붙임2

연구논문/작품 중간보고서

2021 학년도 제 2 학기

제목	Verilog를 이용한 Convolutional Neural Network 구현	○ 논문(　　) 작품(✓) ※해당란 체크
GitHub URL	https://github.com/boaaaang/CNN-Implementation-in-Verilog	
평가등급	지도교수 수정보완 사항	팀원 명단
A, B, F중 택1 (지도교수가 부여)	○ ○ ○	강 보 영　(인) 학번: 2017312548

2021 년 09 월 16일

지도교수 : 이 강 윤 　　서명

■ 요약

본 과제에서는 신경망을 통한 학습 과정 중 하나인 합성곱 신경망(Convolutional Neural Network)을 Verilog 언어를 사용하여 회로적으로 구현해보려 한다. CNN은 이미지 처리/인식에 탁월한 성능을 보이는 신경망 아키텍쳐으로, 수동으로 특징을 추출할 필요가 없고 스스로 학습하기 때문에 딥러닝에서 널리 사용된다. 본 연구에서는 설계한 CNN을 통해 MNIST 데이터셋 인식률을 확인해 볼 것이다. 파이토치(PyTorch) 프레임워크를 이용하여 데이터셋으로부터 추출해 낸 가중치(weight)와 편향(bias) 등의 데이터를 설계한 CNN에 적용함으로써 설계를 검증하고, 적정률(인식률)을 확인할 것이다.

Keyword : CNN, Convolution, Verilog, PyTorch, MNIST, Deep Learning

■ 서론

1) 제안배경 및 필요성

최근 인공지능에 관한 관심이 증가함에 따라 인간의 두뇌를 모방한 뉴로모픽(Neuromorphic) 기술이 차세대 반도체 기술로 대두되고 있다. 뉴로모픽 기술은 기존의 폰 노이만(Von-Neumann) 구조의 병목현상(Bottleneck)을 해결하고 저전력으로 보다 방대한 양의 연산을 병렬적으로 빠르게 수행 가능하다는 강력한 장점이 있다. 이러한 점에서 뉴로모픽 기술은 많은 양의 컴퓨팅 자원 및 전력 소모 요구하는 기존의 딥러닝 모델을 경량화할 수 있는 해결책이라고 볼 수 있다.

지난 2016년 기계가 넘볼 수 없는 분야로 여겨졌던 바둑을 딥러닝으로 학습해 인간을 이긴 인공지능(AI, Artificial Intelligence) 알파고의 등장은 전 세계인을 충격에 빠뜨렸다. 1,202개의 CPU와 176개의 GPU로 이루어진 알파고는 소프트웨어적인 측면에서는 당시 비약적인 발전으로 인정받았지만, 하드웨어적인 측면에서는 전력 소모가 매우 크다는 치명적인 단점 또한 가지고 있다. 기존의 폰 노이만(Von-Neumann) 구조는 순차적 계산은 속도가 빠르지만, 인공지능기술이 요구하는 병렬계산에는 병목현상(Von Neumann Bottleneck)이 발생하기 때문에 적절하지 않다는 지적을 받는다. 이러한 폰 노이만 구조의 한계를 극복하는 인공지능 기술의 발전을 위한 하나의 대안으로 뉴로모픽(Neuromorphic) 기술이 새로이 조명받고 있다.

인간의 뇌는 천억 개가 넘는 신경세포, 즉 뉴런이 시냅스라는 연결고리를 통해 다른 뉴런과 신호를 주고받으며 정보를 처리하고 저장한다. 따라서 정보의 저장과 처리(연산)를 같은 장소에서 진행하기 때문에 저전력 고속 동작이 가능하다. 이에 반해 기존의 컴퓨터는 연산을 담당하는 CPU와 저장을 담당하는 메모리가 따로 존

