**Temat projektu**

Rozpoznawanie cyfr z użyciem sieci neuronowej i interfejsu graficznego (GUI)

**Określenie celu projektu i oczekiwanych rezultatów**

Celem projektu jest stworzenie aplikacji, która będzie w stanie **rozpoznawać cyfry zapisane w formie obrazów przy użyciu wytrenowanej sieci neuronowej.** Użytkownicy będą mogli łatwo trenować model, ładować obrazy i uzyskiwać przewidywania poprzez intuicyjny interfejs graficzny. Oczekiwanym rezultatem jest aplikacja, która skutecznie i dokładnie rozpoznaje cyfry od 0 do 9.

**Opis funkcjonalności projektu**

1. **Trenowanie modelu**:   
   Użytkownik może wytrenować sieć neuronową na zbiorze danych MNIST.
2. **Ładowanie obrazu**:   
   Użytkownik może załadować obraz, który chce poddać rozpoznaniu.
3. **Przewidywanie cyfry**:   
   Aplikacja wyświetla przewidywaną cyfrę na podstawie załadowanego obrazu.
4. **Interfejs graficzny (GUI)**:   
   Aplikacja posiada przyjazny dla użytkownika interfejs umożliwiający łatwe korzystanie z powyższych funkcjonalności.

**Krótka instrukcja uruchomienia projektu**

1. **Pobierz projekt**:   
     
   Upewnij się, że masz wszystkie wymagane pliki: create\_model.py, predict\_image.py, main\_gui.py, oraz plik mnist.npz zawierający dane MNIST.
2. **Zainstaluj wymagane biblioteki**:  
   * **tensorflow**
   * **numpy**
   * **opencv-python-headless**
   * **pillow**
   * **tkinter**
3. **Uruchom aplikację**:   
     
   Aby uruchomić aplikację z interfejsem graficznym, otwórz terminal lub wiersz poleceń, przejdź do katalogu, w którym znajdują się skrypty, i uruchom plik main\_gui.py
4. **Korzystanie z aplikacji**:  
   * Kliknij "Create Neural Network", aby wytrenować model.
   * Kliknij "Load Image", aby załadować obraz do rozpoznania.
   * Kliknij "Predict", aby uzyskać przewidywanie cyfry.

**Opis technologii użytych w projekcie**

* **Python**: Język programowania używany do implementacji logiki aplikacji oraz interfejsu graficznego.
* **TensorFlow**: Biblioteka do uczenia maszynowego używana do tworzenia, trenowania i używania sieci neuronowej.
* **NumPy**: Biblioteka do obliczeń numerycznych używana do przetwarzania danych.
* **OpenCV**: Biblioteka do przetwarzania obrazów używana do ładowania i przetwarzania obrazów przed podaniem ich do modelu.
* **Tkinter**: Biblioteka do tworzenia interfejsów graficznych w Pythonie.
* **Pillow**: Biblioteka do przetwarzania obrazów używana do walidacji załadowanych obrazów w GUI.

**Wymagania funkcjonalne**

1. **Trenowanie modelu**:  
   * Aplikacja powinna umożliwiać użytkownikowi uruchomienie procesu trenowania modelu sieci neuronowej na zbiorze danych MNIST.
   * Proces trenowania powinien być wizualizowany za pomocą paska postępu, który pokazuje aktualny status trenowania.
2. **Ładowanie obrazu**:  
   * Aplikacja powinna umożliwiać użytkownikowi załadowanie obrazu w formatach JPEG, PNG i innych popularnych formatach graficznych.
   * System powinien sprawdzać, czy załadowany plik jest prawidłowym obrazem, i wyświetlać odpowiednie komunikaty w przypadku błędów.
3. **Przewidywanie cyfry**:  
   * Aplikacja powinna umożliwiać użytkownikowi uzyskanie przewidywania cyfry na podstawie załadowanego obrazu.
   * Przewidywana cyfra powinna być wyświetlana na ekranie.
4. **Interfejs użytkownika (GUI)**:  
   * Aplikacja powinna posiadać intuicyjny interfejs graficzny umożliwiający użytkownikowi łatwe korzystanie z dostępnych funkcji.
   * GUI powinno zawierać przyciski do trenowania modelu, ładowania obrazu oraz uzyskiwania przewidywania.
5. **Zapis i ładowanie modelu**:  
   * Aplikacja powinna zapisywać wytrenowany model do pliku.
   * Aplikacja powinna umożliwiać ładowanie zapisanego modelu w celu dalszego użycia.

**Wymagania niefunkcjonalne**

1. **Wydajność**:  
   * Proces trenowania modelu powinien być wydajny, a czas odpowiedzi aplikacji powinien być akceptowalny dla użytkownika.
   * Przewidywanie cyfry powinno być wykonane w czasie rzeczywistym po załadowaniu obrazu.
2. **Użyteczność**:  
   * Interfejs użytkownika powinien być prosty i intuicyjny, aby użytkownicy mogli łatwo korzystać z aplikacji bez konieczności dodatkowego szkolenia.
3. **Przenośność**:  
   * Aplikacja powinna działać na różnych systemach operacyjnych, takich jak Windows, macOS i Linux, z odpowiednimi wersjami Pythona i zainstalowanymi bibliotekami.
4. **Dokumentacja**:  
   * Aplikacja powinna być dobrze udokumentowana, w tym instrukcja uruchomienia oraz opisy poszczególnych funkcji.
5. **Niezawodność**:  
   * Aplikacja powinna być niezawodna, tj. nie powinna się zawieszać ani wyłączać w trakcie normalnego użytkowania.
   * Powinna obsługiwać błędy użytkownika, takie jak próba przewidywania bez załadowanego obrazu.

