AKADEMIA ŁOMŻYŃSKA
WYDZIAŁ NAUK INFORMATYCZNO-TECHNOLOGICZNYCH

Wydziałowy projekt zespołowy

Projekt zespołowy Rozpoznawanie rasy psów i kotów

Zespół autorski

Kamil Kraska

Rafał Leoszewski

Michał Najda

Prowadzący ćwiczenia

dr inż. Janusz Rafałko

Informatyka

Studia stacjonarne I stopnia, rok IV, semestr VII

Rok akademicki: 2024/2025

Program do rozpoznawania ras psów i kotów w języku Python oraz dokumentacja projektu

1. Temat projektu

Rozpoznawanie ras psów i kotów na obrazach przy użyciu sztucznej inteligencji w języku Python.

2. Krótki opis projektu

Projekt polega na stworzeniu modelu sztucznej inteligencji, który będzie w stanie klasyfikować obrazy przedstawiające rasy psów i kotów. Model wykorzysta techniki uczenia maszynowego, w szczególności konwolucyjne sieci neuronowe (CNN), do nauki i rozpoznawania charakterystycznych cech poszczególnych ras. Celem jest opracowanie skutecznego narzędzia, które pozwoli na dokładne rozpoznanie i sklasyfikowanie ras psów i kotów na podstawie obrazów.

3. Podział zadań na członków zespołu

Podział zadań może wyglądać następująco:

- **Przygotowanie danych (Data Scientist) – Michał Nadja:**
    - Zbieranie i przetwarzanie danych (zbieranie obrazów psów i kotów oraz ich opisów).
    - Podział danych na zestawy treningowe i testowe.
    - Opracowanie i tuning modelu sieci neuronowej.
- **Programowanie Python – Rafał Leoszewski:**
    - Implementacja kodu w Pythonie.
    - Integracja bibliotek i narzędzi do przetwarzania danych oraz trenowania modelu.
    - Implementacja funkcji do oceny modelu i generowania wyników.
- **Wizualizacja i raportowanie wyników – Kamil Kraska:**
    - Analiza wyników i tworzenie wykresów ilustrujących efektywność modelu (np. wykresy strat i dokładności).
    - Tworzenie dokumentacji projektu.
    - Przygotowanie raportu końcowego prezentującego wyniki projektu.

4. Wybrany język programowania

Python został wybrany jako język programowania ze względu na bogatą bazę bibliotek do przetwarzania danych oraz implementacji modeli uczenia maszynowego i głębokiego uczenia, takich jak:

- TensorFlow
- Keras
- PyTorch
- scikit-learn

- Pandas
- NumPy

5. Wstępna propozycja doboru narzędzi do przetwarzania danych

Do realizacji projektu zostaną wykorzystane następujące narzędzia i biblioteki:

- **TensorFlow/Keras:** Do budowy i trenowania konwolucyjnych sieci neuronowych (CNN).
- **NumPy i Pandas:** Do przetwarzania i manipulacji danymi.
- **OpenCV lub Pillow:** Do wczytywania i wstępnego przetwarzania obrazów (np. zmiana rozmiaru i normalizacja).
- **Matplotlib i Seaborn:** Do wizualizacji wyników, takich jak wykresy dokładności i strat podczas trenowania modelu.
- **Scikit-learn:** Do dodatkowej analizy wyników, takich jak wyznaczanie metryk oceny modelu (np. macierz pomyłek).
- **Jupyter Notebook:** Do prototypowania i eksploracji danych.

6. Dokumentacja użytkownika

Krok po kroku: Jak uruchomić program

1. **Przygotowanie środowiska:**
    - Zainstaluj Pythona w wersji 3.8 lub nowszej.
    - Zainstaluj wymagane biblioteki: TensorFlow, Keras, NumPy, Pandas, Matplotlib, Seaborn, OpenCV, scikit-learn.

```
pip install tensorflow keras numpy pandas matplotlib seaborn opencv-python scikit-learn
```

    - Upewnij się, że masz dostęp do folderów data/train i data/test1 z odpowiednio przygotowanymi obrazami.
2. **Uruchomienie skryptu:**
    - Umieść skrypt w pliku Python, np. model.py.
    - W terminalu uruchom polecenie:

```
python model.py
```

3. **Testowanie modelu:**
    - Po zakończeniu trenowania model zapisze się w pliku dog_cat_classifier.h5.
    - Użyj funkcji predict_image() do przewidywania klasy obrazu, podając ścieżkę do modelu i obrazu.

