

Przetwarzanie Języka Naturalnego

System do odczytywania hieroglifów przy pomocy biblioteki scikit-learn

JK, KK, MK

I. Wstęp

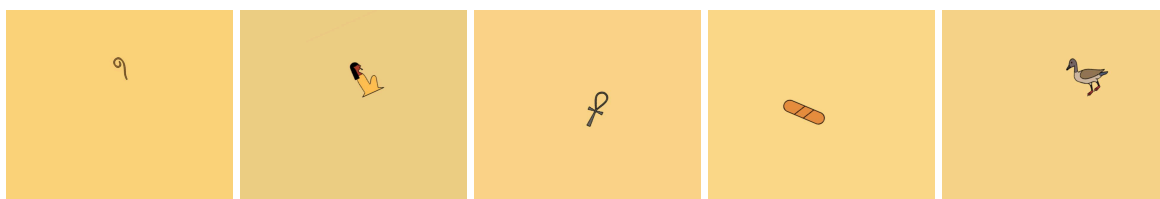
Cel projektu

Celem niniejszego projektu było stworzenie programu potrafiącego poprawnie odczytywać hieroglify ideograficzne oraz zdekodowanie ich na język angielski.

Zakres pracy

Do projektu został użyty zbiór danych Egyptian Hieroglyphics Dataset ze strony <https://www.kaggle.com/datasets/waleedumer/egyptian-hieroglyphics-datasets>. Posiada on 7779 plików do wytrenowania modelu oraz 2100 plików testowych, na których sprawdzane jest, czy model poprawnie rozpoznaje kształty.

Poniżej przedstawione zostały przykładowe hieroglify.



Metodyka

W projekcie wykorzystane zostały następujące procesy:

1. Przygotowanie zbioru danych obejmujące poniższe elementy:
 - a. Zgrupowanie danych w katalogi szkoleniowe i testowe.
 - b. Implementacja klasy ImageDataset do obsługi ładowania obrazów, kodowania etykiet i transformacji.
 - c. Wyodrębnienie i kodowanie etykiet za pomocą LabelEncoder.
 - d. Przekształcanie obrazów przy użyciu transformacji powiązanych ze wstępnie wytrenowanym modelem ResNet-50.
2. Trening modelu - użycie wstępnie wytrenowanego modelu ResNet-50 jako modelu bazowego. Na proces uczenia składa się:

- a. Wykorzystanie pętli szkoleniowej przy określonej ilości epok, korzystając z optymalizatora Adam i funkcji utraty entropii krzyżowej. Przy każdej iteracji parametry były aktualizowane w oparciu o gradienty obliczone na podstawie strat.
 - b. Na koniec każdej iteracji obliczane i drukowane były straty w celu śledzenia postępów modelu i dopasowywania parametrów.
3. Ocena - po zakończeniu uczenia model zostawał poddawany ocenie:
 - a. Wydajność modelu została oceniona na podstawie testowego zestawu danych.
 - b. Dokładność została obliczona przez porównanie przewidywanych etykiet z prawdziwymi.
 - c. Błędnie sklasyfikowane obrazy zostały zidentyfikowane i zapisane do dalszej analizy.
4. Wizualizacja - w celu łatwiejszego wglądu w uzyskane efekty uczenia, zaimplementowane zostały następujące metody wizualizacji:
 - a. Pokazanie zestawu błędnie sklasyfikowanych obrazów z ich prawdziwymi i przewidywanymi etykietami.
 - b. Wyświetlanie losowych obrazów z zestawu testowego wraz z ich prawdziwymi etykietami i najlepszymi przewidywanymi przez model, wraz z prawdopodobieństwem.

II. Część teoretyczna

Wiedza potrzebna do rozpoznawania hieroglifów

Znaki w piśmie hieroglificznym dzielą się na fonetyczne, ideograficzne i determinatywne. Ich prawidłowe rozpoznanie jest kluczowe w zrozumieniu przekazywanej treści.

Projekt skupia się na znakach ideograficznych, które wskazują na konkretne pojęcia, a nie litery.

Uczenie maszynowe może być przydatnym narzędziem do zrozumienia hieroglifów - za pomocą algorytmów zdolnych do samodzielnego uczenia się na podstawie dostarczonych danych możliwe jest uzyskanie wysoce prawdopodobnych wyników. Konieczne jest jednak użycie odpowiednich technik statystycznych i optymalizacyjnych, aby model mógł rozpoznać wzorce i sklasyfikować obrazy.

