ZESZYTY NAUKOWE AKADEMII MARYNARKI WOJENNEJ
ROK XLVII NR 2 (165) 2006

Piotr Szymak
Robert Tomasik
Akademia Marynarki Wojennej

IMPLEMENTACJA PODSYSTEMU DIAGNOSTYCZNEGO OKRĘTOWEGO ZESPOŁU PRĄDOTWÓRCZEGO W ŚRODOWISKU „OPEN SOURCE”

STRESZCZENIE

W artykule przedstawiono zaprojektowany podsystem diagnostyki przewidziany do oceny stanu technicznego okrętowego zespołu prądotwórczego. Do przeprowadzenia analizy dostarczonych danych proponuje się wykorzystać sztuczne sieci neuronowe o architekturze determinowanej przez użytkownika. W celu obniżenia kosztów projektu implementację prezentowanego podsystemu diagnostyki zrealizowano w środowisku „Open Source”, czyli tzw. wolnego oprogramowania.

WSTĘP

Ogromny postęp, jaki dokonał się w budowie układów mikroprocesorowych w ostatnich latach, pociągnął za sobą rozwój wielu dziedzin nauki. Nowoczesne układy pomiarowe zyskały możliwość zbierania, gromadzenia i przetwarzania ogromnej ilości danych w bardzo krótkim czasie. Pozwoliło to na konstruowanie skomplikowanych systemów kontrolno-pomiarowych jednoznacznie określających stan techniczny obiektów. Podczas budowania dużych jednostek pływających projektuje się podsystemy diagnostyczne [7] zawierające procedury wnioskowania, które proces podejmowania decyzji realizują w sposób automatyczny. Niestety, ze względu na bardzo wysokie koszty wspomnianych systemów nie są one wprowadzane do mniejszych oraz starszych jednostek lub są wprowadzane w bardzo ograniczonym zakresie.

W celu minimalizacji kosztów realizacji wspomnianych podsystemów diagnostycznych istotne jest zastanowienie się nad podstawowymi źródłami tych kosztów. Jak się okazuje, jednym z głównych składników ceny całego systemu są środki finansowe wydatkowane na oprogramowanie, w skład którego wchodzi m.in.:

- licencja na system operacyjny;
- licencje na środowiska programistyczne, w których tworzone jest oprogramowanie diagnostyczne;
- licencja na gotowe programy diagnostyczne.

W dalszej części artykułu przedstawione zostanie rozwiązanie aplikacyjne, które pozwoli obniżyć wymienione koszty.

IDEA WOLNEGO OPROGRAMOWANIA

W 1991 roku fiński programista Linus Torvalds stworzył jądro nowego systemu operacyjnego. Informacje o nim opublikował na internetowej liście dyskusyjnej, zaznaczając jednocześnie, że jest ono wydane w oparciu o licencję GNU GPL („GNU's Not UNIX" General Public License). Wiadomość ta spotkała się z ogromnym zainteresowaniem i wkrótce przy rozwoju tego systemu zaczęło pracować wiele osób, a sam projekt zyskał nazwę linux. Proces ów trwa do dziś, a milionowa liczba osób rozwijających oprogramowanie jest wspierana przez gigantów świata informatycznego, takich jak IBM czy Novel [5].

Dynamiczny rozwój systemu był możliwy dzięki licencji GNU GPL, sformułowanej w 1988 roku przez Richarda Stallmana i Ebena Moglena. Dla potrzeb projektu GNU licencja przekazała użytkownikom prawa do uruchamiania programu w dowolnym celu (prawo wolności nr 0), analizowania działania programu i dostosowywania go do swoich potrzeb (prawo wolności nr 1), kopiowania (prawo wolności nr 2) oraz udoskonalania i publikowania własnych poprawek (prawo wolności nr 3), programów i kodu źródłowego tych programów [6]. GPL jest obecnie najpopularniejszą licencją wolnodostępnego oprogramowania.