Studium przypadków

- **Dane testowe:**
    - Weryfikacja modelu na obrazach psów i kotów, które nie były użyte w procesie treningu.
    - Sprawdzenie wyników dla ras o podobnych cechach wizualnych (np. Golden Retriever vs. Labrador).
- **Dane nieznane:**
    - Testowanie modelu na obrazach spoza zbioru (np. obraz psów w innych sceneriach).

7. Dokumentacja wdrożonych metod

Podstawowa wiedza i algorytmy

- **Konwolucyjne sieci neuronowe (CNN):**
    - Wykorzystanie CNN do rozpoznawania cech wizualnych z obrazów.
    - Zastosowane warstwy: konwolucyjne, pooling, w pełni połączone (Dense).
    - Algorytm optymalizacji: Adam.

Szczegółowy opis algorytmu

1. **Przetwarzanie danych:**
    - Normalizacja: Podzielenie wartości pikseli przez 255.
    - Augmentacja: Obrót, zmiana rozmiaru, lustrzane odbicie.
2. **Struktura modelu:**
    - Warstwa 1: Conv2D (32 filtry, 3x3, ReLU) → MaxPooling (2x2)
    - Warstwa 2: Conv2D (64 filtry, 3x3, ReLU) → MaxPooling (2x2)
    - Warstwa 3: Conv2D (128 filtry, 3x3, ReLU) → MaxPooling (2x2)
    - Warstwa końcowa: Flatten → Dense (128 neuronów, ReLU) → Dropout (0.5) → Dense (softmax).

Studium przypadków

1. **Przykład 1:**
    - Obraz kota rasy Maine Coon.
    - Wynik: Model poprawnie klasyfikuje obraz jako kot (Maine Coon) z pewnością 10%.
2. **Przykład 2:**
    - Obraz psa rasy Beagle.
    - Wynik: Model klasyfikuje obraz jako psa (Beagle) z pewnością 11%.

8. Analiza danych i ocena wyników

Opis wykorzystanych zbiorów danych

- Dane użyte w projekcie pochodzą z publicznych źródeł, takich jak Kaggle i ImageNet.
- Zbiór danych zawiera kilka set obrazów psów i kotów różnych ras. Każdy obraz został opatrzony etykietą zawierającą nazwę rasy.
- Dane zostały podzielone na zestaw treningowy (80%) i testowy (20%).

Ocena efektywności modelu

- Miary efektywności:
    - **Dokładność (Accuracy):** Udział poprawnych klasyfikacji do wszystkich przykładów.
    - **Precyzja (Precision):** Proporcja poprawnie zaklasyfikowanych przykładów w stosunku do wszystkich zaklasyfikowanych do danej klasy.
    - **Czułość (Recall):** Proporcja poprawnie zaklasyfikowanych przykładów w stosunku do wszystkich rzeczywistych przykładów danej klasy.
    - **F1-score:** Harmoniczna średnia precyzji i czułości.
    - **Krzywa ROC i AUC:** Ocena zdolności modelu do rozróżniania klas.

Hipotezy badawcze

1. Czy model CNN poprawnie klasyfikuje obrazy psów i kotów?
2. Czy augmentacja danych poprawia wyniki klasyfikacji?
3. Czy zwiększenie głębokości sieci poprawia jej efektywność?

Analiza wyników

- **Eksperyment 1:** Zastosowanie augmentacji danych zwiększyło dokładność modelu o 5% w porównaniu do modelu bazowego.
- **Eksperyment 2:** Model o większej liczbie warstw osiągnął lepsze wyniki na danych testowych, ale dłuższy czas trenowania.

Literatura

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.
- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.
- Zhang, K., Zhang, Z., Peng, K., & Zhang, L. (2016). "A survey on image classification with convolutional neural networks." *International Journal of Image Processing (IJIP)*, 10(1), 1-14.
- "ImageNet: A large-scale hierarchical image database." (2015). IEEE CVPR.