

bokie uczenie maszyngreen6 a^czonagreen7

Wykrywanie wdechu i wydechu na podstawie dźwięku z mikrofonu

Dominik Lau, Mateusz Kowalczyk, Michał Tarnacki

8 maja 2023

1 Wstęp

Celem projektu było określanie chwil na nagraniu, w których osoba bierze wdech i wydech. Dokonano oceny jakościowej za pomocą detekcji oddechu na żywo jak i ilościowej (przy wykorzystaniu dalej wymienionej metryk).

2 Teoria

2.1 DFT

W celu przejścia z dziedziny natężenia od czasu do dziedziny częstotliwości stosujemy dyskretną transformatę Fouriera. DFT przekształca ciąg skończonych próbek sygnału $a_0, a_1, a_2, \dots, a_{N-1}$ w ciąg $A_1, \dots, A_{N-1} \in \mathbb{C}$

$$A_k = \sum_{n=0}^{N-1} a_n e^{-\frac{k n i \pi}{N}}, 0 \leq k \leq N - 1$$

W naszych zastosowaniach będziemy brali pod uwagę tylko część rzeczywistą tego wielomianu

$$R_k = \operatorname{Re}(A_k), 0 \leq k \leq N - 1$$

Naszą funkcję natężenia w czasie dla pewnego okresu $t \in [t_0, t_0 + B]$, gdzie B - rozmiar bloku (przymijmy $B = 1024 \approx \frac{1}{44}s$) przedstawiamy zatem jako

$$I(t) = \Sigma_f I_f(t) \sin(2\pi f t)$$

stąd dla danego przedziału w czasie jesteśmy w stanie stworzyć wektor natężeń

$$\mathbf{I} = [I_{f_1}, I_{f_2}, \dots, I_{f_n}]$$

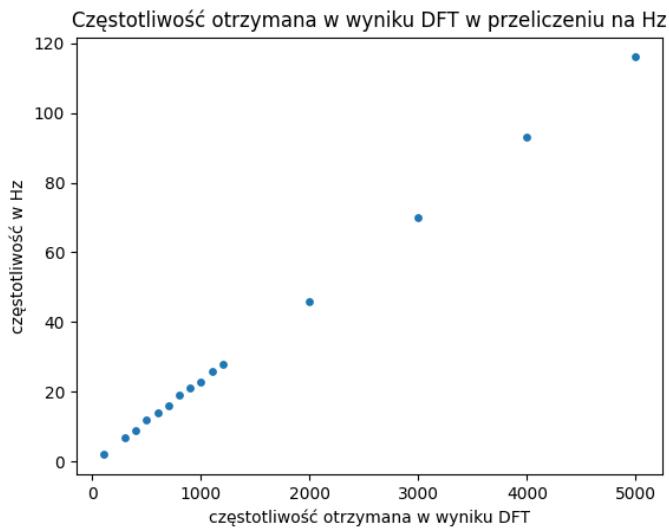
W dalszych rozważaniach wykorzystujemy także średnią częstotliwość, którą liczymy ze wzoru

$$\bar{f} = \frac{\Sigma_f f I_f}{I}$$

Wykorzystujemy implementację transformaty z biblioteki *numpy*.

2.2 Przeliczanie częstotliwości z DFT

W celu wyznaczenia sposobu zamiany częstotliwości otrzymanej za pomocą DFT na znaną jednostkę (Hz) wykonaliśmy kilkanaście nagrań z ustaloną (w Hz) dominującą częstotliwością i odczytaliśmy, której wartości częstotliwości otrzymanej za pomocą DFT odpowiada maksymalne natężenie. Wyniki przedstawia rysunek 1.



Rysunek 1

Po zaobserwowaniu w przybliżeniu liniowego charakteru tej relacji zastosowaliśmy metodę najmniejszych kwadratów do wyznaczenia najlepiej dopasowanej prostej. Otrzymaliśmy następujące wyniki:

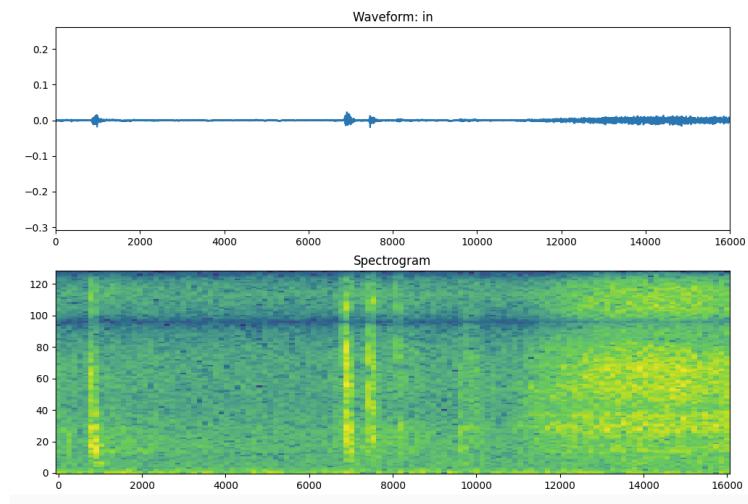
$$f_{Hz} \approx 43,0789 f_{DFT} - 2,8174,$$
$$f_{DFT} \approx 0,0232 f_{Hz} + 0,068,$$

gdzie f_{DFT} to wartość liczbową częstotliwości w jednostkach, w której otrzymuje się wynik DFT, natomiast f_{Hz} to wartość liczbową częstotliwości wyrażonej w Hz.