W obecnej chwili linux ze względu na swoją stabilność, bezpieczeństwo oraz odporność na wirusy i ataki komputerowe, a także niskie koszty utrzymania, ilość oprogramowania oraz możliwość darmowego pobrania zastępuje komercyjne systemy. Jego głównym zastosowaniem są co prawda systemy i serwery sieciowe, ale coraz częściej wykorzystuje się go w innych dziedzinach. Na pracę w tym właśnie systemie zdecydował się między innymi Pentagon oraz rząd Szwecji.

ARCHITEKTURA ZAPROJEKTOWANEGO OPROGRAMOWANIA DIAGNOSTYCZNEGO

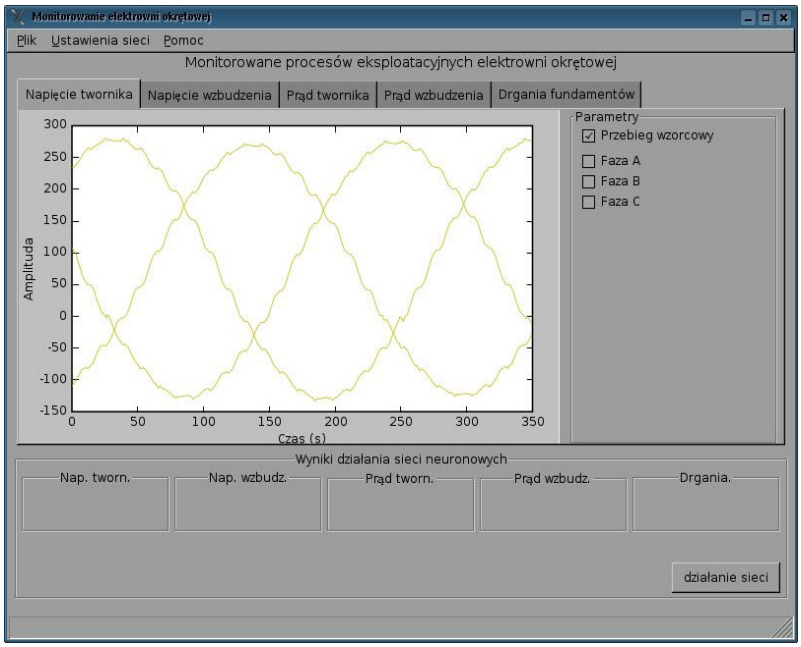
W Zakładzie Elektroniki i Elektrotechniki Akademii Marynarki Wojennej prowadzone są prace nad wdrożeniem układu monitorowania i sterowania systemem

elektroenergetycznym elektrowni okrętowej. Mają one na celu opracowanie całego podsystemu diagnostycznego [6]. Wytypowany został zbiór symptomów oraz przeprowadzono analizę mierzonych parametrów eksploatacyjnych i diagnostycznych. Ustalono, że dla większości przypadków do zdiagnozowania stanu obiektu wystarczające jest dokonanie pomiarów następujących parametrów [2]:

- napięcia twornika;
- napięcia wzbudzenia;
- prądu twornika;
- prądu wzbudzenia;
- drgań fundamentów [3].

Na tej podstawie ustalono stan techniczny obiektu w relacji defekt – symptom. Pomiaru parametrów dokonać można typowymi urządzeniami pomiarowymi, a następnie za pomocą przetwornika analogowo-cyfrowego przesłać do komputera i dokonać niezbędnej analizy. Najczęściej do tego typu analizy wykorzystywane jest wyspecjalizowane komercyjne oprogramowanie, co w bardzo dużym stopniu zwiększa koszty całego systemu diagnostycznego.

