Rozpoznawanie Mówcy za Pomocą Sieci Neuronowych

Marcin Chwedczuk Albert Skłodowski

5 Styczeń 2012

Spis treści

1 Wstęp

Celem projektu było stworzenie systemu identyfikacji osób na podstawie próbek ich głosu, bazującego na sieciach neuronowych. Zagadnienie leży w obszarze zainteresowań bardzo wielu naukowców na całym świecie, o czym świadczą liczne poświęcone mu artykuły i książki naukowe. Bezpośrednią przyczyną takiego stanu rzeczy są bez wątpienia bardzo liczne możliwe sposoby wykorzystania takiej technologii. Mogłaby ona posłużyć m.in. w celu kontroli dostępu do różnego rodzaju usług, np. obsługi kont bankowych przez telefon, dostępu do poufnych danych, sterowania urządzeń za pomocą głosu jedynie przez osoby do tego uprawnione itp.

Można wyróżnić co najmniej dwa podejścia do rozpoznawania głosu. Pierwszy z nich polega na rozpoznawaniu słów pochodzących z ograniczonego ich zbioru, drugi natomiast dotyczy rozpoznawania wypowiedzi dowolnej, czyli swobodnej, z nieograniczonego zbioru słów. Początkowo zakładaliśmy, że zajmiemy się tym pierwszym, prostszym przypadkiem. Wynikało to z obserwacji, że w artykułach naukowych, do których udało nam się dotrzeć (patrz: Bibliografia), dla tego prostszego przypadku uzyskiwano znacznie (można nawet rzec: nieporównywalnie) lepsze wyniki. Ostatecznie jednak zdecydowaliśmy się skierować

naszą uwagę w kierunku rozpoznawania osób po ich dowolnej wypowiedzi. Wynikało to nie tylko z tego, że zagadnienie to wydaje się znacznie ciekawsze, ale także z prośby prowadzącego zajęcia.

W kolejnych punktach niniejszego dokumentu prezentujemy dokładny opis zastosowanej przez nas metody, a także szczegółowe zestawienie otrzymanych za jej pomocą wyników.

2 Dane uczące i dane testowe

Danymi wejściowymi stworzonej przez nas aplikacji są pliki w formacie WAVE. Aby uniknąć problemów ze zniekształceniami sygnału mowy, aplikacja wymaga, by częstotliwość próbkowania wynosiła 44 100Hz przy 16 bitach przeznaczonych na próbkę. Sygnał musi być zapisany w wersji MONO.

Nagrań wypowiedzi, które posłużyły za dane uczące oraz dane testowe podczas przeprowadzonych przez nas prób, dokonaliśmy za pomocą darmowego programu Audacity w wersji 1.2.6, przy wykorzystaniu mikrofonów komputerowych. Próbki po nagraniu nie były przetwarzane w żaden dodatkowy sposób. Zdecydowaną większość próbek zebraliśmy podczas zajęć laboratoryjnych. Pomimo tego, że podczas nagrywania w sali laboratoryjnej często panował dość duży hałas, nie wpłynął on negatywnie na jakość naszych nagrań, ponieważ zastosowane mikrofony były słabo czułe na taki zewnętrzny szum.

Każdy z plików, którego używaliśmy za dane uczące lub dane testowe, zawierał nagraną krótką wypowiedź pojedynczej osoby - każdą z nich prosiliśmy o wypowiedzenie frazy: „tlen, kasza, żyzny, mini, ćma, ultra, krew, house, felicja, komputer". Każdą osobę nagrywaliśmy co najmniej sześć razy. W kilku przypadkach zebrane za pierwszym razem dane były zbyt słabej jakości, co zmusiło nas do ponownego nagrywania tych samych osób.

Tak uzyskane dane zostały przez nas losowo podzielone na dane służące do uczenia sieci oraz na dane służące do ich testowania. Danych uczących było więcej niż danych testowych. Z sześciu nagranych plików dla każdej osoby, cztery służyły do uczenia, a dwa do testowania sieci neuronowych.