2.3 STFT

Dla sieci neuronowej zamiast posługiwać się zwykłym DFT (na całym przedziale), używamy STFT, czyli krótko-czasowej transformacji Fouriera. Jest to zadaniczo transformacja Fouriera jednak wykonywana w mniejszych oknach czasowych. Dzięki temu otrzymujemy także informację o zmianie widma w czasie,

czyli jednocześnie posiadamy informację o właściwościach zarówno w dziedzinie czasu jak i częstotliwości. Najistotniejszym parametrem tej metody jest rozmiar okna. Ponieważ iloczyn jego szerokości w dziedzinie czasu i szerokości w dziedzinie częstotliwości jest stały dla danego okna, poprawiając rozdzielczość czasu, pogarszamy rozdzielczość częstotliwości. Użyliśmy gotowej funkcji z biblioteki tensorflow, nie będziemy więc dokładniej opisywać sposobu implementacji STFT. W naszym przypadku, za szerokość okna ustawiliśmy wartość 255. Rysunek 2 reprezentuje wykres natężenia dźwięku w zależności od czasu, oraz spektrogram stworzony na podstawie niego omawianą metodą.



Rysunek 2

2.4 Bramka szumów

Główna metoda odszumiana, z której korzystamy to *Spectral Gating* (rodzaj bramki szumów). Pierw wyznaczany jest spektrogram sygnału za pomocą *DFT* i tworzony jest próg szumu dla każdej jego częstotliwości. Częstotliwości progowe służą do stworzenia maski, której potem używamy do usunięcia niechcianych dźwięków. Następnie ze spektrogramu tworzymy z powrotem natężenie $I(t)$. W projekcie wykorzystujemy gotową implementację bramki szumów z biblioteki *noisereduce*¹. Przykładowe wartości osiąganych SNR w porównaniu z innymi algorytmami usuwania szumu²:

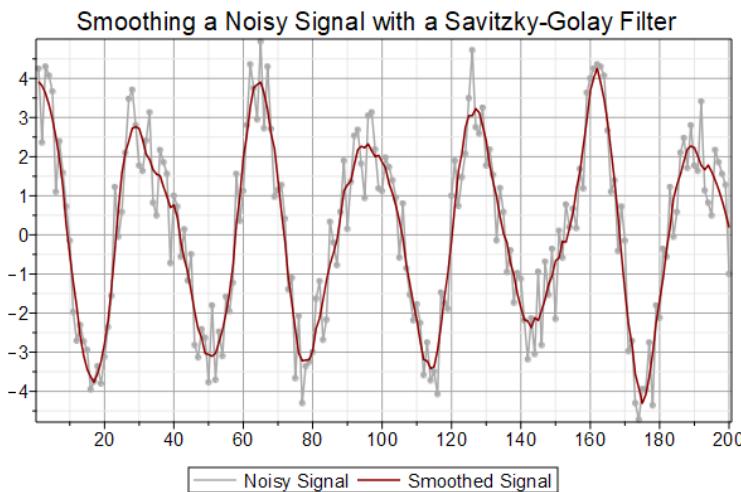
¹<https://pypi.org/project/noisereduce/>

²źródła [3]

algorytm	SNR
<i>LMS</i>	2
<i>Kalman</i>	6
<i>Spectral Gating</i>	14

2.5 Filtr Savitzky'ego-Golaya

Filtru Savitzky'ego-Golaya używamy do wygładzienia spektrogramu po przeprowadzeniu DFT. Polega on na dopasowywaniu zbioru sąsiadujących punktów do wielomianu niskiego stopnia za pomocą metody najmniejszych kwadratów. Te fragmenty wielomianów są potem łączone w wygładzoną funkcję. Korzystamy z gotowej implementacji z biblioteki *scipy*. W przypadku sieci neuronowej korzystamy natomiast z warstw konwolucyjnych, nakładających odpowiednie filtry na obraz.



