AKADEMIA ŁOMŻYŃSKA

WYDZIAŁ NAUK INFORMATYCZNO-TECHNOLOGICZNYCH



# KLASYFIKACJA KWIATÓW

Dokumentacja

## Wydziałowy projekt zespołowy

**Zespół autorski**

Dariusz Nowacki Damian Leszczyński

Michał Szuba

## Prowadzący

dr inż. Janusz Rafałko

Informatyka Studia stacjonarne I stopnia, rok IV, semestr VII

Rok akademicki: 2024/2025

# Opis

Naszym zadaniem będzie stworzenie 2 modeli uczenia maszynowego, zdolnych do klasyfikacji gatunku kwiatu na podstawie zdjęcia. *Analysis of Convolutional Neural Network based Image Classification Techniques* [*1*](#_heading=h.gjdgxs)w tej pracy, opisany jest sposób implementacji CNN oraz wyjaśnienie jego działania. *Automatic Flower Detection and Classification System Using a Light-Weight Convolutional Neural Network*[*2*](#_heading=h.30j0zll) porównuje zastosowanie różnych modeli sieci neuronowych, w tym ResNet50, w celu znajdowania na ekranie i klasyfikacji kwiatów.

# Podział obowiązków:

* + Przygotowanie i obróbka danych - Dariusz Nowacki
  + Klasyfikator przy użyciu ResNet50 – Damian Leszczyński
  + Klasyfikator przy użyciu prostego CNN – Michał Szuba

# Technologie

Większość projektu zostanie wykonany przy użyciu języka Python. W celu stworzenia sieci neuronowych, zostanie wykorzystana biblioteka PyTorch. Interfejs graficzny użytkownika (GUI) tego projektu zostanie zaimplementowany w języku C#. Będzie on służyć do prezentacji wyników działania modeli oraz umożliwi użytkownikom łatwe testowanie modeli na nowych zdjęciach kwiatów. Rozwiązanie to pozwoli na dynamiczne wybieranie zdjęć za pomocą przycisków i wyświetlanie przewidywanych kategorii kwiatów.

1 [https://www.researchgate.net/profile/Milan-](https://www.researchgate.net/profile/Milan-Tripathi/publication/352726526_Analysis_of_Convolutional_Neural_Network_based_Image_Classification_Techniques/links/613864f0637a811d6d587719/Analysis-of-Convolutional-Neural-Network-based-Image-Classification-Techniques.pdf?origin=journalDetail&_tp=eyJwYWdlIjoiam91cm5hbERldGFpbCJ9)

[Tripathi/publication/352726526\_Analysis\_of\_Convolutional\_Neural\_Network\_based\_Image\_Classification\_Tec](https://www.researchgate.net/profile/Milan-Tripathi/publication/352726526_Analysis_of_Convolutional_Neural_Network_based_Image_Classification_Techniques/links/613864f0637a811d6d587719/Analysis-of-Convolutional-Neural-Network-based-Image-Classification-Techniques.pdf?origin=journalDetail&_tp=eyJwYWdlIjoiam91cm5hbERldGFpbCJ9) [hniques/links/613864f0637a811d6d587719/Analysis-of-Convolutional-Neural-Network-based-Image-](https://www.researchgate.net/profile/Milan-Tripathi/publication/352726526_Analysis_of_Convolutional_Neural_Network_based_Image_Classification_Techniques/links/613864f0637a811d6d587719/Analysis-of-Convolutional-Neural-Network-based-Image-Classification-Techniques.pdf?origin=journalDetail&_tp=eyJwYWdlIjoiam91cm5hbERldGFpbCJ9) [Classification-Techniques.pdf?origin=journalDetail&\_tp=eyJwYWdlIjoiam91cm5hbERldGFpbCJ9](https://www.researchgate.net/profile/Milan-Tripathi/publication/352726526_Analysis_of_Convolutional_Neural_Network_based_Image_Classification_Techniques/links/613864f0637a811d6d587719/Analysis-of-Convolutional-Neural-Network-based-Image-Classification-Techniques.pdf?origin=journalDetail&_tp=eyJwYWdlIjoiam91cm5hbERldGFpbCJ9)

