논문 2009-46CI-6-6

병렬 프로세서 기반의 패턴 분류 기법을 이용한 유전자 발현 데이터 분석

(Gene Expression Data Analysis Using Parallel Processor based Pattern Classification Method)

최 선 욱*, 이 종 호**

(Sun-Wook Choi and Chong Ho Lee)

요 약

최근 활발히 연구가 진행 중인 마이크로어레이로부터 얻어지는 유전자 발현 데이터를 이용한 질병 진단은, 데이터를 직접적으로 분석하기 힘들기 때문에 일반적으로 기계 학습 알고리즘을 사용하여 이루어져왔다. 그러나 유전자 발현 데이터를 분석함에 있어서 유전자들 간의 상호작용을 고려하는 분석이 필요하다는 최근의 연구 결과들은 기존 기계 학습 알고리즘들을 이용한 분석에 한계가 있음을 의미한다고 볼 수 있다. 본 논문에서는 특징들 사이의 고차원 상관관계를 고려 가능한 하이퍼네트워크 모델을 이용하여 유전자 발현 데이터의 분류를 수행하고 기존의 기계 학습 알고리즘들과 분류 성능을 비교한다. 또한 기존 하이퍼네트워크 모델의 단점을 개선 한 모델을 제안하고, 이를 병렬 프로세서 상에서 구현하여 처리 성능을 비교한다. 실험 결과, 제안 된 모델은 기존의 기계 학습 방법들과의 비교에서도 경쟁력 있는 분류 성능을 보여주었고, 기존 하이퍼네트워크 모델 보다 안정적이고 향상된 분류 성능을 보여주었다. 또한 이를 병렬 프로세서 상에서 구현 할 경우 처리 성능을 극대화 할 수 있음을 보였다.

Abstract

Diagnosis of diseases using gene expression data obtained from microarray chip is an active research area recently. It has been done by general machine learning algorithms, because it is difficult to analyze directly. However, recent research results about the analysis based on the interaction between genes is essential for the gene expression analysis, which means the analysis using the traditional machine learning algorithms has limitations. In this paper, we classify the gene expression data using the hypernetwork model that considers the higher-order correlations between the features, and then compares the classification accuracies. And also, we present the new hypernetwork model that improve the disadvantage of existing model, and compare the processing performances of the existing hypernetwork model based on general sequential processor and the improved hypernetwork model implemented on parallel processors. In the experimental results, we show that the performance of our model shows improved and competitive classification performance than traditional machine learning methods, as well as, the existing hypernetwork model. We show that the performance is maximized when the hyperntwork model is implemented on our parallel processors.

Keywords: hypernetwork model, microarray, gene expression, pattern classification, parallel processing

I. 서 론

생체 내에서 유전자가 mRNA의 형태로 나타나는 현상을 유전자 발현(gene expression)이라 한다. 이러한 유전자 발현은 항상 일어나는 것이 아니고, 생체의 생명현상을 유지해야 하거나 조절이 필요한 상황에서 이를 위한 단백질을 만들기 위해 발현하는 것이다. 이러한 유전자 발현 양상은 개별적으로 발생하는 것이 아니고 여러 유전자들 간의 상호작용으로 인해 발생하므로 특정 유전자의 발현으로 인해 다른 유전자의 발현이 촉

* 학생회원, ** 평생회원, 인하대학교 정보공학과
(Dept. of Information Engineering, Inha University)
※ 이 논문은 인하대학교의 지원에 의하여 연구되었음.
접수일자: 2009년9월28일, 수정완료일: 2009년11월2일

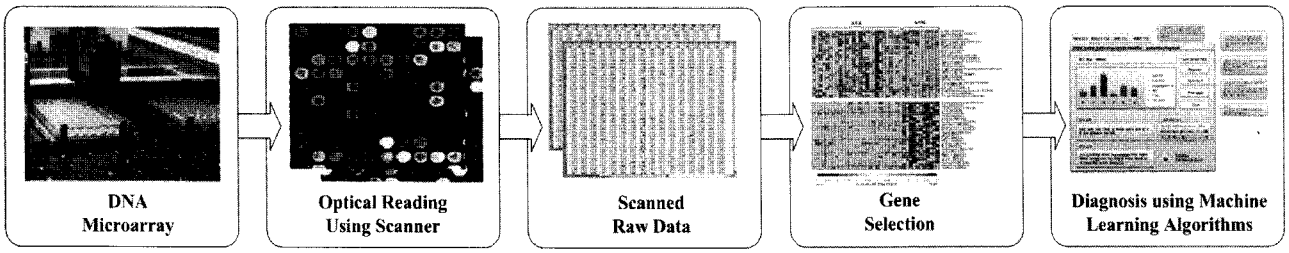


그림 1. 마이크로어레이로부터 얻어진 유전자 발현 데이터를 분석하기 위한 절차
Fig. 1. Procedure of analysis of gene expression data obtained from microarray.

진 되거나 억제 될 수도 있다[1]. 따라서 생체 내에서 발생하는 유전자들 간의 상호작용을 기반으로 하는 생물학적 메커니즘을 이해하기 위해서는, 그와 관련된 모든 유전자들을 동시에 조사하는 것이 가능해야 한다.

최근 분자 생물학과 공학 기술의 발달로 기존 유전자 발현 연구의 한계에서 벗어나 한 번의 실험으로 수천에서 수만 여개의 유전자 발현을 동시에 측정 하는 것을 가능하게 해주는 DNA 마이크로어레이(Microarray)가 개발 되었고, 이를 이용한 연구들이 활발히 진행 중에 있다[2]. 이와 관련 하여, 생체 내에서 발생하는 암이나 유전병과 같은 질병들 또한 특정 유전자들에 생긴 돌연변이에 등에 의해 유발 되므로, 마이크로어레이를 통해 얻어진 유전자 발현 정보를 이용하여 관련 유전자들의 발현 양상을 살펴, 질병을 진단하거나 질병과 관련 된 유전자들을 찾아내는 연구 또한 활발히 진행되고 있다.

앞서 살펴보았듯이 유전자의 발현 양상은 유전자들 간의 상호작용을 통하여 나타남으로, 유전자 발현 데이터를 이용한 질병 진단이나 관련 유전자 탐색에는 이러한 유전자들 간의 상호작용을 효과적으로 고려 할 수 있는 분석 방법이 필수적이다. 그러나 지금까지의 유전자 발현 데이터를 이용한 분석 방법들은 일반적으로 SVM, 신경망, 결정 트리, k-NN 등과 같은 기계 학습 알고리즘들을 사용해 왔다. 이들 방법들은 유전자들 간의 상호작용을 고려하지 못하고, 개별 유전자들을 기반으로 하여 고차원 공간상에 사상시키거나, 결정 경계를 이용한 분류 방법들이기 때문에 유전자들 간의 상호 작용을 고려하는 분석에 적합하지 않은 방법들이라고 할 수 있다.

최근 Zhang 등은 하이퍼그래프(Hypergraph)를 기반으로 특징들 간의 고차원 상관관계를 고려 가능한 기계학습 알고리즘인 하이퍼네트워크(Hypernetwork) 모델을 제안 하였다[3]. 하이퍼네트워크는 학습 데이터들로부터 확률적으로 추출 된 하이퍼에지(Hyperedge)들을 기반으로 구성 된 확률 라이브러리를 이용하여 분류를 수행하며, 필기체 숫자 인식, 얼굴 인식, 유전자 발현 데이터 분류 등 다양한 영역에 효과적으로 적용되었고, 기존의 기계 학습 알고리즘들과 비교하여 경쟁력 있는 분류 성능을 보여주었다. 이러한 하이퍼네트워크 모델을 유전자 발현 데이터 분류에 적용 할 경우 분류 성능에서의 장점뿐만 아니라, 분류 과정에서 매칭 된 하이퍼에지들을 분석하여 기존의 기계 학습 알고리즘들에서 얻기 힘들었던 유전자들 간의 상호 작용을 분석 하는 것에도 이용 할 수 있다. 그러나 하이퍼네트워크 모델은 분류를 수행하는데 있어, 학습 데이터들로부터 추출 된 다수의 하이퍼에지들과 입력 데이터 간의 비교를 반복적으로 수행하는 방법을 사용하기 때문에 많은 연산 시간을 필요로 하게 된다.

