Klasyfikator ruchów czujnika IMU na podstawie rekurencyjnej sieci neuronowej

Mateusz Woźniak
wozniakmat@student.agh.edu.pl

Maciej Pawłowski
maciekp@student.agh.edu.pl

Abstrakt

Przedmiotem tego artykułu jest omówienie realizacji zadania klasyfikacji aktywności ruchowych człowieka rejestrowanych przy pomocy czujnika IMU (Inertial Measurement Unit). Czujnik IMU mierzy przyśpieszenia postępowe i kątowe używając żyroskopu, akcelerometru i magnetometru. IMU jest powszechnie stosowane w lotnictwie, robotyce, wirtualnej rzeczywistości i medycynie. W lotnictwie umożliwia precyzyjne sterowanie statkami powietrznymi, a w robotyce wspomaga autonomiczne poruszanie się robotów. Jednym z zastosowań klasyfikatora ruchów może być detekcja przeciągnięcia samolotu. Z kolei w motoryzacji, IMU znajduje użycie w systemach kontroli stabilności pojazdów oraz w zaawansowanych systemach wspomagania kierowcy, które poprawiają bezpieczeństwo. Proponowana jest realizacja klasyfikatora pięciu z góry ustalonych ruchów używając rekurencyjną sieć neuronowej.

1 Wstęp

2 Zbiór danych

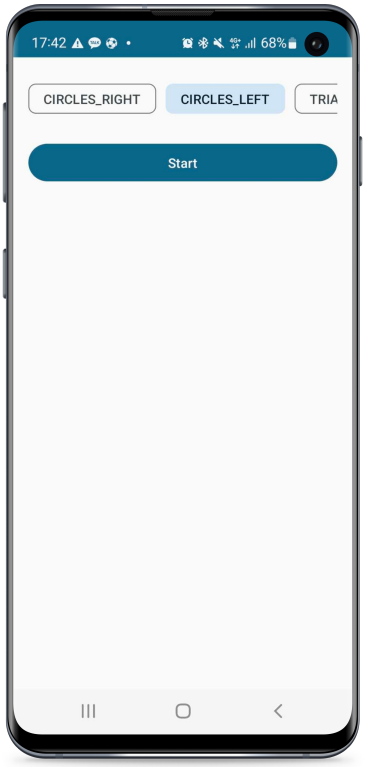
Dane zostały zebrane z urządzenia Samsung Galaxy S10e 2019. Każdy ruch został zebrany ręcznie a seria danych z czujników była zapisywana do pliku csv o ilości wierszy takiej jak ilość kroków czasowych. Interwał próbkowania pomiaru z czujnika został ustawiony na $50ms$. To znaczy, że w ciągu każdej sekundy trwania ruchu następowało 20 odczytów.

Ruchy były wykonywane przy włączonej aplikacji mobilnej napisanej w Kotlinie (rys. 2).

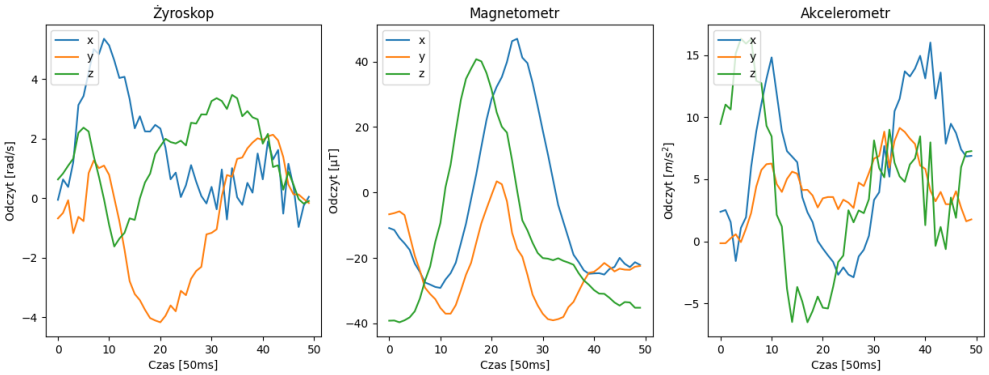
Dane były wysyłane na zdalny serwer na którym był uruchomiony mikroserwis HTTP napisany w Go. Mikroserwis zbierał dane i zapisywał je na dysk. Dzięki temu zbieranie danych odbywało się sprawnie, a mikroserwis dawał nam informacje o ilości ruchów dla każdej z klas.