재하고 정보가 상호전송 되는 과정이 존재하여 이 과정에서 병목현상이 발생한다 [1]. 뉴로모픽 시스템은 인간의 두뇌를 구성하는 신경 시스템을 모사하는 시스템으로서, 뉴런 소자와 시냅스 소자가 고밀도 네트워크로 연결된 구성을 갖는 시스템이다 [4]. 외부에서 입력된 자극을 전기적 신호로 전환하는 뉴런(neuron)과 전기적 신호를 인접한 뉴런으로 전달하면서 신호 전달 능력을 의미하는 시냅스 가중치(weight)를 변화시켜 적응 학습 및 기억 기능을 수행하는 시냅스(synapse)가 대규모 병렬 구조로 형성되어 방대한 양의 데이터를 효과적으로 처리하는 두뇌를 모사한다.

반도체 기술은 사람을 흉내 내면서 발전해왔다. 예를 들어 이미지 센서는 사람의 눈을, 가스탐지기는 사람의 코를, 터치 센서는 사람의 피부를 모방한다. 사람의 뇌를 흉내 내는 뉴로모픽 기술을 NPU(Neural Processing Unit)이라고도 한다. 물론 기존의 CPU(Central Processing Unit)와 GPU(Graphics Processing Unit)도 인공지능 기술에 사용되지만, 사람의 뇌를 모사하여 사람처럼 생각, 학습하고 추론까지 하는 것은 NPU가 특화돼있다고 볼 수 있다. 물론 CPU와 GPU가 NPU에 비해 오랫동안 사용되어 왔기 때문에 아직까지 많이 사용된다. 그렇기에 적용 가능한 소프트웨어가 다양하게 존재하지만, NPU는 그에 반해 상대적으로 짧은 역사를 가지고 있기 때문에 아직까지 소프트웨어 또는 어플리케이션 개발 영역이 한정되어있다고 볼 수 있다. 화웨이(Huawei)에 따르면 GPU는 CPU의 8배, NPU는 CPU의 무려 50배에 달하는 효율성을 가진다고 한다. 이러한 효율성을 갖는 이유는 뉴런과 시냅스의 병렬적인 연결 구조를 모방한 기술이기 때문일 것이다. 현재 정부와 기업이 뉴로모픽 기술 연구에 많은 관심을 갖고 투자와 참여를 하고 있다. 향후 뉴로모픽 기술 개발에 힘써 그 활용 분야를 넓힌다면 4차 산업 시대에 핵심이 될 수 있는 기술을 선도해 나갈 수 있을 것이다.

2) 연구작품의 목표

본 연구작품에서는 MNIST Dataset에 대해 94% 이상의 적중률을 보이는 Convolutional Neural Network를 Verilog언어로 설계하는 것을 목표로 한다. 또한, 입력 데이터에 대한 학습 시간은 얼마나 소요되는지 측정한다.

MNIST란 손으로 쓴 글자 데이터셋에서 숫자만 따로 뽑아낸 데이터셋으로, 0부터 255 사이의 값을 가지는 흑백 이미지로 구성되어 있으며, 이미지 크기는 28X28로 고정되어 있으며 60,000개의 Training Set로 구성되어 있다. 본 과제에서는 PyTorch라는 프레임워크를 이용하여 MNIST 데이터셋을 학습시킨다. CNN의 각 Layer로부터 가중치와 편향을 추출/보정하여 Verilog 기반 CNN 설계에서 사용한다. CNN은 Mentor사의 Modelsim 환경에서 설계되며, 설계를 검증하기 위해 PyTorch에서 각 Layer의 Output Data 또한 별도의 텍스트 파일로 저장하도록 한다.

3) 연구작품 Overview

본 연구작품에서는 CNN을 하드웨어로 설계하기 위해 크게 2단계로 나누어 프로젝트를 진행하게 된다.