2.6 Ocena modeli

Zastosowaliśmy następujące miary oceny ilościowej modelu:

$$\begin{aligned}
 accuracy &= \frac{|TP_{wdech}| + |TN_{wdech}|}{|TP_{wdech}| + |FP_{wdech}| + |TN_{wdech}| + |FN_{wdech}|} \\
 precision_{wdech} &= \frac{|TP_{wdech}|}{|TP_{wdech}| + |FP_{wdech}|} \\
 recall_{wdech} &= \frac{|TP_{wdech}|}{|TP_{wdech}| + |FN_{wdech}|} \\
 F_{wdech} &= \frac{2 \cdot precision_{wdech} \cdot recall_{wdech}}{precision_{wdech} + recall_{wdech}}
 \end{aligned}$$

oraz analogicznie $precision_{wydech}$, $recall_{wydech}$ i F_{wydech} .

2.7 Standaryzacja

Dane wejściowe do modelu zawsze standaryzujemy, czyli

$$\tilde{X} = \frac{X - EX}{\sigma_X}$$

stosujemy gotową implementację z *scikit – learn* lub warstwę normalizacyjną w przypadku sieci neuronowej.

2.8 SVM

Pierwszym modelem, jakim dokonujemy klasyfikacji, jest SVM. Metoda ta polega na szukaniu optymalnych wag \mathbf{w}, b . Następnie będziemy klasyfikować według funkcji maszyny uczącej

$$f(x) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b)$$

jeśli f zwraca 1 to traktujemy to jako jedną z binarnych decyzji (w naszym przypadku np. wdech) a jeśli zwraca -1 to drugą (czyli np. wydech). \mathbf{x} jest wektorem cech. Znalezienie optymalnych wag będzie polegało na minimalizacji funkcji kosztu

$$J(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{N} \sum_x \max\{0, 1 - y(\mathbf{w}^T \mathbf{x} + b)\}$$

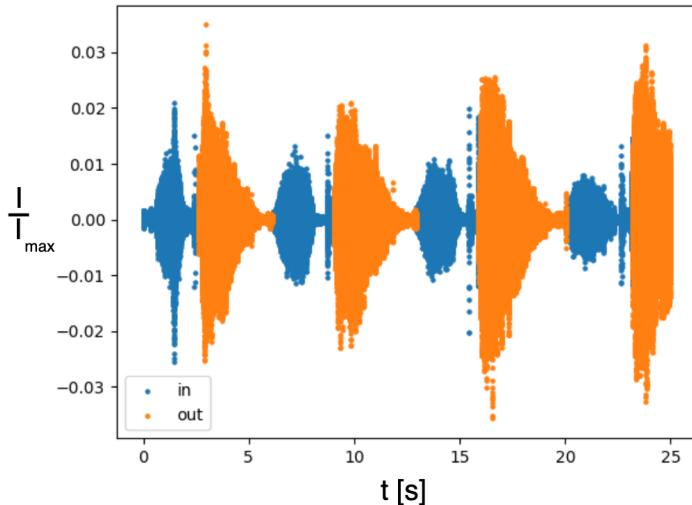
gdzie C jest parametrem uregulowania. Funkcję będziemy minimalizować metodą stochastycznego spadku po gradiencie. W projekcie użyliśmy własnej implementacji SVM-a.

2.9 Sieć neuronowa

Drugim modelem jakim dokonujemy klasyfikacji jest sieć neuronowa. Ze względu na trudność w samodzielnym zaprojektowaniu dobrej sieci neuronowej postanowiliśmy użyć "gotowca" więc stworzonego przez firmę Google Tensorflow'a wraz z interfejsem Keras. TODO opisać jak działa ta sieć

3 Nagrywanie danych oddechowych

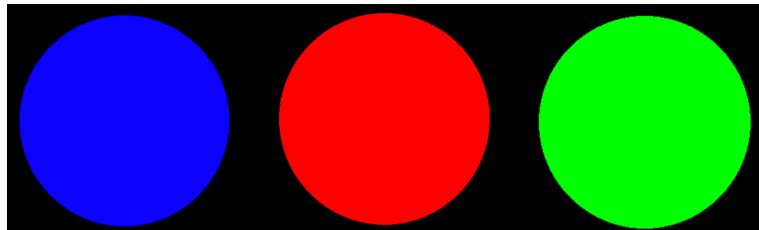
Żeby zapewnić dobre oznaczenie danych, etykietujemy je jeszcze w trakcie nagrywania dźwięku. Osoba nagrywana naciska przycisk, żeby zasygnalizować, że przestała brać wdech i zaczyna wydychać powietrze lub na odwrót. Momenty przejścia z wdechu na wydech i w drugą stronę zapisywane są w pliku .csv, a dźwięk w pliku .wav. Częstotliwość próbkowania ustalamy na 44,1 kHz.



Rysunek 3: Przykładowe nagrane dane, in-wdech, out-wydech. Jak widać dane oznaczone na żywo są bardzo dokładne, być może nie byłobyśmy w stanie osiągnąć takiej dokładności oznaczając ręcznie (lub byłoby to bardzo żmudne).

4 Oddychanie na żywo

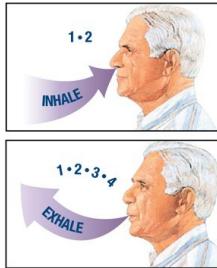
Testowanie modeli do predykcji na żywo odbywało się za pomocą programu wizualizującego aktualnie stwierdzany stan.