Początkowo zakładaliśmy, że - zgodnie z wymaganiami postawionymi przez prowadzącego zajęcia laboratoryjne - sieć zostanie nauczona ok. czterdziestu osób (w tym co najmniej dwudziestu studentów naszego wydziału). Jako że w początkowej fazie testów nie potrzebowaliśmy aż tak wielu danych, zebraliśmy wtedy próbki ok. 15 osób, w większości wchodzących w skład naszej grupy laboratoryjnej. Podczas prac nad aplikacją okazało się, że zastosowana przez nas metoda nie radzi sobie ze zbyt dużym zbiorem danych testowych i dane zebrane dla tych 15 osób były w zupełności wystarczające. Dlatego zaniechaliśmy nagrywania kolejnych osób.

Stworzona przez nas aplikacja umożliwia testowanie nauczonych sieci neuronowych nie tylko na podstawie dostarczonych z góry danych testowych, ale także za pomocą próbek głosu nagrywanych na żywo podczas jej działania. Ta część aplikacji była bardzo trudna do testowania, co wynikało oczywiście z tego, że nie mieliśmy w ciągłej dyspozycji wszystkich osób, których uczyliśmy nasze sieci neuronowe.

3 Opis Zastosowanych Algorytmów

3.1 Opis Wykorzystywanej Sieci Neuronowej

W stworzonej przez nas aplikacji przyjęliśmy następującą architekturę systemu: Dla każdej osoby(oznaczmy ją P) która ma być rozpoznawana przez aplikację zostanie stworzona osobna sieć neuronowa net_P. Sieć ta będzie uczona odpowiedzi 1 dla próbek głosu osoby P oraz odpowiedzi 0 dla próbek głosu pozostałych osób. Wszystkie sieci będą uczone i testowane w ten sam sposób.

Każda z sieci net_P będzie siecią typu perceptron wielowarstwowy oraz będzie posiadać dwie warstwy ukryte. O wielkości warstw ukrytych będą decydować parametry D_1 oraz D_2 w następujący sposób:

$$size(hidden_1) = \frac{size(input)}{D_1}$$

$$size(hidden_2) = \frac{size(input)}{D_2}$$

gdzie $size(hidden_1)$, $size(hidden_2)$ to odpowiednio rozmiary I i II warstwy ukrytej, a $size(input)$ to rozmiar wejścia. Zazwyczaj będziemy przyjmować $1 \leqslant D_1 < D_2$. Każda z sieci będzie posiadała dokładnie jedno wyjście zwracające wartości z przedziału $[0, 1]$.

W opisanych wyżej sieciach będziemy stosować neurony o sigmoidalnej funkcji aktywacji ($f(x) = \frac{1}{1+\exp(-x)}$), dodatkowo będziemy stosować wzmocnienie (bias).

Każda z sieci będzie uczona za pomocą standardowego algorytmu wstecznej propagacji błędów (backpropagation). W algorytmie backpropagation będziemy stosować stopniowy rozpad współczynnika nauki oraz momentu, zgodnie z wzorami:

$$\eta_i = \frac{\eta}{\sqrt{(1+i)}}$$

$$\mu_i = \frac{\mu}{\sqrt{(1+i)}}$$

gdzie η oznacza współczynnik nauki, μ współczynnik momentu, a i to oczywiście numer iteracji. Zazwyczaj będziemy przyjmować $\mu = \frac{\eta}{2}$.

W dalszej części dokumentu gdy nie będzie powiedziane inaczej będziemy przyjmować że $D_1 = 4$, $D_2 = 8$, $\eta = 0.21$ a $\mu = 0.1$.

Sieci neuronowe w naszej aplikacji zostały zaimplementowane za pomocą darmowej do zastosowań niekomercyjnych biblioteki Encog w wersji 3.0.1.

Algorytm nauki pojedynczej sieci neuronowej

Na listingu 1 przedstawiono algorytm nauki pojedynczej sieci neuronowej przeznaczonej do rozpoznawania osoby *person*. W algorytmie występują następujące stałe, będące parametrami nauki:

MAX-ITERATIONS Maksymalna liczba iteracji przeznaczona na naukę pojedynczej sieci, w programie ten parametr przyjmuje wartość 160 tysięcy iteracji.