먼저 PyTorch 기반 딥러닝 데이터셋 학습이다. 그에 앞서, CNN의 기본 연산 방법과 Convolution Layer, Pooling Layer, Fully Connected Layer의 동작 및 원리를 이해한다. Activation Function에는 어떤 종류가 있는지에 대해 학습하고 각 함수의 특징을 조사한다. Convolution Layer, Pooling Layer, Fully Connected Layer에서 각각의 입력데이터에 대한 출력 데이터의 크기는 어떻게 산정되는지 학습하고 다양한 예제를 통해 적용시켜 본다. CNN에 대한 학습을 기반으로 PyTorch 프레임워크에서 그를 적용시키기 위해 먼저 PyTorch 환경을 설정하고 문법을 학습한다. 특히 CNN 구축을 위한 라이브러리/패키지 및 텐서 조작을 위한 파이썬 문법을 위주로 학습한다. CNN 구조를 PyTorch에서 설계하고 학습을 진행하고, MNIST 데이터셋을 로드하여 적용시킨다. 이 단계에서 94% 이상의 적중률을 갖는 CNN 모델에 대한 확보가 필요하다. CNN 모델을 결정한 후에는 Verilog 설계를 위한 가중치와 편향, 각 Layer의 Output Data를 추출하고 보정하여 저장한다.

두 번째 단계는 Verilog 설계이다. CNN의 구조에 따라 Convolution Layer, Max Pooling Layer, Activation Function, Fully-Connected Layer 등의 sub-block을 나누고 설계를 진행한다. 각 모듈에 대한 testbench를 작성하여 동작을 검증하고, 최종적으로 MNIST 이미지 데이터 입력에 대해 몇 퍼센트의 적중률을 갖고, 인식에 소요되는 시간을 얼마인지 측정하도록 한다.

■ 관련연구

딥러닝(Deep Learning)은 오늘날 산업 전반에 걸쳐 큰 혁명을 일으켰다. AI(Artificial Intelligence)와 DL는 비슷하게 이해될 수 있지만, 실질적인 차이가 존재한다. AI, ML(Machine Learning), DL의 관계를 이해하기 위해서는 아래의 그림과 같은 포함관계로 이해함이 바람직하다 [1].

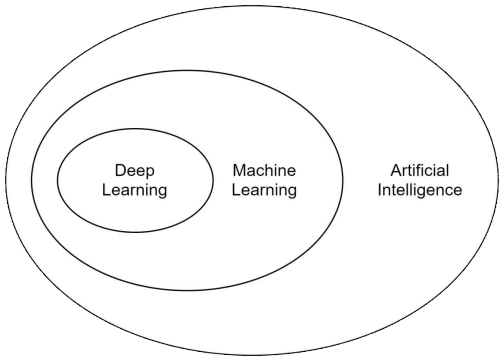


Fig. 1. AI＆ML & DL

AI는 인간의 학습 능력, 추론 능력, 지각 능력, 언어 이해 능력 등을 컴퓨터에 구현한 기술이다 [1]. ML은 과거의 경험을 바탕으로 추론 또는 결정을 하는 방법을 머신에게 알려준다. 즉, 특정 패턴을 파악하고 과거의 데이터를 분석함으로써 그 의미를 추측하고 결론을 도출한다 [1]. 빅데이터를 분석하고 가공하여 새로운 정보를 얻어낸다. DL은 ML의 한 종류로 생각할 수 있다. 패턴으로부터 결과를 분류하고 추측하기 위해, Layer를 통해 입력을 처리하는 것을 머신에게 알려준다. DL은 축적된 데이터를 분석 뿐만 아니라 학습까지 하는 능력을 활용하여 결론을 도출한다 [1]. 딥러닝은 컴퓨터 비전, 음성 인식, 자연어 통역 및 이미지 인식 등 다양한 비형식 데이터 처리 분야에서 다른 ML 알고리즘보다 뛰어난 성능 및 범용성을 보여준다 [1]. 이는 인간의 신경세포와 유사한 원리에 따라 기능을 한다. 딥러닝에서는 분야에 따라 CNN(Convolutional Neural Network), RNN(Recurrent Neural Network), RBM(Restricted Boltzmann Machine), DBN(Deep Belief Network) 등 다양한 학습 모델이 있다.

