

DEKODOWANIE KODÓW ITEROWANYCH Z UŻYCIEM SIECI NEURONOWEJ
DECODING OF ITERATED CODES WITH THE USE OF A NEURAL NETWORK

Oskar Fiedot¹; Marek Blok²

¹ Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, Gdańsk, oskar.fiedot@gmail.com

² Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, Gdańsk, marek.blok@pg.edu.pl

Streszczenie: Nadmiarowe kody iterowane są jedną z prostych metod pozyskiwania długich kodów korekcyjnych zapewniających dużą ochronę przed błędami. Jednocześnie, chociaż ich podstawowy iteracyjny dekodery jest prosty koncepcyjnie oraz łatwy w implementacji, to nie jest on rozwiązaniem optymalnym. Poszukując alternatywnych rozwiązań zaproponowano, przedstawioną w pracy, strukturę dekodera tego typu kodów wspomaganą przez sieci neuronowe. Zaproponowane rozwiązanie pozwala na wykrywanie oraz korekcję błędów w odbieranych ciągach.

Abstract: Redundant iterated codes are one of the simple methods of deriving long correction codes that provide high error protection. At the same time, although their basic iterative decoder is conceptually simple and easy to implement, it is not an optimal solution. Looking for alternative solutions, a neural network-assisted decoder structure for this type of codes was proposed. The solution presented in this paper allows the detection and correction of errors in the received sequences.

Słowa kluczowe: dekodowanie kodów iterowanych, detekcja i korekcja błędów, sieć neuronowa, syndrom

Keywords: iterated code decoding, error detection and correction, neural network, syndrome

1. WSTĘP

W dzisiejszym świecie, w związku z coraz to nowszymi usługami sieciowymi, takimi jak na przykład strumieniowanie gier komputerowych lub aplikacji w czasie rzeczywistym, coraz szybciej rośnie ilość danych przesyłanych pomiędzy różnymi urządzeniami oraz wymagania na szybkość transmisji, jak i odporność na błędy powstające w trakcie transmisji. Wiele usług jest bardzo wrażliwych na tego typu błędy, które mogą w znacznym stopniu wpływać negatywnie na wygodę korzystania z tych usług [2]. Wymaga to projektowania takich systemów, które w jak najlepszym stopniu są w stanie wykrywać oraz poprawiać błędy powstające na różnych etapach przesyłania informacji z jednego urządzenia, na drugie.

Jednym ze sposobów na skuteczne wykrywanie oraz korekcję błędów jest korzystanie z liniowych kodów blokowych [2], które umożliwiają korekcję pewnej liczby błędów. Uzyskuje się to poprzez dodanie do przesyłanego ciągu bitów dodatkowych bitów kontroli parzystości. Wraz z długością kodu rosną jego potencjalne zdolności korekcyjne, jednak zwykle wiąże się to ze znacznym wzrostem złożoności jego dekodowania. Alternatywą jest

złożenie jednego dłuższego kodu z kilku kodów składowych. Jednym z możliwych podejść do tego, jest stosowanie kodów iterowanych [2]. Ograniczeniem kodów iterowanych jest jednak to, że klasyczny, iteracyjny, dekodery tego typu kodów, mimo że bardzo prosty w implementacji, często nie jest w stanie poprawić wszystkich błędów, których poprawę gwarantują wysokie zdolności korekcyjne kodu iterowanego.

Coraz większą popularnością na świecie cieszą się również sieci neuronowe, będące implementacją sztucznej inteligencji, które próbuje się wykorzystywać wszędzie tam, gdzie mogą one wpłynąć pozytywnie na skuteczność projektowanych systemów. Znajdują one również zastosowanie w systemach informacyjnych, a przykładowe próby wytrenowania sieci neuronowych do dekodowania liniowych kodów blokowych zostały opisane w pracach [3] oraz [4].