Algorytm 1 Algorytm nauki pojedynczej sieci neuronowej przeznaczonej dla osoby *person*

```
1: input person
2:
3: network ← InitNetwork()
4: for 0 < i < MAX-ITERATIONS do
5:    expectedAnswer ← 1
6:    trainingData ← DrawNetworkInput(person)
7:    if rnd(0, 1) < TCOEF then
8:       other ← SelectOtherPerson(person);
9:       expectedAnswer ← 0
10:      trainingData ← DrawNetworkInput(other)
11:   end if
12:
13:   η_i ← η / √(i+1)
14:   μ_i ← μ / √(i+1)
15:
16:   backpropagation(network, η_i, μ_i, expectedAnswer, trainingData)
17:
18:   {Every 1000 iterations check if network error}
19:   {is acceptable}
20:   error ← GetNetworkError(network, GetPersonTestSet(person))
21:   if error < MAX-ERROR then
22:      break
23:   end if
24: end for
25:
26: return network
```

TCOEF Współczynnik znajomości, niestety testy wykazały że wartość tego parametru powinna zawsze wynosić 0.5 (przy założeniu że **rnd**$(0, 1)$ zwraca liczbę losową z przedziału $(0, 1)$).

MAX-ERROR Maksymalny dopuszczalny błąd na zbiorze walidacyjnym, jeżeli sieć osiąga mniejszy błąd niż MAX-ERROR to dalsza nauka jest przerywana

Zauważmy że algorytm nauki sieci jest wysoce niedeterministyczny: zarówno osoby jak i wybór próbki uczącej są dokonywane w sposób losowy.

Algorytm obliczania błędu sieci na zbiorze walidacyjnym

Przedstawiony na listingu 2 sposób obliczania błędu sieci neuronowej opiera się na standardowym błędzie średnio kwadratowym, z tym że odpowiedzi sieci które znajdują się zbyt blisko wartości 0.5 nie są uwzględnia przy liczeniu błędu. Co ma sens ponieważ nie są one uwzględniane również przy typowaniu najbardziej prawdopodobnego mówcy.

W wielu miejscach będziemy prezentować błąd aplikacji(czyli całego zestawu sieci) na zbiorze uczącym lub walidacyjnym, błąd ten będzie obliczany jako średnia arytmetyczna z błędów poszczególnych sieci na zadanym zbiorze.

Algorytm 2 Algorytm obliczający błąd sieci neuronowej na zadanym zbiorze testowym

```
1: input testedPerson
2:
3: error ← 0
4: count ← ϵ { Epsilon is a small number e.g. 0.0001}
5: for all person ∈ People do
6:   answer ← 0
7:   if person = testedPerson then
8:     answer ← 1
9:   end if
10:
11:  {testSet - Array containing mfcc coefficients for consecutive voice frames}
12:  testSet ← GetPersonTestSet(person)
13:  for 0 ⩽ start < length(testSet)− FRAMES-COUNT do
14:    netInput ← testSet[start..(start + FRAMES-COUNT−1)]
15:    netAnswer = network_person(netInput)
16:    if netAnswer > T or netAnswer < (1 - T) then
17:      error ← error + (netAnswer - answer)^2
18:      count ← count + 1
19:    end if
20:  end for
21: end for
22:
23: return error / count
```

3.2 Opis Algorytmu Ekstrakcji Cech

Opis Danych Wejściowych Algorytmu

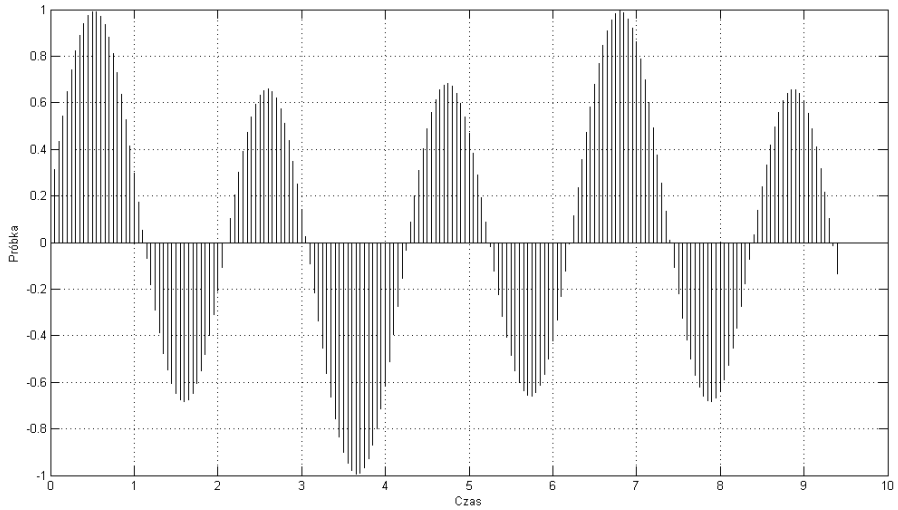
Algorytm ekstrakcji cech operuje na pojedynczych próbkach dźwięku. Każda próbka niezależnie do długości jest reprezentowana jako tablica liczb zmiennoprzecinkowych o wartościach z przedziału $[-1, 1]$. Algorytm zakłada ponadto że próbki zostały otrzymane przez kwantyzacje sygnału ciągłego ze stałą częstotliwością próbkowania równą F. Konwersji plików WAVE na wyżej opisany format dokonaliśmy za pomocą darmowej biblioteki NAudio.