인공 신경망(ANN)은 인간의 신경 세포 즉 뉴런과 유사한 인공 뉴런을 이용하여 복잡한 프로세서로 이루어진다. 시냅스에 해당되는 가중치는 인공 뉴런들 사이에 신호가 지나가는 링크들로 연결되어 있다. 가중치는 인공 신경망 모델에서 기억과 수행한 학습을 바탕으로 뉴런 입출력 강도, 즉 특성 문제에 대한 신경망의 패턴을 표현한다 [2-6]. 이 가중치를 반복적으로 조정함으로써 학습이 이루어진다.

최근 많은 컴퓨터 비전 분야에서 딥러닝을 이용하여 최고의 성능을 도출하고 있다 [7]. 하지만 딥러닝은 연산량과 메모리 사용량이 매우 커서 실시간 구현을 위해서는 하드웨어 가속기가 필수적이다 [7]. 특히, 영상 콘텐츠에 대한 수요가 증가하는 요즈음 촬영과 동시에 유통을 하는 생방송 영상의 지적재산권을 보호할 수 있는 워터마킹 삽입 방법이 요구된다 [7]. 디지털 워터마킹 프로세서의 구조는 주 동작에 따라 크게 데이터 패스부와 제어부로 구분된다 [7]. 데이터 패스부에서는 CNN의 핵심인 Convolution 연산을 하며, 제어부는 메모리 제어부와 메인 제어부로 나뉘게 된다 [7]. CNN은 컴퓨터 비전 분야 중에서 이미지 인식 문제를 해결하기 위해 특화된 딥러닝 학습 모델 중 하나이다 [8]. Convolution block에서는 Filter Register나 Input Feature Map Register를 입력으로 받아 Convolution 연산을 수행한다 [7]. 그 결과값을 Partial Sum Register에 넘겨준다 [7]. 이는 SRAM 메모리에 저장되어 다음 동작에 사용되거나 다음 block으로 넘어가 활성화 함수를 통해 Output Register에 저장된다 [7].

CNN은 대표적인 feedforward neural network로 보통 여러 층의 convolutional layer와 pooling layer, 그리고 fully connected layer를 포함한다 [9]. 이 층들이 input부터 output까지 연결된다 [9]. RTL level로 CNN 연간 가속기 뿐만 아니라 CNN application system의 빌드 속도를 향상시키는 CNN IP core 설계를 위한 sub-block partitioning은 다음과 같다.

A. Convolution module : image feature 추출

Input Feature map에 convolution kernal과 weight로 합성곱연산을 한다. 그 후, 활성화 함수(activation function)를 거쳐 해당 layer의 feature map을 얻는다. 이를 표현한 식은 아래 [9]와 같다.

$$y_j^l = f\left(\sum_{i \in M_j}^{i^{l-1}} x_i^{l-1} * k_{ij}^l + b_j^l\right)$$

$y_j^l = feature\ map\ of\ the\ j-th\ convolution\ kernal\ convolution\ result\ of\ l-th\ layer$
$M_j = selection\ of\ the\ previous\ input\ feature\ map\ by\ the\ current\ convolution$
$x_i^{l-1} = previous\ input\ feature\ map$
$k_{ij}^l = i-th\ weighting\ coefficient\ of\ the\ j-th\ convolution\ \mathrm{kernel}\ of\ the\ l-th\ layer$
$b_j^l = bias\ parameter\ of\ the\ j-th\ convolution\ kernel\ of\ the\ l-th\ layer$
$f = non\ linear\ activation\ function$

B. Pooling module

주로 Convolution module 뒤에 위치하여, input feature map의 사이즈를 줄이는 기능을 한다. Maximum pooling과 average pooling 방법이 있다 [9]. 각각 최댓값과 평균값이 해당 영역의 output data로 쓰이며, 2x2 size를 가장 많이 쓴다.

