POLITECHNIKA KRAKOWSKA
IM. TADEUSZA KOŚCIUSZKI
WYDZIAŁ FIZYKI MATEMATYKI I INFORMATYKI
KIERUNEK INFORMATYKA

PIOTR BORYCZKO

NEUROEWOLUCJA TOPOLOGII SIECI NEURONOWYCH Z WYKORZYSTANIEM ALGORYTMÓW GENETYCZNYCH

PRACA INŻYNIERSKA
STUDIA STACJONARNE

Promotor: Dr inż. Tomasz Gąciarz

Kraków 2011

Spis treści

1 Wstęp

Współcześnie, coraz częściej i w coraz większej ilości problemów wykorzystuje się rozwiązania z dziedziny Sztucznej Inteligencji. Sieci neuronowe i algorytmy genetyczne umożliwiają dostarczenie rozwiązań w przypadkach, gdy brakuje dokładnego algorytmu wykonania zadania. Ich umiejętność dopasowywania działania do sytuacji, znajdują zastosowanie między innymi w sterowaniu, analizie języków i grach.

Zazwyczaj wykorzystuje się algorytmy genetyczne lub sieci neuronowe, jednak synteza obu tych rozwiązań może przynieść ciekawe perspektywy. Jednym z problemów z sieciami neuronowymi jest konieczność ich odpowiedniej nauki oraz doboru topologii sieci. Tu z pomocą mogą przyjść mechanizmy ewolucyjne dostarczane przez algorytmy genetyczne.

1.1 Cel pracy

Celem pracy jest wykorzystanie algorytmów genetycznych do modyfikacji parametrów i topologii sztucznych sieci neuronowych. Zostanie dokonana analiza działania przykładowego algorytmu genetycznego i porównanie wyników jego działania z wynikami standardowej metody nauczania sieci. Doświadczenia oparte będą o zgromadzoną bazę próbek głosowych cyfr w języku polskim.

Praktycznym rezultatem pracy, będzie stworzenie środowiska doświadczalnego w którym możliwe będzie dokonywanie eksperymentów z nauczaniem sieci neuronowych za pomocą algorytmów genetycznych i standardowych metod. Eksperymenty w aplikacji przeprowadzane będą przy pomocy przedstawionych w dalszej części pracy algorytmu genetycznego lub standardowej metody nauki z nauczycielem.

1.2 Plan pracy

Rozdziały pracy omawiają kolejne problemy i zagadnienia z którymi wiązała się realizacja pracy inżynierskiej. Uszeregowane są zgodnie z kolejnością z jaką były rozpoznawane i realizowane.

Na początku krótko zostanie omówiona tematyka neuroewolucji. Pomoże to zrozumieć kontekst tej pracy inżynierskiej.

Rozdział trzeci poświęcony został krótkiemu omówieniu działania sieci neuronowych. Przedstawiono w nim zasadę uczenia sieci algorytmem wstecznej propagacji błędów.

W kolejnym rozdziale zawarty został wstęp dotyczący algorytmów genetycznych. Omówiono w nim sposoby użycia algorytmów genetycznych do nauki i doboru najlepszych sieci neuronowych. Analizowane jest tu dokładne działanie algorytmu NEAT.

Piąty rozdział poświęcony jest próbkom głosowym polskich cyfr. Przedstawię sposób ich zebrania oraz metodę użytą do zwiększenia ich ilości. Wyjaśniono metodę użytą do obróbki tych próbek, tak aby mogły być później przetwarzane przez sieci neuronowe.

W szóstym rozdziale, omówione zostały technologie użyte przy wykonaniu aplikacji. Przedstawię przyczyny konkretnych decyzji, oraz przedstawię pochodzenie używanych zewnętrznych bibliotek lub algorytmów.

W siódmym rozdziale zaprezentowane jest działanie aplikacji i jej interfejs graficzny. Zawiera architekturę programu i stanowi instrukcję jego obsługi.

Rozdział 8 zawiera wyniki doświadczeń. Przedstawiono wyniki rozpoznawania próbek rozpoznawanych przez sieci uczone metodą wstecznej propagacji oraz sieci dobierane przez algorytm genetyczny.

Ostatni rozdział to podsumowanie pracy.

2 Neuroewolucja topologii sieci neuronowych

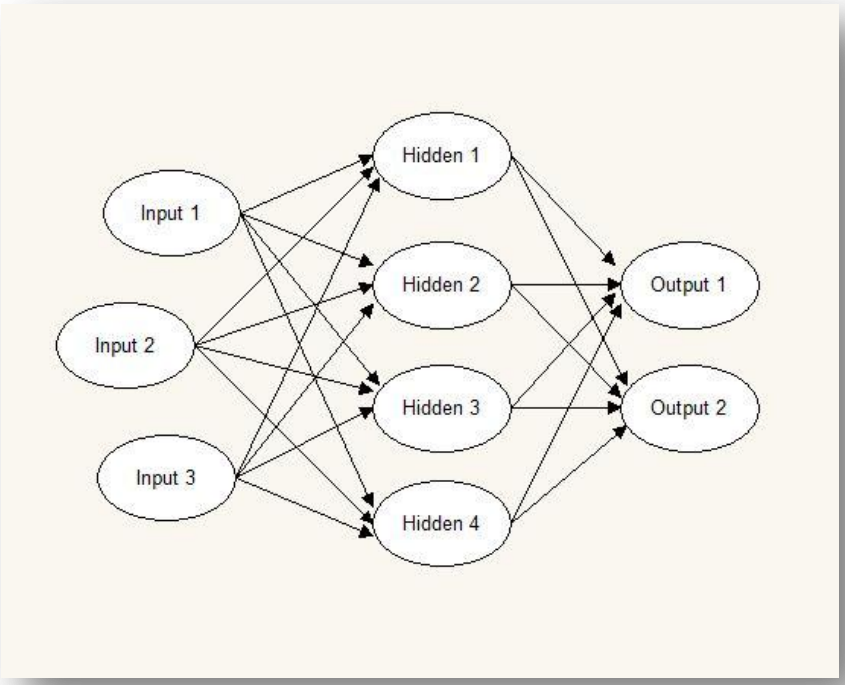
Neuroewolucja, czyli rozwój sztucznej sieci neuronowej za pomocą algorytmów ewolucyjnych jest ostatnio bardzo intensywnie badaną dziedziną uczenia maszynowego [Stanley 3]. Znajduje zastosowania w sytuacjach w których możliwe jest ocenienie poprawności działania generowanych sieci neuronowych, ale trudne lub niemożliwe jest zdefiniowanie stałej struktury sieci, której wymagają standardowe metody nauki z nauczycielem.

Sednem działania algorytmów neuroewolucyjnych jest zakodowanie struktury sieci neuronowej w postaci chromosomu algorytmu genetycznego a następnie postępowanie z nim zgodnie z zasadami działania algorytmów genetycznych. Rozróżniane są dwa typy algorytmów neuroewolucyjnych. W pierwszym zakłada się stałą strukturę sieci neuronowej. Takie podejście bezpośrednio zastępuje wykorzystanie standardowych metod nauki z nauczycielem. W dalszym ciągu problemem pozostaje dobór odpowiedniej struktury sieci dla rozwiązywanego problemu.

Drugi typ algorytmów parametryzuje również kształt sieci neuronowych i wymaga określenia jedynie stałej liczby wejść i wyjść sieci neuronowej. Umożliwia to tworzenie struktur, które w pełnie będą się w stanie adaptować do wymagań. Sieci które podlegają ewolucji zarówno pod względem ich wag jak i topologii określane są jako TWEANN's (*Topology & Weight Evolving Artificial Neural Networks*). Istnieje kilka implementacji takich algorytmów, jednak w ninejszej pracy opisany zostanie algorytm NEAT (*NeuroEvolution of Augmenting Topologies*) .

3 Sztuczne sieci neuronowe

Sztuczne sieci neuronowe to matematyczne lub komputerowe modele, wzorowane ma strukturze i działaniu biologicznych sieci neuronowych. Modele te składają się z grupy powiązanych ze sobą sztucznych neuronów, które przetwarzają sygnały według zadanego wzorca algorytmicznego. Sieci zazwyczaj składają się z warstwy wejściowej, ukrytych i wyjściowej. Możliwe są struktury bez warstw ukrytych. [WWW2]



Rysunek 1.Sieć neuronowa

Nie są one są w stanie modelować rzeczywistej pracy mózgu, ale są odpowiednim narzędziem do reprezentacji skomplikowanych zależności zachodzących pomiędzy bodźcami (sygnałami wejściowymi) a reakcjami (sygnałami wejściowymi) [Tadeusiewicz 2007, s.4-5]. Tworzone są jako struktury adaptacyjne, dostosowujące zachowanie do zadanych warunków. Z uwagi na te cechy są doskonałym narzędziem do analizy mowy. W przypadku niniejszej pracy inżynierskiej – badania mówionych cyfr .

3.1 Nauka i optymalizacja sieci neuronowych

Sztuczne sieci neuronowe stanowią narzędzie, za pomocą którego można rozwiązywać wiele problemów. Jednak ich struktury same w sobie nie pozwolą na żadne efektywne działanie. Konieczne jest wykorzystanie mechanizmów uczenia, które umożliwią dostosowywanie parametrów takich sieci do specyfiki analizowanego problemu. Jest kilka możliwych podejść do tej kwestii.

Najczęściej stosowane są metody w których stosuje się nauczyciela-nadzorcę. Sprawność algorytmu jest po każdym kroku nauczania sprawdzana przez metodę oceniającą i na podstawie tej oceny modyfikowane są odpowiednio wagi neuronów. Przykładem takiego podejścia jest metoda wstecznej propagacji błędów.

3.1.1 Metoda wstecznej propagacji błędów

Wsteczna propagacja błędów (*backpropagation*) to jedna z najpopularniejszych i sztandarowych metod nauki sieci neuronowych. „Metoda ta jest tak popularna, że w większości gotowych programów służących do tworzenia modeli sieci i ich uczenia – stosuje się tę metodę jako domyślną” [Tadeusiewicz 2007, s.220].

Metoda została dobrana jako element odniesienia dla bardziej zaawansowanych metod opierających się na neuroewolucji. Wymaga dostarczenia jej wektora oczekiwanych wartości na wyjściach sieci. W oparciu o różnicę między nimi a wartości uzyskanymi na wyjściach, wyliczany jest błąd. Na jego podstawie, modyfikowane są wartości wag na wcześniejszych połączeniach, rozpoczynając od warstwy poprzedzającej wyjściową a kończąc na połączeniach warstwy wejściowej.

4 Algorytmy genetyczne

Metoda *backpropagation* ma niewątpliwą zaletę prostoty i przewidywalności. Możliwe jest zastosowanie innego podejścia. Do nauczania i transformacji sieci neuronowych można użyć algorytmów genetycznych.

„Algorytmy genetyczne są to algorytmy poszukiwania oparte na mechanizmach doboru naturalnego oraz dziedziczności" [Goldberg 2003]. Podstawową strukturą jest chromosom. Reprezentuje on obiekt lub obiekty podlegające mechanizmom algorytmu genetycznego. Na takim chromosomie kodowane są wartości, struktury tych obiektów. Kolejne wyniki działania algorytmu genetycznego nazywamy populacjami – zbiorami chromosomów przetworzonych przez algorytm. Zbiór początkowy nazywamy populacją początkową. Na osobnikach populacji wykonuje się odpowiedniki zjawisk występujących w przyrodzie. Najważniejszymi z nich są mutacje i krzyżowania (*crossing-over*). Ocenę przydatności danego chromosomu dokonuje się na podstawie funkcji przystosowania.