2

[https://new.eurasip.org/Proceedings/Eusipco/eusipco2019/wpcontent/uploads/2019/08/Automatic\_Flower\_D](https://new.eurasip.org/Proceedings/Eusipco/eusipco2019/wp-content/uploads/2019/08/Automatic_Flower_Detection_and_Classification_System_Using_a_Light-Weight_Convolutional_Neural_Network.pdf) [etection\_and\_Classification\_System\_Using\_a\_LightWeight\_Convolutional\_Neural\_Network.pdf](https://new.eurasip.org/Proceedings/Eusipco/eusipco2019/wp-content/uploads/2019/08/Automatic_Flower_Detection_and_Classification_System_Using_a_Light-Weight_Convolutional_Neural_Network.pdf)

1. Wprowadzenie
   1. **Cel projektu**

Celem projektu jest stworzenie dwóch modeli uczenia maszynowego zdolnych do klasyfikacji gatunku kwiatu na podstawie zdjęcia. Modele mają za zadanie osiągnąć wysoką skuteczność klasyfikacji, korzystając z datasetu zawierającego pięć różnych gatunków kwiatów. Jednym z kluczowych celów projektu jest porównanie wydajności prostego CNN oraz sieci ResNet50.

* 1. **Zakres projektu**

Zakres projektu obejmuje wszystkie etapy niezbędne do stworzenia, przetestowania i oceny dwóch modeli uczenia maszynowego do klasyfikacji kwiatów. Projekt zostanie zrealizowany w następujących krokach:

* **Przygotowanie danych:** Analiza i wstępne przetworzenie datasetu zawierającego zdjęcia pięciu gatunków kwiatów. Obejmuje to normalizację obrazów, ich skalowanie oraz ewentualną augmentację w celu zwiększenia różnorodności danych treningowych.
* **Budowa modeli:** Implementacja dwóch różnych architektur sieci neuronowych:
  + Modelu opartego na lekkiej sieci konwolucyjnej.
  + Modelu wykorzystującego sieć ResNet50.
* **Trenowanie modeli:** Użycie przygotowanego datasetu do nauki obu modeli.
* **Ewaluacja:** Porównanie skuteczności obu modeli za pomocą miar takich jak dokładność (accuracy) oraz F1-score. Uwzględniona zostanie także analiza efektywności czasowej i obliczeniowej.
* **Wnioski:** Na podstawie wyników ewaluacji zostaną sformułowane wnioski dotyczące wydajności analizowanych sieci neuronowych.

Stworzony także został prosty program do testowania obu modeli sieci neuronowych.

1. Zbiór danych
   1. **Opis danych**

Wykorzystany dataset zawiera 3670 zdjęć kwiatów z czego 924 zdjęcia przeznaczone do testowania oraz 2746 do trenowania. Zdjęcia dzielą się na 5 gatunków w następujący sposób:

* 501 zdjęć stokrotek
* 646 zdjęć mniszków lekarskich
* 497 zdjęć róż
* 495 zdjęć słoneczników
* 607 zdjęć tulipanów

Wszystkie zdjęcia są w kolorze oraz różnych wymiarach i kompozycjach. Dataset wzięty był ze strony Kaggle (<https://www.kaggle.com/datasets/imsparsh/flowers-dataset/data>).

* 1. **Przygotowanie danych**

Przygotowanie danych jest kluczowym etapem w procesie uczenia maszynowego. Celem jest przetworzenie surowych danych do formatu odpowiedniego dla modelu oraz zwiększenie jego efektywności uczenia się. W tym projekcie przygotowanie danych obejmuje:

* Uniformizacja danych: Transformacje, takie jak skalowanie i przycinanie, zapewniają, że wszystkie obrazy mają jednakowy rozmiar, co jest wymagane przez model.
* Augmentacja danych (tylko zbiór treningowy): Losowe transformacje (przycinanie, odbijanie, obracanie) zwiększają różnorodność danych treningowych, co zapobiega przeuczeniu modelu i poprawia jego zdolność do generalizacji na nowe, niewidziane wcześniej dane.
* Normalizacja: Przeskalowanie wartości pikseli do określonego zakresu (np. -1 do 1 lub 0 do 1) ułatwia i przyspiesza proces uczenia.
* Konwersja do tensorów: Dane muszą być w formacie tensorów PyTorch, aby mogły być przetwarzane przez model.
* Uproszczenie danych: Konwersja do skali szarości zmniejsza ilość danych wejściowych, co może przyspieszyć uczenie, szczególnie w prostszych modelach, bez znaczącej utraty informacji istotnych dla klasyfikacji.