C. Fully connected module

Fully Connected Layer에서는 이전 layer의 feature map output을 one-dimensional vector로 변환한다. Multiply-accumulate operation으로 적은 양의 연산이지만 많은 양의 가중치 파라미터와 메모리 접근이 요구된다 [9]. 따라서 FC layer parallel computing을 확장시키는 것은 비현실적이므로, 본 선행연구 [9] 에서는 locally parallel acceleration 전략을 사용한다. 이는 $P_x \times P_y$가 previous layer의 FC layer의 output feature map 사이즈, P_n이 채널의 수, M_n이 FC layer의 뉴런 수라고 가정한다 [9]. 이 때, 변환된 벡터의 총 길이는 $L_n = P_x P_y P_n$이고, 각 채널의 길이는 $P_x P_y$이다 [9].

■ 제안 작품 소개

1) CNN이란?

Convolutional Neural Network(CNN)은 이미지를 분석하기 위해 패턴을 찾는 데에 유용한 알고리즘으로 입력데이터로부터 직접 학습하고 패턴을 사용하여 이미지를 분류한다. Fully Connected Layer(FC)로 구성된 신경망은 입력데이터가 1차원 배열 형태로 한정되어있다는 한계가 있다. 하지만 한 장의 컬러 사진은 RGB로 이루어져 있기 때문에 3차원 데이터이다. 또한, 배치 모드에 사용되는 여러 장의 사진은 4차원 데이터이다. 따라서, 사진 데이터로 Fully Connected 신경망을 학습 학습시킬 때, 3차원의 데이터를 1차원으로 평면화해야 한다. 그 과정에서 공간 정보의 손실이 발생한다. 이러한 한계를 극복하고 이미지의 공간 정보를 유지한 상태로 학습이 가능한 모델이 CNN이다. CNN은 각 Layer의 입출력 데이터의 형상을 유지하고, 이미지의 공간 정보를 유지하며 인접 이미지와의 특징을 인식한다는 차별점이 있다. 또한, 필터를 공유 파라미터로 사용하기 때문에 일반 인공 신경망과 비교하였을 때 학습 파라미터가 적다는 특징이 있다.

CNN은 크게 두 가지 파트로 나눌 수 있다. 첫 번째는 이미지의 특징을 추출하는 부분이고 두 번째는 클래쓰를 분류하는 부분이다. 이미지 특징을 추출하는 부분은 Convolution Layer와 Pooling Layer를 여러 겹 싸는 형태이다. Convolution Layer는 입력데이터에 필터를 적용 후 activation Function(활성화 함수)를 반영하는 필수 요소로 필터를 사용하여 공유 파라미터 수를 최소화하면서 이미지의 특성을 찾는다. Pooling Layer는 선택적 레이러로 특징을 강화하고 모은다. Max Pooling, Average Pooling, Min Pooling 등의 방법이 있는데 CNN에서는 주로 Max Pooling을 사용한다. 필터의 크기, stride, padding 적용 여부, pooling 크게 등에 따라 출력 데이터의 shape이 변경되므로 적절히 조절하여 입/출력 데이터 shape을 맞추는 작업이 중요하다. 클래쓰를 분류하는 Fully Connected Layer는 CNN의 마지막에 추가되며, 그 사이에 이미지 형태의 데이터를 배열 형태로 만드는 Flatten Layer가 존재한다.

Channel(채널)은 데이터의 차원을 의미한다. 즉, 컬러 사진은 천연색을 표현하기 위해 RGB 3개의 실수로 표현한 3-channel의 3차원 데이터이고, 흑백사진은 2차원 데이터로 1-channel로 구성된다. Convolution Layer에 유입되는 입력 데이터에는 보통 한 개 이상의 필터가 적용된다. 즉, n개의 필터가 적용되면 출력 데이터는 n개의 channel을 갖게 된다. Filter(필터)는 이미지의 특징을 찾아내기 위한 공용 파라미터로 정사각 행렬이다. 즉, CNN에서 학습의 대상이 필터 파라미터이고, 필터는 입력데이터를 유입하고 학습시키면 자동으로 특징을 인식하여 필터를 만들어 낸다. 입력데이터에 대해 지정된 간격(stride)으로 순회하며 채널별로 합성곱 연산을 하고