Możliwe jest zastosowanie algorytmów genetycznych do uczenia sieci neuronowych poprzez zakodowanie ich parametrów na chromosomach – jest to opisana w rozdziale 2 neuroewolucja. Po tworzeniu kolejnych generacji, odbywa się tworzenie sieci neuronowych na podstawie informacji z chromosomu. Następnie przy pomocy zaprojektowanej funkcji dopasowania, podlegają one ocenie.

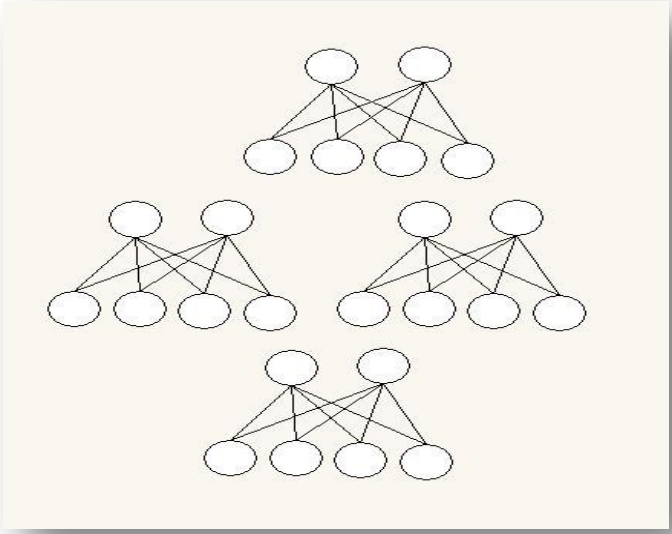
W metodach neuroewolucyjnych ze stałą topologią sieci na chromosomach kodowana jest informacja jedynie o wagach chromosomów. Kolejne dekodowane informacje nanoszone są na z góry ustaloną topologię sieci. Możliwe jest jednak zastosowanie algorytmów ze zmienną topologią sieci.

Pierwszym problemem, który trzeba rozwiązać w przypadku takich metod jest zakodowanie na chromosomie nie tylko wag neuronów ale także struktury powiązań między nimi. Następnie należy opracować odpowiednie algorytmy realizujące operacje genetyczne, tak aby modyfikowały oba elementy. Ostatecznie ważne jest dopracowanie wydajności takiej metody, gdyż do wykonania jest więcej operacji niż w przypadku poprzednich metod. Przykładem algorytmu stosującego takie podejście jest algorytm NEAT.

4.1 Algorytm N.E.A.T.

Algorytm NEAT został stworzony przez dr Kennetha O. Stanley'a. Jest to przykład wykorzystania neuroewolucyjnych metod ze zmienną topologią sieci. Wymagane jest jedynie podanie stałej liczby wejść i wyjść sieci [WWW2].

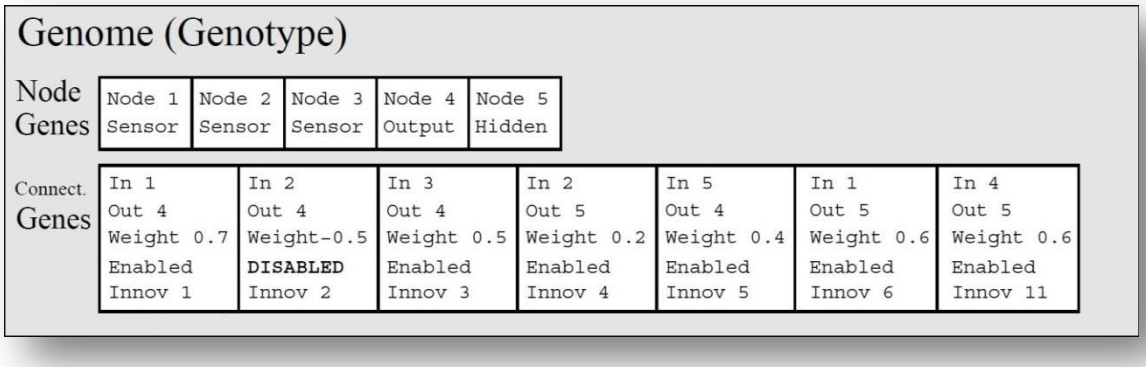
Populacją początkową algorytmu są sieci o dowolnej strukturze wewnętrznej i losowych wagach na węzłach sieci. Jedynym obostrzeniem jest wymóg identycznej struktury topologii sieci dla całej populacji początkowej. Zalecane jest też dobranie na początek prostej struktury, bez warstw ukrytych.



Rysunek 2. Przykładowa populacja początkowa

4.1.1 Kodowanie chromosomów

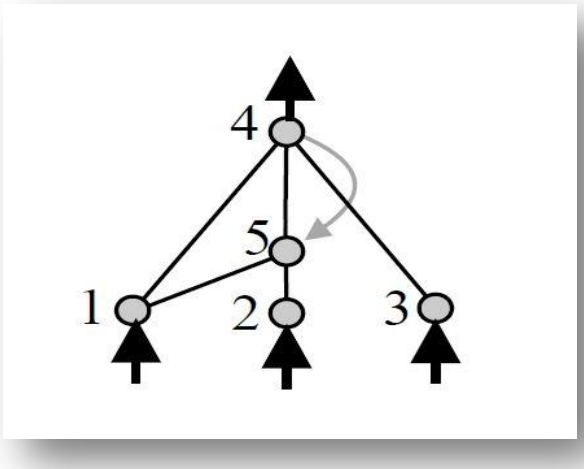
W każdym algorytmie neuroewolucyjnymi, ważne jest opracowanie kodowania sieci. W algorytmie N.E.A.T, ten problem rozwiązano kodując węzły sieci na jednym chromosomie a połączenia pomiędzy nimi na drugim, powiązanym chromosomie. Stworzony w ten sposób genotyp jest wystarczający do reprezentacji sieci neuronowej.



Rysunek 3. Przykładowy genotyp [Stanley 1]

Każdy gen chromosomu reprezentującego węzły sieci ma dwie składowe. Pierwsza to identyfikator węzła, a druga określa gdzie w strukturze sieci znajduje się węzeł. *Sensor* to węzeł wejściowy, *Output*, to węzeł na wyjściu sieci a *Hidden* oznacza, że węzeł znajduje się w warstwie ukrytej.

Struktura genu reprezentującego połączenia pomiędzy węzłami jest bardziej złożona. Pierwsze dwie składowe (*In* i *Out*) odpowiadają za określenie miedzy którymi węzłami jest to połączenie. *Weight* zawiera informację o wadze na połączeniu. Kolejna składowa informuje czy połączenie jest aktywne (*Enabled*) czy nie (*Disabled*). Ostatnia składowa genu – *Innov* określa współczynnik innowacyjności [Stanley 1 str. 8-10].

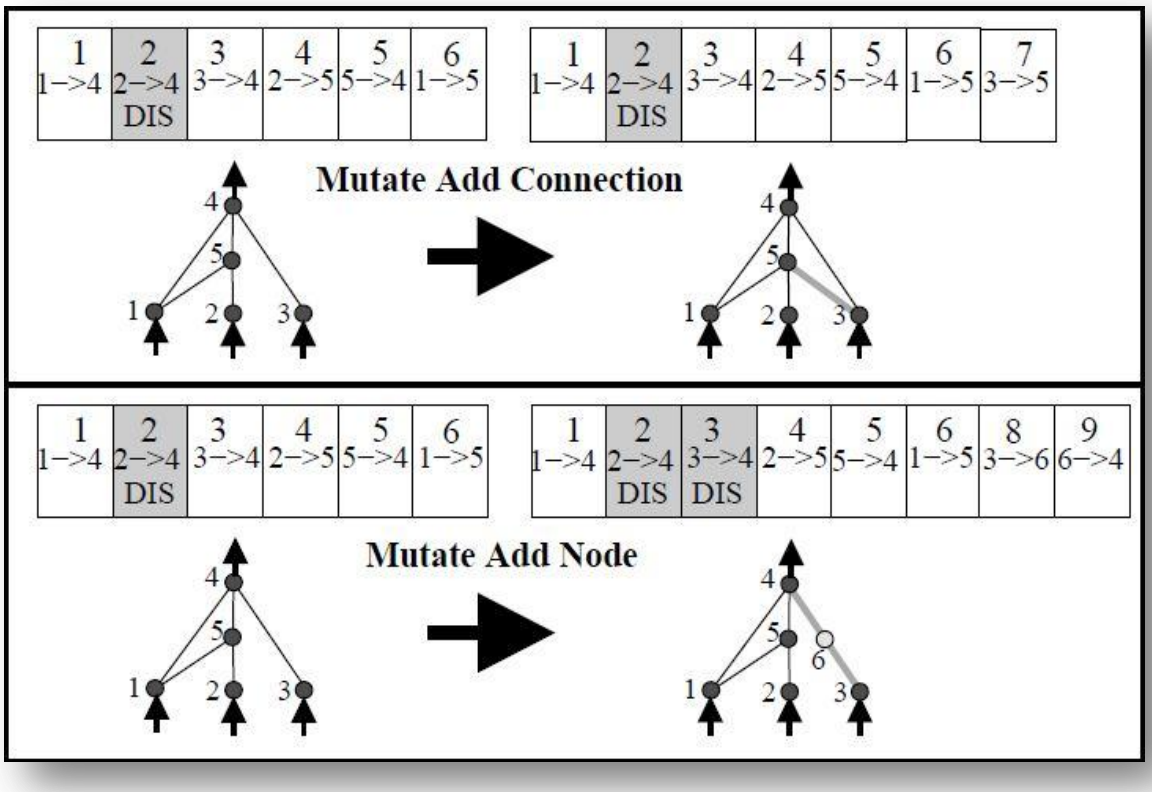


Rysunek 4. Fenotyp odpowiadający genomowi z rys. 6 [Stanley 1].

4.1.2 Mutacja

Pierwszą z dwóch operacji genetycznych, wykorzystanych w algorytmie jest mutacja. Może ona dotyczyć wag połączeń. Losujemy z pewnym prawdopodobieństwem chromosom, zmieniając losowo wagę na połączeniach przez niego reprezentowanych. Ciekawszy jest drugi typ mutacji – dotyczący topologii.

Tu rozróżniane są kolejne dwa typy mutacji. Mutacja może dotyczyć dodania połączenia lub dodania węzła. W pierwszym przypadku mutacja polega na dołączeniu genu do chromosomu odpowiadającego za połączenia. Dodatkowy gen zawiera losową wagę, jest aktywny i łączy losowo wybrane, niepołączone węzły sieci. W przypadku dodania nowego węzła pomiędzy 2 inne połączone węzły, dodaje się dodatkowy gen do chromosomu reprezentującego węzły. Ponadto dołączane są 2 geny reprezentujące połączenia miedzy nowo dodanym węzłem. Gen reprezentujący połączenie w które wstawiony został nowy węzeł zostaje zdezaktywowany.