본 논문에서는 특징들 사이의 고차원 상관관계를 고려 가능한 하이퍼네트워크 모델을 이용하여 유전자 발현 데이터를 이용한 멀티클래스 암 분류에 적용하고, 기존의 기계 학습 알고리즘들과 분류 성능을 비교 분석해 보고자 한다. 또한 기존 하이퍼네트워크 모델의 단점을 개선 한 모델을 제안하고, 이를 FPGA, GPU와 같은 병렬 프로세서 상에서 구현하여 처리 성능을 극대화 할 수 있음을 보이고자 한다.

Ⅱ. 관련 연구

1. DNA 마이크로어레이

DNA 마이크로어레이는 수 ㎠ 정도의 좁은 칩 표면에 수천 개 이상의 프로브(probe)라고도 하는 유전자 서열을 고정시켜 만들어지며, cDNA 칩과 Oligonucleotide 칩의 두 가지 종류 가 있다. 두 칩은 서로 다른 특성들을 가지고 있지만, 표본과 프로브 유전자의 결합 반응을 이용한다는 기본 원리는 동일하다고 할 수 있다.

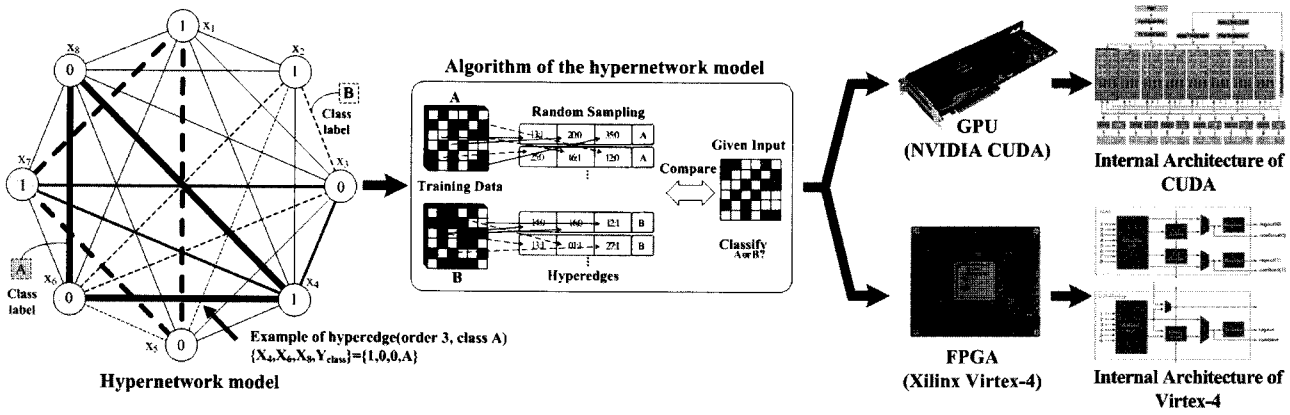


그림 2. 병렬 프로세서 상에서 구현 된 하이퍼네트워크 모델
Fig. 2. Hypernetwork model implemented on Parallel Processor.

또한 각 칩들마다 유전자 발현 양상을 측정하는 다양한 방식들을 갖고 있지만, 일반적으로 널리 이용되는 방식은 그림 1에서와 같이 형광물질에 의해 만들어지는 색을 통해 발현 정도를 나타내는 방식으로, 광학 스캐너를 통하여 읽어 드리는 방식이다. 이를 이용하여 시각적으로 수 천~수 만 개의 유전자의 발현 정도를 한 번에 확인 할 수 있고, 이를 수치화 할 수 있으므로 컴퓨터를 이용한 분석이 가능하게 된다.

DNA 마이크로어레이로부터 얻어진 유전자 발현 데이터를 이용하여, Golub 등은 유전자 선택 기법으로써 Signal-to-noise 방법을 제안하고, 백혈병의 2가지 서브타입을 분류하는데 적용하였다[4]. Khan 등은 PCA와 신경망을 이용하여 SRBCT의 4가지 서브타입을 분류하는데 적용하였다[5]. 이밖에도 대용량의 유전자 발현 데이터 분석을 통한 질병 진단에 대한 연구가 활발히 이루어지고 있다.

2. 병렬 프로세서의 이용

다양한 매체들로부터 쏟아지고 있는 대용량의 데이터들과 수많은 테스크(task)들을 동시에 관리하고 처리해야하는 최근의 컴퓨팅 상황에서, 기존의 단일 프로세서를 통한 연산은 한계에 부딪치게 되었고, 이들의 업무를 분담시키기 위해 병렬 프로세서를 이용한 보조 혹은 전용의 연산 방법들이 각광을 받고 있다. 지금까지 일반적으로 사용되어 왔던 CPU 또한 멀티 코어의 형태로 발전하고 있으며, 부하가 많이 걸리는 테스크를 전용의 하드웨어를 이용하여 처리하는 사례는 더 또한 새로운 일이 아니발전 할 수 있다. 따라서 이들 병렬 프로세서의 풍부한 연산 자원을 활용하여 이를 마이크로어레이로부터 얻어지는 대용량의 데이터를 효과적으로 처리 할 수 있는 방법들에 대한 연구가 필요한 시점이라고 할 수 있다.

이들 병렬 프로세싱의 용도로 사용가능한 플랫폼들 중, FPGA는 사용자의 필요에 따라 설계 변경이 가능한 하드웨어로서, 연산 모듈들을 병렬로 배치하여 대규모의 병렬 연산이 가능하고, 해당 작업에 따라 최적화 된 설계가 가능하므로 범용의 프로세서와 비교하여 고속의 연산이 가능하다. 이러한 장점들로 인해 FPGA를 고속의 처리 속도를 필요로 하는 디지털 필터, 패턴 인식 등의 신호처리에 적용하는 연구들이 진행 되어 왔다. 특히 패턴 인식 문제에 대해, Wee 등은 FPGA를 이용하여 전용의 SVM 하드웨어, 신경망 하드웨어 등을 설계하였고, 이를 실제의 문제에도 성공적으로 적용 한바 있다[6]. Pournara 등은 FPGA를 이용하여 베이지안 학습 기반의 유전자 조절 네트워크 탐색을 위한 하드웨어를 구현 하였다[7].

또한 컴퓨터 상에서 3차원 그래픽 처리를 위해 사용되었던 GPU(Graphics Processing Unit)를 이용한 병렬처리 방법 또한 최근 주목 받고 있다. 반도체 공정의 발달과 GPU 설계 기술의 급격한 발달로 GPU의 부동소수점 연산속도는 이미 CPU의 연산속도를 넘어선 상태로써, 그동안의 3차원 그래픽 처리를 위한 CPU의 보조 프로세서로의 용도가 아닌, 일반적인 프로세싱에 사용하고자하는 GPGPU(General-Purpose GPU)로의 연구가 활발히 진행 중에 있다. 이와 더불어 NVIDIA에서 발표한 CUDA는 그 동안 GPU 사용의 걸림돌이 되어 왔던 쉐이딩 언어와 그래픽 렌더링 파이프라인에 대한 이해가 없이도 GPU의 연산 능력을 이용 할 수 있도록

하는 관련 라이브러리를 제공함으로써 더욱 편리한 개발이 가능하게 되었다. 이들 GPU의 병렬 연산 능력을 이용하여 Chang 등은 유전자 발현 데이터의 계층적 클러스트링을 위한 클러스터 내의 특징들 간 거리들의 고속 연산에 적용하였다[8]. 또한 Manavski 등은 시퀀스 정렬을 위한 Smith-Waterman 알고리즘을 GPU를 이용하여 고속으로 구현하였다[9].