Przykładowy plik csv (oraz jego wizualizacja na rys. 1):

```
gyro_x;gyro_y;gyro_z;magnetometer_x;...
-0.062384613;-0.15294538;-0.32650748;-43.62;...
-0.87361366;0.41210496;-0.49510628;-43.5;...
-1.7758616;0.20685424;-1.4266758;-43.92;...
-1.8662697;-0.46876273;-1.4040737;-44.399998;...
-0.7575493;-0.8077929;-1.488984;-44.28;...
0.06895141;-1.5170075;-2.1273382;-46.14;...
```



Rysunek 1: Aplikacja mobilna do zbierania danych z czujnika IMU



Rysunek 2: Przykładowy odczyt ruchu dla klasy CIRCLES_RIGHT

Tabela 1: Tabela dostępnych ruchów

Klasa	Opis ruchu	Ilość próbek
SQUARE	Ruch imitujący rysowanie kwadratu	203
TRIANGLE	Ruch imitujący rysowanie trójkąta	179
CIRCLES_LEFT	Rysowanie okręgu przeciwnie z ruchem wskazówek zegara	193
CIRCLES_RIGHT	Rysowanie okręgu zgodnie z ruchem wskazówek zegara	191
FORWARD_BACK	Ruch "od siebie - do siebie"	187

Dane zostały załadowane do tensora o kształcie (B, T, C), gdzie

- B - wsad (32)
- T - oś czasu
- C - cechy (9)

W przypadku, gdy próbki były różnej długości, tensor został dopełniony zerami, by T było równe 100. Dzięki temu kształt tensora odpowiadał wejściu sieci neuronowej.

3 Model

Ze względu na to, że zadanie klasyfikacji ruchów ma być niezależne od czasu, tzn. że ruch może zawierać dowolną ilość próbek zdecydowaliśmy się użyć rekurencyjnej sieci neuronowej z komórką LSTM [1]. Long Short-Term Memory (LSTM) to rodzaj architektury sieci neuronowej rekurencyjnej (RNN), zaprojektowanej w celu przezwyciężenia ograniczeń tradycyjnych RNN w przechwytywaniu i uczeniu się długoterminowych zależności w danych sekwencyjnych. LSTMy zostały wprowadzone przez Seppa Hochreitera i Jürgena Schmidhubera w 1997 roku [2]. Chcemy, aby sieć neuronowa nauczyła się zaleności w czasie pomiędzy zmianami w przyśpieszeniach czujnika [3] [4]. W związku z tym w pierwszym podejściu zdecydowaliśmy sprawdzić architekturę składającą się z 32 komórek LSTM oraz dwóch warstw gęstych o wielkości 24 i 5. Na ostatniej warstwie została zastosowana funkcja aktywacji $softmax$ by uzyskać dystrybucję prawdopodobieństwa klas (realizuje to $CrossEntropyLoss$)

Po kilku próbach znaleźlimy hiperparametry modelu, które dają najlepsze zachowanie sieci. LSTM musi mieć 22 komórki, a warstwa gęsta musi mieć 32 neurony. Przeszukiwanie przestrzeni hiperparametrów zostało zrealizowane metodyką gridsearch. Finalny model ma następującą architekturę:

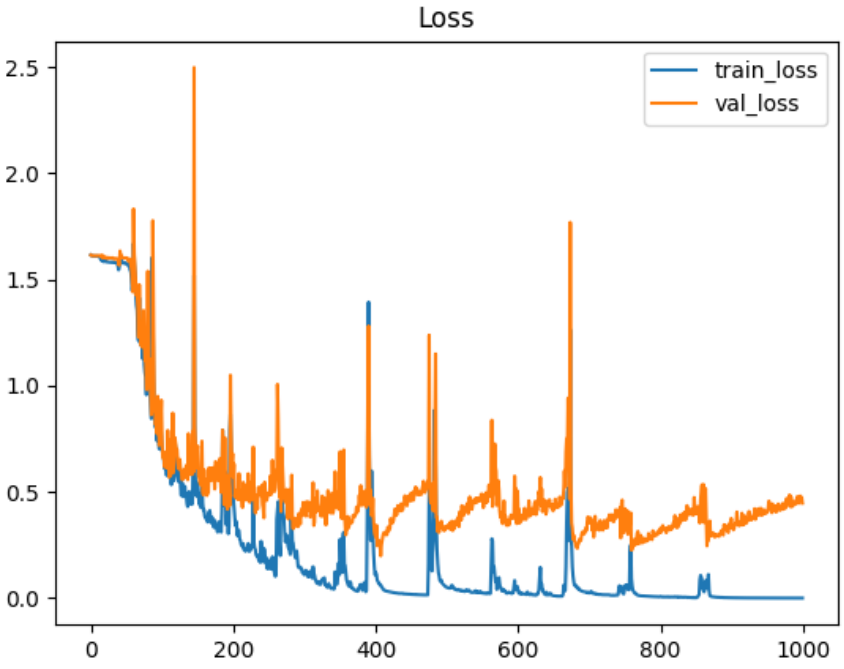
```
import torch
import torch.nn as nn

class Net(nn.Module):
  def __init__(self, input_size):
      super(Net, self).__init__()
      self.lstm = nn.LSTM(input_size, 22, batch_first=True)
      self.fc1 = nn.Linear(22, 32)
      self.fc2 = nn.Linear(32, 5)

  def forward(self, x):
      _, (h_n, _) = self.lstm(x)
      x = h_n[-1, :, :]
      x = self.fc1(x)
      x = self.fc2(x)
      return x
```

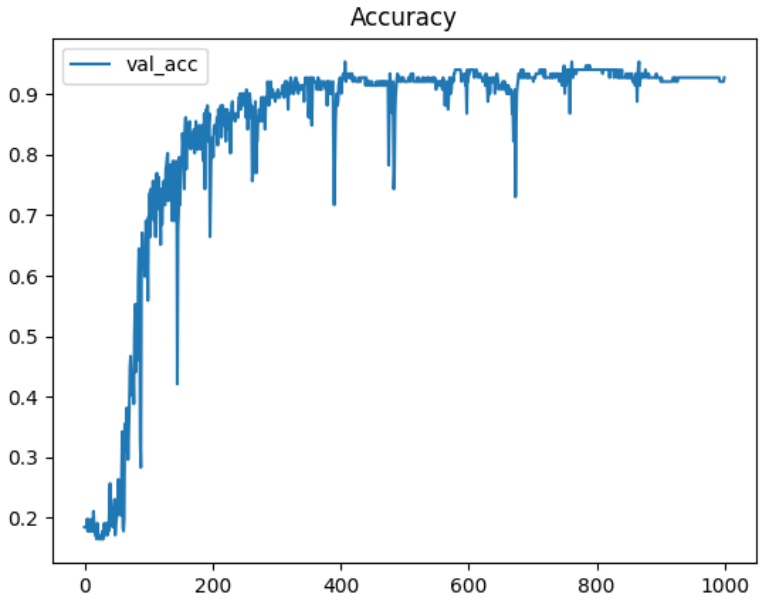
4 Obserwacje

Model jest w stanie rozwiązać problem klasyfikacji i dostarcza zadowalających rezultatów. Zadanie znajdowania zależności między odczytami daleko od siebie nie jest trywialne, aczkolwiek optymalizator jest w stanie znaleźć odpowiedni kierunek by dostosować wagi sieci (rys. 3).