Rysunek 5. Mutacje struktury [Stanley 1]

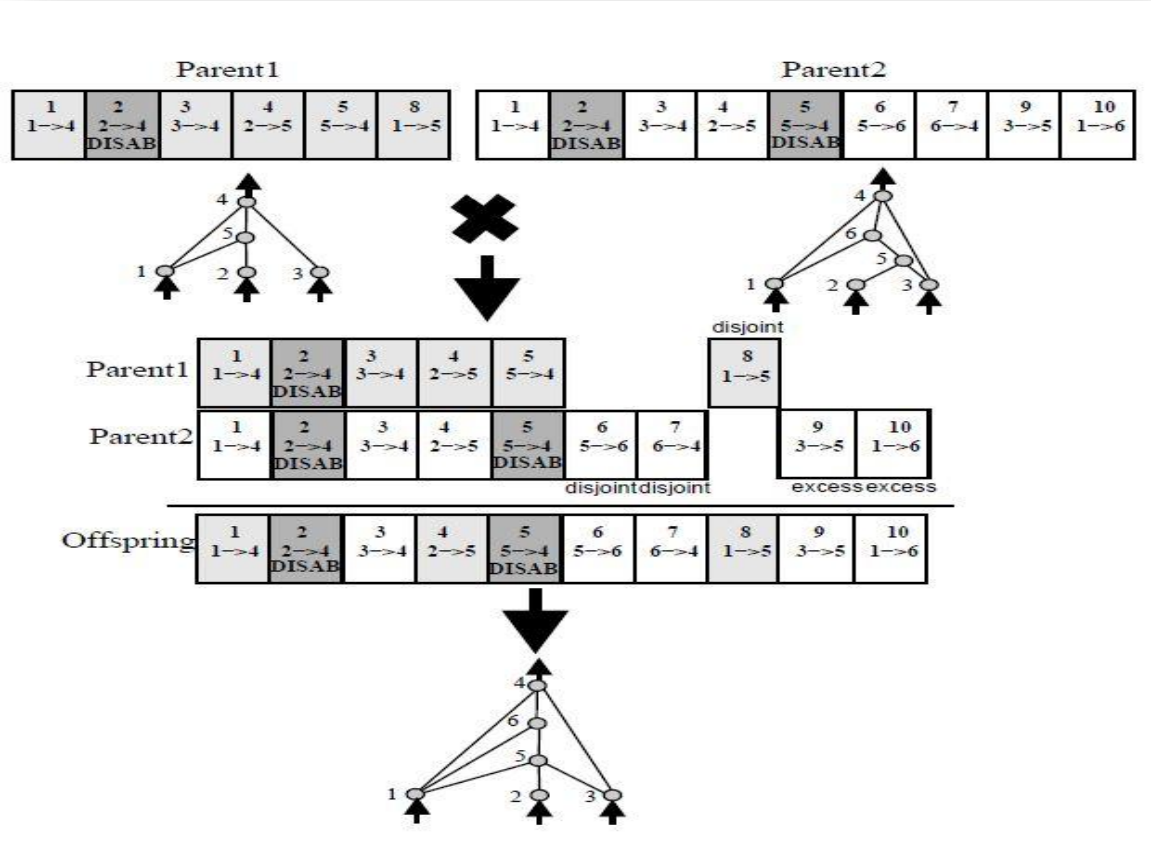
Dzięki mutacjom, sieć neuronowa rozwija się. Jednak sieci o różnych rozmiarach, sprawiają, że nietrywialnym zagadnieniem staje się krzyżowanie chromosomów [Stanley 1 str. 9-11].

4.1.3 Krzyżowanie

Bezpośrednie krzyżowanie 2 genotypów, reprezentujących różne struktury sieci nie jest możliwe. Aby taka operacja była możliwa, wykorzystywany jest współczynnik innowacyjności (*Innov*). Określa on pochodzenie genu. Za każdym razem, gdy przez dodawanie nowego genu, modyfikowana jest struktura sieci, inkrementuje się globalny współczynnik innowacji. Jego nową wartość przypisuje się do dodanego genu. Dzięki temu, bez skomplikowanych obliczeń możliwe jest obliczanie pochodzenia genu.

Gdy następuje krzyżowanie 2 chromosomów, geny dzielone są na 3 kategorie, według współczynnika innowacyjności. Chromosomy obu rodziców są układane wzdłuż siebie. Kolejne geny, o wspólnym współczynniku innowacyjności określa się jako **geny dopasowane** (W). Kolejne geny o niezgodnym współczynniku są genami **rozłącznymi** (D) lub **nadmiarowymi** (E) w zależności od tego czy znajdują się wewnątrz czy na zewnątrz względem zakresu drugiego rodzica.

W przypadku genów dopasowanych, potomek otrzymuje losowo wybrane genu od obu rodziców. W przypadku pozostałych genów, potomek otrzymuje geny bardziej dopasowanego rodzica. Geny D i E mniej dopasowanego rodzica przekazywane są wtedy, gdy pasują do struktury potomka. Takie podejście do krzyżowania rozwiązuje problem różnych topologii. [Stanley 1 str. 9-12]



Rysunek 6. Krzyżowanie sieci o różnych topologiach [Stanley 1]

4.1.4 Populacje

Do rozwiązania pozostaje jeden problem. Sieci o mało skomplikowanej strukturze są w stanie szybciej osiągnąć wysokie dopasowanie niż te o bardziej złożonej topologii. Spowodowane jest to dłuższym czasem na naukę wymaganym dla bardziej złożonych struktur. W celu ochrony innowacyjnych rozwiązań algorytm N.E.A.T wykorzystuje mechanizm gatunków. Chromosomy reprezentujące sieci o podobnej topologii grupuje się w gatunki. Przynależność do gatunku określa współczynnik odległości gatunkowej γ:

$$\gamma = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \bar{W}$$

N oznacza liczbę gatunków, $\bar{W}$ – średnią różnicę wag wśród genów dopasowanych, a współczynniki c pozwalają dopasować wagę dla poszczególnych składników. Każdy gatunek posiada współczynnik γ_t, gdzie t to numer identyfikacyjny gatunku. Przyporządkowując

nowy genotyp do populacji, wyliczamy jego współczynnik γ względem losowego przedstawiciela gatunku. Nowy genotyp zostaje przyporządkowany do gatunku, gdy γ jest mniejszy od γ_t [Stanley 1 str. 13].

5 Zbiór próbek głosowych

Aby móc przystąpić do jakichkolwiek badań, potrzebne są próbki na których można je przeprowadzać. Z uwagi, że niniejsza praca inżynierska, opiera się o badanie mówionych cyfr w języku polskim, konieczne było zebranie dostatecznie dużego zbioru nagrań. Zebrano około 800 próbek głosowych. W tej liczbie na każdą cyfrę przypadło, około 80 próbek. Każda próbka była osobnym plikiem WAVE. Próbki były monofoniczne, 32-bitowe i nagrane z częstotliwością 44100 Hz. Z taką liczbą nagrań, przystąpiono do dalszego procesu przygotowywania ich do dalszego użytku.

5.1 Poprawa próbek

Nagrane próbki miały pewne wady. Szczególnie problematyczne były fragmenty ciszy oraz szumy. Do odpowiedniego przetworzenia próbek, użyto darmowego narzędzia **Audacity**. Przy jego pomocy wycięto ciszę z początku i końca nagrań.

W dalszym ciągu rozwiązać należało problem szumów w niektórych nagraniach. W tym przypadku również przydatny był edytor **Audacity**. Wykorzystując opcję odszumiania, możliwe było poprawienie jakości próbek.

5.2 Zwiększenie ilości próbek

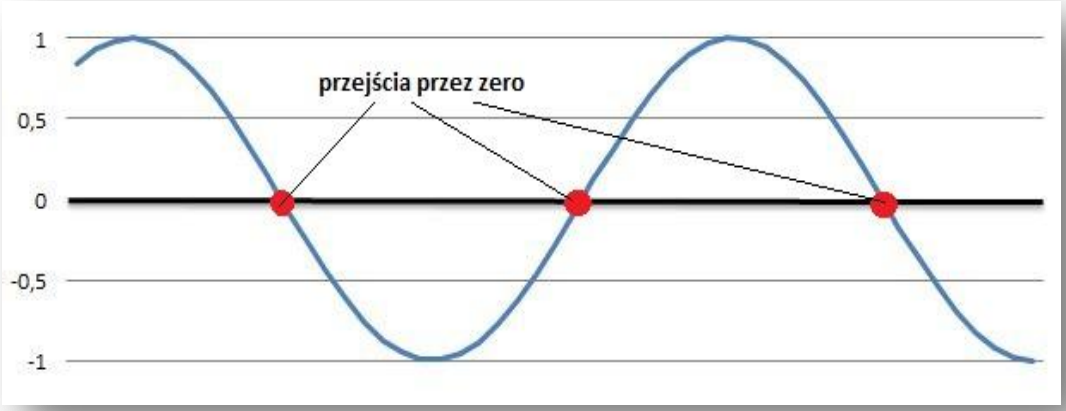
Liczba 80 początkowych próbek przypadających na każdą cyfrę, nie była wystarczająca. W celu otrzymaniu bardziej realnych wyników w późniejszym etapie pracy, konieczne było uzyskanie większej liczby nagrań. Z uwagi na trudności w uzyskaniu kolejnych nagrań od kolejnych osób, konieczne było użycie metod programistycznych. Użyto algorytmu, który zmodyfikował istniejące próbki. Dla pozyskania większej ilości próbek, przepuszczono je przez filtr preemfazy. Na skutek jego działania, górne pasma przenoszenia są wzmacniane. W rezultacie otrzymywany jest sygnał o mniejszej dynamice, stłumionych niższych częstotliwościach a podbitych wyższych [WWW1]. Z jego pomocą udało się podwoić liczbę nagrań i w dalszej pracy możliwe było wykorzystanie około 1600 próbek głosowych.

5.3 Zbadanie przebiegu fali

Próbki głosu, są zbyt różnorodne, aby możliwe było ich ocenianie na podstawie samego kształtu fali dźwiękowej. Dwa, na pozór zupełnie inne, przebiegi mogą reprezentować te same cyfry. Dlatego też konieczne jest wykorzystanie metod badania częstotliwości i określania na jej podstawie charakterystycznych cech próbki.. Możliwe jest wykorzystanie prostszych metod, takich jak badanie „przejść przez zero” („zero-crossing”).

5.3.1 Metoda zliczania „przejść przez zero”

Metoda polega na podziale próbki na zadaną liczbę przedziałów i zliczenia sytuacji w których wartość fali zmienia znak.



Rysunek 7. Badanie przejść przez zero

Często zdarza się, że tego typu badanie generuje błędy. Mogą one wynikać np. z szumów lub małej wartości wahań fali w pobliżu zera. W celu uniknięcia błędów wynikłych z tego powodu, konieczne jest modyfikowanie metody.

Pierwszą modyfikacją jest badanie przejść nie przez zero a przez wartość średnią próbki. Próbki mogą mieć wartości przesunięte względem zera. Dla skrajnie złego przypadku można otrzymać dla przebiegu fali brak przejść przez zero, podczas gdy dla innego bardzo podobnego przebiegu, ilość przejść przez zero jest prawidłowa.

.

Kolejnym problemem są wahania fali wokół średniej wartości. niedoskonałość metody można zrekompensować uwzględniając współczynnik redukcji **α**. Po jego uwzględnieniu, przejście przez zero jest zliczane jedynie w sytuacji, gdy wartość bezwzględna kolejnej wartości jest większa od współczynnika **α**.

W spółczynnik redukcji jest zadawany dla całego zestawu próbek. Jego uelastycznienie i dostosowanie do każdej próbki z osobna, jest możliwe poprzez korelację jego wartości z amplitudą fali.

6 Wybór technologii

6.1 Język programowania

Po za poznaniu się z tematem, pierwszą decyzją którą należało podjąć był dobór języka programowania i środowiska w którym wykonywana będzie praca. Rozważono kilka języków programowania jednak ostatecznie wybór padł na język C++. Dzięki swojemu stosunkowo długiemu istnieniu, posiada wiele gotowych bibliotek i implementacji algorytmów. Jego kolejną niewątpliwą zaletą jest wydajność. Szybkość wykonywania algorytmów w C++ jest w pesymistycznych przypadkach równa wykonaniu algorytmów w języku Java czy C#, a często jest większa. Istotny wpływ na wybór języka miała też liczba implementacji algorytmu NEAT w C++ . Postanowiono wykorzystać jedno z zintegrowanych środowisk programistycznych (IDE – Integrated Development Environment).