Ⅲ. 하이퍼네트워크 모델

1. 하이퍼네트워크 모델

하이퍼네트워크는 그래피컬 모델의 일종으로써 하이퍼그래프(hypergraph)와 유사하게 설명 되어 질 수 있다[3]. 하이퍼그래프는 에지(edge)가 널(null) 값을 가지지 않는 정점들로 연결 된 비방향성 그래프 $G=(V,E)$ 이다. 여기에 가중치 W를 추가하여 하이퍼네트워크 $H=(V,L,W)$ 로 나타낼 수 있는데, V는 특징(feature)들의 집합을 나타내고, L은 하이퍼에지들의 집합으로써, 하이퍼네트워크를 구성하는 라이브러리를 나타낸다.

하이퍼에지는 하나 이상의 특징들을 포함할 수 있는 에지로써 중복을 허용한다. 라이브러리 L을 구성하는 각 하이퍼에지 $l_i=\{l_{i_1},l_{i_2},...,l_{i_j},y_i\}$는 n개의 특징과 클래스 레이블(label)로 구성 되고, j는 특징 공간의 인덱스 값으로써, $1\le j\le k$의 값을 갖고, 여기서 k는 특징 공간의 차원(dimension)을 의미한다.

하이퍼에지의 가중치 W는 라이브러리 L 내에서 동일한 구성요소를 갖는 하이퍼에지들의 수로써 표현가능하고, 하이퍼에지가 포함하는 특징들의 수는 차수(order)라고 한다.

라이브러리 L을 구성하는 하이퍼에지들을 얻기 위해서는 특징 공간상의 값들을 랜덤 샘플링하는 과정을 거치게 되는데, 일반적으로 이 과정은 균등한 분포를 갖는 난수 생성기를 통하여 얻어진 난수들을 이용하여 차수 k 만큼의 가로 크기를 갖는 배열들을 생성하여 저장한다. 이들 배열을 랜덤 마스크라 부른다. 하이퍼에지와 입력 데이터를 비교하는 과정은 랜덤 마스크로부터 얻어진 인덱스 값들을 통해 특징 공간상의 해당 위치의 값들을 반복적으로 비교하며 이루어진다.

하이퍼네트워크 모델을 이용한 패턴 분류는 테스트 데이터가 주어졌을 때, 각각의 클래스 레이블들에 대한

표 1. 하이퍼네트워크 모델의 분류 과정
Table 1. Classification procedure of the hypernetwork model.

1. 라이브러리 L을 학습 데이터로부터 랜덤 샘플링을 통해 추출 된 하이퍼에지들을 이용하여 구성
   - L은 입력 패턴 x와 출력 클래스 Y의 결합확률 $P(x,Y)$를 경험적 확률 분포로 나타냄
2. 테스트 할 입력 패턴 x가 주어짐
3. 입력 패턴 x와 매치 되는 모든 하이퍼에지를 라이브러리 L로부터 추출
4. 추출 된 하이퍼에지들의 클래스 레이블을 해당 M^n에 각각 카운트
   - 클래스 레이블이 $Y=0$ 인 경우 M^0로 카운트
   - 클래스 레이블이 $Y=1$ 인 경우 M^1로 카운트
   . . .
   - 클래스 레이블이 $Y=n$ 인 경우 M^n로 카운트
5. 가장 많은 수가 카운트 된 클래스를 입력 패턴 x의 클래스로 분류

조건부 확률로서 설명 할 수 있다[10]. 테스트 데이터 x가 주어진다면, 예측 된 클래스 y^*는 각각의 클래스의 조건부 확률의 계산에 의해 예측되어 진다. 이 때 선택되어지는 클래스는 식 1에서와 같이 가장 높은 조건부 확률값을 갖는 클래스의 레이블 값이다.

$$y^* = \underset{Y\in\{0,1,...,n\}}{\arg max}\ P(Y|x)=\frac{P(Y,x)}{P(x)} \tag{1}$$

여기서 $P(Y,x)=P(Y|x)P(x)$이고, Y는 후보 클래스를 나타낸다. 하이퍼네트워크 모델을 이용하여 입력 데이터의 클래스를 예측하기 위한 과정은 다음 표 1과 같이 요약 할 수 있다.

위의 단계 3 에서는 입력 데이터 x와 매치 되는 모든 하이퍼에지를 추출하여, 각 클래스 별로 추출 되는 수를 M^n에 카운트 한다. 이를 통해 입력 데이터와 매치되는 하이퍼에지를 발견할 확률을 구할 수 있다.

$$\frac{c(x)}{|L|}=\frac{|M|}{|L|}\approx P(x) \tag{2}$$

단계 4 에서는 입력 데이터 x가 주어졌을 때, 이에 대한 각 클래스에서의 발생 정도를 나타내는 $c(Y|x)$

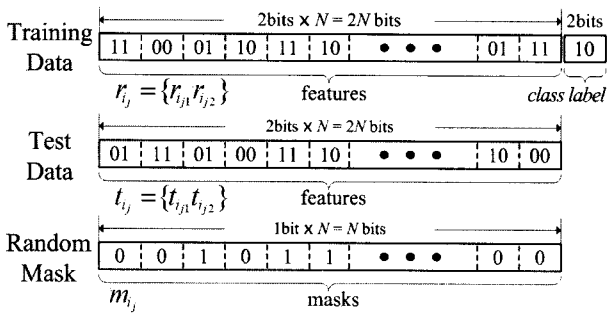


그림 3. 제안 된 멀티 레벨 데이터의 구조 (2비트)
Fig. 3. Structure of the proposed multi-level data (2bit).

를 구한다. 이 값을 이용해 입력 데이터가 주어졌을 때 예측 될 클래스 레이블들에 대한 조건부 확률인 사후 확률을 아래와 같이 근사할 수 있다.

$$\frac{c(Y|x)}{|M|}=\frac{|M^Y|}{|M|}\approx P(Y|x) \tag{3}$$

단계 5 에서는 가장 많은 수가 카운트 된 클래스를 입력 데이터의 클래스로 예측하게 되는데, 이는 식 1의 조건부 확률이 최대가 되게 하는 클래스 레이블을 찾는 것과 같다고 생각 할 수 있다. 더 자세한 이론적 배경은 다음에서 참조 할 수 있다[3, 10].

