Wspomaganie rehabilitacji

z użyciem wizyjnego śledzenia sylwetki

Aleksandra Mazur, Mateusz Buta

Wstęp

Celem projektu jest przygotowanie aplikacji, która rozpoznaje, zlicza i ocenia poprawność wykonania ćwiczeń rehabilitacyjnych na podstawie śledzenia sylwetki z kamery RGB . Aplikacja będzie mieć postać webową.

Metody

Zliczanie i ocena poprawności wykonania ćwiczeń sprowadza się do czterech kroków.

Detekcja pozy

Do wykrywania postawy na podstawie obrazu wykorzystaliśmy bibliotekę PoseNet, która pozwala wykrywać położenie poszczególnych części ciała w czasie rzeczywistym w przeglądarce przy użyciu TensorFlow.js. Biblioteka wykrywa 17 punktów ciała, wśród których większość stanowią stawy, czyli ruchome połączenie między składnikami szkieletu. To kluczowe punkty, niezbędne do analizy postawy. Estymowana poza oraz każdy wykryty punkt mają przypisany wskaźnik wiarygodności.

Porównywanie pozy

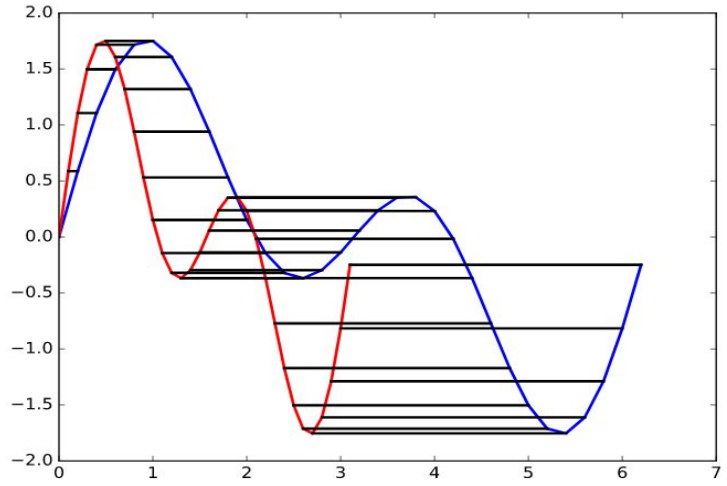
Do porównania pozy skorzystaliśmy z miary podobieństwa consinusowego wektorów 17 punktów ciała. Miara ta jest dodatkowo ważona wskaźnikami wiarygodności. Przed porównaniem wektory są skalowane i transponowane, tak aby najmniejszy prostokąt otaczający wszystkie punkty miał wymiary 1000x1000. Następnie wartości są normalizowane, aby miały porównywalne wartości. Po dokonaniu tych operacji detekcja pozy jest niewrażliwa na lokalizację i rozmiar ćwiczącej osoby.

Detekcja ćwiczenia

Zakładamy, że ćwiczenie jest wykonywane w podobnym tempie, dlatego do dalszej analizy kierowana jest sekwencja klatek zebrana w czasie wykonywania ćwiczenia wzorcowego.