6.2 Środowisko pracy

Wybrano środowisko Microsoft Visual Studio 2010. Środowisko to zapewnia wygodną pracę i solidne nowoczesne kompilatory. Dodatkowym atutem była możliwość wykorzystania możliwości platformy .Net za pośrednictwem języka C++/CLI. Dodatki zawarte w tym rozszerzeniu języka dostarczanym przez Microsoft, wymagają użycia kodu zarządzanego. Istnieje jednak metoda łączenia kodu niezarządzanego z kodem zarządzanym. Przy użyciu flagi kompilacji mieszanej możliwe jest łączenie obu styli programowania. Zmniejsza ona w pewnym stopniu wydajność rozwiązania, ale na współczesnych maszynach nie jest to zbyt odczuwalne. Ponadto pisany program nie wymaga obróbki sygnału w czasie rzeczywistym. Wykorzystanie dodatków zarządzanych ułatwiło znacznie budowę interfejsu graficznego użytkownika, oraz pewne operacje systemowe.

6.3 Implementacje algorytmów i struktur

6.3.1 Implementacja sieci neuronowych

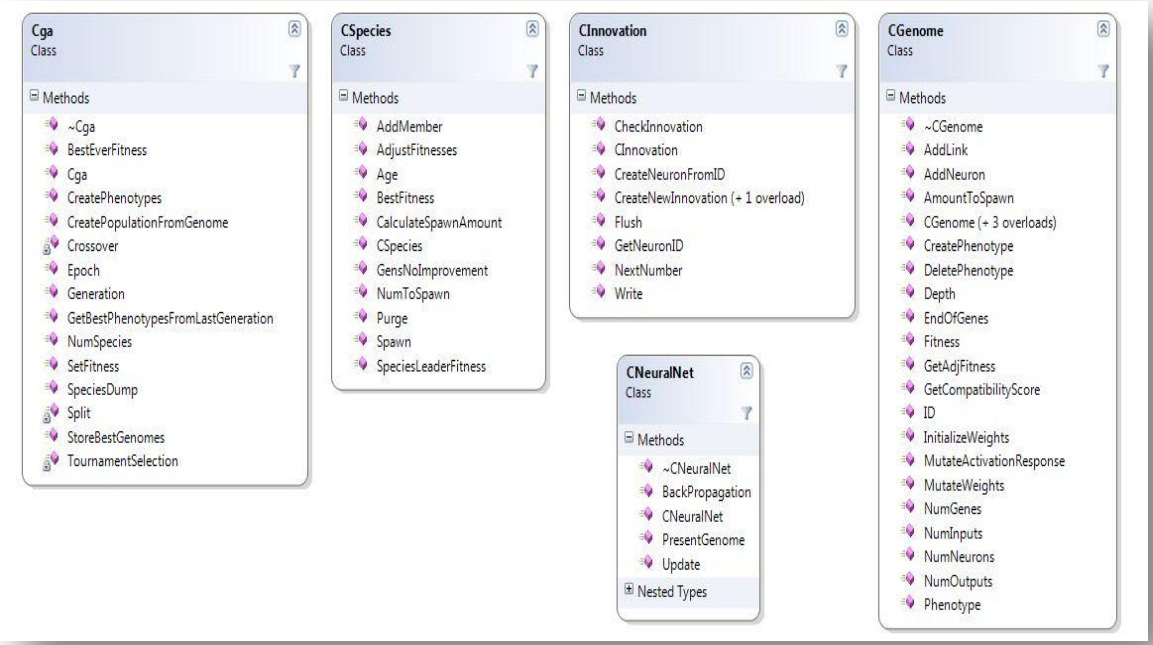
Wykorzystywane są 2 implementacje sieci. Pierwsza prostsza wykorzystywana przy algorytmie wstecznej propagacji jest oparta o tablice. Struktura sztucznej sieci neuronowej, po jej początkowym zadeklarowaniu, jest zawsze stała. Dlatego też aby zwiększyć szybkość przetwarzania, sieć oparta jest o tablice a dokładniej macierz 3-wymiarową. Każdy element macierzy zawiera wagę na wyjściach konkretnych neuronów. Dodatkowe tablice zawierają poprzednie wartości wag neuronów, wyjścia neuronów oraz wartości błędów na wyjściach.

Druga sieć neuronowa wykorzystywana w aplikacji to zmodyfikowana na potrzeby pracy, klasa używana w implementacji algorytmu N.E.A.T. Oparta jest o szablonową klasę wektora. Wykorzystanie takiej sieci jest dodatkowo uzasadnione jej dynamicznymi zmianami podczas działania algorytmu genetycznego.

6.3.2 Implementacja algorytmu NEAT

W pracy inżynierskiej wykorzystano implementację algorytmu NEAT stworzoną przez Mata Buckland'a [WWW3]. Oparł on swoje klasy o wzorcową implementację Stanley'a. Z uwagi, że kod pisany był pod środowiskiem Visual Studio 2005, możliwa była jego migracja i użycie w Aplikacji do przeprowadzania doświadczeń. Kod klas udostępniony jest na wolnej licencji GNU GPL.

Buckland w swojej przykładowej implementacji zajął się problemem omijania przeszkód, przez obiekty sterowane sieciami neuronowymi. Część klas dotycząca tych zagadnień została usunięta z projektu. Zlikwidowane zostały nadmiarowe funkcje służące do rysowania i obsługi okien WINAPI. Konieczne były też poprawki w wyrażeniach sterujących. Zastosowana przez autora składnia, generowała na używanym środowisku błędy kompilacji powodowane przez użycie zmiennej poza zakresem. .W miarę pracy zostały dodane brakujące zabezpieczania dotyczące, przekroczenia zakresu wektorów, których brakło w oryginalnej implementacji.



Rysunek 8. Najważniejsze klasy implementacji algorytmu NEAT

Implementacja oparta jest o kilka klas, z których najważniejszą jest klasa ***Cga***. Zawarte są w niej implementacje operacji genetycznych, metody do podziału populacji na gatunki oraz pobranie informacji o stanie algorytmu.

Metoda *Epoch()* wykorzystywana jest do tworzenia kolejnych pokoleń chromosomów. Jako parametr przekazuje się do niej wektor z wartościami funkcji dopasowania dla kolejnych chromosomów. Każde wywołanie metody zwraca wektor zawierający fenotypy w postaci stworzonych sieci neuronowych. Metoda *GetBestPhenotypeFromLatGeneration()* umożliwia otrzymanie *n* najlepszych sieci z ostatniego pokolenia. Parametr *n* tak jak i inne parametry (prawdopodobieństwa poszczególnych mutacji, krzyżowania, przetrwania określonych osobników z wcześniejszych generacji) ustalane są w klasie ***CParams***. Jest to klasa składająca się z statycznych pól i metod umożliwiających wczytanie parametrów z pliku.

Każdy neuron jest obiektem klasy ***SNeuron*** . Struktura ta posiada wektory połączeń przychodzących i wychodzących wraz z ich wagami, typ neuronu, jego identyfikator i wartość wyjścia.

6.4 Technologia użyta w GUI

Interfejs użytkownika został zrealizowany w technologii Microsoft WindowsForms. Do rysowania sieci, wykorzystano bibliotekę GDI+.

6.5 Program do plików z próbkami cyfr

Za pomocą prostej aplikacji napisanej w C# (z wykorzystaniem bibliotek .NET), możliwe jest wygenerowanie plików *.digits*, które zawierają przetworzoną treść próbek głosowych z wykorzystaniem opisywanego algorytmu. Program wywołuję się z konsoli poleceniem:

./transform.exe katalog_z_próbkami ilość_przedziałow α nazwa_pliku ,

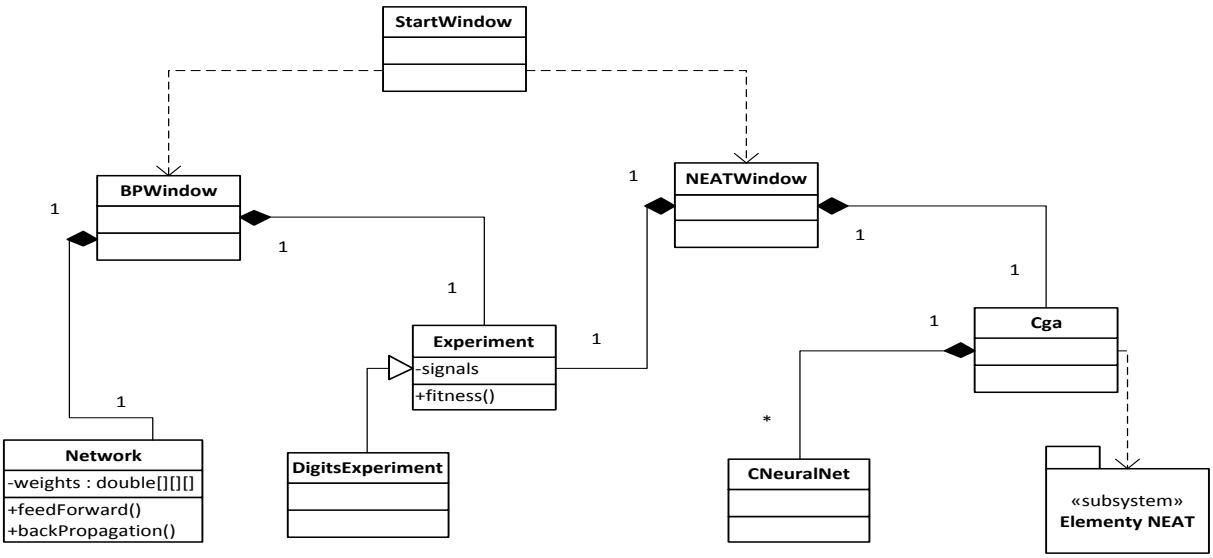
gdzie nazwa pliku, to nazwa pliku w którym zapisane zostaną wartości przejść przez zero dla próbek, a α to wartość współczynnika redukcji opisanego w rozdziale 5.

7 Aplikacja

W celu umożliwienia przeprowadzania badań związanych z wymienionymi wcześniej metodami nauki sieci, została stworzona aplikacja okienkowa.

7.1 Architektura

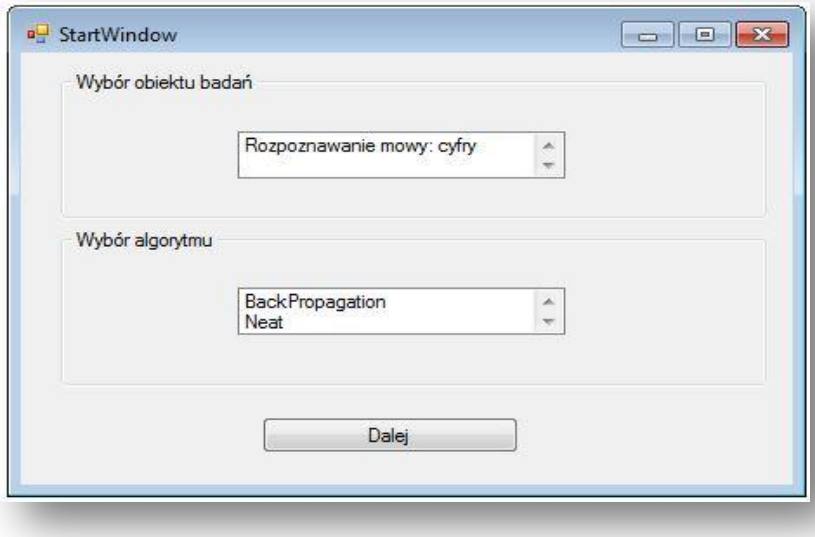
Program posiada dosyć prostą architekturę. Zbudowany jest z klas reprezentujących okna aplikacji i wykonujących operacje związane z logiką ich działania. Okna zbudowane są w oparciu o framework WindowsForms. Obiekty **Samples** reprezentują próbki cyfr. Klasa Network odpowiada za realizację zadań związanych z metodą wstecznej propagacji. Klasy **NeatOne** i **NeatTen** stanowią fasady dla metod wykonywanych przez klasę **Cga** i umożliwiają przeprowadzanie doświadczeń z neuroewolucją. Poza tym aplikacja posiada obiekty związane z użytą implementacją algorytmu NEAT.