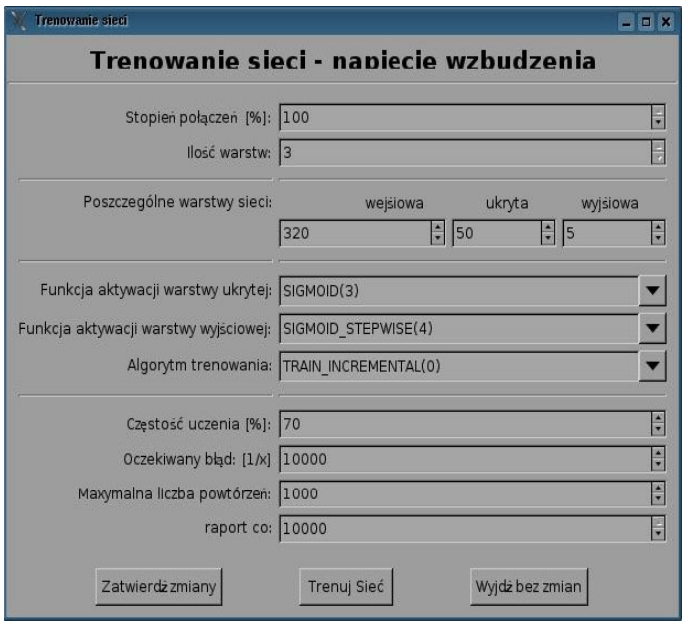
Właśnie dlatego proponuje się realizację wszystkich zadań związanych z archiwizacją i analizą danych przeprowadzić przy wykorzystaniu programu diagnostycznego zaprojektowanego w środowisku „Open Source” [5].



Rys. 1. Główne okno zaprojektowanej aplikacji diagnostycznej

Jako środowisko pracy zaprojektowanego programu diagnostycznego (rys. 1.) wykorzystano dystrybucję PLD Linux AC zaprojektowaną w 1998 roku przez polskich programistów. Jako środowisko programistyczne zastosowano natomiast Pythona, czyli interpretowany, interaktywny język programowania zaproponowany w 1990 roku przez Guido van Rossuma. Dodatkowo wykorzystano następujące biblioteki:

1) wxPython – pakiet dla języka Python umożliwiający tworzenie interfejsów graficznych zaimplementowany jako nakładka na bibliotekę wxWidgets;
2) matplot – zorientowana obiektowo biblioteka do elastycznego tworzenia wykresów 2D z danych wczytywanych z plików;
3) Fann – biblioteka napisana w języku C z implementacją Pythona do tworzenia sztucznych sieci neuronowych.



Rys. 2. Okno modyfikacji architektury i parametrów trenowania sztucznych sieci neuronowych

Przy wykorzystaniu opisanego wcześniej oprogramowania powstała aplikacja, która pozwala na tworzenie sztucznych sieci neuronowych o zadanej architekturze, wybór parametrów procesu, uczenie się sieci neuronowej (rys. 2.), dokonanie procesu trenowania tych sieci oraz analizę otrzymanych danych (rys. 3.) [4]. Efektem wyjściowym jest informacja o stanie technicznym diagnozowanego obiektu.

Dodatkowo na generowanych wykresach można dokonać porównania przebiegów przyjętych za wzorcowe z przebiegami rzeczywistymi.

W trakcie przygotowywania procesu uczenia aplikacja umożliwia wybór funkcji aktywacji osobno dla każdej warstwy neuronów. W rozwijalnym polu wyboru dostępnych jest czternaście funkcji aktywacji:

1) LINEAR – funkcja liniowa;
2) THRESHOLD – funkcja progowa;
3) THRESHOLD_SYMMETRIC – symetryczna funkcja progowa;
4) SIGMOID – funkcja sigmoidalna;
5) SIGMOID_STEPWISE – skokowa funkcja sigmoidalna;
6) SIGMOID_SYMMETRIC – symetryczna funkcja sigmoidalna;
7) SIGMOID_SYMMETRIC_STEPWISE – skokowa symetryczna funkcja sigmoidalna;
8) GAUSSIAN – funkcja Gausa;
9) GAUSSIAN_SYMMETRIC – symetryczna funkcja Gausa;
10) GAUSSIAN_STEPWISE – skokowa funkcja Gausa;
11) ELLIOT – funkcja Elliota (zaprojektowana przez Davida Elliota na bazie sigmoidalnej funkcji aktywacji);
12) ELLIOT_SYMMETRIC – symetryczna funkcja Elliota;
13) LINEAR_PIECE – liniowa krokowo funkcja aktywacji;
14) LINEAR_PIECE_SYMMETRIC – symetryczna, liniowa krokowo funkcja aktywacji.