W pracy przedstawiono strukturę dekodera kodów iterowanych, wykorzystującego możliwości sieci neuronowych do poprawy skuteczności dekodera w zakresie korekcji błędów oraz wyniki testów skuteczności tego rozwiązania.

2. TRENING I TESTY SKUTECZNOŚCI ZAPROPONOWANEJ SIECI

2.1. Struktura kodu iterowanego

Kod iterowany, dla którego poszukiwana była optymalna struktura sieci jest kodem podwójnie iterowanym, składającym się z kodów składowych zdolnych poprawić pojedynczy błąd. Do kodowania wierszy wybrano kod Hamminga o następującej macierzy generującej:

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (1)$$

natomiast do kodowania kolumn, został wybrany kod o macierzy generującej:

$$G_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad (2)$$

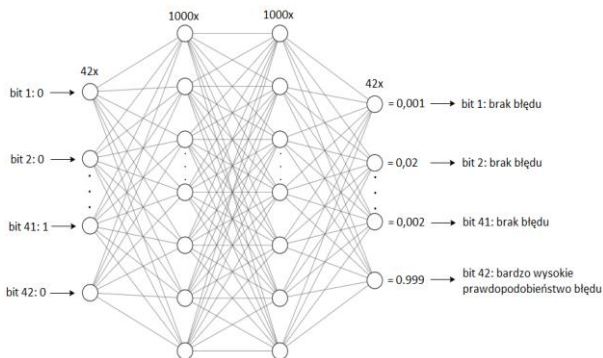
Kody te cechują kolejno 4 bity informacyjne i długość 7 oraz 3 bity informacyjne i długość 6, co oznacza, że wynikowy kod cechuje 12 bitów informacyjnych oraz łączna długość 42 bity, co daje 30 dodatkowych bitów kontroli parzystości. Zawartość informacyjna złożonego

kodu iterowanego jest równa 0,285, co oznacza, że zaledwie 28,5% wszystkich bitów w kodzie przenosi informację bezpośrednio. Zdolności korekcyjne kodu gwarantują poprawę 4 błędnych bitów w ciągu odebranym.

2.2. Struktura sieci neuronowej

Zaproponowany dekodery opiera się o współpracę dwóch sieci neuronowych. To podejście jest wynikiem niewystarczająco dobrych rezultatów osiągniętych przez dekodery oparte na działaniu pojedynczej sieci neuronowej. Dekodowanie początkowo było przeprowadzane tylko przez jedną z dwóch obecnie wykorzystywanych sieci, której zadaniem jest określanie czy poszczególne bity uległy przekłamaniu. Taka sieć posiada liczbę wejść oraz wyjść równą liczbie wszystkich bitów w kodzie iterowanym. W przypadku omawianego kodu iterowanego (rys. 1) oznacza to 42 neurony wejściowe oraz 42 neurony wyjściowe w sieci neuronowej. Sieć posiada również dwie warstwy ukryte, w której znajduje się po 1000 neuronów. Taka liczba neuronów sprawia, że sieć połączeń pomiędzy nimi jest bardzo skomplikowana, umożliwiając sieci neuronowej przetwarzanie oraz poznawanie złożonych relacji pomiędzy poszczególnymi bitami w ciągach kodowych oraz pozwala na szybkie osiągnięcie stosunkowo dobrych wyników predykcji sieci.

Każdy neuron na wyjściu oraz wejściu sieci odpowiada pojedynczemu bitowi w ciągu odebranym. Sieć na wejście przyjmuje ciąg zer i jedynek, będący ciągiem odebranym z wprowadzonymi błędami, natomiast zwraca na wyjściu ciąg wartości z zakresu od 0 do 1, reprezentujących rozmytą decyzję, czy poszczególne bity uległy przekłamaniu.