**Interfejs użytkownika i funkcjonalności systemu**

1. **Główne okno aplikacji**:  
   * **Tytuł okna**: "Digit Recognizer"
   * **Etykieta tytułowa**: "Digit Recognizer" wyświetlana na górze okna.
   * **Przycisk "Create Neural Network"**:  
     + Funkcja: Uruchomienie procesu trenowania modelu.
     + Po kliknięciu rozpoczyna się proces trenowania, a pasek postępu pokazuje status trenowania.
   * **Przycisk "Load Image"**:  
     + Funkcja: Otwiera okno dialogowe do wyboru pliku obrazu.
     + Po załadowaniu prawidłowego obrazu, użytkownik jest informowany o sukcesie, a przycisk "Predict" zostaje aktywowany.
   * **Przycisk "Predict"**:  
     + Funkcja: Uruchamia proces przewidywania cyfry na podstawie załadowanego obrazu.
     + Przycisk jest aktywowany dopiero po załadowaniu prawidłowego obrazu.
   * **Etykieta z wynikiem przewidywania**:  
     + Wyświetla przewidywaną cyfrę po wykonaniu przewidywania.
   * **Pasek postępu**:  
     + Funkcja: Pokazuje postęp trenowania modelu.
     + Pasek jest aktualizowany na bieżąco podczas trenowania.
2. **Funkcjonalności systemu**:  
   * **Trenowanie modelu**:  
     + Po kliknięciu przycisku "Create Neural Network" uruchamiany jest proces trenowania modelu.
     + Pasek postępu jest aktualizowany w trakcie trenowania modelu, pokazując użytkownikowi postęp.
     + Po zakończeniu trenowania użytkownik otrzymuje komunikat informujący o pomyślnym zakończeniu procesu.
   * **Ładowanie obrazu**:  
     + Po kliknięciu przycisku "Load Image" otwiera się okno dialogowe pozwalające na wybór pliku z obrazem.
     + Po pomyślnym załadowaniu obrazu użytkownik jest informowany o sukcesie, a przycisk "Predict" zostaje ”aktywowany”.
     + Jeśli plik nie jest prawidłowym obrazem, użytkownik otrzymuje komunikat o błędzie.
   * **Przewidywanie cyfry**:  
     + Po kliknięciu przycisku "Predict" aplikacja wyświetla przewidywaną cyfrę na podstawie załadowanego obrazu.
     + Jeśli model nie został wcześniej wytrenowany, użytkownik otrzymuje komunikat o konieczności wytrenowania modelu.
   * **Obsługa błędów**:  
     + Aplikacja obsługuje błędy, takie jak brak modelu, niepoprawny plik obrazu itp., wyświetlając odpowiednie komunikaty użytkownikowi.

**Wnioski**

Projekt "Digit Recognizer" jest dobrze zorganizowany i spełnia wymagania funkcjonalne i niefunkcjonalnych. Korzysta z klasyfikatora sieci neuronowej do rozpoznawania cyfr na podstawie obrazów, umożliwiając użytkownikowi zarówno trenowanie modelu, jak i przewidywanie cyfr na podstawie załadowanych obrazów. Interfejs użytkownika jest intuicyjny, co ułatwia korzystanie z aplikacji nawet osobom niezaznajomionym z programowaniem.

**Możliwe usprawnienia projektu**

1. **Rozszerzenie funkcjonalności**:   
     
   Można rozważyć dodanie dodatkowych funkcji, takich jak możliwość zapisywania historii przewidywań, interaktywnych narzędzi do analizy danych itp.
2. **Rozwinięcie obsługi błędów**:   
     
   Warto rozszerzyć obsługę błędów, aby aplikacja była jeszcze bardziej odporna na różne scenariusze użytkowania, np. lepsze komunikaty o błędach dla użytkowników.
3. **Usprawnienie interfejsu użytkownika**:   
     
   Można rozważyć wprowadzenie bardziej zaawansowanego interfejsu użytkownika z dodatkowymi funkcjami, rysowanie cyfr, zapisywanie i wczytywanie obrazów itp.
4. **Zwiększenie zbioru danych treningowych:**  
    Aktualnie model jest trenowany na standardowym zbiorze danych MNIST. W przyszłości można rozszerzyć zbiór danych treningowych o dodatkowe, bardziej zróżnicowane przykłady. Może to obejmować dane pochodzące z różnych źródeł, zawierające cyfry napisane różnymi stylami, w różnych warunkach oświetleniowych, na różnych tłach itp. Dzięki temu model będzie bardziej wszechstronny i dokładniejszy w rozpoznawaniu cyfr w różnych kontekstach.

*Przy realizacji tych usprawnień projekt "Digit Recognizer" mógłby stać się jeszcze bardziej użyteczny i wszechstronny dla użytkowników.*