모든 채널의 합성곱의 합을 Feature Map으로 만든다. 필터는 입력받은 데이터에서 그 특성을 가지고 있으면 큰 값이 나오고 그렇지 않다면 0으로 수렴하는 양상이 있다. 즉, 데이터가 그 특성을 가지고 있는지 없는지의 여부를 합성곱 연산의 결과로부터 예상 가능하다. 입력데이터가 여러 채널을 가질 경우 필터는 각 채널별로 순회하며 합성곱을 계산한 후 각 채널별로 Feature Map을 만든다. 그 후, 각 채널의 Feature Map을 합산하여 최종적으로 한 개의 Feature Map을 반환한다. 합성곱 연산을 통해 얻은 Feature Map은 비선형성 추가를 위해 활성화 함수를 지나게 되는데, CNN에서는 주로 ReLU 함수를 사용한다. Padding(패딩)이란 Convolution Layer의 출력 데이터 크기가 줄어드는 것을 방지하기 위한 방법으로 입력 데이터의 외각에 지정된 픽셀만큼 특정값으로 채워 넣게 된다. 보통 0으로 채워 넣으며 이를 zero padding이라고 한다. Padding을 통해 출력 데이터의 사이즈 조절 뿐만 아니라 이미지 외각을 인식하는 학습의 효과도 얻을 수 있다. Bias(편향)은 필터를 적용한 후에 더해지는 파라미터이다.

2) 환경 설정

① Python 설치
https://www.python.org/

② Anaconda 설치
https://www.anaconda.com/

③ PyTorch 설치
https://pytorch.org/

파이토치는 앞서 설치한 아나콘다의 버전, OS, 플랫폼에 따라 설치 명령어가 다르므로 위 링크에서 확인 후 설치하도록 한다. 설치 후 아래 명령을 통해 아나콘다 프롬프트에서 버전 확인이 가능하다.

```
import torch
torch.__version__
```

■ 구현 및 결과분석

1) PyTorch를 이용한 CNN 설계 예제

Convolution Layer의 Output Data 크기는 아래의 식을 통해 계산할 수 있다.

$$(Output\ Height) = \frac{H + 2P - FH}{S} + 1$$
$$(Output\ Width) = \frac{W + 2P - FW}{S} + 1$$

H: Input Data Height
W: Input Data Width
FH: Filter Height
FW: Filter Width
S: Stride
P: Padding

Max Pooling Layer의 출력 데이터 크기는 아래의 식을 통해 계산할 수 있다.

$$(Output\ Height) = \frac{H}{Pooling\ Size}$$
$$(Output\ Width) = \frac{W}{Pooling\ Size}$$

가로가 31, 세로가 39, 채널이 1인 입력 데이터에 4번의 Convolution Layer와 3번의 Max Pooling Layer를 통과시키고 마지막 출력층에서 Fully Connected Layer를 통해 학습시킨 예제이다. Padding은 없다고 가정했으며, 아래의 그림/표와 같이 각 Layer를 구성한다.

