Łukasz Rychter
Maciej Sikora

10.06.2012r.

Zaawansowane programowanie w C++ (ZPR)

Dokumentacja końcowa

Projekt pod kierownictwem dr inż. Rafała Biedrzyckiego:

„Zaimplementować algorytm indukcji klasyfikatora bazujący na sieci neuronowej”

1. Wstęp

Aplikacja bazując na sieci neuronowej realizuje rozpoznawanie dźwięków z kilku kategorii na podstawie ich widma, tworzonego za pomocą szybkiej transformaty Fouriera (FFT).
Próbki dla kilku kategorii z zestawów uczących i weryfikujących dostarczane są jako posegregowane zbiory plików WAVE (*.wav).
Interfejs użytkownika ma postać linii poleceń. Użytkownik ma możliwość wskazania pliku z dźwiękiem do rozpoznania. Wyniki podawane są jako procentowe dopasowanie próbki do każdego z zestawu wyuczonych dźwięków.

2. Wybrany zestaw uczący

Sieć jest uczona rozpoznawania jednego z 6 dźwięków pustych strun gitary strojonej do dźwięku E (E, A, D, G, B, e1). W tym celu został przygotowany szereg nagrań w postaci plików WAVE każdego z tych dźwięków.
Aplikacja jest w stanie rozpoznawać również inne zestawy dźwięków, w tym innej ilości kategorii, łatwo rozróżnialne na podstawie ich widma. Jednakże należałoby stworzyć dodatkowe zbiory testujące. Nie powstały takie zbiory na potrzeby projektu.

3. Moduły aplikacji

Aplikacja składa się z 3 głównych części:

- Media – zbiór klas służących do przetwarzania danych audio
    - Moduł do wczytywania dźwięków z nieskompresowanych plików WAVE, zawierający prostą funkcjonalność dostosowującą dane w dowolnym formacie (częstotliwość próbkowania, liczba bitów na próbkę, stereo/mono) do wymaganego formatu. Interfejs pozwala dołączyć moduł wczytujący dane także z innych źródeł, np. mikrofonu.
    - Moduł wykonujący FFT.
    - Bufory

- Neur – zbiór klas odpowiedzialnych za sieć neuronową.
  - Moduł 3-warstwowej, jednokierunkowej sieci neuronowej, uczonej za pomocą propagacji wstecznej błędu.

- Learn – zbiór klas służących do przeprowadzenia nauki sieci neuronowej.

- Moduł interfejsu użytkownika, monitorujący przebieg uczenia się i prezentujący weryfikację próbek, z podawaniem wyników.

4. Cechy aplikacji

- Język implementacji: C++
- Zgodność z systemem Linux oraz Windows
- Narzędzie, które służy do automatyzacji budowania kodu - SCons
- Komentarze dokumentujące, zgodne z narzędziem Doxygen (możliwość zbudowania dokumentacji technicznej przy pomocy Doxygen)
- Wykorzystane biblioteki/kod boost oraz STL - testy jednostkowe `boost_unit_test_framework`, zarządzanie pamięcią (sprytne wskaźniki – `auto_pointer, shared_pointer`), kontenery STL (`map, list, vector`), wielowątkowość `boost_thread`
- Wykorzystanie innych bibliotek – FFTW: multiplatformowa biblioteka open-source do FFT (www.fftw.org)
- (Dodatkowo) Projekt znajduje się do pobrania (kod, zbiory testowe i wszystkie inne powiązane pliki) jako repozytorium GIT pod adresem https://code.google.com/p/zpr-rozpoznawanie-dzwieku

5. Wnioski/napotkane problemy w trakcie implementacji

- Klasa SimpeBuffer(sources/media/include/simpleBuffer.h), mogłaby nie być zaimplementowana przy pomocy szablonów oraz kontenerów STL – jest to implementacja mało wydajna jak na bufor danych.
- Kalkulacja FFT mogła by być szybsza dzięki wielokrotnemu użytkowaniu tych samych buforów i obiektów biblioteki.
- Sieć neuronowa – mimo poprawnej implementacji i działania – nie uczy się poprawnego rozpoznawania dźwięków. Wykazuje ona tendencję do reagowania na dowolne wejścia wyjściem o największym pobudzeniu dla ostatnio uczonego wzorca. Potrzeba wielu testów aby dostosować wszystkie parametry sieci do poprawnego działania. Być może konieczny byłby także preprocessing danych audio.
- Wysoce obiektowa implementacja sieci neuronowej – mimo iż czytelna i wygodna w użyciu jest mało wydajna pamięciowo i czasowo – należałoby szukać optymalizacji – zwłaszcza w realizacji połączeń między neuronami, których mogą być setki tysięcy.

6. Szczegółowa dokumentacja techniczna

Dokumentacja techniczna sporządzona została za pomocą programu Doxygen. Znaleźć ją można w doc/html/index.html