KONFERENCJA RADIOKOMUNIKACJI
I TELEINFORMATYKI
KRiT 2023

DEKODOWANIE KODÓW ITEROWANYCH Z UŻYCIEM SIECI NEURONOWEJ

DECODING OF ITERATED CODES WITH THE USE OF A NEURAL NETWORK

Oskar Fiedot[1]; Marek Blok[2]

[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 dekoder 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, dekoder 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:

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

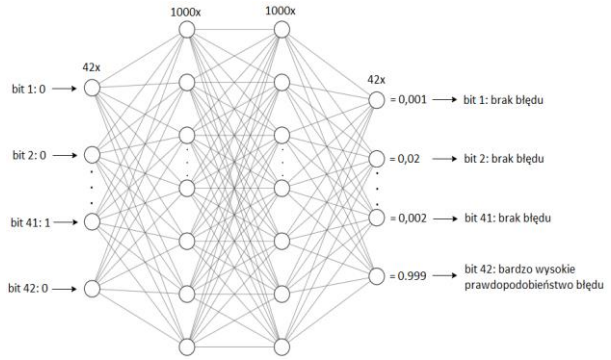
Kody te cechują kolejno 4 bity informacyjne i długość 7 oraz 3 bity informacyjne i długość 6, co oznacza, że wynikowy kod cechuje 12 bitów informacyjnych oraz łączna długość 42 bity, co daje 30 dodatkowych bitów kontroli parzystości. Zawartość informacyjna złożonego

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

2.2. Struktura sieci neuronowej

Zaproponowany dekoder opiera się o współpracę dwóch sieci neuronowych. To podejście jest wynikiem niewystarczająco dobrych rezultatów osiąganych przez dekoder oparty 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 osiąganie 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 niezmienione. 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 poskutkował 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 bezbłędnymi 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 %
```

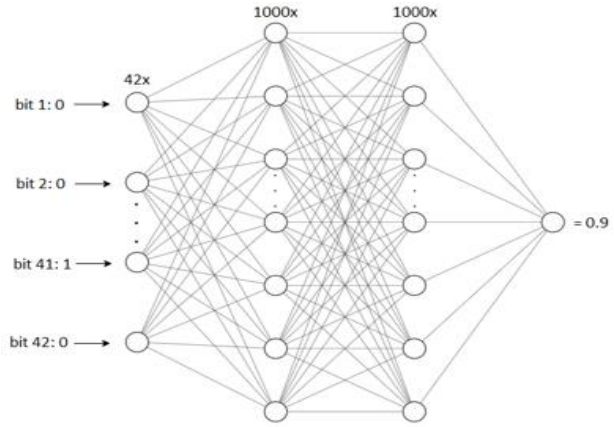
Rys. 2. Skuteczność sieci dla dekodowania ciągów, w których wystąpiły różne liczby błędów

```
Message received:
[0 1 0 0 1 0 1 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 1 1 0 1 1 0 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 1 0 0 0 0 0 0 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 1 0 0 1 0 0 0 0 0 0 0 0 0 1 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)

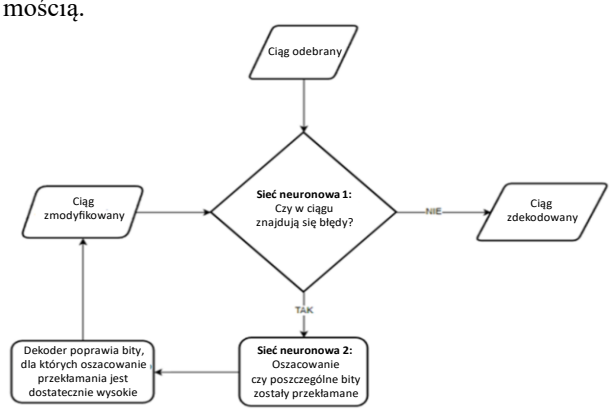
Aby naprawić problem niepoprawnego dekodowania bezbłędnych ciągów oraz wykorzystać zaletę zaproponowanej sieci neuronowej w postaci zdolności do po-

prawiania części błędnych bitów, zdecydowano się wytrenować dodatkową sieć neuronową, której zadaniem jest określanie czy w ciągu odebranym 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 odebranym (rys. 4).