Jeżeli koło zwiększa się i jest niebieskie to model wykrywa wdech, jeśli zmniejsza się i jest czerwone to jest to wydech, natomiast kolor zielony oznacza, że natężenie dźwięku nie przekracza pewnego progu dobieranego eksperymentalnie w zależności od mikrofonu. Dalej klasyfikacja na żywo nazywana jest też testem jakościowym.

5 Przyjęty model oddechu

Na początku przyjmujemy model sportowego oddychania, czyli wdech nosem i wydech ustami. Uproszczenie to polega na tym, że wydawane dźwięki są dosyć różne, dopiero potem sprawdzamy jak nasze podejście będzie się sprawować przy innych metodach oddychania.



Rysunek 4: Ilustracja przyjętego modelu, źródło

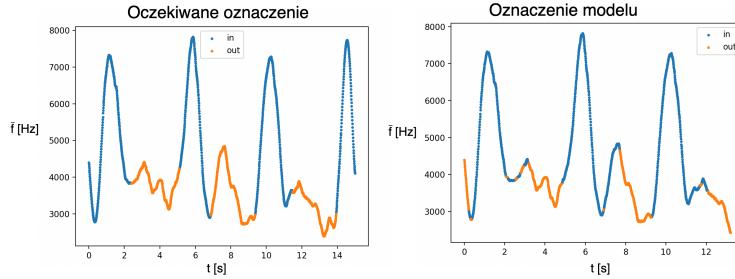
6 Średnia częstotliwość w czasie

6.1 Metoda

Pierwotnie przyjętym założeniem było, że podczas wdechu średnia częstotliwość dźwięku jest wyższa niż gdy osoba wydycha. Z pliku w formacie .wav pobieramy natężenie, które następnie **odszumiamy** za pomocą *noisereduce* (przy testach jakościowych stosujemy bramkę szumów dla 3 ostatnich sekund). **Dzielimy próbki na bloki, dla których tworzymy spektrogramy** i obliczamy średnią ważoną częstotliwość. Cechami, na podstawie których dokonywana jest predykcja są

$$\mathbf{x}(t_n) = [\bar{f}(t_0), \bar{f}(t_1), \dots, \bar{f}(t_n)]$$

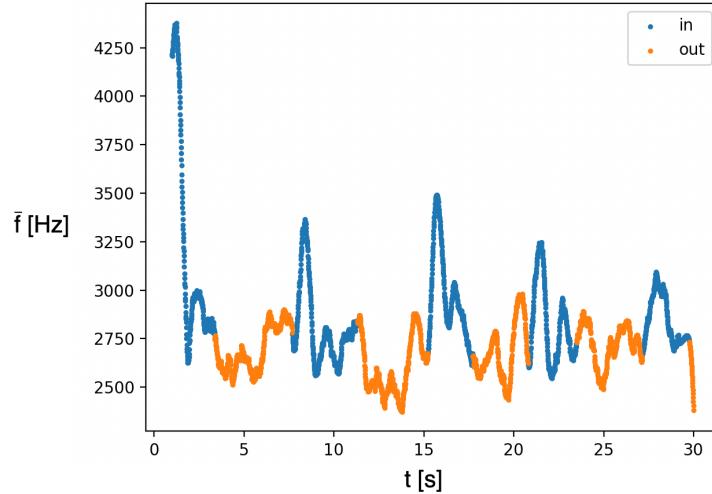
czyli średnie częstotliwości z kilku przeszłych chwil. Dla danych testowych spełniających powyższe założenie i modelu wytrenowanego SVM-em dawało nam to dokładność $\sim 60\%$.



Rysunek 5: Przykładowe działanie modelu dla danych spełniających założenie: wyższy wdech, niższy wydech. W pewnych momentach podobnie wyglądające fragmenty krzywej powinny być wdechem, w innych wydechem, mamy więc możliwy underfitting.

6.2 Problemy

Podejście to jednak mocno ograniczna nasze dalsze pole do rozwoju. Zmniejszenie całego spektrogramu do pojedynczej wartości częstotliwości daje nam mniej informacji, przez co być może ograniczylibyśmy się tylko do przyjętego przez nas modelu (tj. wdech - nos, wydech - usta) - w innych modelach różnica średnich częstotliwości między wdechem a wydechem może nie być taka wyraźna. Ponadto, dla niektórych osób zaobserwowaliśmy, że zależność między wdechem a wydechem jest niekoniecznie tak prosta jak założyliśmy. Obserwowana metoda była również bardzo niestabilna jeżeli chodzi o testy jakościowe (klasyfikację na żywo).

