Tomato


Krzywe ROC - CNN
True Positive Rate
False Positive Rate
Bean (AUC=1.000)
Bitter_Gourd (AUC=1.000)
Bottle_Gourd (AUC=1.000)
Brinjal (AUC=1.000)
Broccoli (AUC=1.000)
Cabbage (AUC=1.000)
Capsicum (AUC=1.000)
Carrot (AUC=1.000)
Cauliflower (AUC=1.000)
Cucumber (AUC=1.000)
Papaya (AUC=1.000)
Potato (AUC=1.000)
Pumpkin (AUC=1.000)
Radish (AUC=1.000)
Tomato (AUC=1.000)
Losowy klasyfikator


Krzywe ROC - Random Forest
True Positive Rate
False Positive Rate
Bean (AUC=1.000)
Bitter_Gourd (AUC=1.000)
Bottle_Gourd (AUC=1.000)
Brinjal (AUC=1.000)
Broccoli (AUC=1.000)
Cabbage (AUC=1.000)
Capsicum (AUC=1.000)
Carrot (AUC=1.000)
Cauliflower (AUC=1.000)
Cucumber (AUC=1.000)
Papaya (AUC=1.000)
Potato (AUC=1.000)
Pumpkin (AUC=1.000)
Radish (AUC=1.000)
Tomato (AUC=1.000)
Losowy klasyfikator


Krzywe Precision-Recall - CNN
Precision
Recall
Bean (AUC=0.997)
Bitter_Gourd (AUC=0.998)
Bottle_Gourd (AUC=1.000)
Brinjal (AUC=0.999)
Broccoli (AUC=1.000)
Cabbage (AUC=1.000)
Capsicum (AUC=1.000)
Carrot (AUC=1.000)
Cauliflower (AUC=0.999)
Cucumber (AUC=1.000)
Papaya (AUC=1.000)
Potato (AUC=1.000)
Pumpkin (AUC=0.998)
Radish (AUC=1.000)
Tomato (AUC=0.999)


Krzywe Precision-Recall - Random Forest
Precision
Recall
Bean (AUC=0.998)
Bitter_Gourd (AUC=0.998)
Bottle_Gourd (AUC=1.000)
Brinjal (AUC=0.999)
Broccoli (AUC=0.999)
Cabbage (AUC=1.000)
Capsicum (AUC=1.000)
Carrot (AUC=1.000)
Cauliflower (AUC=0.998)
Cucumber (AUC=1.000)
Papaya (AUC=0.999)
Potato (AUC=1.000)
Pumpkin (AUC=0.997)
Radish (AUC=1.000)
Tomato (AUC=0.998)

6 Podsumowanie

6.1 Porównanie metod

Zalety CNN:

- Bezpośrednie uczenie: Automatyczna ekstrakcja cech istotnych dla zadania
- End-to-end optymalizacja: Wszystkie parametry optymalizowane jednocześnie
- Wyższa accuracy: Lepsze wyniki na zbiorze testowym
- Reprezentacje hierarchiczne: Uczenie cech na różnych poziomach abstrakcji

Zalety Random Forest:

- Szybkość: Znacznie szybsze trenowanie niż CNN
- Interpretowalność: Możliwość analizy ważności cech
- Mniejsze wymagania obliczeniowe: Brak potrzeby GPU
- Odporność na overfitting: Naturalna regularyzacja poprzez ensemble

6.2 Kluczowe obserwacje

Skuteczność CNN: Model CNN osiągnął lepsze wyniki, co potwierdza skuteczność deep learning wzadaniach klasyfikacji obrazów

Jakość cech: Cechy wyekstrahowane przez CNN zawierały wystarczającą informację dla RandomForest do uzyskania konkurencyjnych wyników

Transfer learning: Możliwość wykorzystania wytrenowanego CNN jako ekstraktora cech dla innychklasyfikatorów

```
======================================================================
PODSUMOWANIE WYNIKÓW EKSPERYMENTU
======================================================================

 INFORMACJE O DANYCH:
   • Liczba klas: 15
   • Klasy: Bean, Bitter_Gourd, Bottle_Gourd, Brinjal, Broccoli, Cabbage, Capsicum, Carrot, Cauliflower, Cucumber, Papaya, Potato, Pumpkin, Radish, Tomato
   • Rozmiar obrazów: 224x224 ==resize==>> 64x64 pikseli
   • Augmentacja danych: Tak (ColorJitter, RandomHorizontalFlip)

 ARCHITEKTURA MODELU CNN:
   • Liczba parametrów: 287,568
   • Warstwy konwolucyjne: 4 (48→64→128→256 kanałów)
   • Normalizacja: BatchNorm2d + BatchNorm1d
   • Regularyzacja: Dropout (0.2), Weight Decay (1e-4)
   • Optimizer: AdamW (lr=0.001)

  TRENOWANIE:
   • Czas trenowania: 28.94 minut
   • Liczba epok: 30
   • Early stopping: Tak (patience=3)
   • Najlepsza val_loss: 0.0312

 WYNIKI KLASYFIKACJI:
   ┌─────────────────┬──────────┬──────────┐
   │     Metryka     │   CNN    │    RF    │
   ├─────────────────┼──────────┼──────────┤
   │ Accuracy        │ 0.9887   │ 0.9877   │
   │ AUC (macro)     │ 0.9999   │ 0.9999   │
   │ PR-AUC (śred.)  │ 0.9993   │ 0.9990   │
   └─────────────────┴──────────┴──────────┘

 NAJLEPSZE KLASY (CNN):
   1. Bottle_Gourd: 1.0000
   2. Carrot: 1.0000
   3. Cucumber: 1.0000

 NAJGORSZE KLASY (CNN):
   13. Cauliflower: 0.9750
   14. Pumpkin: 0.9750
   15. Tomato: 0.9550

 PORÓWNANIE MODELI:
   • CNN przewyższa Random Forest o 0.0010 punktów accuracy
```

Projekt wykazał skuteczność obu badanych podejść do klasyfikacji obrazów warzyw. Convolutional NeuralNetwork osiągnął lepsze wyniki bezpośredniej klasyfikacji, podczas gdy Random Forest z cechami wyekstrahowanymi przez CNN stanowił efektywną alternatywę o mniejszych wymaganiach obliczeniowych.

Projekt potwierdził przewagę metod deep learning w zadaniach klasyfikacji obrazów, jednocześniedemonstrując wartość klasycznych metod ML w połączeniu z nowoczesnymi technikami ekstrakcji cech.

7. Bibliografia

1 Vegetable Image Dataset. Kaggle. https://www.kaggle.com/datasets/misrakahmed/vegetable-image-dataset

Kod źródłowy projektu: `eml_lukasz_kielczyk_projekt.py`

Środowisko: Google Colab, Python 3.x, PyTorch, scikit-learn