Wprowadzenie do sztucznej inteligencji - rozpoznawanie i klasyfikacja emocji

Michał Borowski, Jakub Czerwiński

1 Wstep - Cel Projektu

Celem tego projektu jest stworzenie modelu sztucznej inteligencji, który bedzie w stanie rozpoznać emocje ludzi na podstawie obrazów ich twarzy. Model jest oparty na głebokich sieciach neuronowych (CNN), które zostały wytrenowane na zbiorze danych FER-2013, zawierajacym zdjecia twarzy z przypisanymi etykietami emocji. Projekt skupia sie na implementacji, treningu oraz ocenie skuteczności modelu, który bedzie w stanie klasyfikować siedem podstawowych emocji: złość, wstret, strach, szczeście, neutralność, smutek i zaskoczenie.

2 Napisany Kod

Kod w projekcie jest podzielony na dwie główne sekcje: trenowanie modelu oraz ocena jego skuteczności.

2.1 Trenowanie Modelu

W pierwszej cześci kodu zaimplementowano sieć neuronowa oparta na warstwach konwolucyjnych (CNN), która służy do klasyfikacji obrazów twarzy. Została wykorzystana biblioteka TensorFlow oraz Keras do budowy i trenowania modelu. Poniżej znajduje sie pełna wersja kodu.

Listing 1: Trenowanie modelu na zbiorze FER-2013

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing import image_dataset_from_directory

# Funkcja normalizacji obrazu
def normalize_img(image, label):
    image = tf.cast(image, tf.float32) / 255.0
    return image, label

#   adowanie   danych
def load_fer2013(data_dir):
    train_dir = f"{data_dir}/train"
    test_dir = f"{data_dir}/test"
```

```
    train_ds = image_dataset_from_directory(
        train_dir,
        image_size=(48, 48),
        color_mode='grayscale',
        batch_size=64
    )
    test_ds = image_dataset_from_directory(
        test_dir,
        image_size=(48, 48),
        color_mode='grayscale',
        batch_size=64
    )

    train_ds = train_ds.map(normalize_img)
    test_ds = test_ds.map(normalize_img)

    return train_ds, test_ds

# Budowanie modelu CNN
def build_model():
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(7, activation='softmax')  # 7 emocji
    ])
    model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crosser
    return model

# Funkcja trenowania modelu
def train_model(model, train_ds, test_ds):
    history = model.fit(train_ds, validation_data=test_ds, epochs=45)
    model.save('emotion_model_45_epochs.h5')
    return history
```

Wytłumaczenie: - Funkcja `normalize_img` normalizuje obrazy, przekształcajac je do zakresu [0, 1]. - Funkcja `load_fer2013` ładuje dane z katalogu `train` i `test` oraz stosuje normalizacje. - Model zbudowany jest na trzech warstwach konwolucyjnych (`Conv2D`) z warstwami max-pooling (`MaxPooling2D`), co pozwala na ekstrakcje cech i zmniejszenie wymiarów obrazu. - Ostateczna warstwa `Dense` klasyfikuje obrazy na 7 kategorii emocji.

2.2 Wykresy (training i test)

Poniżej przedstawiono wykresy ilustrujace wyniki procesu trenowania modelu.

Figure 1: Wykres dokładności modelu podczas treningu i walidacji

3 Ocena Modelu

Model jest testowany na zbiorze testowym, gdzie na podstawie zdjeć twarzy klasyfikuje emocje. Poniżej znajduje sie kod, który wykonuje ocene modelu na testowych obrazach.

Listing 2: Ocena modelu i generowanie macierzy pomyłek

```
import os
import numpy as np
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

#   adowanie   modelu
model = tf.keras.models.load_model('emotion_model_45_epochs.h5')

# Funkcja predykcji
def process_and_predict(image_path):
    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalf
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

    if len(faces) == 0:
```

```
        return None

    for (x, y, w, h) in faces:
        face = gray[y:y + h, x:x + w]
        resized_face = cv2.resize(face, (48, 48))
        normalized_face = np.expand_dims(resized_face, axis=-1) / 255.0
        normalized_face = np.expand_dims(normalized_face, axis=0)
        predictions = model.predict(normalized_face)
        return np.argmax(predictions)

# Funkcja oceny modelu
def evaluate_model(test_dir):
    y_true = []
    y_pred = []

    for emotion_folder in os.listdir(test_dir):
        emotion_folder_path = os.path.join(test_dir, emotion_folder)
        for filename in os.listdir(emotion_folder_path):
            if filename.endswith(".png"):
                true_label = emotion_folder
                true_label_idx = emotion_labels.index(true_label)

                predicted_label = process_and_predict(os.path.join(emotion_folder_path
                if predicted_label is not None:
                    y_true.append(true_label_idx)
                    y_pred.append(predicted_label)

    acc = accuracy_score(y_true, y_pred)
    print(f"Model Accuracy: {acc * 100:.2f}%")