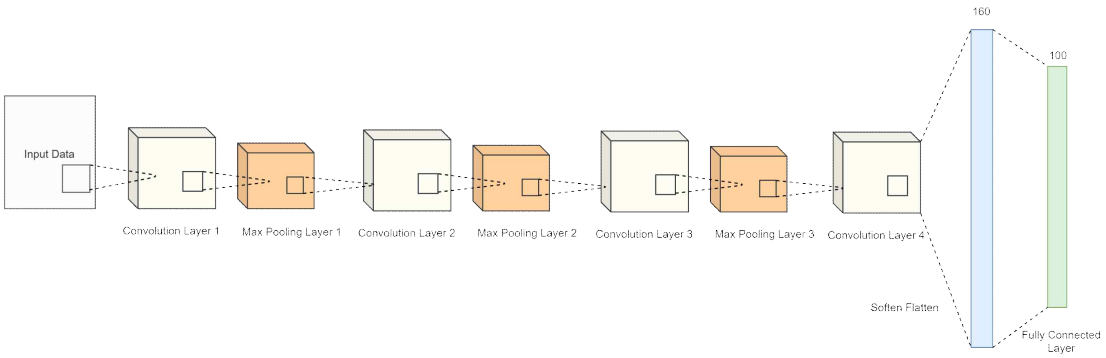
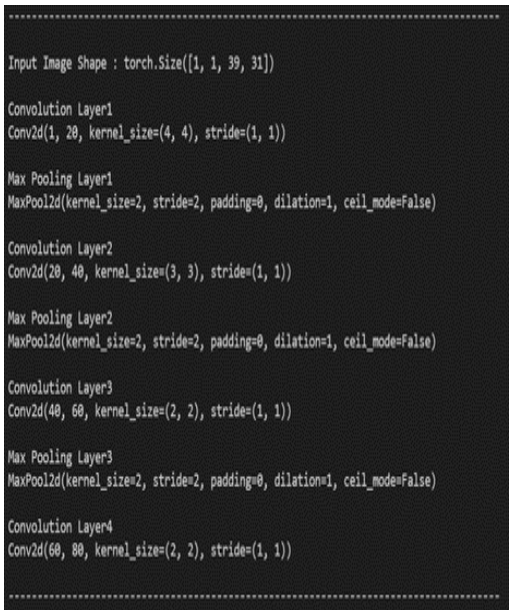


Fig. 1. 4-Layer CNN Structure

Table. 1. Parameter Information

Layer	Filter	Stride	Pooling	Activation Function	Input Shape	Output Shape
Convolution Layer1	(4, 4)	1	X	ReLU	(39, 31, 1)	(36, 28, 20)
Max Pooling Layer1	X	X	(2, 2)	X	(36, 28, 20)	(18, 14, 20)
Convolution Layer2	(3, 3)	1	X	ReLU	(18, 14, 20)	(16, 12, 40)
Max Pooling Layer2	X	X	(2, 2)	X	(16, 12, 40)	(8, 6, 60)
Convolution Layer3	(2, 2)	1	X	ReLU	(8, 6, 40)	(6, 4, 60)
Max Pooling Layer3	X	X	(2, 2)	X	(6, 4, 60)	(3, 2, 60)
Convolution Layer4	(2, 2)	1	X	ReLU	(3, 2, 60)	(2, 1, 80)
Flatten	X	X	X	X	(2, 1, 80)	(160, 1)
Fully Connected Layer	X	X	X	X	(160, 1)	(100, 1)

아래의 그림은 PyTorch에서 위 Layer에 따라 구축한 CNN에서 Input Data, Convolution Layer, Max Pooling Layer의 Shape과 각 층을 통과한 후 Output Data의 Shape과 최종 Data의 Shape 결과를 출력한 것이다.