Rys. 4. Diagram sieci neuronowej szacującej czy w ciągu odebranym 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 przekłamane. 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, dekoder zwraca ciąg w bieżącej postaci, będący już zdekodowaną wiadomością.

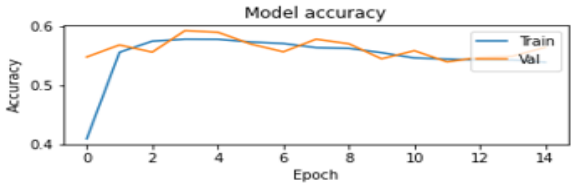


Rys. 5. Diagram przedstawiający algorytm postępowania w ramach zaproponowanego dekodera

2.3. Przebieg treningu

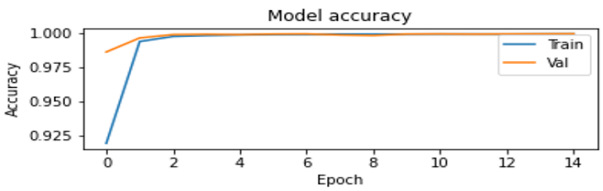
Do wytrenowania sieci dekodujących opisany kod iterowany konieczne było wygenerowanie odpowiednio dużego zbioru danych treningowych. Do treningu obydwu sieci wygenerowano po około milionie danych treningowych, w postaci ciągów kodowych z losowo wprowadzonymi błędami. Do tych danych dołączane są oczekiwane 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 odebranym 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

około 98%. Skuteczność sieci utrzymywała się na równym poziomie przez cały czas trwania treningu podobnie jak w przypadku poprzedniej sieci neuronowej. Na rys. 8 można zauważyć, że wytrenowana sieć neuronowa osiąga skuteczność na poziomie ponad 99% dla ciągów z losową liczbą błędów (w zakresie zdolności korekcyjnych kodu) na losowych pozycjach, co jest znakomitym wynikiem.

```
4688/4688 [==============================] - 10s 2ms/step - loss: 0.0037 - accuracy: 0.9995
The accuracy achieved by the neural network: 99.95 %
```

Rys. 8. Skuteczność sieci neuronowej służącej do określania, czy w wiadomości wystąpiły błędy

W ten sposób, możliwe było wytrenowanie dwóch sieci neuronowych, które mogą zostać wykorzystane w zaproponowanym dekoderze, przeprowadzającym dekodowanie wiadomości w oparciu o współpracę tych dwóch sieci neuronowych. Na rys. 9 przedstawiono przykład ciągu zawierającego 4 błędy poprawnie zdekodowanego z użyciem zaproponowanego dekodera.

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

Error vector:
[0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 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]

Message corrected by a neural network-based decoder:
[0 1 1 1 0 0 1 1 0 1 1 0 1 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 0 0 0 0 0 0 0 1 1
 0 0 0 1 1]
```

Rys. 9. Przykład poprawnie zdekodowanej wiadomości

3. Porównanie z dekoderem klasycznym

Z punktu widzenia użyteczności dekodera opartego na sieciach neuronowych, istotne jest również porównanie jego skuteczności ze skutecznością dekodera klasycznego. Wyniki porównania dla danych z losową liczbą błędów przedstawiono na rys. 10. Jak można zauważyć, obydwa dekodery osiągnęły bardzo wysokie i podobne skuteczności na zadanym zbiorze danych testowych. Przewagę dekodera opartego na sieciach neuronowych, można zauważyć przy porównaniu skuteczności dekodowania ciągów, w których wystąpiły 4 błędy. To porównanie zostało przedstawione na rys. 11. W tym przypadku, dekoder oparty na sieciach neuronowych osiągnął skutecznością lepszą o niecałe 5 punktów procentowych. Jednak w przypadku liczb błędów mniejszych niż 4 dekoder klasyczny osiągał zawsze skuteczność równą 100%, natomiast dekoder oparty na sieciach neuronowych ponad 99%, co, mimo że jest to minimalna przewaga, daje jednak pewność, że dekoder klasyczny poprawnie zdekoduje ciągi, w których wystąpiły te liczby błędów, podczas gdy tej pewności nie ma przy stosowaniu dekodera, opartego na sieciach neuronowych. Na podstawie tych wyników, można wysnuć wniosek, że dekoder, oparty na sieciach neuronowych, w większości przypadków nie gwarantuje 100% skuteczności dekodowania, jednak może zapewnić skuteczność na poziomie około 99% dla każdej możliwej do poprawy liczby błędów, w przeciwieństwie do dekodera klasycznego, który choć zapewnia 100% skuteczności dla dekodowania ciągów, w których wystąpiło do trzech błędów, cechuje się znacznie niższą skutecznością (choć nadal wysoką) podczas dekodowania ciągów, w których wystąpiły cztery błędy, mimo że zdolności korekcyjne kodu iterowanego powinny zapewniać poprawę takiej liczby błędów.