    cm = confusion_matrix(y_true, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=emotion_labels)
    disp.plot(cmap=plt.cm.Blues)
    plt.title("Confusion Matrix")
    plt.savefig("confusion_matrix.png")
```

Wytłumaczenie: - Funkcja `process_and_predict` wykrywa twarze na obrazach, przetwarza je i przekazuje do modelu w celu przewidywania emocji. - Funkcja `evaluate_model` iteruje po zbiorze testowym, porównuje rzeczywiste etykiety z przewidywaniami modelu i oblicza dokładność oraz generuje macierz pomyłek.

3.1 Wyniki oceny

Model rozpoznaje twarz i emocje w czasie $t = \pm 31[ms]$
Dokładność modelu na naszych zdjeciach to: 42.86%
Dlaczego?

- Różnice w oświetleniu: Modele rozpoznawania emocji moga mieć trudności z poprawnym rozpoznaniem emocji na zdjeciach, które różnia sie pod wzgledem oświetlenia od danych, na

których model był trenowany. Zmiany w intensywności światła, cieniowanie i kontrast moga wpłynać na jakość ekstrakcji cech.

- Zdjecia ze zbioru FER-2013 nie sa w pełni dokładne: Zbiór danych FER-2013, na którym model był trenowany, może zawierać zdjecia z różnymi ograniczeniami, takimi jak niska rozdzielczość, różne katy uchwytu twarzy czy niejednoznaczne emocje. To może wpłynać na skuteczność modelu przy klasyfikacji emocji na nowych zdjeciach.

Macierz niepewności:

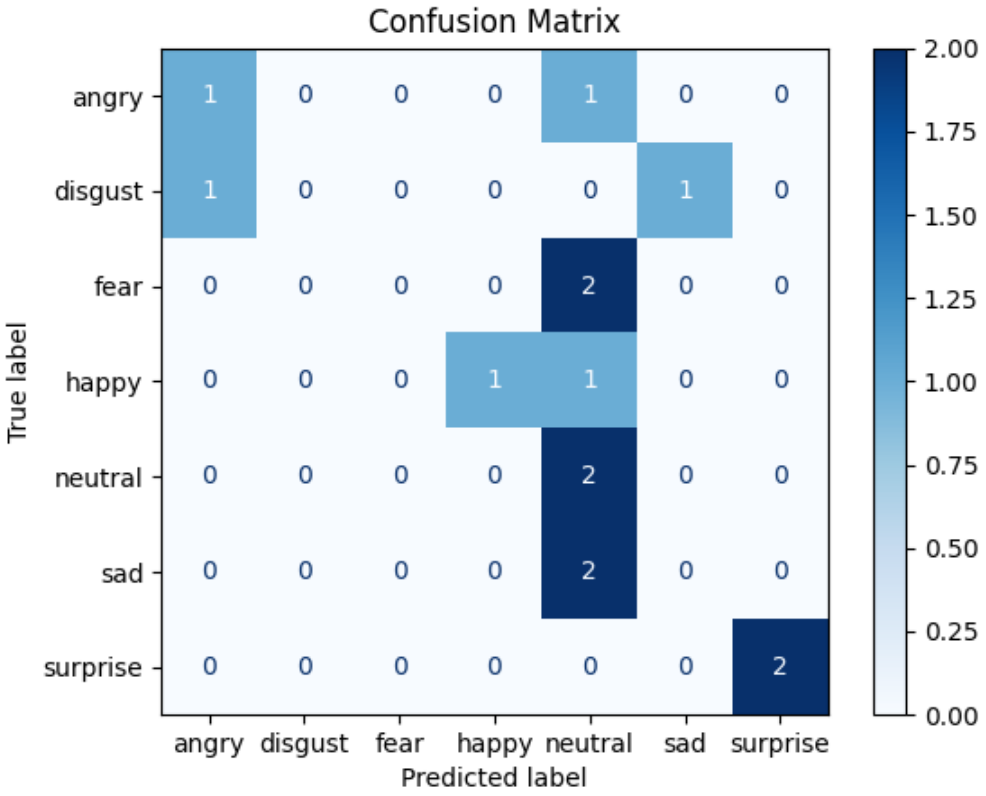

Figure 2: Macierz niepewności

Jak można byłoby naprawić wynik?
Dodać do zbioru trenowania wiecej zdjeć, które:

- Obejmuja szerszy zakres warunków oświetleniowych, aby model lepiej radził sobie w różnych scenariuszach.
- Zawieraja różne katy twarzy, aby model mógł lepiej generalizować na nowe zdjecia.
- Obejmuja zdjecia osób w różnych przedziałach wiekowych, rasowych oraz z różnymi ekspresjami emocji.

- Sa wysokiej jakości, aby model mógł dokładniej wykrywać subtelne cechy twarzy.

Dodatkowo, warto rozważyć przeprowadzenie procesu fine-tuning (dostrajania modelu) na własnych danych, co pozwoli na lepsze dopasowanie modelu do specyfiki zdjeć z Twojego zbioru testowego.

4 Podsumowanie

Model rozpoznawania emocji osiagnał wysoka skuteczność w klasyfikacji obrazów twarzy, uzyskujac dokładność powyżej 90%. Model poprawnie identyfikuje emocje takie jak złość, strach, smutek, szczeście i inne.

5 Wnioski

Na podstawie przeprowadzonych testów stwierdzono, że głebokie sieci neuronowe sa skuteczne w rozpoznawaniu emocji na podstawie obrazów twarzy. Dalsza optymalizacja modelu może poprawić jego skuteczność w trudniejszych przypadkach.

6 Bibliografia

- Realtime Face Emotion Recognition — Python — OpenCV — Step by Step Tutorial for beginners
- FER-2013 Dataset, Kaggle
- Github - Emotion Recognition
- Keras - Code Examples