Normowanie Sygnału Wejściowego

Okazuje się że w zależność od mówcy (a także jego nastroju, czy zmęczenia), oddalenia od mikrofonu oraz innych czynników można otrzymać dość znaczne różnice w amplitudzie nagrywanych sygnałów dźwiękowych. Aby wyeliminować wpływ amplitudy sygnału na jakość klasyfikacji wszystkie przetwarzane sygnały przed ekstrakcją cech są poddawane normowaniu. Algorytm normujący sygnał wejściowy został przedstawiony na listingu 3.

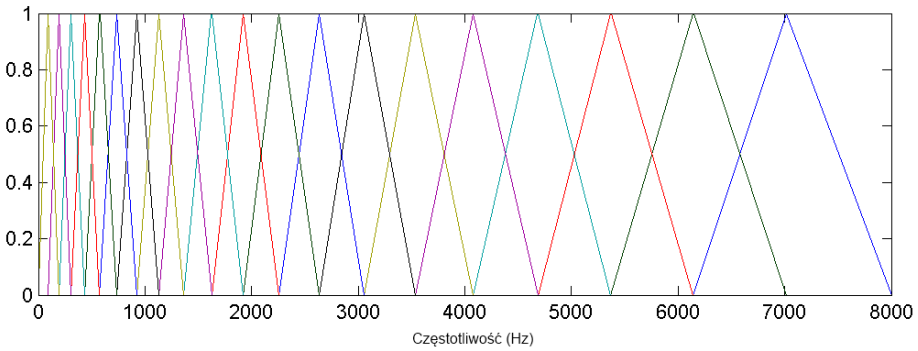
Algorytm Ekstrakcji Cech

W tej części dokumentu przedstawimy algorytmy służące do wyodrębniania charakterystycznych cech mowy, omówimy również towarzyszące im parametry.



Rysunek 1: Reprezentacja sygnału ciągłego za pomocą tablicy liczb zmiennoprzecinkowych. Kolejne elementy tablicy zawierają wartości kolejnych próbek sygnału, i tak element o indeksie 0 zawiera wartość 0.0, element o indeksie 1 wartość 0.31, element o indeksie 2 wartość 0.43 itd.

Tabela 1 zawiera opis parametrów algorytmów, listingi 4, 5 oraz 6 prezentują pseudokody wykorzystywanych algorytmów.



Rysunek 2: Położenie i przepustowość banku filtrów trójkątnych rozłożonych zgodnie ze skalą MEL na przedziale częstotliwości 0Hz - 8kHz

Algorytm 3 Algorytm wykorzystywany do unormowania wartości sygnału wejściowego

1: $\{samples_i$ - Array containing voice samples$\}$
2: empth $\leftarrow 1.0$
3: max $\leftarrow max_i(samples_i)$
4: min $\leftarrow min_i(samples_i)$
5:
6: **if** $|min| > \epsilon$ **then**
7: emph $\leftarrow \max(emph, \frac{1}{|min|})$
8: **end if**
9: **if** $|max| > \epsilon$ **then**
10: emph $\leftarrow \min(emph, \frac{1}{|max|})$;
11: **end if**
12:
13: **for all** i **do**
14: $samples_i \leftarrow emph \cdot samples_i$
15: **end for**