W projekcie wykorzystane zostały biblioteki:

- PyTorch - dostarczająca zestaw narzędzi do pracy z tensorami, a także definiowania modeli sieci neuronowych i operacji optymalizacyjnych.
- Torchvision - pakiet biblioteki PyTorch, zawierający narzędzia do pracy z zestawami danych obrazowych oraz zestawy narzędzi do przetwarzania obrazów.
- Scikit-learn i scikit-image - do konwersji etykiet do postaci liczbowej oraz wczytywania i zapisywania obrazów.

Opis działania i implementacja

Klasa `ImageDataset` została zaimplementowana do przetwarzania własnych zestawów danych złożonych z obrazów. Jest rozszerzeniem klasy `Dataset` z biblioteki Py-torch. W konstruktorze odczytujemy pliki JPG z katalogu. Następnie zapisywane są ich ścieżki i kodowane etykiety za pomocą `LabelEncoder` z biblioteki scikit-learn.

Metoda `__getitem__()` zwraca obraz z możliwą jego transformacją.

W kolejnej części następuje inicjalizacja wstępnie wytrenowanego modelu ResNet-50, używając domyślnych dla niego wag. Model ustawiany jest w tryb ewaluacji.

Inicjalizowane są obiekty klas ImageDataset oraz DataLoader. Obiekty są osobne dla zbiorów danych treningowych i testowych.

Ostatnia, wynikowa warstwa modelu składa się z liczby neuronów równej liczbie osobnych klas hieroglifów z naszego zbioru danych.

W celu obliczenia strat wykorzystana zostanie funkcja CrossEntropyLoss(), a do optymalizacji - optymalizator Adam z współczynnikiem uczenia równym 0.00005.

Funkcja train() używana jest do trenowania modelu. Dla danej liczby epok (11) wykonywane są następujące kroki:

- ustawienie modelu w tryb uczenia,
- iteracja przez bloki danych:
 - reset gradientu optymalizatora,
 - przejście przez model w celu uzyskania predykcji,
 - obliczanie straty i przejście wsteczne, obliczenie gradientu,
 - aktualizacja parametrów modelu na podstawie gradientu,
 - akumulacja łącznej straty,
- ustawienie modelu w tryb ewaluacji,
- przejście przez bloki danych i odczyt % poprawnych predykcji
- zapisanie modelu do pliku

Funkcja test_and_show_misclassifications() odczytuje zapisany model i ustawia go w tryb ewaluacji.

Następnie wykonywane jest przejście przez zestaw danych i sprawdzenie, czy przewidziana etykieta odpowiada właściwej. W przypadku nieodpowiedniego wyniku przypadek zapisywany jest jako niedopasowanie.

W kolejnym kroku niedopasowania są wyświetlane graficznie, za pomocą biblioteki matplotlib, po uprzedniej odwrotnej transformacji w celu uzyskania etykiet.

Funkcja show_random_predictions() wykorzystana została do wybrania kilku obrazów z testowego zestawu danych, dokonania predykcji wykorzystując wytrenowany model, a następnie do wyświetlenia na wykresie najbardziej prawdopodobnych wyników.

Na początku model jest odczytywany i ustawiony w tryb ewaluacji, a zdefiniowana ilość próbek jest wybierana z zestawu. Dla każdej próbki odczytywany jest tensor obrazu i jego pozostałe parametry. Wykonywane jest przejście przez model w celu

uzyskania predykcji, następnie obliczane są prawdopodobieństwa przy pomocy funkcji `softmax()`. Wektor prawdopodobieństwa jest przeniesiony do procesora i przekonwertowany na tablicę, wybierając pierwszy element.

W kolejnym kroku uzyskane wyniki są sortowane malejąco, transformacja odwrotna pozwala na uzyskanie etykiet, a 5 najlepszych predykcji jest wyświetlana graficznie.

III. Podsumowanie

- Cel projektu został osiągnięty.
- System poprawnie tłumaczy losowe obrazy.
- Program można w łatwy sposób rozbudować o kolejne znaki, które będą rozpoznawane.
- Do rozpoznawania hieroglifów nie trzeba koniecznie używać biblioteki scikit-learn.
- Model może z powodzeniem zostać użyty do rozpoznawania różnych obrazów. Wystarczy przygotować odpowiedni zbiór danych uczących.

Bibliografia

<https://www.kaggle.com/datasets/waleedumer/egyptian-hieroglyphics-datasets> - dostęp 30.05.2024

<https://ii.pk.edu.pl/~rkycia/classes/2015/files/Projekty/Hieroglify/HieroglifyPrezentacja.pdf> - dostęp 30.05.2024

<https://pytorch.org/> - dostęp 30.05.2024

https://pl.wikipedia.org/wiki/Pismo_hieroglificzne - dostęp 30.05.2024