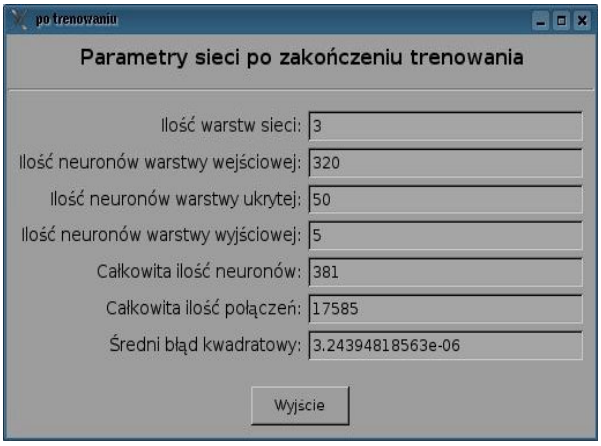
Można także dokonać wyboru jednego z czterech algorytmów trenowania:

1) TRAIN_INCREMENTAL – algorytm uczenia metodą przyrostową;
2) TRAIN_BATCH – standardowy algorytm wstecznej propagacji błędów;
3) TRAIN_RPROP – zaawansowany algorytm trenowania;
4) TRAIN_QUICKPROP – zmodyfikowany algorytm trenowania.

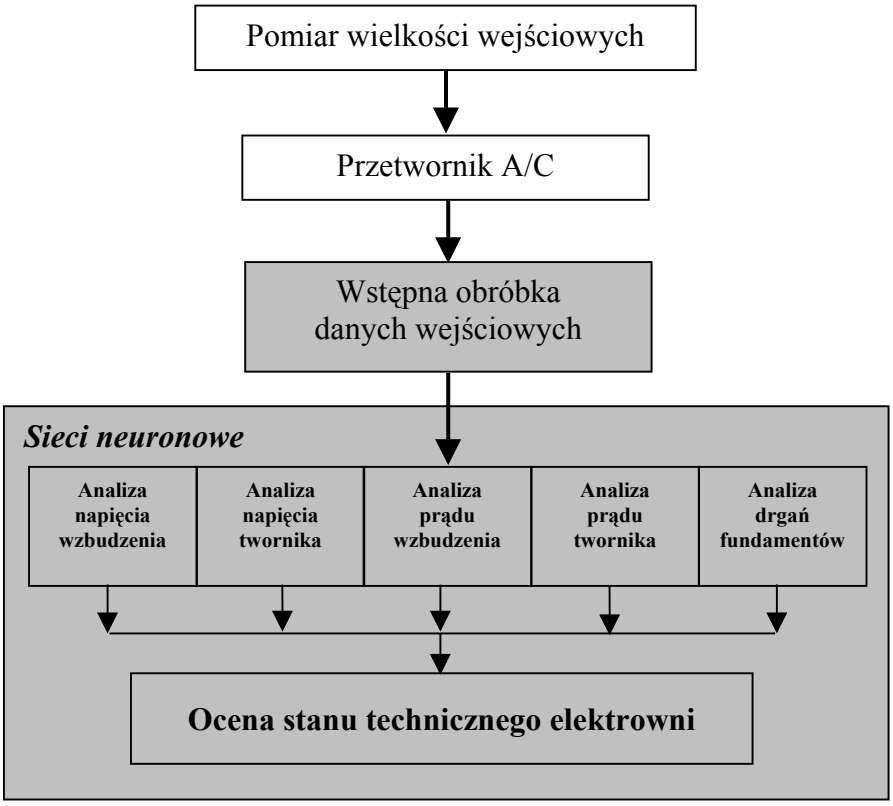
Dodatkowo możliwe jest ustawienie takich parametrów, jak:

1) liczba neuronów w poszczególnych warstwach;
2) stopień połączeń sieci;
3) częstość uczenia;
4) oczekiwany średni błąd kwadratowy;
5) maksymalna liczba powtórzeń.

Zakończenie procesu uczenia wykonywane jest po zadanej wcześniej liczbie powtórzeń lub po osiągnięciu założonego średniego błędu kwadratowego, co minimalizuje możliwość wystąpienia przetrenowania sieci.