2. 개선 된 하이퍼네트워크 모델

서론에서 언급했던 것과 같이, 앞서 설명 된 하이퍼네트워크 모델은 경쟁력 있는 분류 성능과 특징들 간의 고차원 상관관계를 고려할 수 있는 분류 모델이라는 장점에도 불구하고, 반복적인 랜덤 샘플링과 비교 연산을 수행해야하기 때문에 연산 시간의 증가로 인해 실제 문제에 적용하는 것에 한계가 존재 한다. 이 연산의 시간 복잡도는 $O(N_r \cdot N_h \cdot N_o)$ 로써 학습 데이터의 수(N_r)가 증가 되거나, 비교 되는 하이퍼에지의 수(N_h) 그리고 하이퍼에지의 차수(N_o)가 증가 할수록 급격한 연산 시간의 증가를 가져온다. 또한 기존의 하이퍼네트워크 모델에서는 하이퍼에지들과 입력 데이터의 비교에 있어서 이진화(binaryzation) 된 데이터들만을 처리 가능했었다. 즉, 각 특징 값들을 전처리 과정을 통해 이진화 하고, 상응하는 각 특징 값들이 동일 할 경우 매칭 되는 것으로 처리 하는 방식이었다. 그러나 데이터의 이진화는 필연적으로 정보의 손실을 발생시킨다. 또한 랜덤 샘플링을 통하여 구성 된 라이브러리를 사용함으로써 분류기의 분류 정확도는 큰 분산을 갖게 되고, 이는 분류 결과와 분류기의 신뢰성을 떨어트리는 결과를 초래하게 된다.

본 논문에서는 기존 하이퍼네트워크 모델의 한계를 개선하기 위해 먼저 멀티 레벨의 데이터를 처리 할 수 있도록 매칭 테이블을 기반으로 하는 새로운 비교 연산 방법을 제시하여 정보 손실을 최소한으로 줄 일 수 있도록 하고, 반복적으로 이루어지는 랜덤 샘플링과 비교 연산과정을 초 병렬 연산이 가능한 병렬 프로세서 상에서 구현함으로써 하이퍼네트워크 모델의 수행 성능을 효과적으로 개선하고자 한다.

먼저, 멀티 레벨 데이터의 사용을 위해 새로운 데이터 구조를 그림 3과 같이 제안한다. 그림 3은 2비트의 특징 값을 갖는 4레벨 데이터 구조의 예이다. k차원의 입력 공간에 대해 학습 데이터 집합 $R=\{r_1,r_2,...,r_c\}$ 을 구성하는 c개의 데이터 $r_i=\{r_{i_1},r_{i_2},...,r_{i_k},y_i\}$는 n 비트의 특징 기반으로 하는 k개의 구성요소와 해당 데이터의 클래스 레이블 y_i를 포함한다.

예를 들어, 2비트 특징 값들을 기반으로 하는 데이터인 경우 $r_{i_j}\in\{'00','01','11','10'\}$와 같이 표현되어진다. 새로운 데이터 구조는 d개의 데이터를 포함하는 테스트 데이터 집합 $T=\{t_1,t_2,...,t_d\}$ 에도 동일하게 적용 된다. n비트 기반의 멀티 레벨 데이터를 인코딩 하기위해, 각 데이터의 특징값들이 해당 레벨의 구간에 해당하는 값을 갖도록 이산화(discretization) 한다.

제안 된 방법에서는 새롭게 하이퍼에지를 구성하는 각 멀티 레벨 데이터의 매치 여부를 결정하기 위해, 매칭 테이블(matching table)을 이용하는 방법을 제안 한다. 그림 4는 2비트 데이터를 처리 할 수 있는 2비트 매칭 테이블을 나타낸다. 행과 열은 각각 학습 데이터와 테스트 데이터의 특징 값을 나타내고, 행과 열이 교차하는 구간의 값은 서로 상응 하는 특징들의 매치 여부

$t_{i_1}t_{i_2}$ \ $r_{i_1}r_{i_2}$	00	01	11	10
00	1	1	0	0
01	1	1	1	0
11	0	1	1	1
10	0	0	1	1

$$Matching\_result=r_{i_1}t_{i_1}+r_{i_2}t_{i_2}+r'_{i_1}t'_{i_1}$$

그림 4. 제안 된 매칭 테이블의 구조 (2비트)
Fig. 4. Structure of the proposed matching table (2bit).

를 의미하는 값이다. ('1'일 경우 매치 되었음을 의미) 이러한 매칭 테이블을 사용할 경우 인접해 있지만 이산화 과정에서 다른 레벨로 할당 된 값들에 대해서도 매치 되는 것으로 처리가능 하기 때문에, 잡음에 민감하지 않고, 새로운 데이터가 제시 되었을 때도 일반화된 분류 성능을 제공 할 수 있다.

Ⅳ. 병렬 프로세서 기반 하이퍼네트워크 모델

1. FPGA 기반 하이퍼네트워크 모델 구현

앞 절에서 설명 된 개선 된 하이퍼네트워크 모델을 FPGA 상에서 구현하기 위해, 본 논문에서는 그림 5에서 보여 지는 것과 같이 테스트 데이터 레지스터, 제어 블록과 학습 데이터 메모리, 랜덤 마스크 생성기, 비교 블록, 가산 블록을 포함하고 있는 프로세싱 블록들 그리고 다수 투표 블록으로 구성 된 하드웨어 모델을 제안 한다.

각 비교 블록 내에서 동시적으로 이루어지는 초 병렬 연산은 하이퍼네트워크 모델의 하드웨어가 빠른 수행 속도를 얻기 위한 핵심 블록이라고 할 수 있다. 프로세싱 블록은 각각의 클래스의 학습 데이터에 대해 분리되어 동작 하도록 구성 되어 있는데, 이러한 특성으로 인해 서로 다른 클래스 수를 갖는 새로운 데이터 집합이 분류를 위해 주어졌을 경우에도, 프로세싱 블록의 확장을 통해 최소한의 설계 변경만으로도 해당 데이터를 처리 할 수 있는 확장성을 얻을 수 있다. 제안 된 하드웨어의 동작 단계는 데이터 로딩, 비교, 카운팅 및 예측의 단계로 이루어져 있다. 데이터 로딩과 비교 단계에서, 테스트 데이터는 공통 된 버스(bus)통하여 학습

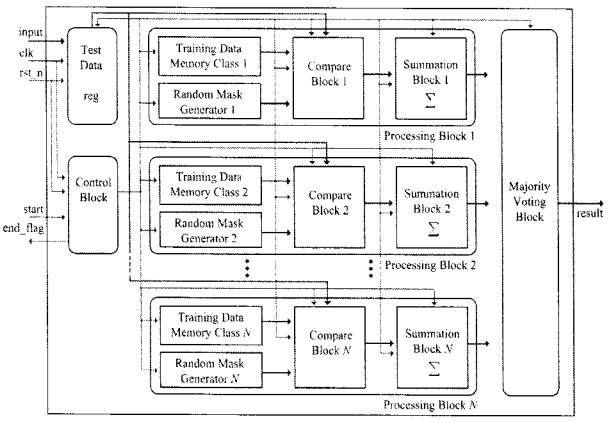


그림 5. 하이퍼네트워크 모델 하드웨어의 구조
Fig. 5. Scheme of the hypernetwork model hardware.

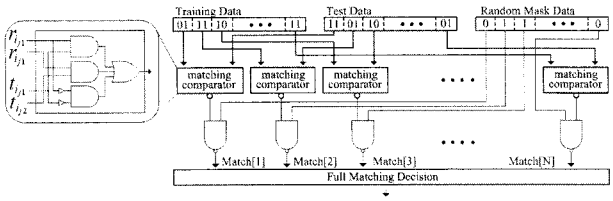


그림 6. 비교 블록 구조
Fig. 6. Structure of the compare block.

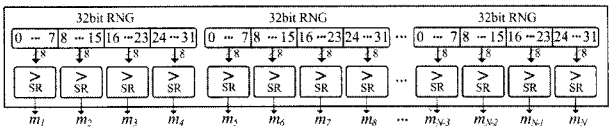


그림 7. 랜덤 마스크 생성기 구조
Fig. 7. Structure of random mask generator.

