POLITECHNIKA WROCŁAWSKA

Wydział Podstawowych Problemów Techniki

Praca inżynierska

Rozwój mobilnej aplikacji do rozpoznawania języka migowego z wykorzystaniem transformerów

Development of mobile application for sign language recognition using transformers

Autor: Maciej Grzesik asda asds

Opiekun: Prof. uczelni dr hab. Sebastian Kraszewski

Wrocław 2024

Moim ...

Spis treści

1 Wprowadzenie **7**
1.1 Motywacja 7
1.2 Cel 8
1.3 Zakres 8

2 Teoria **9**
2.1 Architektura 10
2.2 Zasada działania 13
2.2.1 Scaled dot-product 13
2.2.2 Multi-Head Attention 13

Bibliografia **15**

Rozdział 1

Wprowadzenie

1.1 Motywacja

Wykluczenie społeczne osób głuchoniemych stanowi poważny problem w społeczeństwie, wynikający z braku mozliwości komunikacji werbalnej w sferze publicznej. Uszkodzenie słuchu, szczególnie w życiu codziennym, powoduje trudności w komunikacji z otoczeniem, które opiera się na konwencjonalnej mowie. Rodzi to problem z mozliwością partycypacji w prostych aktywnościach poczynając od uczestniczenia w zyciu społecznym do m.in. jakości otrzymywanych usług medycznych.

Nalezy zaznaczyć, iz znajomość języka migowego wsród ludzi zdrowych ogranicza się do sytuacji w których osoby im bliskie są dotknięte problemem uszkodzonego słuchu. Język ten, będący podstawową formą komunikacji dla osób głuchoniemych nie jest szeroko nauczany a jego znajomość często wymaga zaangazowania się w dodatkowe kursy tudziez szkolenia, nierzadko płatne.

Na podstawie badań Pauliny Malczewskiej wynika, że aż 90% ankietowanych głuchoniemych doświadcza alienacji oraz dyskryminacji ze strony słyszących [1]. Zjawiska te niezaprzeczalnie przyczyniają się do przewlekłego obnizenia nastroju, lęku przed byciem postrzeganym przez społeczeństwo oraz ogólnym spadkiem poczucia własnej wartości i samooceny. Te aspekty stawiają solidny grunt pod rozwój chorób psychicznych tj. depresja, epizody nastroju depresyjnego, zaburzenia adaptacyjne czy fobii społecznej co potwierdzają poszczególne wytyczne diagnostyczne zawarte w DSM-5 [2].

Istotnym jest zaadresowanie tego problemu za pomocą aplikacji wspierającej osoby głuchonieme. Nie tylko ułatwi to komunikację, ale także przyczyni się do zwiększenia inkluzywności społecznej, zapewniając równe szanse i redukując poziom dyskryminacji.

1.2 Cel

Głównym celem pracy jest stworzenie algorytmu opertego na transofrmerach czyli na architekturze głębokiego uczenia maszynowego, którego zadaniem będzie klasyfikacja znaków języka migowego. Istotnym jest zasięgnięcie do rozwiązań opartych na uczeniu maszynowym gdyz rozwiązania oparte na algorytmice nie sprawdzają się w przypadkach dotyczących widzenia komputerowego, które jest znaczącym elementem tej pracy. [elaborate on this] Umozliwi to tłumaczenie języka migowego na tekst w czasie rzeczywistym, co ułatwi komunikację osobom głuchoniemym.

Sam algorytm zaimplementowany zostanie w aplikacji mobilnej docelowo dedykowanej na smartfony z systemem oparacyjnym Android oraz iOS. Mozliwe jest to dzięki wykorzystaniu środowiska Flutter, które pozwala na tworzenie oprogramowania opartego o język Dart, a następnie na kompilowanie kodu przy wykorzystaniu natywnych narzędzi kompilacyjnych (Android Studio dla systemu Android; Xcode dla systemu iOS).

1.3 Zakres

W ramach pracy inżynierskiej zaprojektowany zostanie przyjazny interfejs użytkownika, który umożliwi proste korzystanie z aplikacji zarówno przez osoby głuche, jak i słyszące. Wykorzystanie środowiska Flutter pozwoli na implementacje interaktywnego interfejsu użytkownika przy wykorzystaniu wbudowanych funkcjonalności oraz zewnętrznych modułów utrzymywanych przez społeczność programistów. Dodatkowo zostanie wyeliminowana potrzeba pisania kodu w natywnym dla danego środowiska języku, co zapewnia aplikacji możliwość działania na różnych mobilnych systemach operacyjnych.