Rysunek 9. Ogólny diagram klas aplikacji

7.2 Obsługa aplikacji.

Po uruchomieniu aplikacji, wyświetlane jest okno (rysunek 12) w którym wybieramy obiekt doświadczeń i używany algorytm. Na chwilę obecną wybór jest ograniczony do jednego obiektu badań. Możliwe jest eksperymentowanie tylko na próbkach głosowych. Wybieramy też jeden z dwóch algorytmów : wstecznej propagacji albo NEAT. Aby przejść dalej konieczne jest wybranie po jednym elemencie z każdej listy.



Rysunek 10. Wybór eksperymentu i algorytmu

7.2.1 Okno do algorytmu NEAT

Po wyborze algorytmu NEAT (rysunek 13), wyświetlone zostaje okno składające się z trzech części.

Sekcja **Parametry algorytmu**:

Eksperyment –umożliwia wybór pomiędzy dwoma dostępnymi doświadczeniami z NEAT. Są to rozpoznanie cyfry na podstawie próbki lub wykrywanie czy próbka zawiera zadaną cyfrę.

Rozmiar populacji - określa jaką liczbę chromosomów tworzymy dla kolejnych populacji.

Neurony max. – parametr wyznaczający maksymalną ilość neuronów, które mogą znaleźć się w sieci neuronowej.

Przetrwanie – prawdopodobieństwo przetrwania organizmu z poprzedniej generacji.

Nowe poł. i **Nowy węzeł** - prawdopodobieństwo mutacji polegającej na dodaniu połączenia lub dodania węzła sieci (neuronu).

Mutacja - prawdopodobieństwo mutacji wagi na połączeniach pomiędzy neuronami.

Zmiana wag – procentowy zakres zmian wagi podczas mutacji.

Krzyżowanie - prawdopodobieństwo, z którym dochodzi do operacji *crossing-over*, pomiędzy dwoma osobnikami populacji.

Max. liczba gatunków - maksymalna liczba gatunków jakie mogą zostać stworzone podczas ewolucji. Gatunki chronią różnorodność populacji.

Sekcja **Sterowanie**:

Ewolucja – uruchamia algorytm dla zadanej liczby pokoleń

Pokaż najlepszą sieć – wyświetla okno z najlepszą siecią i jej parametrami.

Cyfra – wybór konkretnej sieci rozpoznającej cyfrę. Dostępne tylko po wyborze odpowiedniej opcji na liście rozwijalnej **Eksperyment.**

Liczba pokoleń – określa ile kolejnych generacji ma zostać stworzonych po wciśnięciu przycisku **Ewolucja.**

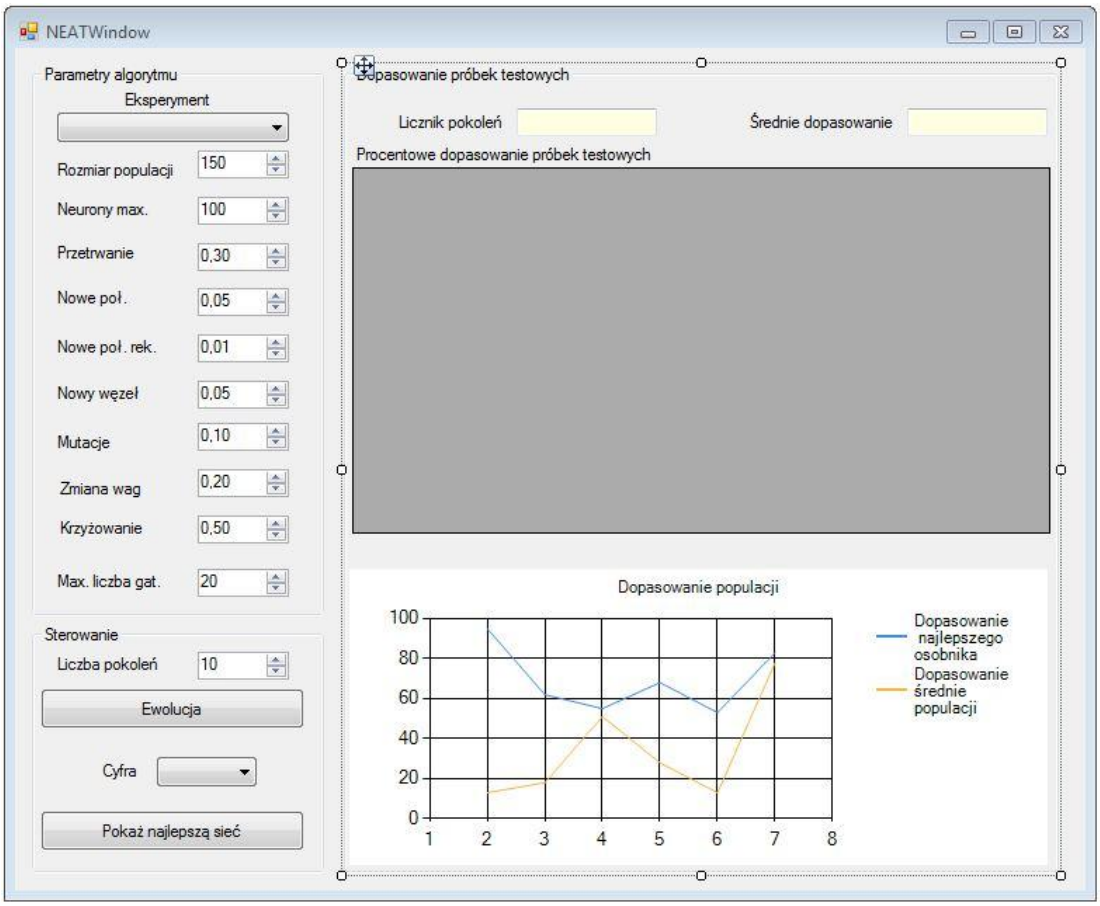
Sekcja **Dopasowanie**:

Licznik pokoleń – wyświetla liczbę stworzonych pokoleń.

Średnie dopasowanie – zawiera średnie dopasowanie sieci w nauce.

Procentowe dopasowanie próbek testowych – zawiera procentowe rozpoznanie cyfr zawartych w próbkach testowych. Jeżeli doświadczenie dotyczy rozpoznania cyfry na podstawie próbki, to wiersze reprezentują rzeczywiste wartości próbek a kolumny wartości rozpoznane. Gdy wybrano wykrywanie czy próbka zawiera zadaną cyfrę, kolumny reprezentują sieci mające rozpoznać kolejne cyfry.

Dopasowanie populacji – zawiera wykres z średnim dopasowaniem populacji i dopasowaniem jej najlepszego osobnika, podczas trwania eksperymentu. Dla drugiego typu doświadczenia wykres jest nieaktywny.



Rysunek 11. Okno z algorytmem NEAT

7.2.2 Okno do metody backpropagation

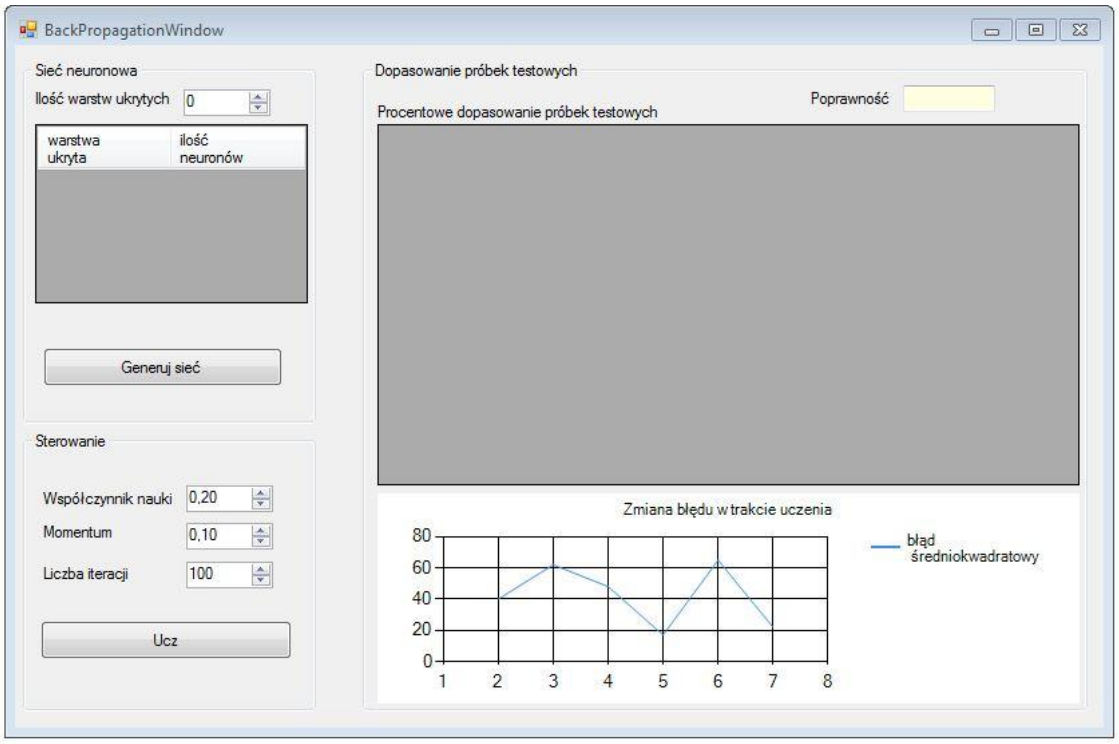
Jeżeli wybrany na początku został algorytm wstecznej propagacji, wyświetlane jest inne okno (rys. 14). Zawiera ono elementy przydatne przy wykonywaniu doświadczeń z nauką sieci. Składa się z trzech sekcji.

Sekcja **Sieć neuronowa**:

Ilość warstw ukrytych – umożliwia ustalenie ilości warstw ukrytych w sieci neuronowej.

Ilość neuronów – tabela umożliwiająca ustalenie liczby neuronów na każdej z warstw ukrytych.

Generuj sieć – tworzy nową sieć neuronową na podstawie tabeli **Ilość neuronów** i liczby przedziałów zadanych dla próbek głosowych.



Rysunek 12. Okno z algorytmem wstecznej propagacji

Sekcja **Sterowanie**:

Współczynnik nauki – daje możliwość określenie siły z jaką odbywać się będzie nauka algorytmu.

Momentum – umożliwia spowolnienie tempa nauki, ale chronię (w pewnym stopniu) sieć przed przeuczeniem.

Liczba iteracji – definiuję ile razy w każdym kroku zostanie powtórzona operacja wstecznej propagacji błędów.

Ucz – rozpoczyna naukę sieci. Po rozpoczęciu nauki, przycisk zmiana kolor na niebieski i zostaje opisany etykietą **Stop.** Nauka zatrzymana zostaje wraz z ponownym wciśnięciem przycisku.

Sekcja **Dopasowanie próbek testowych**:

Procentowe dopasowanie próbek testowych – tabela zawierająca procentowe rozpoznanie cyfr zawartych w próbkach testowych. Wiersze reprezentują rzeczywiste wartości cyfr, kolumny – wartość wykrytą przez sieć.

Poprawność – wyświetla współczynnik poprawności sieci na podstawie rozpoznania próbek testowych.