Zbiór treningowy (data/train): Obrazy są losowo przycinane, odbijane, obracane, konwertowane do skali szarości, transformowane do tensorów PyTorch i normalizowane (średnia 0.5, odchylenie standardowe 0.5).

Zbiór testowy (data/test): Obrazy są skalowane, przycinane centralnie, konwertowane do tensorów i normalizowane z użyciem statystyk ImageNet. Etykiety testowe są ustawione na -1, ponieważ nie są dostępne w oryginalnym zbiorze.

Przetworzone dane (zbiór treningowy, obrazy i pseudo-etykiety testowe) są zapisywane do plików .pt w celu późniejszego wykorzystania.

1. Modele uczenia maszynowego
   1. **Prosty CNN**
      1. **Opis sieci neuronowej**

W projekcie wykorzystano prostą konwolucyjną sieć neuronową (CNN) zaprojektowaną w celu klasyfikacji obrazów przedstawiających kwiaty. Sieć ta została skonstruowana w sposób umożliwiający efektywne przetwarzanie obrazów wejściowych, transformację ich cech oraz klasyfikację na jeden z pięciu gatunków kwiatów. Poniżej przedstawiono strukturę i działanie zastosowanego modelu:

* **Warstwy konwolucyjne i poolingowe**:  
  Sieć rozpoczyna się od dwóch warstw konwolucyjnych (nn.Conv2d), które odpowiadają za wyodrębnianie cech z obrazów wejściowych.
  + Pierwsza warstwa (c1) przyjmuje obrazy w skali szarości (jeden kanał, 1x224x224) i generuje 6 map cech za pomocą filtrów o wymiarach 5x5.
  + Druga warstwa (c2) przetwarza wyniki poprzedniej, tworząc 16 bardziej złożonych map cech.
  + Każda warstwa konwolucyjna jest połączona z warstwą poolingową (nn.MaxPool2d), która zmniejsza rozmiar map cech o połowę, co redukuje wymiar danych i poprawia efektywność.
* **Warstwy w pełni połączone (fully connected)**:  
  Wydobyte cechy są następnie spłaszczane i przetwarzane przez trzy warstwy w pełni połączone:
  + Pierwsza warstwa (f1) redukuje wymiar do 256 jednostek.
  + Druga warstwa (f2) zmniejsza liczbę neuronów do 128.
  + Ostatnia warstwa (f3) mapuje wyniki na pięć klas odpowiadających gatunkom kwiatów.
* **Funkcje aktywacji**:  
  Każda warstwa konwolucyjna i w pełni połączona (oprócz ostatniej) wykorzystuje funkcję aktywacji ReLU, która nadaje nieliniowość i umożliwia efektywne trenowanie. Ostateczne wyniki są przetwarzane przez funkcję softmax, co pozwala na interpretację wyjść modelu jako prawdopodobieństwa przynależności do każdej klasy.
* **Normalizacja wag**:  
  Wagi modelu są inicjalizowane z wykorzystaniem zaawansowanych technik, takich jak kaiming\_normal\_ dla warstw konwolucyjnych i xavier\_normal\_ dla warstw w pełni połączonych. Dzięki temu model zaczyna trenowanie z dobrze dobranymi wartościami początkowymi, co zwiększa jego stabilność i przyspiesza proces uczenia.
* **Trenowanie i optymalizacja**:  
  Model wykorzystuje algorytm optymalizacji SGD (Stochastic Gradient Descent) z momentum, co pomaga uniknąć lokalnych minimów i stabilizuje proces optymalizacji. Funkcją straty jest CrossEntropyLoss, która mierzy różnicę między przewidywaniami modelu a rzeczywistymi etykietami.

Wykorzystany model CNN jest prosty, ale wystarczająco efektywny, aby przetwarzać obrazy i dokonywać klasyfikacji w zadaniu z ograniczoną liczbą klas. Dzięki odpowiedniej normalizacji, augmentacji danych oraz użyciu zaawansowanych funkcji inicjalizacji wag, sieć jest dobrze przygotowana do skutecznego rozpoznawania gatunków kwiatów.