데이터와 각각의 클래스의 랜덤 마스크와 함께 각각의 비교 블록들로 동시에 입력된다. 그리고 각각의 비교 블록들로부터 얻어진 매칭 결과들은 가산 블록에서 카운트 되어 진다. 마지막으로 가장 많은 값이 카운트 된 클래스가 예측 된 클래스로써 다수 투표 블록을 통해 출력 된다.

그림 5에서 볼 수 있듯이, 제안하는 시스템은 각각의 클래스의 학습 데이터들을 저장하기 위한 N개의 메모리를 가지고 있다. 학습 데이터들은 비교 블록 내의 제안 된 n비트 매칭 비교기들을 사용하여 주어진 테스트 데이터와 비교되어 진다. 3.2절에서 언급되어진 것처럼, 만약 n비트 매칭 비교기가 인접한 입력 값을 갖는다면, 출력 결과는 '1'과 같다. 그렇지 않을 경우 출력 결과는 '0' 이다. 이러한 비교 연산들은 비교 블록 상에서 초 병렬적으로 처리되어 진다. 예를 들어, 분류를 위한 데이터가 N개의 클래스로 구성 되어 있고, 각각의 데이터가 M개의 특징으로 구성 되어 있다면, $M \times N$의 비교 연산들이 동시에 처리되어 진다. 그 후 비교 된 출력 결과들은 그림 6에서와 같이 랜덤 마스크 생성기에 의해 만들어진 랜덤 마스크들을 이용하여 선택되어 진다. 이는 하드웨어로 구성 된 하이퍼네트워크 모델에서만 볼 수 있는 연산 특징으로써, 랜덤 마스크로부터 얻어진 값에 위치하는 특징들을 메모리에서 읽어와 비교하는 것이 아닌, 모든 특징들에 대해 비교 연산을 수행 한 후, 필요한 위치의 값들에 대해서만 선택적으로 NAND 게이트 상에서 활성화 되도록 하는 방식이다. 랜덤 마스크 생성기는 미리 설정 된 선택 비율값을 이용하여 설정 된 차수와 동일한 개수의 값들을 선택 할 수 있도록 설계되었다. 랜덤 마스크 생성기는 LFSR과 CASR

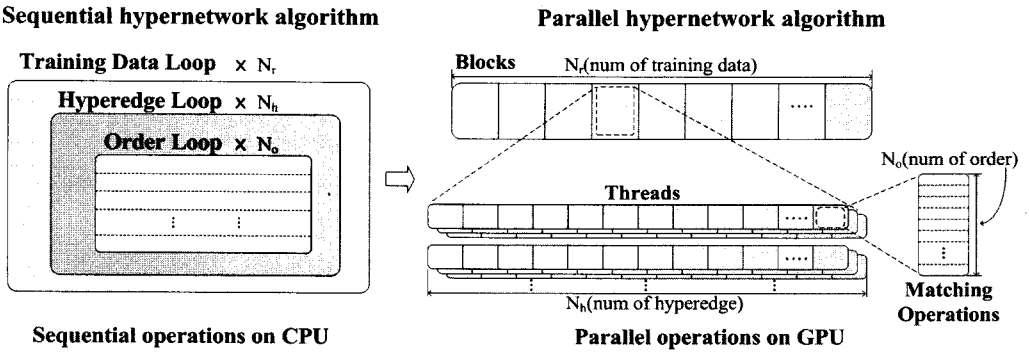


그림 8. GPU 상에서 구현 된 하이퍼네트워크 모델의 연산 구조
Fig. 8. Operation structure of the hypernetwork model implemented on GPU.

을 조합한 32비트의 난수 생성기로 구성 되어 있다[12] (그림 7 참조). 이 때, 생성 된 각각의 난수는 8비트 길이의 출력(0 ~ 255) 값을 갖는다. 따라서 120비트 길이의 랜덤 마스크를 생성하기 위해서는 30개의 32(8×4) 비트 난수 생성기가 필요하다. 이러한 랜덤 마스크 생성기를 이용함으로써, 제안 된 시스템은 랜덤 마스크를 미리 생성하여 메모리에 저장할 필요가 없으므로 메모리 사용량을 감소시킬 수 있다는 장점이 있다.

매칭 비교기로부터 출력 된 결과들은, 각각 반전되어 진 후 랜덤 마스크의 값과 함께 각각의 NAND 게이트로 할당 된다. 이는 인덱스 값들을 통해 서로 상응하는 하이퍼에지의 값들을 비교 하는 것을 모델링 하는 동작이다. 만약 NAND 게이트로 부터 출력 된 모든 값들이 동일하게 '1'의 값을 갖는다면, 그것은 주어진 테스트 데이터와 해당 하이퍼에지가 매치되었음을 의미한다. 가산 블록들로부터 얻어진 각각의 클래스에 해당하는 카운팅 된 매칭 결과들은 다수 투표 블록에서 분류 결과를 예측 하기위해 사용 된다. 프로세싱 블록의 비교와 카운팅 연산들은 학습 데이터의 수와 각각의 클래스에 대해 생성 된 랜덤 마스크의 수를 곱한 만큼 반복되어 진다.

2. GPU 기반 하이퍼네트워크 모델 구현

개선 된 하이퍼네트워크 모델을 GPU를 이용하여 구현하기 위해 NVIDIA에서 제공하는 CUDA(Compute Unified Device Architecture)를 사용하였다. CUDA는 GPU가 일련의 스레드(Thread)들로 구성 된 블록(Block)과 블록들의 집합인 그리드(Grid)를 통해 병렬 연산을 구현 하도록 한다. 일반적으로 NVIDIA의 G80 계열의 GPU에서는 한 개의 블록 당 최대 512개의 스레드를 사용 할 수 있도록 허용하기 때문에 이에 유의하여 커널을 구성해야하고, 또한 메모리 액세스 타임을 줄이기 위해 생성 된 각 스레드에서는 블록 내 스레드 간 공유가 가능하고 빠른 액세스 타임을 보장하는 공유 메모리로 불러와 사용할 수 있도록 하는 것이 효과적이다.

GPU를 이용한 하이퍼네트워크 모델의 구현을 위해 그림 8의 N_r번 반복 되는 학습 데이터의 루프를 구현하기 위해 하나의 그리드 내의 N_r개의 블록으로 대치하여 각 루프에서 이루어지는 연산들이 병렬적으로 동작 할 수 있도록 한다. 입력 데이터가 주어졌을 때, 하이퍼에지들과 입력 데이터에 대해 랜덤 마스크 데이터로부터 차수 k의 크기만큼 얻은 인덱스 값들을 통해 해당 위치의 상응하는 각각의 값들을 비교 연산한다. 또한 이를 통해 매칭여부를 결정하는 2중의 루프 연산은, 각 블록 내의 N_h개의 스레드와 각 스레드 내에서 매칭 여부를 판단을 위해 매칭 테이블로부터 매칭 결과를 얻어오는 N_o번의 연산을 사용하여 구현 한다. 이러한 일련의 과정들은 GPU 상에서 초 병렬적으로 동작하기 때문에 고속의 처리 속도를 얻을 수 있다.