Zmiana błędu podczas nauki – wykres zmiany błędu sieci podczas nauki. Błąd to błąd średniokwadratowy obliczany podczas nauki.

8 Wyniki doświadczeń

Rezultatem działania aplikacji były wyniki doświadczeń dla prostego algorytmu wstecznej propagacji oraz dla algorytmu N.E.A.T. W bieżącym rozdziale przedstawione zostaną wyniki tych badań dla obu metod. Dokonana zostanie analiza i ocena rezultat działania algorytmu wykorzystującego neuroewolucję topologii sieci neuronowych.

8.1 Przedmiot badań

Badaniu podlegały sieci, mające poprawnie interpretować nagrane próbki cyfr w języku polskim. Przeprowadzone zostaną trzy doświadczenia.

W pierwszym badania dotyczyć będą rezultatów nauki metodą wstecznej propagacji błędów. Trenowane sieci mają poprawnie rozpoznawać jakie cyfry reprezentowane są przez nagrane próbki. Sieci będą posiadały regularną strukturę. Dowolny neuron z danej warstwy będzie łączył się z każdym z neuronów z warstw sąsiadujących.

Takie samo zadanie jak w doświadczeniu pierwszym będzie postawione przed sieciami w kolejnym badaniu. Będą one jednak trenowane za pomocą algorytmu NEAT. Topologia sieci będzie zmienna, jak wynika to z założeń metody.

Trzecie doświadczenie również przeprowadzone będzie przy pomocą algorytmu NEAT. Badane będą sieci, które rozpoznawać mają czy próbka zawiera, zadana odgórnie sieci, cyfrę.

8.2 Dane dla sieci

Próbki są zawarte w 2 plikach test.digits oraz teach.digits (próbki testowe i próbki do nauki). Pliki wygenerowane są na podstawie zebranych i obrobionych próbek (rozdział 5).

Dane w plikach wygenerowane zostały z pomocą algorytmu przejść przez zero z następującymi parametrami :

Wartość współczynnika alfa : 0,08

Ilość przedziałów : 20.

Zastosowana ilość przedziałów implikuje ilość wejść w używanych sieciach neuronowych. Jest ona równa 20. Każdy neuron warstwy wejściowej otrzyma na wejściu współczynnik przejścia przez zero w zadanym przedziale.

Nagrania cyfr podzielono na zbiór próbek uczących i zbiór próbek testowych Zgodnie z przyjętymi normami, baza próbek została podzielona w stosunku 2:1. W zbiorze przeznaczonym do nauki, znalazła się taka sama liczba wszystkich cyfr. Biorąc pod uwagę zebraną liczbę próbek, w zbiorze uczącym znalazło się około 100 próbek każdej cyfry.

Próbki uczące dobrane zostały losowo - jednorazowo dla wszystkich doświadczeń. Ich dobór jest wspólny dla wszystkich doświadczeń i obu metod. Dzięki temu na wyniki doświadczeń nie ma wpływu mniej lub bardziej korzystny dobór próbek do zbioru testowego.

8.3 Środowisko przeprowadzanych badań

Doświadczenia zostały wykonane na komputerze klasy PC o następującej konfiguracji:

- Intel Xeon 3440 2.53Ghz (4 rdzenie, 8 wątków)
- 8 GB RAM DDR3
- system Windows 7
- .NET 4.0

8.4 Wyniki dla metody wstecznej propagacji

Na potrzeby doświadczeń zbadano kilka struktur. Zostały przyjęte 4 przykładowe topologie. Wartość siły uczenia i momentum była zmieniana w czasie. Na początku, momentum ustalane było na 0, a siła uczenia na 0,25. W miarę kolejnych iteracji zwiększano momentum stopniowo do 0,1 . Zmniejszano również siłę uczenia. Przy ostatnich wykonywanych iteracjach, najczęściej siła nauki była równa 0,15. Działanie algorytmu wstecznej propagacji pokazuje, że zmiana parametrów w tych kierunkach, jest korzystna dla rezultatu.

Sieć posiada 20 wejść. Z uwagi na to że rozpoznajemy 10 cyfr, siec posiada 10 wyjść. Uznaje się że sieć uznaje próbkę za reprezentującą cyfrę **n**, jeżeli wartość **n-tego** wyjścia sieci jest większa od pozostałych.

8.4.1 Brak warstw ukrytych

Na początku rozpatrzono najprostszy model sieci. Liczbę iteracji w kroku ustalono na 1000 Parametry algorytmu zmieniane były tak jak opisane to zostało w poprzednim akapicie.

Algorytm działał do momentu w którym, błąd średniokwadratowy ustabilizował się na stałym poziomie. Dłuższe działanie algorytmu zaczęło powodować naukę sieci na pamięć i nie przynosiło korzystnych rezultatów. Rozpoznanie próbek testowych kształtowało się na poziomie 63% . Błąd średniokwadratowy wynosił w zaokrągleniu 0,27. Tabela 1 zawiera wyniki badań. Odpowiedź sieci zostaje uznana za poprawną, gdy sieć uznaje próbkę odpowiadającą cyfrze, którą ta rzeczywiście reprezentuje.

Cyfra	Procent poprawnych
0	36%
1	72%
2	48%
3	81%
4	65%
5	69%
6	81%
7	33%
8	84%
9	61%

Tabela 1. Wyniki dla sieci bez warstw ukrytych

8.4.2 Duża ilość neuronów w warstwach ukrytych

Dla porównania przetestowano działanie sieci dla dużej ilości warstw ukrytych. Stworzono 5 warstw ukrytych. W pierwszej i drugiej umieszczono po 10 neuronów, trzecia składała się z 5 neuronów. Wygenerowało to dosyć dużą i skomplikowana sieć. Spowodowało to stworzeni dużej ilości możliwych kombinacji wag. Rezultaty eksperymentu zawarte są w tabeli 2.

Zwiększył się około dwukrotnie czas w jakim sieć uzyskała swoje najlepsze wyniki. Poprawność wyniosła 68%. Błąd średniokwadratowy dla próbek uczących ustabilizował się na poziomie około 0,23

Cyfra	Procent poprawnych
0	58%
1	58%
2	41%
3	75%
4	67%
5	76%
6	78%
7	66%
8	80%
9	78%

Tabela 2. Wyniki dla sieci z dużą liczbą neuronów w warstwie ukrytej

8.4.3 Średnia ilość neuronów w warstwach ukrytych

W kolejnym doświadczeniu sprawdzono, jak uczenie metodą wstecznej propagacji przebiega dla sieci o bardziej umiarkowanej ilości neuronów w warstwie ukrytej w stosunku do doświadczenia z 25 neuronami warstwy ukrytej. Dla obrania optymalnej liczby neuronów w warstwie użyto wzoru $N_{ukryte} = \sqrt{N_{wejscie} \times N_{wyjscie}}$. Biorąc pod uwagę że liczbę neuronów na wyjściu i wejściu sieci, zaproponowano stworzenie dwóch warstw ukrytych z których każda posiadać będzie po siedem neuronów. Tabela 3 zawiera wyniki rozpoznawania mowy dla nauczonej sieci.

Cyfra	Procent poprawnych
0	50%
1	64%
2	64%
3	70%
4	65%
5	66%
6	82%
7	66%
8	83%
9	67%

Tabela 3. Wyniki dla wyśrodkowanej ilości neuronów w warstwach ukrytych

Czas nauki zwiększył się w porównaniu do sieci bez warstw ukrytych ale był znacząco mniejszy od czasu w jakim uczyła się sieć o 50 neuronach w warstwach ukrytych. Błąd średniokwadratowy próbek uczących ustabilizował się na poziomie 0,23. Poprawność wzrosła o 2% i wyniosła 65% poprawnych detekcji dla zbioru testowego.

8.4.4 Eksperymentalny dobór ilości neuronów

W kolejnych doświadczeniach eksperymentalnie zmieniano liczbę neuronów w warstwach ukrytych i liczbę tych warstw. Najlepszą wygenerowaną siecią była sieć z dwoma warstwami. W pierwszej znalazło się 10 neuronów, a w drugiej 8. Poprawność sieci wyniosła 70% . Kolejne próby z innymi topologiami sieci nie przyniosły lepszych rezultatów. Rozpoznawalność cyfr dla takiej sieci zaprezentowana jest w tabeli 4.

Cyfra	Procent poprawnych
0	59%
1	67%
2	62%
3	68%
4	60%
5	76%
6	78%
7	71%
8	81%
9	77%

Tabela 4. Rozpoznanie dla empirycznie dobranej ilości neuronów warstw ukrytych

8.4.5 Wnioski

Otrzymano dosyć przeciętne rezultaty. Prawdopodobnie przy innym, lepszym zbiorze próbek uczących i bardziej dokładnej metodzie badania przebiegu fali na próbkach dźwiękowych, można by osiągnąć jeszcze lepsze rezultaty.

Niedogodnością jest konieczność dopasowywania topologii sieci uczonej samodzielnie. Zbyt proste topologie charakteryzują się szybkim czasem nauki, ale ograniczonymi możliwościami wykrywania cyfr. Z kolei zbyt duże sieci mogą uczyć się dłużej a nie osiągać wystarczająco dobrych rezultatów. Konieczne jest znalezienie optimum. Wzory są pomocne tylko w określeniu pewnego początkowego punktu z którego można rozpocząć poszukiwania optymalnej topologii.

Po przeprowadzonych doświadczeniach widać też, że najtrudniejsza do rozpoznania jest cyfra 0. Wartości jej rozpoznania nie przekraczały 50 %. Najlepiej rozpoznawana była ósemka. Jej rozpoznawalność oscylowała wokół 80%.

8.5 Wyniki dla algorytmu NEAT

8.5.1 Sieć rozpoznająca jak cyfra jest reprezentowana przez probkę

Początkowa populacja składała się z neuronów o 20 wejściach i 10 wyjściach. Nie posiadały one warstw ukrytych. Dodatkowo w sieci umieszczany był neuron odpowiedzialny za bias.

Ustalono następujące parametry algorytmu:

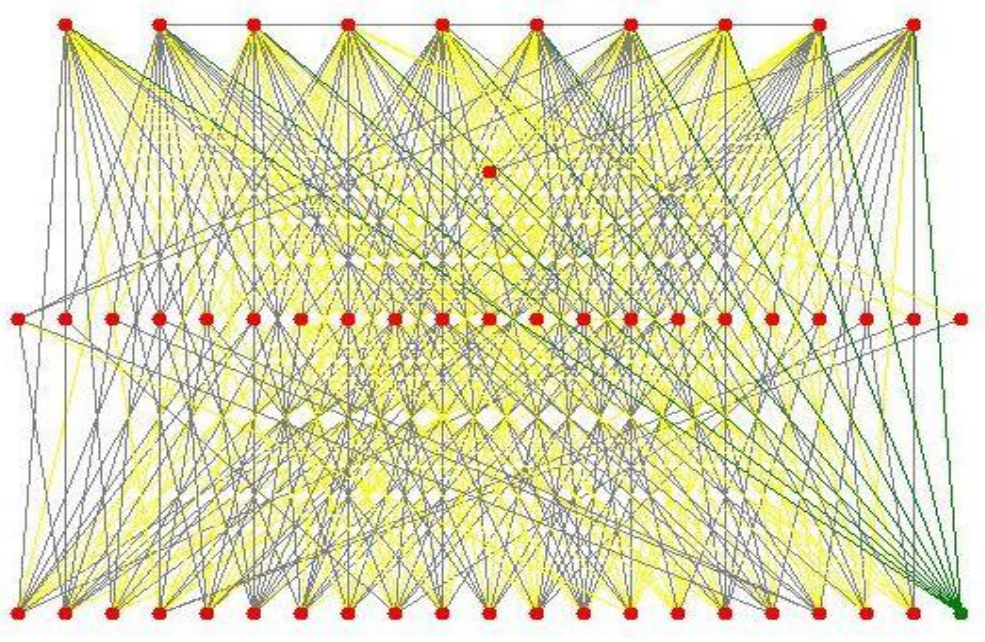
- Każda populacja algorytmu składa się ze 150 chromosomów
- Sieci tworzone przez algorytm mogą posiadać maksymalnie 100 neuronów
- Współczynnik przetrwania: 30%
- Prawdopodobieństwo dodania węzła : 5%
- Prawdopodobieństwo dodania zwykłego połączenia : 5%
- Prawdopodobieństwo połączenia zwrotnego: 1%
- Prawdopodobieństwo mutacji wagi: 10%
- Prawdopodobieństwo krzyżowania: 50%
- Wahania wag: 20% początkowej wartości
- Maksymalna liczba gatunków: 20

Z tak dobranymi parametrami rozpoczęto działanie algorytmu NEAT. Po 200 pokoleniach istniało 5 gatunków genotypów. Uzyskano skuteczność rozpoznania próbek testowych na poziomie 36%. Wyniki zawarte są w tabeli 5.