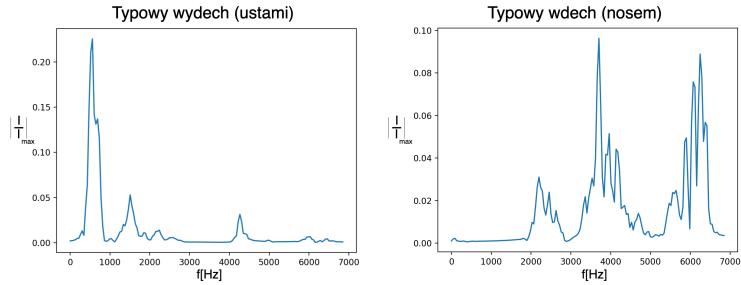


Rysunek 6: Nieoczywista zależność między wdechem a wydechem.

7 Dane wejściowe ze spektrogramu + SVM

7.1 Założenie

Kolejnym przyjętym przez nas podejściem było wzięcie całego spektrogramu (a przynajmniej jego części) jako dane wejściowe do SVM-a. Metoda pochodzi od przypuszczenia, że człowiek rozpoznaje i rozróżnia wdech/wydech na podstawie barwy dźwięku.



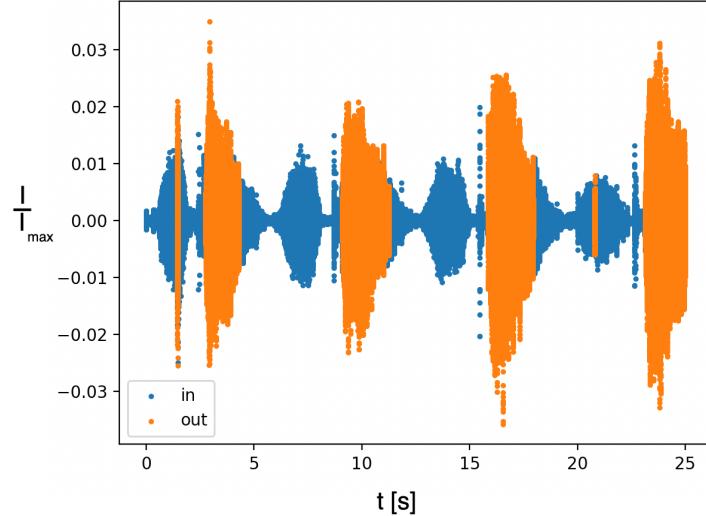
Rysunek 7: Nie są istotne dokładne kształty powyższych spektrogramów, tylko fakt, że wydech jest pojedynczym pikiem $\in (0, 1)$ kHz a wdech częstotliwościami z natężeniem o mniejszym odchyleniu $\in (2, 7)$ kHz. Zbliżoną zależność zaobserwowano dla paru osób.

7.2 Metoda

Podobnie jak ostatnio pierw odszumiamy sygnał z pliku za pomocą *noisereduce*. **Dzielimy dane na bloki rozmiaru 1024 próbek, tworzymy dla każdego bloku spektrogram** ale tym razem nie liczymy średniej tylko zostawiamy całą taką klatkę. Wektor cech ma więc postać

$$\mathbf{x} = [I_{f_1}, I_{f_2}, \dots, I_{f_n}]$$

Nie są nam potrzebne wszystkie częstotliwości zwracane przez algorytm DFT, więc stosujemy ograniczenie podobne do tego stosowanego w telefonii komórkowej - **używamy 160 częstotliwości z przedziału do 6,9 kHz**. Stosując to podejście dla danych testowych otrzymaliśmy $\sim 70\%$ dokładności. W testach jakościowych jednak zaobserwować było można "przeskakiwanie" jednej wartości na drugą i spowrotem np. w połowie brania wdechu, wydychania.



Rysunek 8: Dane z rysunku 1. oznaczone, jak widać, występują przeskoki z jednej wartości na drugą.

7.3 Poprzedni stan

W celu pozbycia się "przeskakiwania" i ogólnej poprawy działania dane z poprzedniego podpunktu rozszerzyliśmy o informację o **wcześniej przewidzianym stanie**, czyli

$$\mathbf{x} = [I_{f_1}, I_{f_2}, \dots, I_{f_{160}}, s]$$

gdzie $s = 1$ jeśli poprzednio stwierdzono wdech, $s = -1$ jeśli poprzednio stwierdzono wydech. Dodatkowo przeprowadziliśmy jeszcze **wygładzenie spektrogramu filtrem Savitzky-Golay**. Takim sposobem otrzymaliśmy dokładność $\sim 99\%$ dla danych testowych w sytuacji idealnej - jako wartość s stanu poprzedniego stosowaliśmy tę wziętą z danych testowych, czyli zakładaliśmy za każdym razem, że nasz model przewidzi dobrze poprzedni stan. Przypadek ten nie jest realny, ponieważ model nie będzie znał poprawnej poprzedniej klasyfikacji tylko zakładał, że to co sam stwierdził jest poprawne. Biorąc poprzedni stan nie z danych testowych a z poprzedniej klasyfikacji modelu otrzymujemy mniejszą dokładność $\sim 80\%$. Wniosek z tego jest taki, że model z wysokim prawdopodobieństwem przewidzi dobrze kolejny stan, jeżeli dobrze przewidział poprzedni. Warto dodać, że rzeczywiście metoda z poprzednim stanem pozwoliła nam ograniczyć "przeskakiwanie" w testach jakościowych.