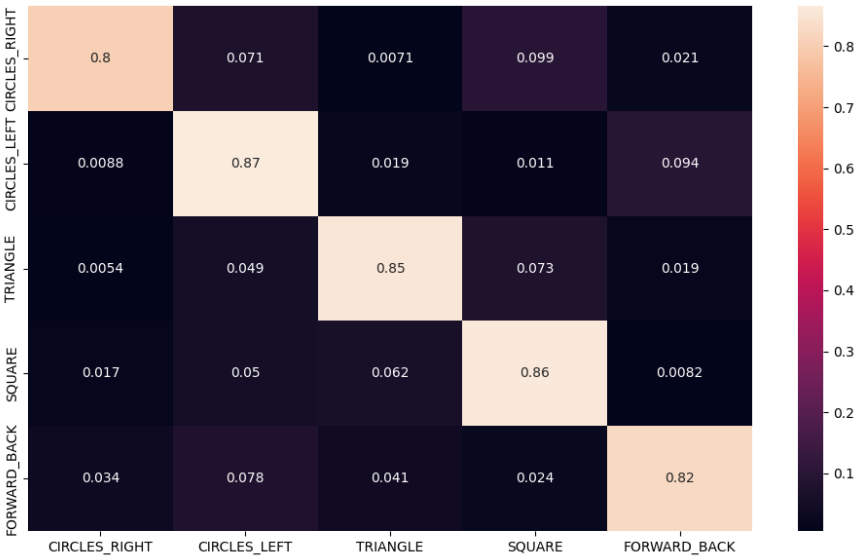


Rysunek 3: Wykres funkcji straty

Użyty optymalizator to $Adam$ ze współczynnikiem uczenia 0.001 Uzyskana precyzja klasyfikacji ruchów to 96% (rys. 4). Sieć wykorzystuje wszystkie 9 cech (3 osie żyroskopu, 3 osie akcelerometru, 3 osie magnetometru). Macierz błędów widoczna na rys. 5.



Rysunek 4: Wykres precyzji na zbiorze walidacyjnym



Rysunek 5: Wykres precyzji na zbiorze walidacyjnym

Co więcej, dostrzegliśmy różnice w zakresach odczytów używając kilku czujników IMU. Pomiary znacząco różniły się od siebie pomiędzy markami i modelami smartfonów. W związku z tym całe badanie oparliśmy na jednym urządzeniu Samsung Galaxy S10e.

5 Ewaluacja modelu

Ze względu na to, że ewaluacja na smartfonie w przypadku rekurencyjnych sieci neuronowych jest trudna, sieć neuronowa została przeniesiona na serwer VPS. Zastosowaliśmy iteracyjną metodologię prac nad projektem:

1. Wytrenuj model na lokalnym sprzęcie (laptop).
2. Zbuduj obraz Dockera z wagami zawartymi w środku i wypchnij go do rejestru
3. Na serwerze zdalnym: ściągnij obraz i uruchom serwer Flask

Dzięki temu byliśmy w stanie szybko ewaluować jakość modelu. Aplikacja mobilna wykonuje żądanie HTTP POST do serwera VPS, by ten zwrócił wyniki.

6 Wnioski

Na podstawie wykresu funkcji straty i wykresu funkcji precyzji modelu można stwierdzić, że decyzja o wykorzystaniu rekurencyjnej sieci neuronowej była trafna. Zaproponowana sieć neuronowa cechuje się wysoką skutecznością w zadaniu klasyfikacji ruchów z urządzenia IMU.

7 Sprzęt

Do pomiarów wykorzystaliśmy smartfon Samsung Galaxy S10e 2019 wyposażony w czujnik IMU. Implementację modelu wykonaliśmy we frameworku PyTorch. Trening sieci neuronowej była wykonywany na Apple Macbook M1 16GB.

Literatura

[1] R. C. Staudemeyer and E. R. Morris, "Understanding lstm–a tutorial into long short-term memory recurrent neural networks," *arXiv preprint arXiv:1909.09586*, 2019.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] P. Rivera, E. Valarezo, M.-T. Choi, and T.-S. Kim, "Recognition of human hand activities based on a single wrist imu using recurrent neural networks," *Int. J. Pharma Med. Biol. Sci*, vol. 6, no. 4, pp. 114–118, 2017.

[4] S. Ashry, R. Elbasiony, and W. Gomaa, "An lstm-based descriptor for human activities recognition using imu sensors," in *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics, ICINCO*, vol. 1, 2018, pp. 494–501.