Porównywanie ćwiczeń

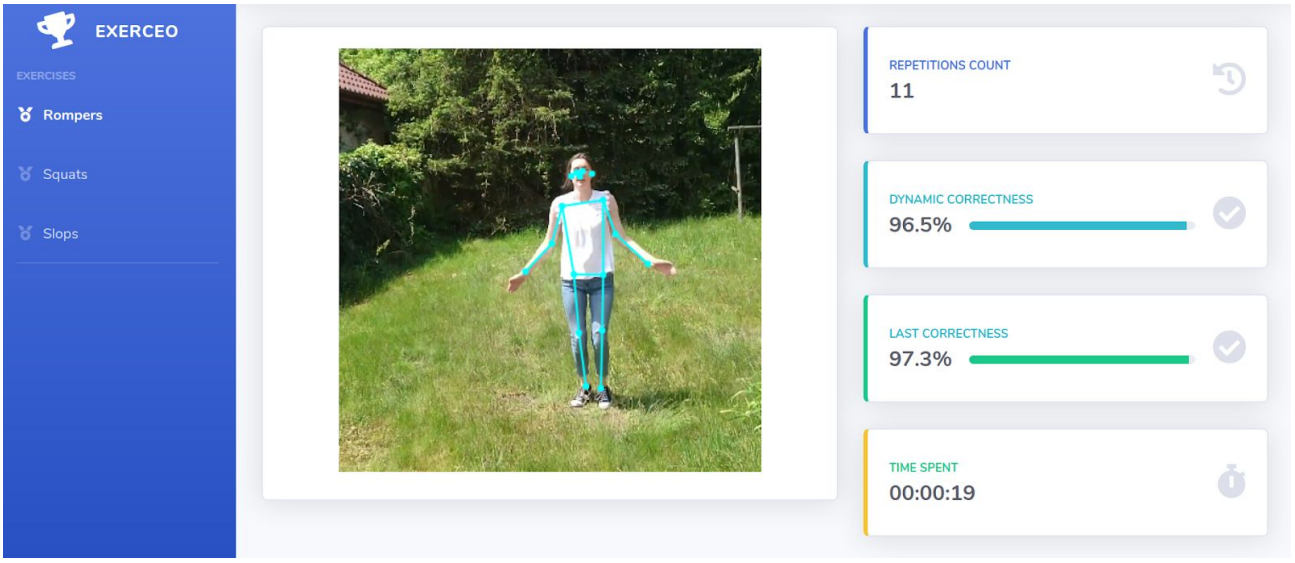
Do porównywania sekwencji obrazów, czyli ćwiczenia wykorzystaliśmy dynamiczną transformatę czasową (DTW Dynamic Time Warping). To algorytm działający w oparciu o programowanie dynamiczne, jest wykorzystywany do mierzenia podobieństwa między dwoma sekwencjami sygnałów czasowych o różnej długości. Algorytm dynamicznej transformaty czasowej wykorzystany w naszej aplikacji pozwala bardzo dobrze dopasować kilkanaście wyłapanych obrazów z kamery do ćwiczenia wzorcowego, które jest próbkujemy kilkanaście razy dokładniej.



Dopasowanie dwóch sekwencji czasowych różnej długości przez DTW.

Opis Implementacji

Realizacja jest oparta o aplikację webową napisaną w języku Javascript. Nasza aplikacja pozbawiona jest pełnoprawnego back-endu, i razem z biblioteką PoseNet działa w przeglądarce. Wygląd naszej aplikacji oparty jest o szablon Bootstrap.



Wygląd aplikacji. Po lewej menu z dostępnymi ćwiczeniami, w środku projekcja ćwiczenia lub obraz z kamerki, a po prawej wyświetlana jest liczba powtórzeń i statystyki poprawności.

Użytkownik wybiera jedno z dostępnych ćwiczeń, Na początku program dokonuje projekcji wzorcowego ćwiczenia spowolnionego dziesięciokrotnie, równocześnie zapamiętuje sekwencję klatek. Nasz program zapamiętuje dziesięciokrotnie więcej klatek ćwiczenia wzorcowego niż później jest w stanie zebrać na podstawie obrazu z kamerki. Następnie program przechodzi do analizy obrazu z kamerki, porównując go z zapisaną sekwencją ćwiczenia.

Opis testów

Testy wykonywaliśmy pod kątem dwóch kryteriów: poprawność działania na osobach o różnej posturze oraz wydajność działania aplikacji.

Testy poprawności działania

W początkowej fazie rozwoju aplikacji różne propozycje algorytmu testowane były tylko na jednej osobie. Dzięki temu mogliśmy na bieżąco kontrolować wpływ zmian wprowadzanych przez nas w algorytmie i tym samym stworzyć obiecujący prototyp rozwiązania, które można dalej rozwijać i szerzej testować. Finalna wersja aplikacji została przetestowana na 4 osobach. Podczas doboru osób staraliśmy się zwrócić uwagę, by ich sylwetki były zróżnicowane, szczególnie pod względem wzrostu, co potencjalnie mogło stanowić pewne utrudnienie dla algorytmu porównującego. Przetestowane zostały trzy ćwiczenia - pajacyki, przysiady, i skłony. Zebraliśmy również materiały o innych ćwiczeniach, które sprawdzały ocenę poprawności źle wykonywanego ćwiczenia.

Testy wydajnościowe aplikacji

Testowaliśmy różne kombinacje trzech parametrów odpowiadających za wydajność aplikacji:

a. *mobileNetArchitecture* - istnieje kilka modeli PoseNet różniących się wielkością i dokładnością. Model 1.01 jest największy, ale będzie najwolniejszy. Natomiast model 0,50 jest najszybszy, ale najmniej dokładny.
b. *outputStride* - wpływa na wysokość i szerokość warstw w sieci neuronowej. Im niższa wartość wyjściowego kroku tym większa dokładność, ale mniejsza prędkość. Z kolei im większa wartość tym większa prędkość, ale mniejsza dokładność.
c. *ImageScaleFactor* - współczynnik przeskalowania obrazu przed podaniem do sieci neuronowej. Im mniejszy obraz, tym szybciej zostanie przetworzony, ale mniej dokładnie z powodu zgubienia szczegółów.