7.4 Problemy

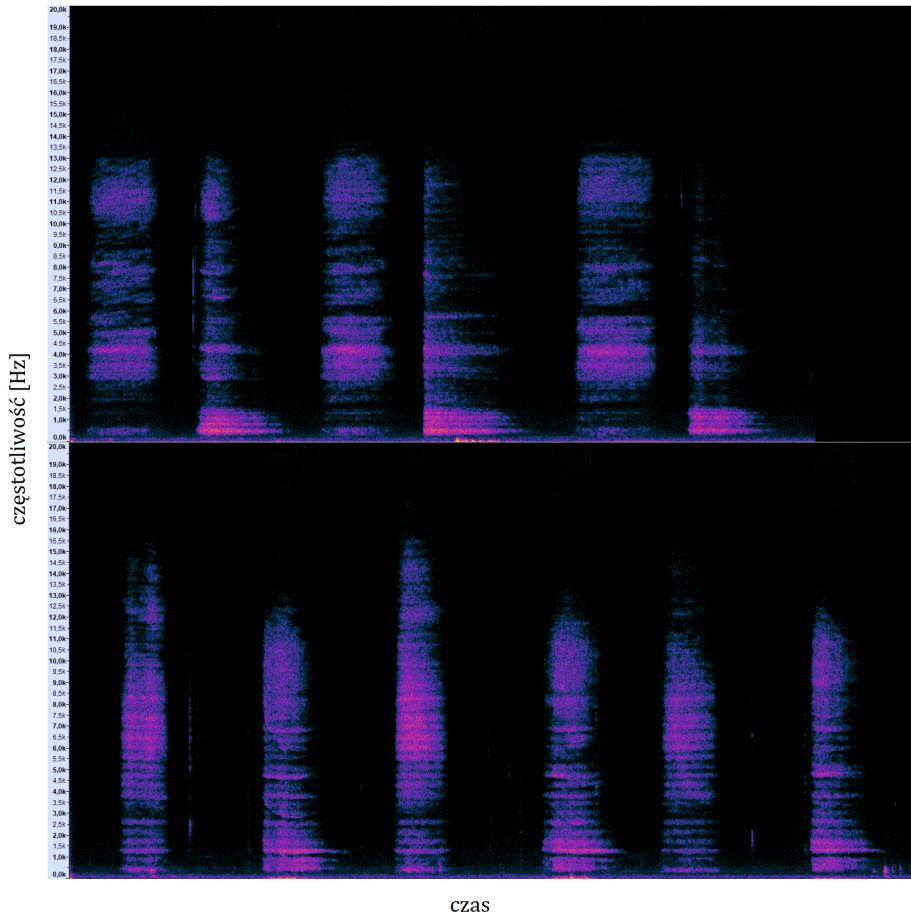
W poprzednim podpunkcie wyszedł główny mankament tej metody - konieczność założenia, że poprzednio przewidziany stan jest dobry co nie zawsze jest prawdą. W testach jakościowych zaobserwować można było "wariowanie" modelu, czyli sekwencję źle przewidzianych stanów.

7.5 Szerszy spektrogram

W celu poprawy działania modelu postanowiliśmy dokładniej przeanalizować, które częstotliwości są kluczowe dla poprawnej oceny fazy oddechu. W oparciu o obserwację spektrogramów wybraliśmy kilka możliwych zakresów, w których różnice między wdechem i wydechem są dobrze widoczne:

- 0–1 300 Hz i 3 400–22 000 Hz,
- 0–14 000 Hz,
- 0–1 300 Hz i 3 400–14 000 Hz,
- 0–8 500 Hz i 10 000–14 000 Hz,
- 0–16 000 Hz,
- 0–1 300 Hz i 3 400–16 000 Hz,
- 0–1 300 Hz, 3 400–8 500 Hz i 10 000 Hz–16 000 Hz,
- 0–8 500 Hz i 10 000–16 000 Hz.

Spektrogram dźwięku oddechu



Rysunek 9: Spektrogramy dwóch nagrani oddechu z wdechem nosem i wydechem ustami uzyskane przy użyciu programu Audacity.

Dla każdego wyboru przeprowadziliśmy testy. W przypadku ostatniego uzyskaliśmy wzajemny wzrost dokładności predykcji o $\sim 8\%$ względem początkowego podejścia. Taki wybór częstotliwości przyjęliśmy zatem dla modelu rozpoznającego wdech nosem i wydech ustami.

