Dokumentacja

Monika Adamczyk
Dawid Bukowski
Aleksandra Gawrońska
Jarosław Małucha

Github Repo

Prezentacja

Wprowadzenie

Ten dokument przedstawia szczegółowe aspekty techniczne projektu aplikacji do klasyfikacji ras psów. Aplikacja wykorzystuje technologie uczenia maszynowego do analizowania zdjęć psów w celu identyfikacji ich rasy. Projekt ma na celu zademonstrowanie, że można stworzyć funkcjonalną aplikację, która umożliwia szybkie i dokładne rozpoznawanie różnych ras psów poprzez prosty interfejs.

Cel Projektu

Głównym celem projektu jest opracowanie systemu identyfikacji ras psów na podstawie zdjęć. System będzie korzystał z obszernej bazy danych zawierającej obrazy i informacje dotyczące różnych ras psów. W ramach tego celu zamierzamy zmodyfikować algorytm sztucznej inteligencji zdolny do analizowania zdjęć i identyfikacji rasy z wysoką dokładnością.

Wybrana technologia

Do budowy i trenowania sieci neuronowych wybraliśmy bibliotekę TensorFlow, która jest open source i specjalizuje się w obliczeniach numerycznych. Dzięki niej można skutecznie trenować i implementować modele uczenia maszynowego. Aplikacja webowa została zbudowana przy użyciu Streamlit, który umożliwia tworzenie interaktywnych aplikacji webowych w Pythonie.

Metoda

Przygotowanie Danych

Projekt wykorzystuje dwa zbiory danych obrazów psów: zestaw treningowy oraz zestaw testowy. Każdy obraz w tych zestawach posiada unikalny identyfikator pliku, który służy jako identyfikator obrazu. Dane są ładowane i przetwarzane przy użyciu biblioteki *'pandas'* oraz '*tensorflow.keras.preprocessing.image.ImageDataGenerator*', który pomaga w augmentacji danych i przygotowaniu zbiorów treningowych i walidacyjnych.

```
# Wczytywanie danych
labels_df = pd.read_csv('Data\\labels.csv')
labels_df['id'] = labels_df['id'].apply(lambda x: f"{x}.jpg")

# Filtrowanie dostępnych obrazów
train_dir = 'Data\\train'
valid_images = []
for img in labels_df['id']:
    if os.path.isfile(os.path.join(train_dir, img)):
        valid_images.append(img)
labels_df = labels_df[labels_df['id'].isin(valid_images)]
datagen = ImageDataGenerator(validation_split=0.2, rescale=1./255)

# Przygotowanie generatorów danych
train_generator = datagen.flow_from_dataframe(
    labels_df,
    directory=train_dir,
    x_col='id',
    y_col='breed',
    subset='training',
    class_mode='categorical',
    target_size=(224, 224),
    batch_size=32
)

validation_generator = datagen.flow_from_dataframe(
    labels_df,
    directory=train_dir,
    x_col='id',
    y_col='breed',
    subset='validation',
    class_mode='categorical',
    target_size=(224, 224),
    batch_size=32
)
```

Architektura Modelu

Do klasyfikacji ras psów wykorzystaliśmy głęboką sieć neuronową (CNN) opartą na pretrenowanej sieci MobileNetV2. Do modelu dodano dodatkowe warstwy dostosowane do naszego zadania.

```
# Przygotowanie przetrenowanego modelu MobileNetV2
base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(len(train_generator.class_indices),
activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

# Zamrożenie warstw bazowego modelu
for layer in base_model.layers:
    layer.trainable = False

# Kompilacje modelu
model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
```

Ewaluacja Modelu

Model został oceniony na podstawie zbioru walidacyjnego, co pozwoliło na monitorowanie jego wydajności oraz dokładności w klasyfikacji ras psów. Wyniki te wskazują na skuteczność zastosowanej architektury modelu.

Parametry modeli ML

W ramach tego projektu dostarczono dwa zbiory danych obrazów psów: zestaw treningowy oraz zestaw testowy. Każdy obraz w tych zestawach posiada unikalny identyfikator pliku, który służy jako identyfikator obrazu. Model trenowany był na zbiorze danych zawierających zdjęcia 120 różnych gatunków psów, a proces trenowania trwał 10 epok.

```
# Trenowanie modelu
model.fit(train_generator,validation_data=validation_generator,
epochs=10)

# Zapisywanie modelu
model.save('dog_breed_classifier.h5')

# Zapisanie indeksów klas
with open('class_indices.pkl', 'wb') as f:
    pickle.dump(train_generator.class_indices, f)
```

Opis funkcjonalności

Aplikacja umożliwia użytkownikom przesyłanie zdjęć, które są automatycznie klasyfikowane pod kątem rasy psa. Użytkownik może wgrać zdjęcia bezpośrednio ze swojego urządzenia i uzyskać informację o rasie w ciągu kilku sekund.

Aplikacja jest zaprojektowana tak, aby była łatwa w użyciu i zapewniała szybkie oraz dokładne wyniki klasyfikacji.

```
import streamlit as st
import numpy as np
import pandas as pd
import tensorflow as tf
from PIL import Image
import pickle

# Wczytanie wytrenowanego modelu
model = tf.keras.models.load_model('dog_breed_classifier.h5')

# Wczytanie indeksów klas
with open('class_indices.pkl', 'rb') as f:
    class_indices = pickle.load(f)##
index_to_class = {v: k for k, v in class_indices.items()}

# Funkcja do przetwarzania obrazu
def preprocess_image(image):
```

```
    image = image.resize((224, 224))
    image = np.array(image) / 255.0
    image = np.expand_dims(image, axis=0)
    return image

# Interfejs użytkownika w Streamlit
st.title("Dog Breed Classifier")
st.write("Upload an image of a dog, and the model will predict
its breed.")

uploaded_file = st.file_uploader("Choose an image...",
type="jpg")

if uploaded_file is not None:
    image = Image.open(uploaded_file)
    st.image(image, caption='Uploaded Image', width=200)
    st.write("Classifying...")

# Przetwarzanie obrazu i predykcja
    image = preprocess_image(image)
    predictions = model.predict(image)[0]

# Uzyskanie top 5 predykcji
    top_indices = predictions.argsort()[-5:][::-1]
    top_breeds = [(index_to_class[idx], predictions[idx]) for idx
in top_indices]

# Wyświetlenie top 5 predykcji
    st.write("### Top 5 Predicted Breeds:")
    for i, (breed, prob) in enumerate(top_breeds):
        if i == 0:
            st.write(f"{breed}: {prob*100:.3f} %")
        else:
            st.write(f"{breed}: {prob*100:.3f} %")
```