Parametr	Optymalna wartość	Opis parametru
WINDOW-SIZE	512 - 1024	Określa ile próbek tworzy pojedynczą ramkę. Wielkość okna powinna być tak dobrana żeby czas trwania ramki wynosił od 10 do 30 milisekund. Wartość tego parametru powinna być potęgą liczby 2
WINDOW-OVERLAP	256 - 512	Określa ile próbek współdzielą ze sobą sąsiadujące ramki
SPEAK-POWER-THRESHOLD	0.02 - 0.06	Parametr określa minimalną energie jaką musi posiadać sygnał dźwiękowy żeby został uznany za część wypowiedzi
MFCC-COUNT	6 - 16	Ilość używanych współczynników MFCC
$\lvert \triangle \rvert$	24 - 27	Ilość używanych filtrów trójkątnych. Filtry pokrywają pasmo częstotliwości od 0 Hz do 8 kHz zgodnie ze skalą MEL. Aby utworzyć zbiór filtrów wymagana jest znajomość częstotliwości próbkowania F, oraz rozmiar ramki WINDOW-SIZE. Rysunek 2 przedstawia przepustowość oraz rozmieszczenie przykładowego zestawu filtrów
$\triangle_i^n$		Przepustowość filtra o numerze n dla częstotliwości odpowiadającej i-temu wyjściu transformacji Furiera

Tabela 1: Opis parametrów algorytmu ekstrakcji cech

Algorytm 4 Algorytm wykorzystywany do ekstrakcji cech z sygnału wejściowego

```
1: input samples : array [0..(N − 1)] of double
2:
3: {Type mfcc is used to represent array of MFCC coefficients}
4: type mfcc = array [0..(MFCC-COUNT−1)] of double
5: features : list of mfcc
6: start ← 0
7: delta ← WINDOW-SIZE − WINDOW-OVERLAP
8:
9: while start < N - WINDOW-SIZE do
10:    frame ← ExtractFrame(samples, start)
11:    power ← ∑_i |frame_i|/ WINDOW-SIZE
12:    if power > SPEAK-POWER-THRESHOLD then
13:       mfcc ← GetMFCC(frame)
14:       append mfcc to features
15:    end if
16:    start ← start + delta
17: end while
18:
19: return features
```

Algorytm 5 ExtractFrame - Procedura pomocnicza

```
1: input samples : array [0..(N − 1)] of double
2: input start : integer
3:
4: frame : array [0..(WINDOW-SIZE−1)] of double
5: for 0 ⩽ i < WINDOW-SIZE do
6:    frame_i = samples_{start+i}
7: end for
8:
9: return frame
```

Algorytm 6 GetMFCC - Algorytm obliczania współczynników MFCC dla danej ramki

input frame : **array** $[0..(N-1)]$ **of double**

for $0 \leqslant i < N$ **do**
  $hamming_i \leftarrow 0.54 - 0.46\cos(\frac{2\pi i}{N-1})$
  $frame_i \leftarrow hamming_i \cdot frame_i$
end for

$furier \leftarrow$ **FFT** $(frame)$
for $0 \leqslant i < N$ **do**
  $furier_i \leftarrow |furier_i|$
end for

for $0 \leqslant i < |\triangle|$ **do**
  $sum_i \leftarrow 0$
  for $0 \leqslant j < N$ **do**
    $sum_i \leftarrow sum_i + \triangle_j^i \cdot furier_j$
  end for
  $sum_i \leftarrow \log_{10}(sum_i)$
end for

{Return only first MFCC-COUNT elements of array returned by DCT}
return **DCT**(sum)

3.3 Opis Algorytmu Rozpoznawania Mówcy

Listing 7 zawiera pseudokod algorytmu wykorzystywanego do wytypowania najbardziej prawdopodobnego mówcy, tabela 2 zawiera opis parametrów występujących w algorytmie.

Parametr	Optymalna wartość	Opis parametru
T	unknown	Próg wiarygodności sieci neuronowej. Odpowiedz sieci będzie uznana za znaczącą wtedy gdy będzie należeć do sumy przedziałów $[0, 1-T]$ oraz $[T, 1]$
Features		Tablica współczynników MFCC dla ramek rozpoznawanego sygnału mowy
People		Zbiór osób rozpoznawanych przez klasyfikator
$network_{person}$		Sieć neuronowa nauczona rozpoznawania osoby $person$ oraz odrzucania pozostałych osób
FRAMES-COUNT	6 - 16	Parametr określający ile kolejnych ramek (a raczej odpowiadających im tablic współczynników MFCC) jest podawanych na wejście sieci neuronowej. Wielkość wejścia sieci można policzyć mnożąc FRAMES-COUNT przez MFCC-COUNT