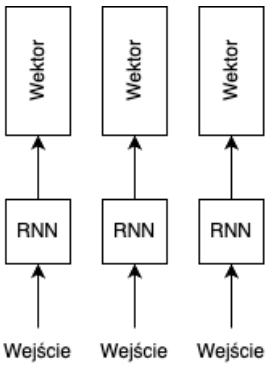
Zaimplementowany zostanie również model mający na celu klasyfikowanie odpowiednich filmów wideo do odpowiadającego im tesktu. W tym etapie projektu zostaną zastosowane odpowiednie algorytmy, które umożliwią skuteczne uczenie modelu na podstawie zebranych danych dotyczących gestów języka migowego. Następnie za pomocą odpowiednio napisanego potoku dane zostaną przesłane do modelu tym samym zaczynając proces uczenia. Proces ten będzie obejmował zarówno fazę wstępną, w której model będzie dostosowywany do charakterystyki danych, jak i fazę walidacji, w której oceni się jego dokładność i zdolność do generalizacji gestów języka migowego. Istotnym jest również przeprowadzenie testów funkcjonalności modelu w warunkach rzeczywistych aby potwierdzić jego poprawne działanie lub wprowadzanie ewentualnych poprawek. Tak skonstruowany i wyuczony model następnie zostanie zkonwertowany do odpowiedniego formatu który następnie zostanie zintegrowany z aplikacja mobilną.

Rozdział 2

Teoria

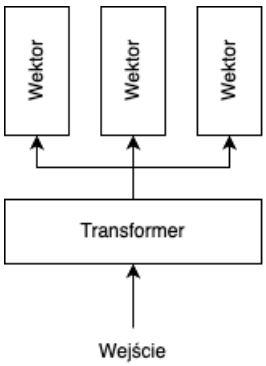
Kluczowym aspektem pracy, na której bazuje duża część funkcjonalności jest model uczenia maszynowego oparty na architekturze transformerów. Przy jego pomocy aplikacja będzie w stanie rozpoznawać znaki języka migowego w czasie rzeczywistym. Transformery stanowią szczególną część szeroko pojętej dziedziny uczenia maszynowego, znajdują one zastosowanie w przetwarzaniu języka naturalnego, wizji komputerowej, ale także w audio i przetwarzaniu multimedialnym. W odróżnieniu od poprzednio stosowanych modeli uczenia maszynowego opierających się na podejściu rekurencyjnym i mechanizmie uwagi, naukowcy pracujący dla firmy Google zaproponowali rozwiązanie wykorzystujące jedynie mechanizm uwagi. Jest to istotny element, który róznicuje transformery od rekurencyjnych modeli uczenia maszynowego. Wyzbycie się rekurencyjności w procesie uczenia przyczyniło się do zwiększenia równoległości, skutkiem czego jest znaczne przyspieszenie czasu uczenia modelu przy zachowaniu wysokiej precyzji [3].

W tradycyjnych modelach rekurencyjnych sieci neuronowych (RNN) dane przetwarzane są sekwencyjnie tzn., że są analizowane krok po kroku (patrz 2.1) [4] co skutkuje ograniczeniami m.in. w równoległości przetwarzaniu danych.



Rysunek 2.1: Przepływ informacji wejściowych w rekurencyjnych sieciach neuronowych

Rozwiązania inżynieryjne wykorzystane w transformerach pozwalają modelom na przetwarzanie wszystkich elementów sekwencji jednocześnie (patrz 2.2).