Cyfra	Procent poprawnych
0	0%
1	95%
2	0%
3	82%
4	27%
5	54%
6	45%
7	70%
8	0%
9	10%

Tabela 5. Ilość poprawnie rozpoznanych próbek głosowych przy pomocy najlepszej sieci otrzymanej po 200 generacjach

Sieć miała dość prostą budowę. Składała się z 4 warstw, w tym dwóch ukrytych. Warstwa ukryta posiadała 22 neurony w warstwie ukrytej. Topologia najlepszej sieci po 200 pokoleniach przedstawiona jest na rysunku 15. Na zielono zaznaczone są połączenia z neuronu biasu. Szarym kolorem zaznaczone są połączenia o wadze dodatniej, a na żółto - ujemnej. Ciekawe są połączenia pomiędzy neuronami warstwy wyjściowej.



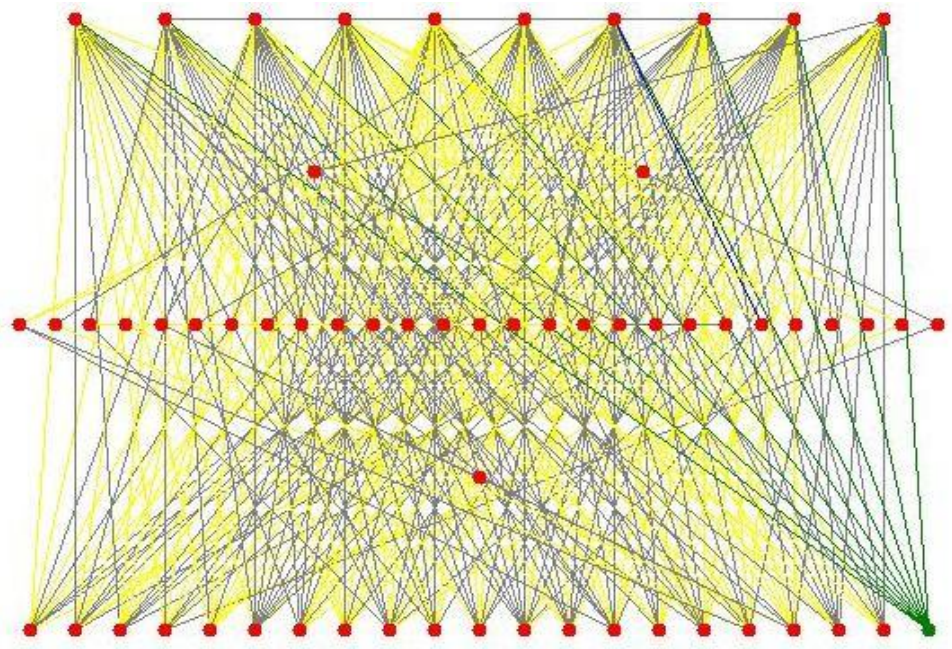
Rysunek 13. Topologia najlepszej sieci po 200 pokoleniach

Po 300 pokoleniach dokonano kolejnego przeglądu populacji. Istniało już 7 gatunków. Ilość rozpoznanych próbek wzrosła do 42% całego zbioru. Sieć nie rozpoznaje cyfr : 0,3 i 4. Widoczna jest jednak poprawa w rozpoznawaniu cyfr: 2,6,8 i 9.

Cyfra	Procent poprawnych
0	0%
1	30%
2	52%
3	0%
4	0%
5	46%
6	70%
7	56%
8	84%
9	86%

Tabela 6. Rozpoznanie cyfr po 300 generacjach algorytmu NEAT

Sieć nieco zmieniła budowę. Posiada teraz trzy warstwy. Najbardziej wewnętrzna posiada wciąż najwięcej neuronów. Warstwa najniższa ma 1 neuron a poprzedzająca wyjścia z sieci – dwa. Struktura zaprezentowana jest na rysunku 14.



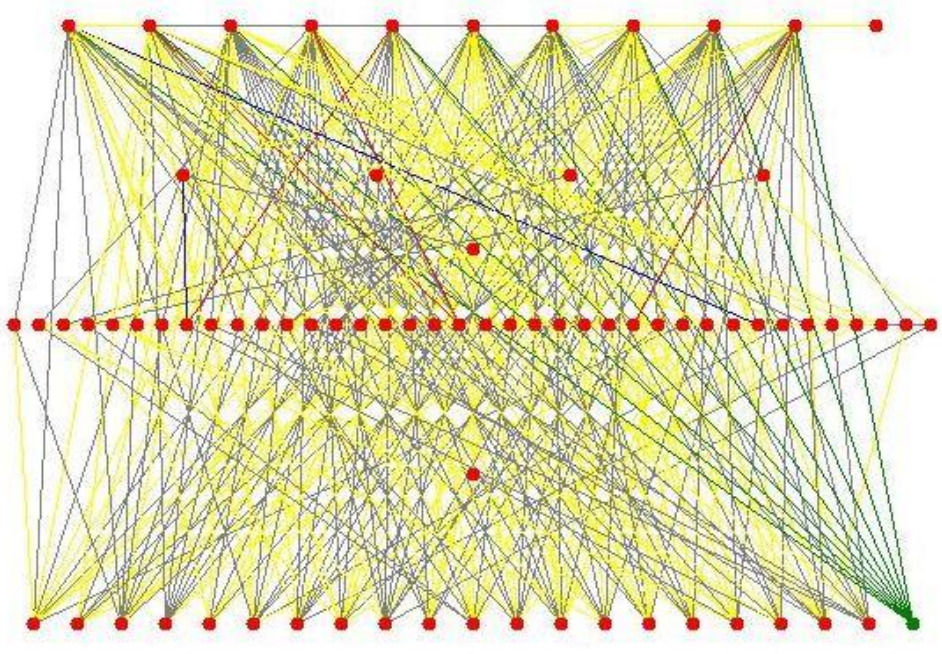
Rysunek 14. Topologia najlepszej sieci po 300 pokoleniach

Kolejną obserwację wykonano po 500 generacjach algorytmu. Zwiększyła się liczba gatunków – było ich 8. Współczynnik rozpoznawania cyfr wzrósł do 44% zbioru testowego. Wyniki znajdują się w tabeli 7.

Cyfra	Procent poprawnych
0	0%
1	0%
2	58%
3	64%
4	4%
5	60%
6	70%
7	74%
8	82%
9	30%

Tabela 7. Rozpoznanie cyfr po 500 generacjach algorytmu NEAT

Cyfry 0 i 1 nie są w żadnym stopniu wykrywane. Bardzo słabo (4%) rozpoznawana jest też liczba 4. Ogólna wykrywalność wrosła do poziomu 44%. Zmieniła się struktura sieci. Dodana została kolejna warstw ukryta w porównaniu do poprzedniego badania. Jest w niej tylko 1 neuron. Znacznie wzrosła liczba połączeń. W sieci pojawiły się połączenia zwrotne. Na niebiesko zaznaczone są połączenia zwrotne o wadze ujemnej a na czerwono – dodatniej. Topologię sieci po 500 pokoleniach przedstawia rysunek 15.



Rysunek 15. Topologia najlepszej sieci po 500 pokoleniach

Kolejne pokolenia nie zwiększały poziomu dopasowania.

8.5.2 Sieć odróżniająca zadaną cyfrę od pozostałych

Aby zbadać jak działa algorytm NEAT dla mniejszych sieci, przeprowadzono kolejne doświadczenie. Zbadano trenowanie sieci, mających rozpoznać, czy próbka reprezentuje określoną dla sieci cyfrę. Stworzono 10 populacji sieci neuronowych. Każda z populacji niezależnie od pozostałych miała rozróżniać czy próbka reprezentuje zadaną cyfrę czy też nie. Każda populacja rozróżniała inna cyfrę.

Początkowe populacje składały się z neuronów o 20 wejściach i 2 wyjściach. Nie posiadały one warstw ukrytych. Tak jak w poprzednim doświadczaniu w sieci umieszczany był neuron odpowiedzialny za bias.

Ustalono następujące parametry algorytmu:

- Każda populacja algorytmu składa się ze 100 chromosomów
- Sieci tworzone przez algorytm mogą posiadać maksymalnie 60 neuronów
- Współczynnik przetrwania: 30%
- Prawdopodobieństwo dodania węzła : 5%
- Prawdopodobieństwo dodania zwykłego połączenia : 5%
- Prawdopodobieństwo połączenia zwrotnego: 1%
- Prawdopodobieństwo mutacji wagi: 10%
- Prawdopodobieństwo krzyżowania: 50%
- Wahania wag: 20% początkowej wartości
- Maksymalna liczba gatunków: 10

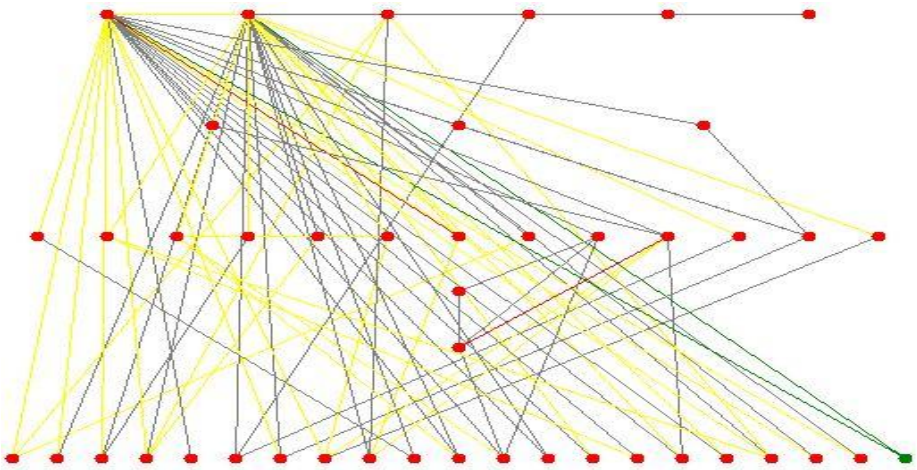
Zbadano rezultat działania algorytmu, po 500 pokoleniach. Większość sieci osiągała wysokie współczynniki dopasowania. Wyniki zaprezentowane są na rysunku. W kolumnach przedstawiane są wyniki dla kolejnych sieci. Wiersze reprezentują cyfry zapisane na próbkach.