* + 1. **Pseudokod CNN**

|  |
| --- |
| definicja funkcji main(num\_epochs):  inicjalizuj logger  ustaw urządzenie (cuda:0 lub cpu)  załaduj model i ustaw tryb trenowania  zdefiniuj funkcję straty (CrossEntropyLoss)  zdefiniuj optymalizator (SGD z lr=0.001, momentum=0.9)  załaduj dane treningowe i wymieszaj losowo  dla każdej epoki w num\_epochs:  zainicjalizuj stratę epoki i licznik poprawnych przewidywań  dla każdego obrazu i etykiety w zestawie danych:  przenieś obraz na urządzenie  wyzeruj gradienty optymalizatora  uzyskaj wyjście modelu  oblicz stratę jeśli przewidywanie  jest poprawne:  zwiększ licznik poprawnych przewidywań  wykonaj propagację wsteczną  zaktualizuj wagi modelu  dodaj stratę do całkowitej straty epoki    oblicz i wypisz średnią stratę i dokładność  zapisz wyniki do loggera  zapisz wytrenowany model do pliku  jeśli \_\_name\_\_ == '\_\_main\_\_':  uruchom funkcję main |

* 1. **ResNet 50**
     1. **Opis sieci neuronowej**

Model ResNet50 jest bardziej zaawansowaną konwolucyjną siecią neuronową, zbudowaną z myślą o lepszej zdolności do rozpoznawania złożonych obrazów i funkcjonowania na większych zbiorach danych. ResNet50 to wariant architektury ResNet (“Residual Network”), która jest znana ze swojego wyrafinowanego mechanizmu warstw resztkowych, pozwalających na efektywne uczenie się bardzo głębokich sieci neuronowych bez problemu zanikającego gradientu. Poniżej przedstawiono strukturę i działanie zastosowanego modelu:

* **Warstwy początkowe**: ResNet50 zaczyna się od standardowej warstwy konwolucyjnej, która pobiera obrazy RGB (trzy kanały) i generuje wiele map cech za pomocą filtrów o rozmiarze 7x7, a następnie stosuje warstwę poolingową w celu redukcji wymiarów map cech.
* **Bloki resztkowe**: Główna część modelu składa się z kilku zestawów bloków resztkowych. Bloki te obejmują po kilka warstw konwolucyjnych, połączonych ze sobą za pomocą połączeń przeskakujących, co poprawia stabilność i efektywność procesu trenowania.
* **Warstwy w pełni połączone**: Na końcu sieci znajduje się warstwa globalnego average poolingu, która spłaszcza dane z bloków resztkowych do jednowymiarowego wektora cech. Następnie warstwa w pełni połączona mapuje te cechy na klasy, wykorzystując funkcję softmax do wyznaczania prawdopodobieństw przynależności do poszczególnych kategorii.
* **Funkcje aktywacji**:Każdy blok resztkowy zawiera funkcje aktywacji ReLU (Rectified Linear Unit), które wprowadzają nieliniowość i pomagają modelowi lepiej aproksymować złożone funkcje. Po warstwach konwolucyjnych stosowana jest także normalizacja wsadowa (Batch Normalization), która stabilizuje proces uczenia.
* **Inicjalizacja wag i trenowanie**: W modelu ResNet50 stosuje się zaawansowane metody inicjalizacji wag, takie jak kaiming initialization, aby zapewnić skuteczny początkowy stan parametrów i przyspieszyć proces uczenia. Model jest trenowany z wykorzystaniem optymalizatora SGD (Stochastic Gradient Descent) z momentum, co stabilizuje optymalizację i pomaga uniknąć utknięcia w lokalnych minimach funkcji straty.

Dzięki zastosowaniu bloków resztkowych, ResNet50 jest w stanie efektywnie uczyć się bardzo głębokich struktur, co czyni go znacznie bardziej precyzyjnym niż proste sieci konwolucyjne. Model ten jest stosowany w zadaniach klasyfikacji, detekcji obiektów i innych problemach analizy obrazów, szczególnie tam, gdzie wymagane jest rozpoznawanie złożonych wzorców.

1. Implementacja

Implementacja projektu została podzielona na dwie główne części: model uczenia maszynowego w Pythonie oraz interfejs graficzny użytkownika w C#.