Rysunek 2.2: Przepływ informacji wejściowych w transformerach

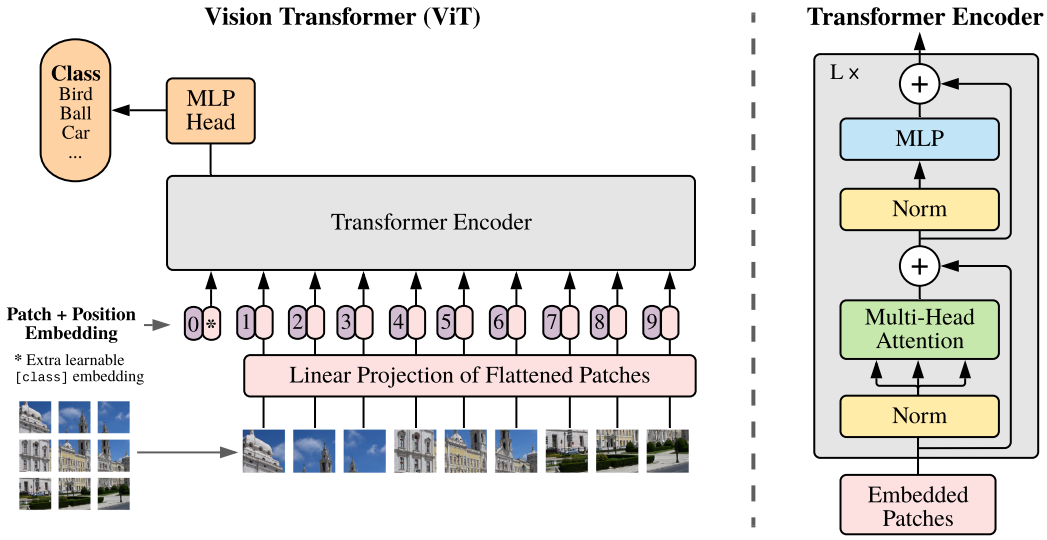
W praktyce oznacza to, że uczenie transformerów przebiega w określonej stałej ilości $O(1)$ sekwencyjnie wykonywanych operacji, modele RNN wymagają natomiast $O(n)$ sekwencyjnie wykonywanych operacji.

2.1 Architektura

Na architekturę transformera składają się dwa główne komponenty: koder oraz dekoder.

Koder składa się z warstwy osadzeń (eng. embedding layer) oraz kodowania pozycyjnego, po których następuje wiele warstw kodera. Warstwa osadzeń zamienia dane wejściowe (słowa, obrazy) na ich reprezentacje w postaci wektorów. Zazwyczaj reprezentacja jest wektorem o wartości rzeczywistej, który koduje znaczenie słowa w taki sposób, że oczekuje się, że słowa znajdujące się bliżej w przestrzeni wektorowej będą miały podobne znaczenie [5]. Osadzanie słów można uzyskać za pomocą modelowania języka i technik uczenia się cech, w których słowa lub frazy ze słownictwa są mapowane na wektory liczb rzeczywistych.

W przypadku modeli Vision Transformer (ViT) proces osadzania obrazów wygląda inaczej niż w przypadku osadzania słów. Obrazy są dzielone na małe, prostokątne fragmenty (eng. patches), które następnie przekształcane są na wektory. Zatem zamiast słow przetwarzane są fragmenty obrazów które dodatkowo mają przypisaną pozycje co pozwala modelowi rozróżnić położenie każdego fragmentu w konteście całego obrazu. W celu przeprowadzenia klasyfikacji jako parametr wejściowy sekwencji podaje się poza odpowiednio wydzielonymi fragmentami obrazu równiez „token klasyfikacyjny” (eng. classification token). Proces ten przedstawiono na schemacie 2.3 [6], a struktura sieci transformerowej jak podnoszą autorzy jest mozliwie najblizsza do struktury zaprezentowanej w artykule „Attention is All You Need”.