Rys. 1. Diagram sieci neuronowej oszacowującej czy poszczególne bity uległy przekłamaniu

Neurony w warstwach ukrytych mają funkcje aktywacji ReLU, która zeruje ujemne wartości, a wartości dodatnie przekazuje niezmiennie. Warstwa wyjściowa wykorzystuje natomiast funkcję aktywacji Sigmoid. Funkcja ta zwraca zawsze wartości z zakresu od 0 do 1, przez co nadaje się do reprezentowania oszacowania binarnej decyzji i odpowiednio skaluje wartości na wyjściu sieci neuronowej. Jako optymalizator do treningu został wybrany algorytm Adama [1], będący metodą stochastycznego spadku gradientu. Jest to metoda wydajna obliczeniowo, o niewielkich wymaganiach pamięciowych i dobrze nadająca się do problemów, które są skomplikowane pod względem ilości danych oraz parametrów [1].

Trening opisywanej sieci poskutkowało bardzo ciekawymi wynikami. Skuteczność sieci dla dekodowania

ciągów, w których wystąpiły różne liczby błędów, została przedstawiona na rys. 2. Jak widać, sieć jest w stanie osiągnąć 100% skuteczności dla dekodowania ciągów odebranych, w których wystąpił jeden błąd, 52% skuteczności dla dekodowania ciągów z dwoma błędami, około 40% dla ciągów z trzema błędami oraz około 30% dla ciągów z czterema błędami. Spadająca skuteczność dla coraz większych liczb błędów nie jest zaskakująca, natomiast zaskakujące jest jak niską skuteczność osiąga sieć neuronowa podczas dekodowania ciągów, w których nie wystąpił żaden błąd. Skuteczność sieci w takim przypadku to niecałe 3%. Co ciekawe, sieć o wyżej opisanej strukturze, osiąga niemal 100% skuteczności dla dekodowania ciągów, w których wystąpił jeden błąd, nawet gdy trening został przeprowadzony z wyłączeniem ciągów, zawierających tylko pojedynczy błąd. Skuteczność sieci dla dekodowania bezbłędnych ciągów, nie wzrosła nawet wtedy, gdy znacznie zwiększono ilość danych treningowych, będących bezbłędными ciągami.

Do opracowania zaproponowanego rozwiązania kluczowe okazało się spostrzeżenie, że mimo zaledwie 33% skuteczności dla dekodowania ciągów z czterema błędami, prawie zawsze sieci udaje się poprawić przynajmniej jeden z czterech błędów (ponieważ skuteczność sieci dla poprawiania pojedynczych błędów to 100% dla danych testowych). Taka sytuacja została przedstawiona na rys. 3, gdzie sieci neuronowej nie udało się poprawić wszystkich czterech błędów w ciągu odebranym, jednak prawidłowo poprawiła dwa błędne bity z czterech (błędne bity: 12, 15, 26, 40; poprawione bity: 15, 40). Podobne sytuacje, w których sieć poprawia słusznie pewną część błędnych bitów, zachodzą również dla trzech oraz dwóch błędów w ciągu odebranym.

```
Testing the neural network on a set of messages containing 0 errors....
313/313 [=====] - 1s 2ms/step - loss: 4.7084e-06 - accuracy: 0.0266
The accuracy achieved by the neural network: 2.66 %

Testing the neural network on a set of messages containing 1 errors....
313/313 [=====] - 1s 2ms/step - loss: 1.0711e-05 - accuracy: 1.0000
The accuracy achieved by the neural network: 100.0 %

Testing the neural network on a set of messages containing 2 errors....
313/313 [=====] - 1s 2ms/step - loss: 1.2015e-04 - accuracy: 0.5238
The accuracy achieved by the neural network: 52.38 %

Testing the neural network on a set of messages containing 3 errors....
313/313 [=====] - 1s 2ms/step - loss: 0.0015 - accuracy: 0.3953
The accuracy achieved by the neural network: 39.53 %

Testing the neural network on a set of messages containing 4 errors....
313/313 [=====] - 1s 2ms/step - loss: 0.0194 - accuracy: 0.3374
The accuracy achieved by the neural network: 33.74 %
```