Zadan	Roz.0	Roz.1	Roz.2	Roz.3	Roz.4	Roz.5	Roz.6	Roz.7	Roz.8	Roz.9
0	0,84	0,30	0,20	0,10	0,36	0,18	0,04	0,38	0,04	0,00
1	0,64	0,64	0,26	0,02	0,10	0,22	0,18	0,14	0,08	0,10
2	0,74	0,36	0,62	0,04	0,26	0,18	0,24	0,12	0,18	0,12
3	0,20	0,30	0,00	0,96	0,46	0,04	0,02	0,82	0,10	0,00
4	0,76	0,10	0,02	0,20	0,84	0,00	0,10	0,72	0,02	0,02
5	0,98	0,06	0,32	0,00	0,14	0,98	0,14	0,00	0,00	0,86
6	0,92	0,08	0,34	0,00	0,26	0,24	0,80	0,06	0,00	0,32
7	0,62	0,12	0,00	0,66	0,88	0,00	0,20	0,92	0,02	0,00
8	0,16	0,08	0,24	0,10	0,34	0,04	0,00	0,04	0,88	0,14
9	0,94	0,00	0,06	0,00	0,26	0,78	0,16	0,00	0,02	0,94

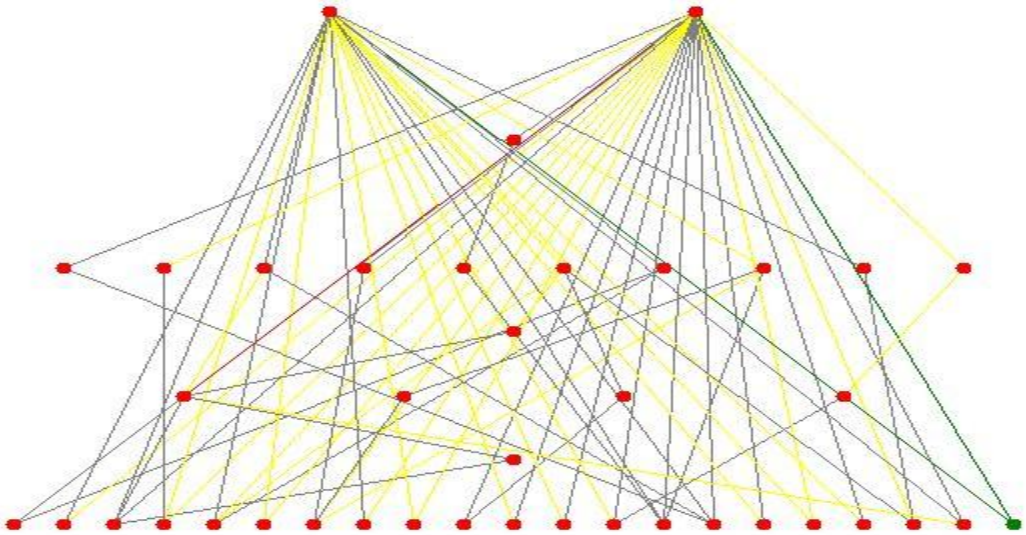
Rysunek 16. Wyniki najlepszych sieci po 500 pokoleniach

Jak widać większość sieci z wysokim prawdopodobieństwem uznają próbkę z określoną dla siebie cyfrą za „swoją”. Mylą jednak niektóre cyfry. Szczególnie złe wyniki generuje sieć dla cyfry „0”. Prawie wszystkie próbki uznaje za reprezentujące zero. Sieci odpowiedzialne za cyfry posiadają przeciętne wyniki. Najlepiej działa sieć wyszukująca cyfry „8”. Dosyć dobrze działa też sieć odpowiedzialna za cyfry „3”. Myli jednak je często z cyfrą „7”.

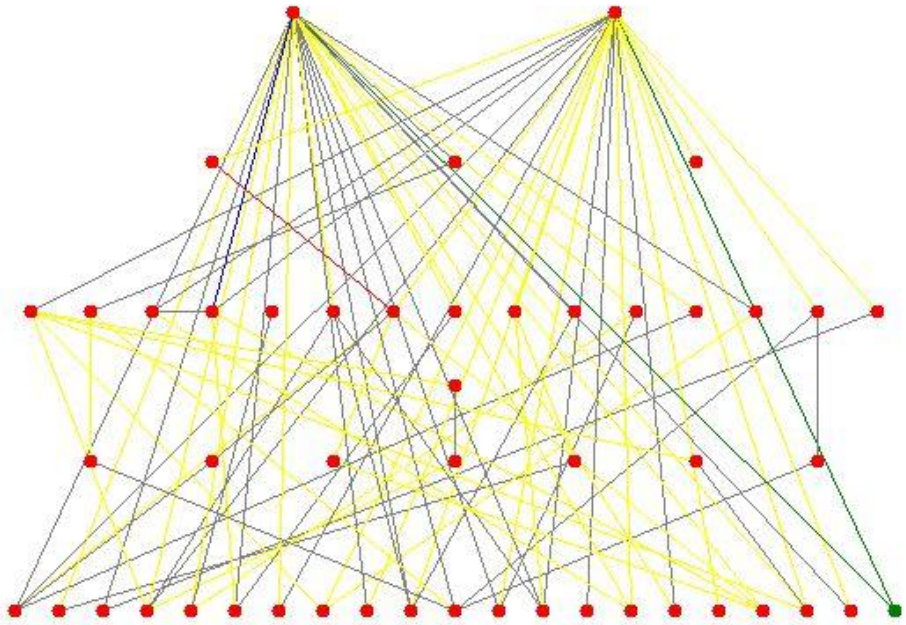
Na rysunkach 17, 18 i 19 przedstawiono trzy topologie powstałe po 500 pokoleniach. Reprezentują one kolejno najlepsze sieci do wykrywania cyfr „0”, „3” i „8”. Sieci znacznie różnią się od siebie. Najsłabiej wykonująca swoje zadanie sieć do rozpoznawania zer stworzyła najbardziej nietypową sieć z zbędnymi neuronami umieszczonymi w okolicach warstwy wyjściowej. Sieci o najlepszym współczynniku dopasowania posiadają bardziej ustrukturyzowane topologie.



Rysunek 17. Budowa sieci do rozpoznawania cyfry 0



Rysunek 18. Struktura sieci do rozpoznawania cyfry "3"



Rysunek 19. Topologia sieci do rozpoznawania cyfry "9"

8.6 Analiza wyników

Algorytm neuroewolucyjny stworzył sieci rozpoznające cyfrę na podstawie próbki o niższej skuteczności od, sieci trenowanych przez algorytm wstecznej propagacji. Wyniki osiągane przez analogiczne sieci różnią się o 24%.

Jednym z możliwych powodów takiej różnicy jest niedokładna funkcja dopasowania. W trakcie implementacji testowane były również inne sposoby oceniania dopasowania od użytego. Żadna jednak nie dała lepszych wyników rozpoznawania. Niektóre metody takie jak

błąd średniokwadratowy, sprawiały, że sieci tworzone przy pomocy algorytmu rozpoznawały poprawnie najczęściej 1/10 próbek testowych. Zastosowanie lepszej funkcji dopasowania mogłoby poprawić wyniki.

Kolejną możliwą przyczyną jest zbyt mały zbiór próbek uczących. Algorytm neuroewolucyjny może potrzebować większej liczby danych dla sensorów, aby było możliwe stworzenie odpowiedniego optimum dla sieci do rozpoznawania próbek głosowych.

Podczas drugiego doświadczenia z pojedynczymi sieciami, działanie algorytmu dawało dużo lepsze rezultaty. Sieci tworzone w kolejnych iteracjach osiągają coraz lepsze wartości funkcji dopasowania i bliskie są czasem osiągniecia maksimum.

We wszystkich doświadczeniach widać problemy z rozpoznawaniem próbek z „0". Algorytm uczący metodą *backproapgation* osiąga najsłabsze wyniki właśnie dla 0. Również wyniki doświadczeń z algorytmem neuroewolucyjnym wskazują na problemy właśnie z tą cyfrą. Przyczyną mogą być najsłabsze próbki, lub charakterystyka przebiegu fali dźwiękowej dla tej cyfry.

Podczas badania budowy najlepszych neuronów w populacjach algorytmu NEAT, widać pewne zależności. Wzrost efektywności rozpoznawania mowy wiąże się z dodawaniem nowych połączeń i węzłów. Sieci ewoluują w coraz bardziej skomplikowane struktury. Mimo możliwości powstania sieci o połączeniach zwrotnych, nie pojawiają się one podczas kolejnych pokoleń. Istnieje wiele gatunków, jednak mimo możliwości usuwania niedopasowanych gatunków, ich liczba rośnie wraz z kolejnymi generacjami. Najlepiej dopasowane neurony, w ciągu kilkudziesięciu populacji, należą do grupy kilku gatunków. Nie różnią się też zbytnio od siebie.

Podsumowanie

Celem pracy było stworzenie oprogramowania umożliwiającego przeprowadzenie doświadczeń z neuroewolucją sieci neuronowych. Program opierając się o zgromadzone próbki cyfr mówionych w języku polskim, generował sieci neuronowe o jak najoptymalniejszym działaniu. Dodatkowo, konieczne było przygotowanie zbioru próbek do doświadczeń. W rezultacie przedstawiono i przeanalizowano wyniki przeprowadzonych doświadczeń.

W pracy zostały zaprezentowane mechanizmy przydatne w rozpoznawaniu mowy. Przedstawiono działanie prostego algorytmu wstecznej propagacji oraz bardziej zaawansowane mechanizmy algorytmów genetycznych. Omówiono działanie algorytmu neuroewolucyjnego N.E.A.T. Za ich pomocą trenowano sieci neuronowe, tak aby osiągały jak najlepsze wyniki w rozpoznawaniu elementów ludzkiej mowy a w przypadku tej pracy inżynierskiej – polskich cyfr.

Zaproponowano dostosowanie algorytmu zliczania przejść przez zero, tak aby jak najlepiej reprezentował badane próbki mowy. W ramach przyszłego rozwoju programu, możliwe jest zastosowanie bardziej zaawansowanych technik takich jak wykorzystujące szybkie transformaty Furiera. Istotne jest też poszukiwanie innych lepszych metod oceny sieci.

Zaprezentowano i przeanalizowano wyniki doświadczeń przeprowadzonych zarówno prostą metodą wstecznej propagacji oraz z wykorzystaniem algorytmu N.E.A.T.. Jego wyniki bardzo znacząco odstawały od rezultatów, prostszej metody. Przedstawiono kilka możliwych wyjaśnień tego faktu. Przeprowadzono też doświadczenia na prostszych sieciach mających rozróżniać zadaną cyfrę od pozostałych. Wyniki były zachęcające.

Neuroewolucja w kontekście zmiennych topologii sztucznych sieci neuronowych jest złożonym ale i interesującym zagadnieniem. Jest godna zainteresowania w kontekście problemów w których nie jest znana optymalna topologia sieci. Warto w dalszym ciągu prowadzić nad nią badania i szukać miejsc, gdzie może znaleźć zastosowania.

Spis tabel i rysunków

Spis tabel

Spis rysunków

Bibliografia

[Goldberg 2003] Dawid E. Goldberg: *Algorytmy genetyczne i ich zastosowania*

[Tadeusiewicz 2007] R. Tadeusiewicz, T Gąciarz: *Odkrywanie właściwości sieci neuronowych*

[Stanley 1] Kenneth O. Stanley and Risto Miikkulainen: *Evolving Neural Networks Through Augmenting Topologies*

[Stanley 2] Kenneth O. Stanley and Risto Miikkulainen: *Efficient Reinforcement Learning Through Evolving Neural Network Topologies*

[Stanley 3] Kenneth O.Stanley, RISTO Miikkulainen.: *Real-Time Neuroevolution in the NERO Video Game*

[WWW1] http://www.koder.yoyo.pl/

[WWW2] http://en.wikipedia.org/

[WWW3] http://nn.cs.utexas.edu/