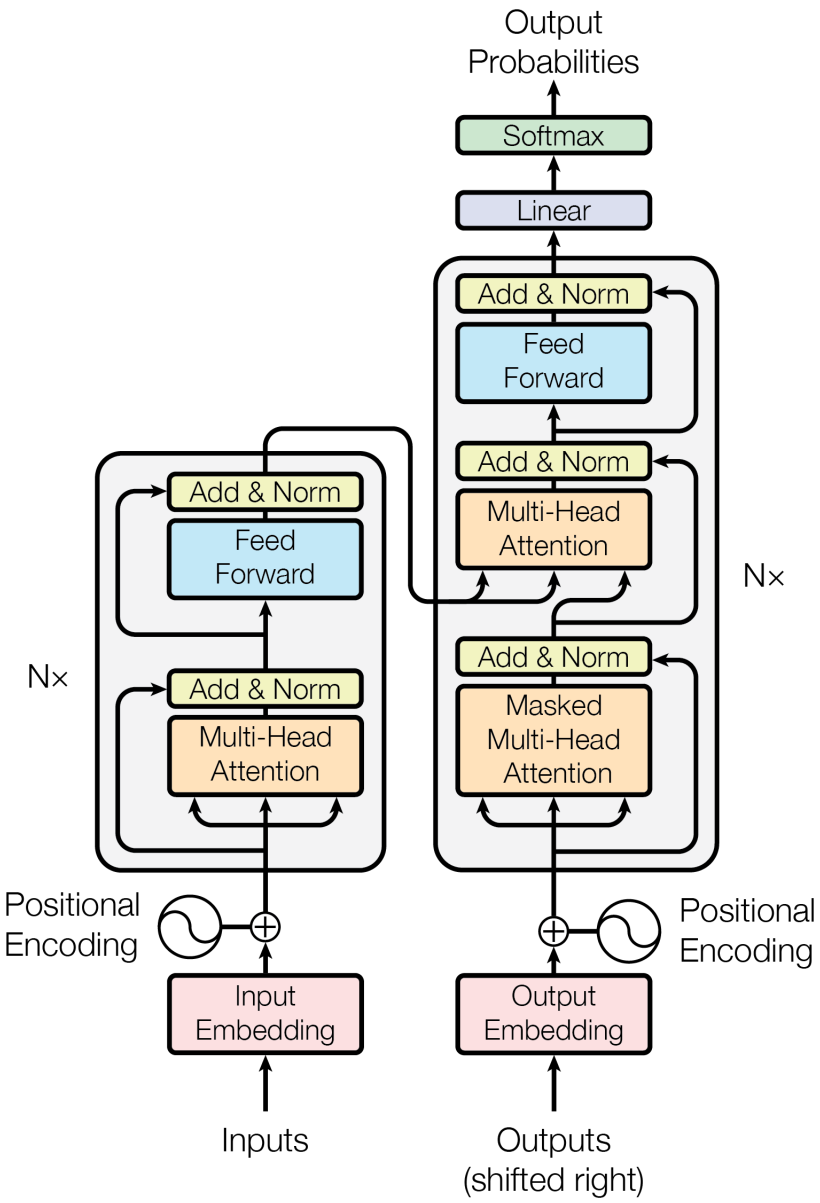
Rysunek 2.3: Schemat uczenia modelu Transformer Vision

Głównym zadaniem kodera jest przekształcanie wejściowych tokenów w kontekstowe reprezentacje. W przeciwieństwie do wcześniejszych modeli, które przetwarzały tokeny niezależnie, koder w transformerze przechwytuje kontekst każdego tokena w odniesieniu do całej sekwencji. Ponieważ transformery nie mają mechanizmu rekurencji, takiego jak RNN, używają kodowania pozycyjnego dodanego do warstwy osadzeń w celu dostarczenia informacji o pozycji każdego tokena w sekwencji. Każda z warstw kodera składa się z dwóch głownych komponentów: mechanizmu samo-uwagi i sieci neuronowej typu feed-forward. Koder pobiera dane wejściowe jako sekwencję wektorów wejściowych, stosuje mechanizm samo-uwagi aby utworzyć pośrednią sekwencję wektorów, a następnie stosuje warstwę feed-forward dla każdego wektora indywidualnie. Warstwy kodera są ułożone w stos, gdzie pierwsza warstwa kodera pobiera sekwencję wektorów wejściowych z warstwy osadzania, tworząc sekwencję wektorów. Ta sekwencja wektorów jest przetwarzana przez drugi koder i tak dalej. Dane wyjściowe z ostatniej warstwy kodera są następnie wykorzystywane przez dekoder.

Dekoder podobnie do kodera, składa się z warstwy embedding layer oraz kodowania pozycyjnego, po której następuje wiele warstw dekodera. Rola dekodera opiera się na tworzeniu sekwencji tekstowych. Każdy dekoder składa się z trzech głównych elementów: maskowanego mechanizmu samo-uwagi, mechanizmu uwagi krzyżowej i sieci neuronowej typu feed-forward. Dekoder działa w podobny sposób jak koder, ale wstawiony jest dodatkowy mechanizm uwagi, który zamiast tego czerpie istotne informacje z kodowań generowanych przez kodery. Mechanizm ten można również nazwać uwagą kodera-dekodera. Podobnie jak pierwszy koder, pierwszy dekoder pobiera informacje pozycyjne i osadzenia sekwencji wyjściowej jako dane wejściowe, a nie kodowania. Transformer nie może wykorzystywać bieżącego lub przyszłego wyjścia do przewidywania wyjścia, więc sekwencja wyjściowa