* 1. **Model w Pythonie**

Model został zaimplementowany przy użyciu biblioteki PyTorch. Skrypt zawiera funkcje do wczytywania modelu, przetwarzania obrazu wejściowego oraz generowania predykcji. Kluczowe elementy implementacji to:

* Wczytywanie modelu: Zarówno model ResNet50, jak i prosty CNN są wczytywane z zapisanych plików .pt przy użyciu funkcji torch.load().
* Predykcja: Funkcja GetLabels() przyjmuje ścieżkę do obrazu, przetwarza go i zwraca predykcje obu modeli.
* Obsługa argumentów wiersza poleceń: Skrypt akceptuje ścieżkę do obrazu jako argument wiersza poleceń, co umożliwia integrację z interfejsem graficznym.
  1. **Interfejs graficzny w C#**

Interfejs graficzny został stworzony w WPF (Windows Presentation Foundation) z wykorzystaniem biblioteki WPF UI. Aplikacja zapewnia użytkownikowi następujące funkcjonalności:

* Instalacja Pythona i bibliotek: Przycisk "Zainstaluj Python" automatycznie pobiera i instaluje interpreter Pythona wraz z biblioteką pip. Przycisk "Zainstaluj zależności" instaluje wymagane biblioteki PyTorch.
* Wybór obrazu: Przycisk "Wybierz obraz" otwiera okno dialogowe, umożliwiające użytkownikowi wybór pliku obrazu z dysku. Dodatkowo zaimplementowano mechanizm "przeciągnij i upuść" (drag and drop) dla wygodniejszego wyboru pliku.
* Wyświetlanie obrazu: Wybrany obraz jest wyświetlany w podglądzie.
* Uruchomienie modelu i wyświetlanie wyników: Przycisk "Sprawdź obraz" uruchamia skrypt Pythona test\_labeling.py z wybranym obrazem jako argumentem. Wyniki predykcji obu modeli są wyświetlane w polu tekstowym. Dodatkowo, ewentualne błędy wykonywania skryptu są przechwytywane i wyświetlane w osobnym polu tekstowym.
* Wskaźnik postępu: Pasek postępu informują użytkownika o postępie operacji, takich jak pobieranie plików czy uruchamianie skryptu Pythona.
  1. **Integracja Pythona i C#**

Integracja Pythona i C# odbywa się poprzez uruchamianie skryptu Pythona jako osobnego procesu. Interfejs graficzny wykorzystuje klasę Process do uruchomienia skryptu i przechwytywania jego standardowego wyjścia oraz standardowego wyjścia błędów. Argumenty są przekazywane do skryptu poprzez właściwość Arguments klasy ProcessStartInfo.

1. Wyniki

Modele były testowane pod względem ich celności oraz mierzone było f1-score dla każdej kategorii oraz pojedynczy wynik biorący pod uwagę różne ilości obrazków w każdej z kategorii.

* 1. **ResNet 50**

Model ResNet50 okazał się być zbyt złożony dla danego zadania, co prowadzi do nadmiernego dopasowania lub nieefektywnego uczenia się ze względu na jego nadmierną głębokość i dużą liczbę parametrów w stosunku do prostoty i wielkości zbioru danych. Udało się osiągnąć 41,51% celności, ważone f1-score wynosi 0,3411, f1-score dla poszczególnych kategorii wyniosło:

* Stokrotka – 0,2938
* Mniszek lekarski – 0,4981
* Róża – 0,4509
* Słonecznik – 0
* Tulipan – 0,4556
  1. **Prosty CNN**

Po 10 epokach trenowania, prosty CNN osiąga celność w okolicach 89,84%. Ważony f1-score wynosi 0,8983, f1-score dla poszczególnych kategorii to odpowiednio:

* Stokrotka – 0,8781
* Mniszek lekarski – 0,9158
* Róża – 0,8926
* Słonecznik – 0,902
* Tulipan – 0,898

1. Wnioski

W trakcie realizacji projektu zrozumieliśmy, jak kluczowe znaczenie ma właściwy dobór narzędzi i modeli do specyfiki danego zadania. Przeprowadzając eksperymenty, zauważyliśmy, że zastosowanie zaawansowanego modelu ResNet50, choć bardzo skutecznego w bardziej skomplikowanych zadaniach, okazało się nieoptymalne w przypadku prostego zadania klasyfikacji kwiatów. Jego głęboka architektura i liczba parametrów prowadziły do problemów z generalizacją, co skutkowało niższą dokładnością i F1-score w porównaniu z prostym modelem CNN.

Prosty CNN lepiej dopasował się do charakterystyki zadania, które wymagało mniej złożonych obliczeń i mniej zaawansowanego wyodrębniania cech wizualnych. Wyniki projektu podkreślają, że wybór bardziej zaawansowanego modelu nie zawsze gwarantuje lepsze wyniki — czasami prostsze rozwiązania są bardziej efektywne i skuteczne.