개선 된 하이퍼네트워크 모델의 매칭 테이블은 하이퍼에지와 입력 데이터 내에서 서로 상응하는 특징 값들을 두 개의 입력으로 갖는다. 이 값들을 매칭 테이블에 접근 하기위한 인덱스 값으로 사용하여 해당 위치로부터 리턴 되는 값을 통해 매칭 여부를 결정 하게 된다. 이 매칭 테이블은 모든 블록 내의 스레드들에서 동일하게 사용 되는 것으로 커널을 호출하기 전 상수 메모리에 저장 하여 각 스레드에서 빠르게 접근 할 수 있도록 한다. 아래 표 2의 코드는 4레벨 매칭 테이블을 CUDA를 이용하여 구현한 예로써, 테이블 상의 각 값들은

표 2. CUDA를 이용해 구현한 4레벨 매칭 테이블
Table 2. 4-level matching table implementation using CUDA.

```
__constant__ __device__ int t[4*4] = { 1, 1, 0, 0,
                                       1, 1, 1, 0,
                                       0, 1, 1, 1,
                                       0, 0, 1, 1 };
```

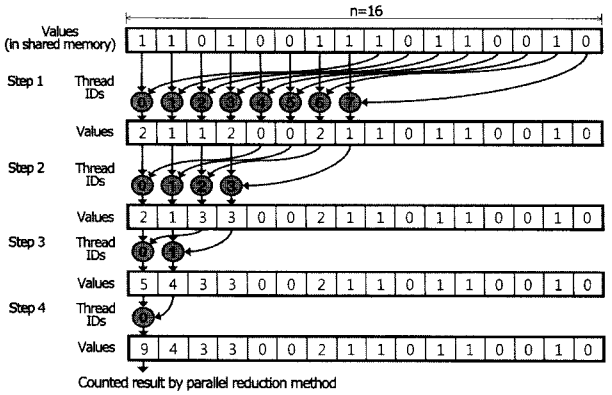


그림 9. 병렬 감소 기법을 이용한 카운팅의 예
Fig. 9. Example of count using parallel reduction method.

$r_i \times 4 + t_i$로 구해지는 배열의 인덱스 값으로 접근 가능하다. 이와 같은 매칭 테이블에 대한 접근은 그림 8에서 볼 수 있듯이 한 스레드 내에서 하이퍼에지의 차수 N_o만큼 반복 되게 되는데, 이는 루프 언롤링(loop unrolling) 기법을 통하여 구현하는 것이 효율적이다.

하이퍼네트워크의 각 하이퍼에지들과 주어진 테스트 데이터가 매치 되는지를 확인 하는 과정은 전체 학습 데이터 수와 랜덤 마스크를 이용한 하이퍼에지들의 수의 곱($N_r \times N_h$)과 같은 수의 스레드를 통해 병렬적으로 이루어진다. 여기서 얻어진 결과들은 각 클래스 별로 카운트 된다. 카운트 과정에서는 빠른 계산을 위해 매칭 결과들을 순차적으로 더하지 않고, 블록 내의 각 스레드들에서 최적화 된 트리 기반의 병렬 감소(parallel reduction) 기법을 사용해 고속으로 카운트 결과를 얻을 수 있도록 한다[12]. CUDA를 이용한 병렬 감소 기법 구현에 있어 고려해야 할 점은 32비트 보다 작은 단위의 데이터를 사용 할 경우 블록 내의 인접 스레드들에서 동일한 뱅크에 접근 하게 되기 때문에, 뱅크 충돌(bank conflict) 문제가 발생하게 된다. 따라서 병렬 감소 기법을 구현함에 있어서 배열의 기본 단위로 32비트의 변수를 사용하였다. 또한 공유 메모리가 16개의 뱅크로 나뉘어 접근 할 수 있도록 되어 있기 때문에 공유 메모리에 접근 할 때에는 32개의 스레드가 묶인 warp가 두 개의 half-warp 단위로 나뉘어 접근 하게 된다. 이로 인해서도 블록 내 스레드들에서의 메모리 접근 방식에 따라 서로 다른 스레드에서 동일 뱅크에 접근하게 되는 뱅크 충돌 현상이 발생 할 수 있다. 따라서 본 논문에서는 그림 9에서와 같이 순차 어드레싱 방식의 병렬 감소 기법을 사용하여 뱅크 충돌 인한 지연을 최소화 할 수 있도록 구현하였다[12].

이렇게 각 클래스 별로 카운트 된 결과를 이용하여, 다시 앞의 병렬 감소 기법을 이용하여 3.1절의 식 3과 같이 가장 많은 수가 카운트 된 클래스를 분류 결과로 출력되도록 다수 투표(majority vote)를 통한 최종 분류 알고리즘을 구현 하였다.

V. 실험 및 결과

본 논문에서는 개선 된 하이퍼네트워크 모델을 이용한 분류 정확도 비교와 병렬 프로세서 상에서 구현 된 하이퍼네트워크 모델의 분류 수행 성능을 평가하기 위해, 다중 클래스 암 분류를 위한 4가지의 공개 된 마이크로어레이 데이터 집합을 이용해 실험을 수행하였다.

사용 된 Subtype of ALL 데이터 집합은 Acute Lymphoblastic Leukemia(ALL)의 6개의 서브타입을 포함하고 있고, Small round blue cell tumors (SRBCT) 데이터 집합은 4가지의 서로 다른 소아암들을 포함하고 있다. Lung cancer 데이터 집합은 폐암의 4가지의 서브타입과 정상 샘플을 포함 하고 있고, Leukemia 데이터 집합은 백혈병의 3가지 서브타입을 포함하고 있다[4~5, 13~14]. 각 데이터에 대한 세부적인 정보는 표 3에 정리 하였다.

1.데이터 전처리 및 실험 설계

표 3에서 볼 수 있는 것과 같이, 마이크로어레이로부터 얻어지는 유전자 발현 데이터 특징들의 수는 대략 수천 개에서 수만 개이고, 이에 반해 실험에 적용 가능한 샘플들의 수는 매우 적다는 것이 암 분류를 위해 사용 되는 유전자 발현 데이터의 일반적인 특징이라고 할 수 있다. 또한 유전자 발현 데이터 상에서 특정 클래스와 연관이 있는 유전자의 수는 본래의 특징들의 수보다 작기 때문에, 유전자 발현 데이터를 이용하여 클래스를 성공적으로 분류하기 위해서는 클래스와 연관성이 높은

표 3. 다중 클래스 암 데이터 집합
Table 3. Dataset of the multi-class cancer.

데이터집합	클래스 수	유전자 수	샘플 수		선택 된 유전자 수
			학습	테스트	
Subtype of ALL	6	12558	163	85	90
SRBCT	4	2308	63	20	120
Lung Cancer	5	12600	136	67	150
Leukemia	3	7129	38	34	90

표 4. 알고리즘들 간의 분류 성능 비교
Table 4. Comparison of the classification performance between algorithms.

데이터집합	하이퍼네트워크 모델(%)			SVM(%)	신경망(%)	k-NN(%)
	1비트(SD)	2비트(SD)	3비트(SD)			
Subtype of ALL	93.57(1.48)	97.25(1.06)	98.76(0.25)	98.00	98.50	97.16
SRBCT	93.80(3.70)	99.60(1.36)	100.0(0.00)	100.0	91.03	86.90
Lung Cancer	89.31(1.47)	94.02(0.00)	95.52(0.00)	96.05	87.80	89.64
Leukemia	94.76(2.11)	96.02(1.40)	99.97(0.29)	97.50	76.61	83.57
평균	92.86(2.19)	96.72(0.95)	98.56(0.13)	97.88	88.49	89.32

유전자를 추출하는 과정이 필요하다.

본 논문에서는 Golub 등이 제안한 signal-to-noise 기법을 사용하여 유전자 선택을 수행 한다[4]. 이 기법은 각 클래스에 대해 유전자 발현 값들의 평균값의 차이는 크면서 동시에 표준 편차의 합이 작은 유전자들을 선택 하도록 되어있다. 이를 이용한 유전자 선택 과정에서는 오버피팅(overfitting) 문제를 피하기 위해 학습 데이터만을 대상으로 수행 하였다. 또한 각각의 샘플의 유전자 발현 값이 처리되기 전에 평균이 0이고, 표준편차가 1인 표준 정규분포 값을 갖도록 표준화 (standardization) 하였다.