```
Comparing the effectiveness of decoders... ———————————————— 100% 0:00:00
Efficiency of a neural network-based decoder: 99.68 %
Efficiency of the classical decoder: 98.98 %
```

Rys. 10. Porównanie skuteczności dekodowania ciągów, w których wystąpiła losowa liczba błędów

```
Comparing the effectiveness of decoders... ———————————————— 100% 0:00:00
Efficiency of a neural network-based decoder: 98.82 %
Efficiency of the classical decoder: 94.04 %
```

Rys. 11. Porównanie skuteczności dekodowania ciągów, w których wystąpiły 4 błędy

4. Podsumowanie

Zaproponowany dekoder złożony z dwóch sieci neuronowych oraz metoda jego treningu pozwala na uzyskanie bardzo wysokiej skuteczności dekodowania i zarazem prostej implementacji. Udało się wykazać, że wytrenowanie sieci neuronowych do dekodowania kodów iterowanych jest możliwe, choć może wymagać bardzo dużej ilości danych treningowych, a dekoder oparty na sieciach neuronowych może być bardzo dobrą alternatywą dla dekodera klasycznego. Trening sieci neuronowych do dekodowania kodów iterowanych może jednak sprawiać pewne problemy. Tak wysoką skuteczność sieci udało się uzyskać dzięki dużej objętości zbioru danych treningowych, który był kilkukrotnie razy większy, niż zbiór wszystkich możliwych słów kodowych kodu iterowanego, dla którego trenowane były sieci. Umożliwiło to wykorzystanie każdego słowa kodowego w trakcie treningu bardzo wiele razy. W przypadku dłuższych kodów iterowanych, zbiór wszystkich słów kodowych jest tak ogromny, że wygenerowanie kilkukrotnie większego zbioru treningowego może być bardzo trudne lub praktycznie nieopłacalne. Oznaczałby to również bardzo długi trening sieci neuronowych i mogłoby wpłynąć na pogorszenie skuteczności. W dalszej perspektywie warto rozważyć połączenie w ramach proponowanej koncepcji dekodera klasycznego z dekoderem neuronowym wykorzystując każdy w zakresie, w którym jest on skuteczniejszy. Pozwoliłoby to na wykorzystanie dużej skuteczności klasycznego dekodera na mniejszej liczby błędów oraz ograniczyć problem stawiany przed siecią neuronową pozwalając na jej uproszczenie.

LITERATURA

[1] Kingma P. Diederik, Ba Jimmy. 2014. „Adam: A method for stochastic optimization”.

[2] Fiedot Oskar. 2023. „Dekodowanie kodów iterowanych z użyciem sieci neuronowej”. *Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki.*

[3] Abdelbaki Hisham, Gelenbe Erol, El-Khamy Said Esmail. 1999. “Random neural network decoder for error correcting codes”. *International Joint Conference on Neural Networks*: 3241-3245.

[4] El-Khamy S. E, Youssef E. A., Abdou H. M., 1995, “Soft decision decoding of block codes using artificial neural network”, *Proceedings IEEE Symposium on Computers and Communications*: 234-240.