8 Dane wejściowe z mikrofonu + Sieć neuronowa

8.1 Założenie

Do przetestowania działania sieci neuronowej przyjęliśmy nieco odmienne podejście. Jako dane wejściowe przyjmujemy (podobnie jak w przypadku SVM) spektrogram jednak w tym przypadku stworzony przy użyciu STFT. Dzięki temu model bierze pod uwagę także zmianę widma w czasie.

8.2 Metoda

Stosując poszczególne warstwy nasz spektrogram jest odpowiednio przetwarzany tak aby uwidoczyć istotne cechy. Wykorzystywane przez nas warstwy (wraz z parametrami) to:

1. **Input** - Punkt wejścia do sieci neuronowej
2. **Resizing** - Zmiana rozmiaru obrazu (32x32)
3. **Normalization** - Normalizacja cech
4. **Conv2D** - Nałożenie odpowiedniego filtra o rozmiarze $n \times m$ (lub $n \times n$) na obraz tak aby uwydątnić jego cechy (32, 3, 'relu'). Wykorzystuje funkcję aktywacji.
5. **Conv2D** - Tak jak wyżej (64, 3, 'relu')
6. **MaxPooling2D** - Downsampling czyli wybranie maksymalnej wartości spośród okna o zadanych wymiarach (2x2)
7. **Dropout** - Ustawienie losowych wejść na 0 z częstotliwością n , przeskalowanie pozostałych cech przez $\frac{1}{1-n}$, pozwala na zredukowanie overfittingu (0.25)
8. **Flatten** - Spłaszczenie wejścia do 1 wymiaru
9. **Dense** - Łączy wszystkie neurony z warstwy poprzedniej ze wszystkimi z warstwy kolejnej, identyfikuje do jakiej klasy przynależy obiekt wejściowy. Wykorzystuje funkcję aktywacji. (128, 'relu')
10. **Dropout** - Tak jak wyżej (0.5)
11. **Dense** - Tak jak wyżej (2)

8.3 Wykorzystywane próbki

W przeciwieństwie do SVM, aktualną predykcję określamy nie na podstawie poprzedniego stanu, lecz dla stanu sprzed pewnej chwili czasowej. W naszym przypadku aktualnie wyświetlany stan pochodzi sprzed około 0,5 s. Oczywiście powoduje to pewne opóźnienie, jednak praktycznie niezauważalne przez użytkownika. Podejście to nie ulega jednak propagacji błędów, którą widzimy w przypadku SVM.

8.4 Problemy

Przy rozpoznawaniu próbki zakładamy przynależność do jednej z dwóch klas. W przypadku gdy jako dane wejściowe mamy jedynie wdechy i wydechy nie stanowi to problemu. Gdy jednak chcemy rozpoznać brak oddychania musimy założyć pewien próg pewności co otrzymanej klasyfikacji. Eksperymentalnie przyjęliśmy iż wdech jest wdechem gdy pewność sieci neuronowej wynosi 99% natomiast co do wydechów przyjęliśmy pewność na poziomie 70%.

9 Przejście na inny model oddychania - SVM

9.1 Wydech nosem

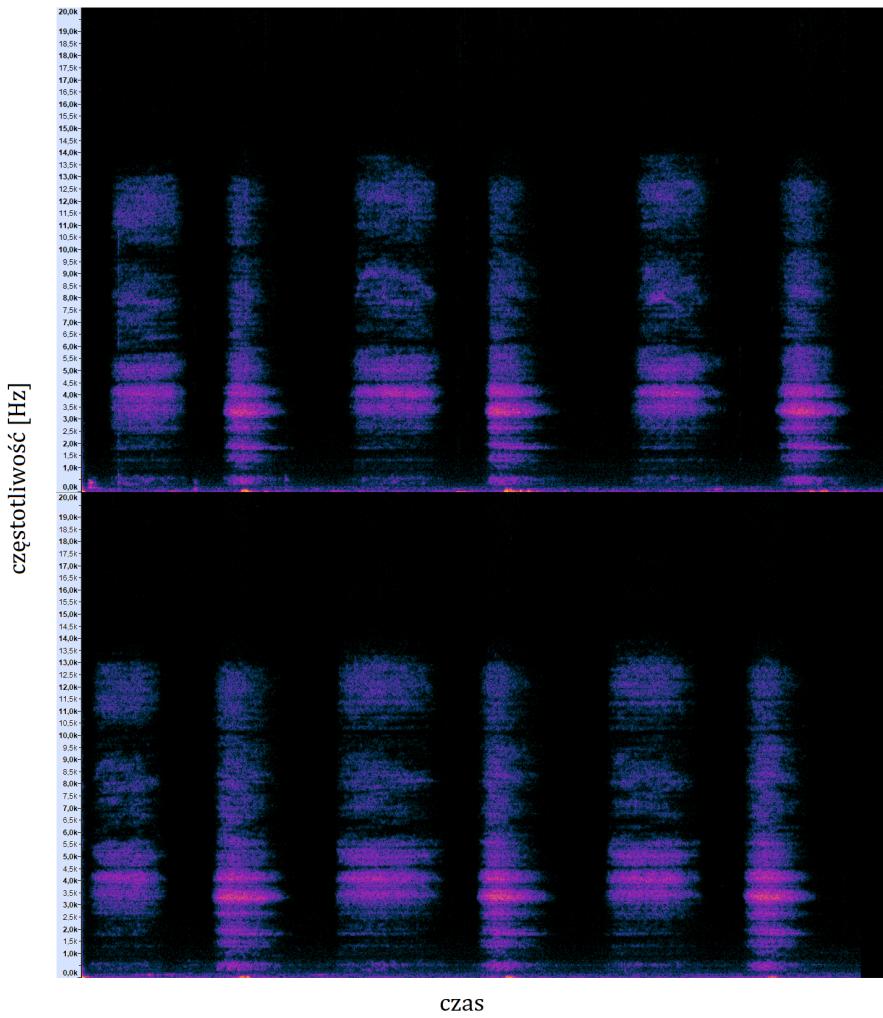
Kolejnym krokiem było przeniesienie wypracowanej metody uczenia na rozpoznawanie wdechu oraz wydechu nosem. W takim przypadku różnice między fazami oddechu są mniej zauważalne, co wymagało kolejnych udoskonaleń.