Tabela 2: Opis parametrów algorytmu wyboru zwycięzcy

Algorytm 7 RecogniseSpeaker - Algorytm rozpoznawania mówcy

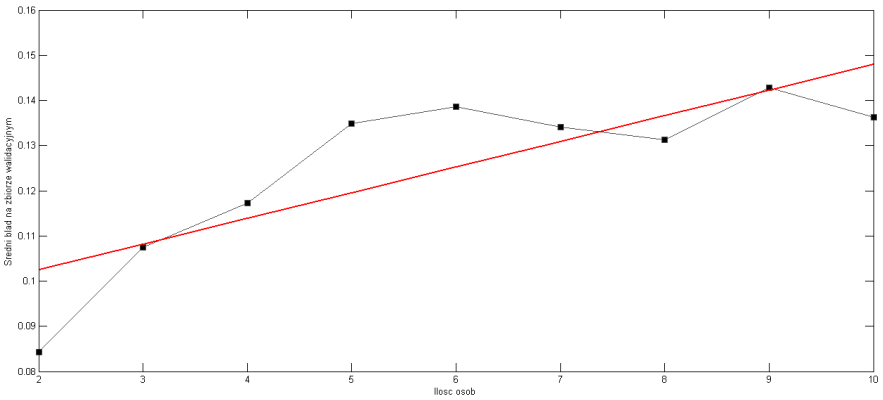
```
1: input Features : array [0..(N − 1)] of mfcc
2:
3: total ← 0
4: for all person ∈ People do
5:    result_person ← 0
6: end for
7:
8: for all 0 ⩽ start < N - FRAMES-COUNT do
9:    {Skopiuj elementy tablicy features do tablicy netinput}
10:   {Kopiujemy również oba indeksy graniczne}
11:   netinput ← Features[start..(start + FRAMES-COUNT−1)]
12:
13:   for all person ∈ People do
14:      answer = network_person(netinput)
15:      if answer > T or answer < (1 - T) then
16:         if answer > T then
17:            result_person ← result_person + answer
18:            total ← total + 1
19:         else
20:            result_person ← result_person + (answer - 1)
21:         end if
22:         {Notice that total is incremented only in the first branch}
23:      end if
24:   end for
25: end for
26:
27: for all person ∈ People do
28:   result_person ← max(0, result_person/total)
29: end for
30: sort descend person by result_person
```

4 Omówienie Uzyskanych Wyników

4.1 Metodologia testowania

4.2 Wpływ Parametrów na Rezultaty

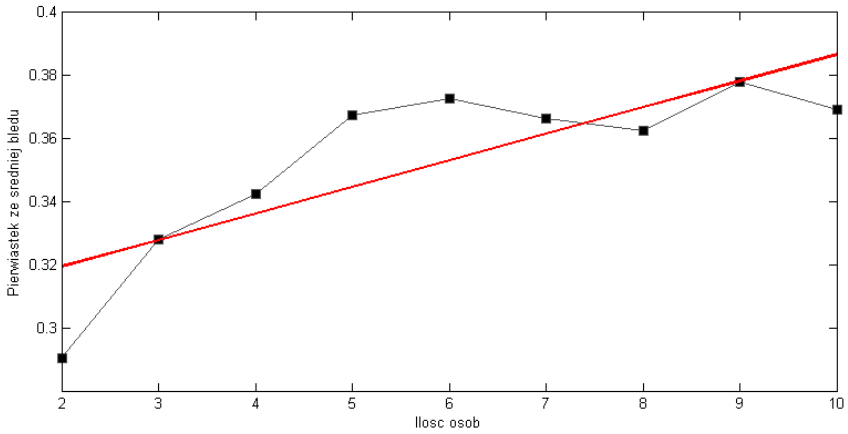
Zależność błędu od ilości uczonych osób



Rysunek 3: Średni błąd sieci na zbiorze walidacyjnym w zależności od liczby uczonych osób. Na czerwono wykreślono linie regresji. Zauważmy że w przyjętej metodologii liczenia błędu błąd rzędu 0.16 powoduje że odpowiedzi sieci różnią się od oczekiwanych o około $\sqrt{0.16} = 0.4$ co praktycznie uniemożliwia działanie aplikacji.

4.3 Uzyskane Wyniki

5 Podsumowanie



Rysunek 4: Pierwiastek błędu sieci na zbiorze walidacyjnym w zależności od liczby uczonych osób. Na czerwono wykreślono linie regresji. Błąd rzędu 0.35 przy THRESHOLD równym 0.65 jest akceptowalny.