Opis uzyskanych wyników

Poprawność działania

Przeprowadzone testy pokazały, że aplikacja radzi sobie dobrze, nawet jeśli osoba wykonująca ćwiczenie ma inny typ sylwetki niż osoba z ćwiczenia wzorcowego. Aplikacja poprawnie zliczała ilość powtórzeń i przerwała zliczanie, gdy jeden z uczestników testów omyłkowo wykonał ćwiczenie w niepoprawny sposób.

Przeprowadzone testy pokazały jednak, że wykorzystana biblioteka PoseNet nie radzi sobie we wszystkich przypadkach. Jest dosyć wrażliwa na tło, dlatego ćwiczenie wzorcowe zostało przygotowane tak, by ćwiczący znajdował się na kontrastowym, możliwie jednolitym tle. Osobom korzystającym z aplikacji należy zwrócić uwagę, by ćwiczyły w dobrze oświetlonym miejscu, na tle innym niż kolor ich ubrań. Kolejną zaskakującą obserwacją było to, że PoseNet rozpoznaje bose stopy jako dłonie, co znacząco pogarsza poprawność wykonywanego ćwiczenia. Dodatkowo, wydaje się że logika PoseNet ogranicza wykrywanie nienaturalnych pozycji ciała. Warto o tych ograniczeniach poinformować użytkowników aplikacji, gdyż rzutuje na jakość stworzonego przez nas algorytmu.

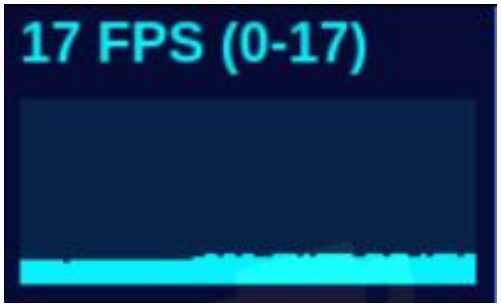

Krzyżowanie nóg oraz gubienie dłoni i stóp to typowe niedoskonałości biblioteki PoseNet.

Wydajność

Przy testowaniu różnych parametrów, nie udało się osiągnąć lepszej wydajności niż 17 klatek na sekundę (tyle ile miały pierwsze filmy w kinie niemym). Przykładowe ćwiczenie, takie jak przysiad lub pajacyk, trwa 2 sekundy, więc jesteśmy w stanie zebrać trochę ponad 30 klatek. Dzięki temu, że ćwiczenie wzorcowe próbkowane jest bardzo dokładnie (ponad 300 klatek), możemy dokonać bardzo dokładnego porównania.

Co bardzo ważne przy testowaniu aplikacji na przygotowanych materiałach filmowych, analiza filmu jest o połowę wolniejsza niż analiza obrazu z kamerki (prawdopodobnie z powodu narzutu związanego z ładowaniem filmu na stronę internetową). Dlatego podczas testowania naszej aplikacji, filmy testowe były odpowiednio spowalnione.

Najlepsza uzyskana wydajność:
Film: 10-11 FPS
Obraz z kamery: 15-17 FPS



Skok wydajności po zakończeniu analizy filmów i przejściu do analizy obrazu z kamerki.

Parametry sieci nie wpłynęły znacząco na działanie aplikacji. Nie zaobserwowano pogorszenia jakości aplikacji, a wydajność poprawiła się nieznacznie. Dlatego proponujemy zostawić domyślny zestaw parametrów.

Podsumowanie

Udało się uzyskać w pełni działającą aplikację, która poprawnie zlicza ćwiczenia i jest w stanie dokonać prawidłowej oceny poprawności wykonywanego ćwiczenia. Aplikacja może posłużyć do wspomagania ćwiczeń rehabilitacyjnych

Aplikacja daje bogate możliwości rozwoju i poprawy działania, pracując nad projektem możemy zaproponować:

- rozbudowa aplikacji o zarządzanie katalogiem ćwiczeń przez rehabilitanta poprzez ładowanie i usuwanie filmów reprezentujących ćwiczenie,
- zapamiętywanie sekwencji klatek każdego ćwiczenia w bazie danych,
- możliwość otrzymywania dedykowanych ćwiczeń przez pacjenta systemu, zapamiętywanie ich wykonywania, przypominanie o ćwiczeniach,
- prezentowanie raportów z wykonanych ćwiczeń pacjentowi i rehabilitantowi,
- poprawa wydajności aplikacji poprzez przeniesienie operacji obliczeniowych na dedykowany backend,
- stworzenie dynamicznego mechanizmu wykrywającego wykonanie sekwencji ćwiczenia w celu dalszej analizy przez algorytm DTW.