앞서 살펴본바와 같이 하이퍼네트워크 모델에 데이터를 적용하기 위해서는 먼저 유전자 발현 값을 이산화 할 필요가 있다. 1비트 데이터만 사용 할 수 있는 기존의 하이퍼네트워크 모델에서 사용 할 이진화 된 데이터를 얻기 위해 유전자 발현 값의 평균값을 문턱값(threshold value)으로 사용하여 이진화 하였다. 그리고 제안된 방법이 적용 된 개선 된 하이퍼네트워크 모델에 대해서는, 2비트 데이터의 경우 (high, mid, low)의 3개의 문턱값을 정하여 유전자 발현 값을 ('00', '01', '11', '10')의 4단계 값으로 이산화 하였고, 3비트 데이터의 경우 7개의 문턱값을 정하여 유전자 발현 값을 ('000', '001', '011', '010', '110', '111', '101', '100')의 8단계의 값으로 이산화 하였다.

예를 들어, 2비트 데이터를 얻기 위한 3개의 문턱 값의 경우 다음과 같이 구해 질 수 있다.

$$\begin{aligned} high &= average + \max \times \frac{1}{2} \\ mid &= average \\ low &= average + \min \times \frac{1}{2} \end{aligned} \tag{4}$$

여기서 문턱 값을 구하기 위해, 먼저 각 샘플 내의의 표준화 된 전체 유전자 값들을 발현 값의 크기에 따라 정렬 하고, 그 후 평균값(average), 최대값(max), 최소값(min) 값을 구한다. 각 유전자의 발현 값들은 이미 표준화 되어 있기 때문에 평균값은 0이 되고, 최대값은 양수를 그리고 최소값은 음수를 갖게 된다.

제안 된 하이퍼네트워크 모델을 이용한 분류 성능을 다른 연구 결과와 비교 평가하기 위해, 본 논문 에서는 데이터를 공개한 원 저자가 사용한 방법들과 동일한 방

법으로 학습 데이터와 테스트 데이터로 분할하여 실험을 수행 하였다. 최종적으로 얻어지는 분류 정확도는 100번의 실험을 반복한 후 평균한 값을 이용하여 구하였다. 또한 하이퍼네트워크 모델의 각 병렬 프로세서 상에서의 분류 수행 속도와 기존의 순차 머신 상에서의 분류 수행 속도를 비교하기 위해 동일한 데이터에 대해 각각의 환경에서 처리 시간을 비교 하였다.

2. 다중 클래스 암 진단 실험 결과

표 4는 하이퍼네트워크 모델과 일반적인 기계 학습 방법을 사용하는 기존 연구결과들의 분류 정확도를 비교하여 보여준다. 또한 기존의 하이퍼네트워크 모델과 개선 된 하이퍼네트워크 모델을 이용한 분류 정확도와 분류 정확도의 표준편차를 비교하여 보여준다. 표 4에서 확인 할 수 있듯이 제안 된 하이퍼네트워크 모델은 2비트, 3비트 데이터를 이용하여 분류 하였을 경우 각각 평균 96.72%와 98.56%의 분류 정확도를 보여주었다. 이것은 기존의 하이퍼네트워크 모델이나 신경망, k-NN 보다 나은 결과이고, SVM과 비교를 했을 경우 Lung cancer 데이터의 경우를 제외 하고, 제안 된 모델이 더 나은 결과를 보였다.

이는 제안 된 하이퍼네트워크 모델을 이용한 유전자 발현 데이터의 분류 성능이, 최근 그 성능과 이론적 완성도로 인해 널리 이용되고 있는 SVM과 비교해서도 경쟁력 있는 성능을 보이고 있음을 의미 한다. 또한 제안 된 하이퍼네트워크 모델은 분류 정확도에 있어서 이전의 모델 보다 일반적으로 낮은 표준편차를 보임을 알 수 있다. 이것은 제안 된 방법을 사용 하였을 경우 안정적이고 신뢰 할 수 있는 성능을 보임을 의미한다.

표 5. 환경 별 SRBCT 데이터 분류 수행 성능 비교
Table 5. Comparison of processing performance.

수행 환경	CPU	GPU	FPGA
	INTEL Core2 Quad Q6600	Geforce GTX 275	Xilinx Virtex4 XC4VLX 200
수행 시간	2700us	190us	27.6us

3. 병렬 프로세서 상에서 구현 결과

병렬 프로세서 상에서의 하이퍼네트워크 모델 구현은 앞의 실험에서 사용 된 마이크로어레이 데이터들 중 SRBCT 데이터에 대해서만 적용 하였고, 2비트 매칭 비교기를 사용하도록 구현 된 모델들을 사용하여 수행 성능을 비교 하였다. 각 모델은 하나의 학습 데이터에 대해 각각 512개의 하이퍼에지를 차수가 $k=30$ 으로 균일 하도록 랜덤 샘플링하여 생성한다.

FPGA 상에서 구현 된 하이네트워크 모델은 Xilinx I SE 9.2i 상에서 VerilogHDL을 이용하여 설계 되었으며, XST를 이용하여 합성하였다. 사용 된 FPGA는 Xilinx 의 Virtex-4 XC4VLX200 모델로써, 89,088개의 사용가능한 Slice를 가지고 있고, 6,048 kbits의 독립적인 메모리 블록(Block RAM)을 사용 할 수 있다. 구현 된 하드웨어는 480(4 class × 120 genes)개의 2비트 매칭 비교기와 32비트 난수 생성기를 각각 30개씩 갖고 있는 랜덤 마스크 생성기 4개와 242 비트 길이의 출력을 갖는 램 4개 등으로 구성 되어 있다.

타겟 디바이스에 대해 합성 하였을 때, 전체의 시스템은 89,088개의 Slice 중 11,512개의 Slice와 336개의 블록 메모리 중 16개의 메모리를 사용 하였다. 이는 각각 전체의 Slice 중 12%, 전체의 Block 메모리 중 4%에 해당 하는 크기이다. 또한 랜덤 마스크의 사용으로 인해 기존의 방법과 같이 랜덤 마스크를 메모리로 저장하고 있지 않아도 되기 때문에 사용된 블록 메모리의 양이 최소화되었음을 알 수 있다. 구현 된 하드웨어의 실제 동작은 ChipScope Pro 9.2i를 이용하여 검증하였다. 최대 동작 주파수는 159.52MHz로서, 구현 된 하드웨어를 100MHz의 클록(clock)에서 동작 시켰을 경우, 3.62×10^4 data/s 의 처리율을 보여준다. 하나의 데이터를 분류 하는데 필요한 클럭 사이클 수는 2760개로서, 약 27.6 마이크로초의 시간이 소요 되었다.

GPU를 이용하여 구현 된 하이퍼네트워크 모델은 NVIDIA의 CUDA 버전 2.3 에서 작성 되었다. 실험에 사용 된 GPU는 NVIDIA의 GeForce GTX 275로서 30개의 멀티프로세서를 가지고 있다. 각각의 멀티프로세서는 8개의 스레드 프로세서를 갖고 있기 때문에 총 240개의 코어로 연산 가능하며, 1.4GHz의 클록에서 동작 한다.