musi być częściowo zamaskowana, aby zapobiec temu odwrotnemu przepływowi informacji. W przypadku dekodowania, uwaga all-to-all jest nieodpowiednia, ponieważ token nie może zwracać uwagi na tokeny, które nie zostały jeszcze wygenerowane. W ten sposób moduł samo-uwagi w dekoderze jest przyczynowo maskowany. W przeciwieństwie do tego, mechanizm uwagi krzyżowej zwraca uwagę na wektory wyjściowe kodera, które są obliczane przed rozpoczęciem dekodowania przez dekoder. W związku z tym nie ma potrzeby maskowania w mechanizmie uwagi krzyżowej.

Poszczególne bloki architektury wraz z przepływem informacji w modelu zostały zaprezentowane na schemacie 2.4 [3].



Rysunek 2.4: Schemat blokowy architektury transformera

2.2 Zasada działania

Działanie transformera opiera się na mechanizmie tzw. self-attention oraz na omówionych warstwach kodujących i dekodujących. W ramach mechanizmu self-attention wykonywane są dwie główne operacje zwane scaled dot-product attention oraz multi-head attention.

2.2.1 Scaled dot-product

Scaled dot-product attention to działanie polegające na obliczeniu wag dla poszczególnych wartości, tym samym określa, jak bardzo dane fragmenty sekwencji są ze sobą powiązane. Sekwencja wejściowa jest dzielona odpowiednio na wektory K (klucze), Q (zapytania) o długości d_k i V (wartości) o długości d_v. Następnie wykonuje się iloczyn skalarny pomiędzy kluczami i zapytaniami, który dzieli się przez pierwiastek z d_k. Na koniec obliczana jest funkcja softmax w celu obliczenia wag. W praktyce macierz wyników obliczana jest na macierzach kluczy, wartosci i zapytań jak pokazano we wzorze 2.1.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Jak podkreślają autorzy, czynnik czynnik faktoryzujący $\frac{1}{d_k}$ przyczynia się do zwiększenia szybkości i jest bardziej efektywny pod względem wykorzystania pamięci, ze względu na możliwość wykorzystania zoptymalizowanych algorytmów wykonujących obliczenia na macierzach. Rozwiązanie to zostało zaproponowane w kontrze do istniejącej już funkcji uwagi zwanej additive attention.

2.2.2 Multi-Head Attention

Proces ten polega na liniowej projekcji zapytań, kluczy i wartości h razy. Każdy z poszczególnych elementów jest rzutowany na mniejsze przestrzenie za pomocą h różnych, uczonych projekcji liniowych, dzięki czemu uzyskuje się wymiar d_k dla kluczy i zapytań oraz d_v dla wartości. Następnie dla każdej z tych zredukowanych projekcji równolegle wykonywana jest funkcja uwagi co skutkuje uzyskaniem na wyjściu wektora o wymiarze d_v. Po zakończeniu obliczeń przez poszczególne głowy ich wyniki łączone są w jeden wektor co zostało pokazane we wzorze 2.2.

$$\text{MultiHead}(Q, K, V) = \text{Concat}\left(\text{head}_1,...,\text{head}_h\right)W^O \tag{2.2}$$

Takie podejście sprawia, że model w procesie uczenia jest w stanie jednoczesnie analizować informacje z różnych podprzestrzeni reprezentacji przy zachowaniu podobnego kosztu obliczeniowego, jak w przypadku pojedycznej uwagi.

Bibliografia

[1] P. Malczewska. Izolacja społeczna osób z uszkodzonym słuchem jako wspólny obszar badań pedagogiki i antropologii. *Pedagogika a etnologia i antropologia kulturowa. Wspólne obszary badań*, page 128, 2011. 7

[2] American Psychiatric Association. *Diagnostic and Statistical Manual of Mental Disorders (DSM-5®)*. American Psychiatric Publishing, 2013. 7

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 9, 12

[4] M. Mamczur. Czym jest i jak działa transformer sieć neuronowa?, March 2020. Dostęp: 23 maj 2024. 9

[5] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 2024. Online manuscript released August 20, 2024. 10

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. 10