Rys. 3. Parametry „wytrenowanej" sztucznej sieci neuronowej



Rys. 4. Ogólna zasada działania zaprojektowanego programu diagnostycznego (bloki z szarym tłem)

Ogólny algorytm działania proponowanego oprogramowania diagnostycznego został zobrazowany na rysunku 4. Aplikacja składa się z sześciu niezależnych dwuwarstwowych sieci neuronowych [4]. Pięć z nich odpowiada za analizę poszczególnych sygnałów:

- napięcia twornika;
- prądu twornika;
- napięcia wzbudzenia;
- prądu wzbudzenia;
- drgań mechanicznych.

Ostatnia, szósta sieć neuronowa po otrzymaniu sygnałów z pięciu poprzedzających sieci dokonuje rozpoznania stanu technicznego badanego obiektu.

Przedstawione rozwiązanie charakteryzuje wiele zalet. Większość defektów ma odwzorowanie w jednym lub dwóch monitorowanych sygnałach. Nie ma więc konieczności uczenia sieci poprawnych sygnałów, może to być wręcz szkodliwe i prowadzić do utraty zbieżności. Znacznie szybszy jest również proces uczenia sieci nowych defektów i większa jest ich odporność na zakłócenia oraz ewentualne uszkodzenia. Przy zaprezentowanej modułowej budowie aplikacji istnieje też możliwość jej łatwiejszej modyfikacji, tzn. w badaniach eksperymentalnych dla oceny jednego lub kilku sygnałów wejściowych zamiast proponowanej metody sztucznych sieci neuronowych można wykorzystać inne metody sztucznej inteligencji lub algorytmy klasyczne. Na uwagę zasługuje rozważana możliwość zastąpienia ostatniej szóstej sieci neuronowej systemem ekspertowym, opartym na metodach logiki rozmytej [1].

PODSUMOWANIE

Zaprezentowana aplikacja diagnostyczna została przetestowana na zarejestrowanych wcześniej i zarchiwizowanych danych. Wyniki przeprowadzonych testów są obiecujące. Całość oprogramowania działa stabilnie, szybko i pewnie. Sztuczne sieci neuronowe podczas procesu trenowania szybko osiągają założony średni błąd kwadratowy i w trakcie pracy pewnie rozpoznają wprowadzone sygnały wejściowe.

W dalszej części prowadzonych prac niezbędne są badania eksperymentalne na rzeczywistym obiekcie, przeprowadzone na danych wczytywanych bezpośrednio z przetworników cyfrowo-analogowych. Już w tej chwili można jednak stwierdzić,

iż wykorzystanie wolnego oprogramowania stwarza możliwość uzyskania taniego i skutecznie działającego narzędzia diagnostycznego, co uzasadnia celowość kontynuowania prac.

BIBLIOGRAFIA

[1] Driankov D., Hellendoorn H., Reinfrank M., *Wprowadzenie do sterowania rozmytego*, Wydawnictwo Naukowo- Techniczne, Warszawa 1996.

[2] Listewnik K., *Symptomy stanów granicznych okrętowej prądnicy synchronicznej w diagnostyce zestawu zasilania elektrycznego*, rozprawa doktorska, Akademia Marynarki Wojennej, Gdynia 2001.

[3] Morel J., *Drgania maszyn i diagnostyka ich stanu technicznego*, Polskie Towarzystwo Diagnostyki Technicznej, Warszawa 1994.

[4] Osowski S., *Sieci neuronowe w ujęciu algorytmicznym*, Wydawnictwo Naukowo-Techniczne, Warszawa 1996.

[5] *System operacyjny GNU*, Internet, http://www.gnu.org/home.pl.html

[6] *Wikipedia*, Internet, http://pl.wikipedia.org/wiki/

[7] Żółtkowski B., *Podsystemy diagnostyczne maszyn*, Bydgoszcz 1996.

ABSTRACT

The paper presents a diagnostic subsystem designed to estimate the technical condition of a shipboard generator set. To analyze the data delivered using neural nets of user-determined architecture is suggested. In order to reduce the costs of the project the diagnostic system presented was implemented in „Open Source" environment.

Recenzent kontradmirał prof. dr hab. inż. Zygmunt Kitowski