실험결과 하나의 데이터를 분류하는데 소요되는 시간은 약 190 마이크로초 정도였다. 전체 GPU 연산 시간의 66.8%는 하이퍼에지와 입력 데이터의 비교 연산을 처리하는 커널을 수행하는데 사용 되었고, 6.34%는 매칭 결과를 카운팅 하는 커널에서 사용 되었다. 2..55%는 최종적으로 예측을 수행하는 커널에서 사용 되었다. 그 밖에 디바이스와 호스트로의 데이터 이동에도 약

23.47%가 소모 되어 매우 많은 시간을 메모리 사이의 데이터 이동에서 소모함을 알 수 있었다. 또한 구현 된 하이퍼네트워크 모델의 비교 연산을 처리하는 커널에서는 497번의 분기가, 매칭 결과를 카운팅 하는 커널에서는 4번의 분기가 발생하였고, 최종적으로 예측을 수행하는 커널에서는 분기가 발생하지 않았다. 각 커널을 수행함에 있어서 if나 for 문에 의해 발생하는 흐름 제어 명령은 스레드를 분기 시키게 되고, 이는 스레드가 순차적으로 처리되도록 하여 수행 속도를 늦추는 원인이 된다. 글로벌 메모리의 초당 처리량은 52.33 GB/s로 계산 되었다.

CPU 기반의 순차 연산 하이퍼네트워크 모델은 MS Visual Studio 2005를 사용하여 컴파일 하였고, 2.4GHz의 클록에서 동작하는 Intel Core 2 Quad Q6600 CPU 상에서 동작시켰다. 해당 CPU는 4개의 멀티코어를 갖고 있으나 비교를 위해 하나의 코어만을 사용하였다.

각 플랫폼 상에서의 연산 수행 결과는 표 5에서 확인할 수 있다. CPU 상에서의 연산 속도에 비해, GPU 상에서 구현 된 하이퍼네트워크 모델은 14.21배, FPGA 상에서 구현 된 하이퍼네트워크 모델은 97.83배 빠르게 연산을 수행했음을 알 수 있다. 이는 병렬 프로세서를 이용하여 하이퍼네트워크 모델을 구현 할 경우, 순차 머신 상에서 동작 했을 때 보다 연산의 효율성을 극대화 할 수 있음을 의미한다. 따라서 제안 된 모델은 대용량의 유전자 발현 데이터들을 고속으로 처리하는데 매우 적합한 분석 방법이라고 할 수 있다.

VI. 결 론

본 논문에서는 마이크로어레이로부터 얻어지는 대용량의 유전자 발현 데이터를 처리 하기위한 기존 기계학습 알고리즘들의 한계를 극복하기 위해, 유전자들 간의 상호 작용을 고려하며 분류를 수행 할 수 있는 개선된 하이퍼네트워크 모델을 제안 하였다. 다중 클래스의 암을 분류하기 위한 데이터를 이용하여 실험한 결과, 기존의 기계 학습 알고리즘들과 비교하여 경쟁력 있는 분류 정확도를 보여주었으며, 제안 된 모델의 수행 성능을 최대화하기 위해 FPGA와 GPU 기반의 병렬 프로세서 상에서 구현 한 후, 수행 성능을 비교하였다. 실험 결과 순차 머신에서 동작 하였을 경우와 비교하여 최대 100배 이상 빠르게 동작함을 확인 할 수 있었다. 이러한 병렬 연산 능력은 최근 급속도로 증가하고 있는 바이오 인포메틱스 영역에서의 대용량 데이터들을 효율적으로 처리하기 위한 대안으로 적합할 것으로 보이며, 향후 지속적인 연구가 필요할 것으로 생각 된다.

참 고 문 헌

[1] Klebanov, L., and Yakovlev, A., "Diverse correlation structures in gene expression data and their utility in improving statistical inference", The Annals of Applied Statistics, 2, pp. 538-559, 2007.

[2] Duggan, D.J. and Bittner, M. and Chen, Y. and Meltzer, P. and Trent, J.M., "Expression profiling using cDNA microarrays", Nature genetics, Vol. 21, pp. 10-14, 1999.

[3] Zhang, B.T., "Hypernetworks: A molecular evolutio nary architecture for cognitive learning and memory", Computational Intelligence Magazine, IEEE 3(3), pp. 49-63, 2008.

[4] Golub, T. and Slonim, D. and Tamayo, P. and Huard, C. and Gaasenbeek, M. and Mesirov, J. and Coller, H. and Loh, M. and Downing, J. and Caligiuri, M., et al., "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring", Science 286(5439), pp. 531-537, 1999.

[5] Khan, J. and Wei, J. and Ringner, M. and Saal, L. and Ladanyi, M. and Westermann, F. and Berthold, F. and Schwab, M. and Antonescu, C. and Peterson, C., et al., "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks", Nature Medicine 7, pp. 673-679, 2001.

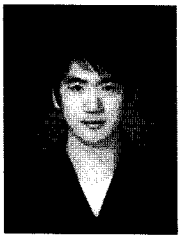
[6] Wee, J.W. and Lee, C.H., "Concurrent Support Vector Machine processor for disease diagnosis", Lecture Notes in Computer Science, vol. 3316, pp. 1129-1134, 2004.

[7] I. Pournara, C.S. Bouganis, G.A. Constantinides, "FPGA-accelerated Bayesian learning for reconstruction of gene regulatory networks", in proceeding of the 15th International Conference on Field Programmable Logic and Applications, pp. 323-328, Tampere, Finland, 2005.

[8] Dar-Jen Chang, Ahmed H. Desoky, Ming Ouyang, Eric C. Rouchka, "Compute Pairwise Manhattan Distance and Pearson Correlation Coefficient of Data Points with GPU", in proceeding of the 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed

Computing, pp. 501 - 506, Daegu, Korea, 2009.
[9] Manavski, S. and Valle, G., "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment", BMC bioinformatics, 9(Suppl 2):S10, 2008.
[10] Zhang, B.T. and Jang, H.Y., "A bayesian algorithm for in vitro molecular evolution of pattern classifiers", Lecture Notes in Computer Science. vol. 3384, pp. 458-467, 2002.
[11] Tkacik, T., "A hardware random number generator", Lecture Notes in Computer Science, vol. 2523, pp. 450-453, 2003.
[12] Harris, M., "Optimizing parallel reduction in CUDA", CUDA Advanced Topics, CUDA ZONE, 2008.
[13] Yeoh, E. and Ross, M. and Shurtleff, S. and Williams, W. and Patel, D. and Mahfouz, R. and Behm, F. and Raimondi, S. and Relling, M. and Patel, A. and et al., "Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling", Cancer Cell, 1(2), pp. 133-143, 2002.
[14] Bhattacharjee, A. and Richards, W. and Staunton, J. and Li, C. and Monti, S. and Vasa, P. and Ladd, C. and Beheshti, J. and Bueno, R. and Gillette, M., et al., "Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses", Proceedings of the National Academy of Sciences, pp. 13790-13795, 2001.

저 자 소 개



최 선 욱(학생회원)
2007년 인하대학교 전자공학과 학사 졸업.
2009년 인하대학교 정보공학과 석사 졸업.
2009년~현재 인하대학교 정보공학과 박사 과정.
<주관심분야 : Evolvable Hardware, Pattern Classification, 3D Computer Vision, Visual SLAM>

이 종 호(평생회원)
1976년 서울대학교 전기공학과 학사 졸업.
1978년 서울대학교 전기공학과 석사 졸업.
1979년~1982년 해군사관학교 전임강사
1986년 미국 아이오와 주립대 전기 및 컴퓨터 공학과 박사 졸업.
1986년~1989년 미국 노틀담 대학교 조교수.
1989년~현재 인하대학교 정보통신공학과 교수.
1997년~1998년 인하대 직접회로설계센터 소장.
2000년~현재 인하대 수퍼지능기술연구소 소장.
2004년~2005년 미국 브라운 대학교 두뇌 및 신경회로망 연구소 방문교수.
2009년~현재 인하대학교 대학원장.
<주관심분야 : Artificial Neural Network Hardware, Evolvable Hardware, Intelligence System>