9.2 Szerszy spektrogram

Ponownie dokonaliśmy obserwacji spektrogramów, tym razem wyznaczonych dla wdechów i wydechów nosem. Zauważymy kilka zakresów, w których różnice między fazami oddechu są zauważalne:

- 0–8 500 Hz i 10 000–16 000 Hz,
- 0–3 000 Hz i 6 000–16 000 Hz,
- 0–16 000 Hz.

Spektrogram dźwięku oddechu



Rysunek 10: Spektrogramy dwóch nagrań oddechu z wdechem i wydechem nosem uzyskane przy użyciu programu Audacity.

Testy pokazały, iż ostatni z powyższych wybór częstotliwości spowodował wzgledny wzrost dokładności o kolejne $\sim 7\%$. Taki zatem pozostawiliśmy do dalszej pracy.

9.3 Usuwanie ciszy

Nietrudno było zaobserwować, iż znaczą część nagrań zajmuje cisza lub dźwięk bliski ciszy.

Rysunek 11: Wykres natężenia dźwięku w nagraniu wddechu i wydechu nosem uzyskany przy użyciu programu Audacity.

Te przerwy między wyraźnymi dźwiękami oddechów pozostawały jednak oznaczone jako wddech lub wydech. Z tego powodu model trenowaliśmy w jednym przypadku informacją, iż cisza występuje podczas wddechu, a w innym – że w trakcie wydechu. Postanowiliśmy usunąć tę niedoskonałość z przypuszczeniem, iż poprawi to wyniki działania programu.

10 Przejście na inny model oddychania - sieć neuronowa

Ze względu na dobre działanie modelu, postanowiliśmy nie implementować dodatkowych usprawnień.

11 Działanie modelu a hałas z otoczenia

11.1 SVM

11.2 Sieć neuronowa

Testy doświadczalne wykazały dobre rozpoznawanie zarówno wddechów i wydechów podczas przebywania w umiarkowanie głośnym pomieszczeniu. Największym problemem okazało się wykrywanie braku oddychania z którym nasza implementacja rozpoznawania na żywo z siecią neuronową sobie nie radzi. Proponowane rozwiązanie - zatrzymanie rozpoznawania w przypadku wykrycia braku aktywności użytkownika (na wzór chociażby Asystenta Google lub Siri).

12 Douczanie

13 Wnioski

Sieć neuronowa lepiej radzi sobie z omawianym w sprawozdaniu zadaniem. Wykrywa to przede wszystkim z faktu skorzystania z gotowego, sprawdzonego już rozwiązania.

14 Źródła

[1] Cormen Thomas H., Leiserson Charles E., Rivest Roland L., Stein Clifford

(1989) *Wprowadzenie do algorytmów*,

[2] https://pl.wikipedia.org/wiki/Dyskretna_transformata_Fouriera

- [3] <https://esezam.okno.edu.pl/mod/book/view.php?id=9&chapterid=77>
- [4] Kumar, E. and Surya, K. and Varma, K. and Akash, A. and Kurapati, Nithish Reddy (2023) *Noise Reduction in Audio File Using Spectral Gattting and FFT by Python Modules*
- [5] <https://www.maplesoft.com/Applications/Detail.aspx?id=154593>
- [6] https://www.deltami.edu.pl/temat/informatyka/sztuczna_inteligencja/2017/12/28/Glebokie_uczenie_maszyn/
- [7] <http://sciagaprogramisty.blogspot.com/2018/03/fully-connected-layer-fc-warstwa-w-peni.html>
- [8] <https://www.tutorialspoint.com/keras>