Rys. 2. Skuteczność sieci dla dekodowania ciągów, w których wystąpiły różne liczby błędów

```
Message received:
[0 1 0 0 1 0 1 1 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1
 1 0 0 1 0]

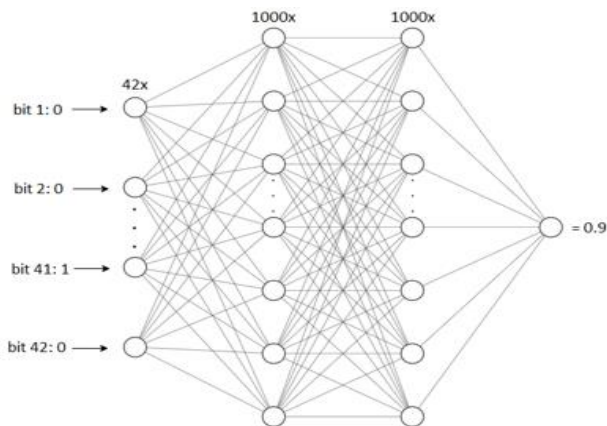
The error vector returned by the neural network:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 0 0]

Expected error vector:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 0 0]
```

Rys. 3. Przypadek, w którym sieć neuronowa poprawnie wskazała dwa błędne bity z czterech (błędne bity: 12, 15, 26, 40; poprawione bity: 15, 40)

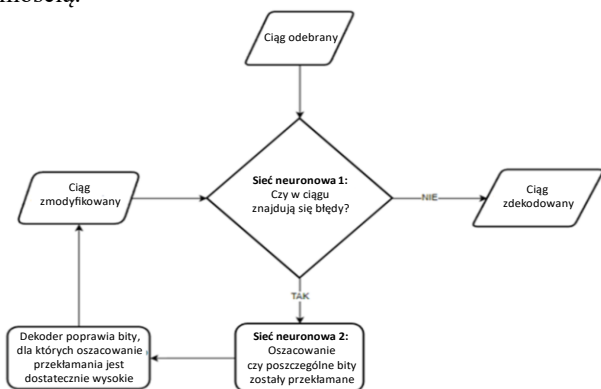
Aby naprawić problem niepoprawnego dekodowania bezbłędnych ciągów oraz wykorzystać zaletę zaproponowanej sieci neuronowej w postaci zdolności do po-

prawiania części błędnych bitów, zdecydowano się wytrenować dodatkową sieć neuronową, której zadaniem jest określanie czy w ciągu odebrany wystąpił co najmniej jeden błąd, bez wskazywania liczby błędów oraz ich pozycji. Ta dodatkowa sieć neuronowa ma taką samą strukturę oraz parametry treningu, jak wcześniej opisywana sieć, z tym wyjątkiem, że warstwa wyjściowa składa się z tylko jednego neuronu, którego wyjście odpowiada rozmytej decyzji dotyczącej wystąpienia błędów w ciągu odebrany (rys. 4).



Rys. 4. Diagram sieci neuronowej szacującej czy w ciągu odebrany wystąpił co najmniej jeden błąd

W ten sposób działanie dekodera zostało ostatecznie oparte na współpracy dwóch sieci neuronowych (rys. 5), podczas której, gdy na wejście dekodera trafia ciąg odebrany, jedna z sieci oszacowuje czy w ciągu pojawiły się błędy. Następnie, jeśli wartość otrzymana na wyjściu sieci jest bliska jedności, to ciąg odebrany przechodzi na wejście następnej sieci, która ocenia dla poszczególnych bitów czy zostały one przekłamanie. Po poprawieniu wykrytych błędów wynikowy ciąg trafia z powrotem na wejście sieci, oszacowującej występowanie błędów w korygowanym ciągu. Jeżeli wartość oszacowania, że w ciągu znajdują się błędy jest dostatecznie niska, dekodery zwraca ciąg w bieżącej postaci, będący już zdekodowaną wiadomością.



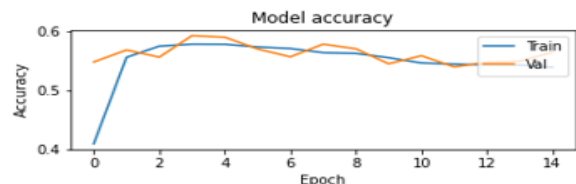
Rys. 5. Diagram przedstawiający algorytm postępowania w ramach zaproponowanego dekodera

2.3. Przebieg treningu

Do wytrenowania sieci dekodujących opisany kod iterowany konieczne było wygenerowanie odpowiednio dużego zbioru danych treningowych. Do treningu oby-

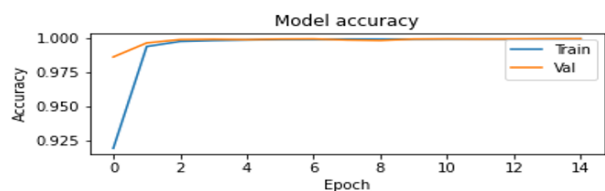
dwu sieci wygenerowano po około milionie danych treningowych, w postaci ciągów kodowych z losowo wprowadzonymi błędami. Do tych danych dołączane są oczekiwane odpowiedzi sieci neuronowych w postaci wektorów błędów oraz jednobitowych informacji o tym czy w ciągu znajdują się błędy. Dane były generowane cyklicznie, co przy liczbie danych równej milionowi, oznacza, że każde słowo kodowe pojawiło się w zbiorze treningowym ponad 200 razy (ponieważ kod posiadający 12 bitów informacyjnych posiada $2^{12} = 4096$ słów kodowych). Takie podejście umożliwia osiągnięcie znacznie lepszych skuteczności treningu sieci neuronowych niż w przypadku danych wygenerowanych całkowicie losowo, gdy nie ma pewności co do tego, czy każde słowo kodowe pojawi się podczas treningu przynajmniej raz.

Najpierw przeprowadzono trening sieci służącej do identyfikacji przekłamania poszczególnych bitów. Dane zostały podzielone na dane treningowe (70%), walidacyjne (15%) oraz testowe (15%). Trening składał się z 15 epok. Na rys. 6 przedstawiono wykres skuteczności wytrenowanej sieci neuronowej. Jak widać, już podczas pierwszej epoki udało się uzyskać skuteczność sieci neuronowej na poziomie ponad 50%. Podczas kolejnych epok skuteczność sieci utrzymywała się na stosunkowo równym poziomie. Więcej informacji na temat skuteczności sieci, może dostarczyć rys. 2. Zważywszy na sposób działania zaproponowanego dekodera, najważniejsza jest skuteczność sieci dla dekodowania ciągów z pojedynczym błędem, ponieważ można ten parametr rozumieć też jako szansę na to, że sieć poprawi przynajmniej jeden błąd ze wszystkich obecnych w ciągu (o ile liczba błędów nie przekracza zdolności korekcyjnych kodu iterowanego). Sieć w tym przypadku osiąga niemal 100% skuteczności, co jest wynikiem jak najbardziej zadowalającym.



Rys. 6. Wykres zmian dokładności sieci służącej do wykrywania błędnych bitów dla kolejnych epok treningu

W analogiczny sposób podzielono dane służące do treningu sieci neuronowej, której zadaniem jest określanie czy w ciągu odebrany pojawił się przynajmniej jeden błąd. Parametry treningu oraz struktura sieci również pozostają takie same z wyjątkiem liczby neuronów w warstwie wyjściowej.



Rys. 7. Wykres dokładności sieci do wykrywania, czy w wiadomości wystąpiły, dla kolejnych epok treningu

Jak widać na rys. 7, już przy pierwszej epoce udało się uzyskać bardzo wysoką skuteczność sieci na poziomie