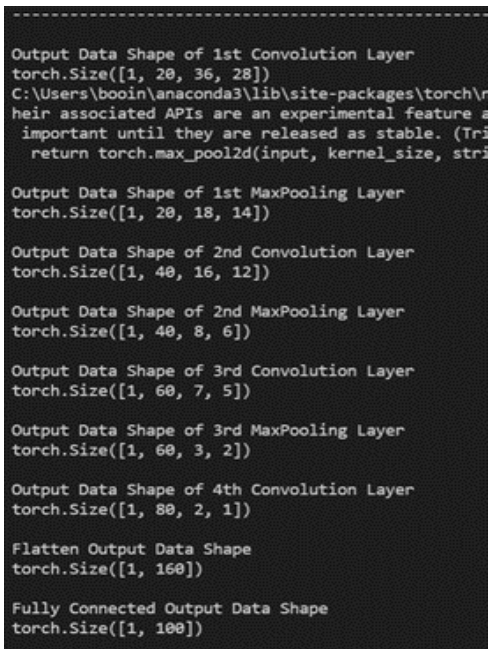


Fig. 2.
(a) Input Image & Convolution Layer Shape (b) Output Data Shape

2) PyTorch를 이용한 MNIST 데이터셋 분석

i) CNN Layer Optimization

PyTorch에서 데이터로더를 사용하여 MNIST 손글씨 데이터셋에 대한 학습을 진행한 결과이다. 파라미터는 아래 표와 같이 설정하였고, 각각 2개, 4개 층의 Layer를 이용해 학습을 진행하고 적중률을 확인하였다.

Table. 2. Parameter Setting for CNN

Batch Size	100
Training Epoch	15
Learning Rate	0.001
Optimizer	Adam
Activation Function	ReLU

(a) 2개의 Layer로 학습한 결과

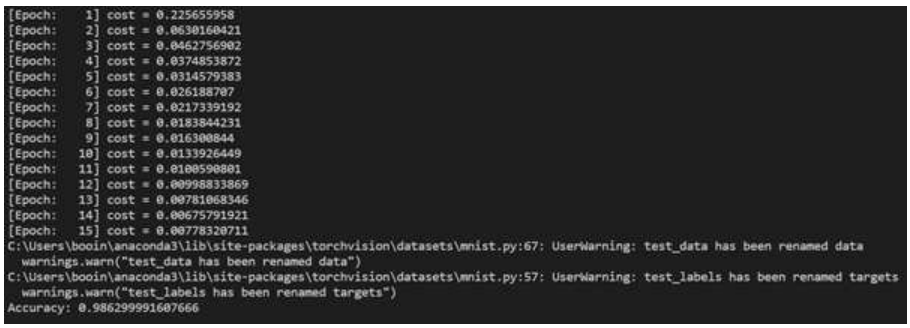


Fig. 3. Test Result of 2-Layer CNN

(b) 4개의 Layer로 학습한 결과

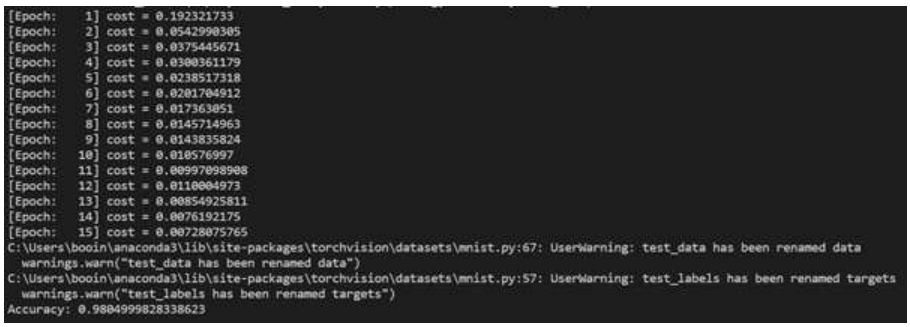


Fig. 4. Test Result of 4-Layer CNN

2개의 Layer로 학습하였을 때는 98.6%, 4개의 Layer로 학습하였을 때는 98%의 적중률을 가졌다. 더 많은 Convolution Layer를 통과시켰지만 오히려 적중률을 0.6% 떨어졌다. 결과적으로 Layer를 여러번 통과시킨다고 적중률이 항상 오르는 것을 아

니기 때문에 효율적으로 Layer를 쌓는 것이 중요하다. 즉, 환경에 따라 적절한 Learning rate, Batch Size, Training Epochs를 결정하여 안정적으로 학습이 가능하도록 하는 것이 바람직하다.

ii) Data Extraction from CNN

최소 94% 이상의 적중률을 가지면서 Layer를 가능한 적게 쌓아 용량이 작은 CNN 구조를 채택하였다. 채택한 CNN의 구조는 2-Layer로 parameter set은 다음과 같다.

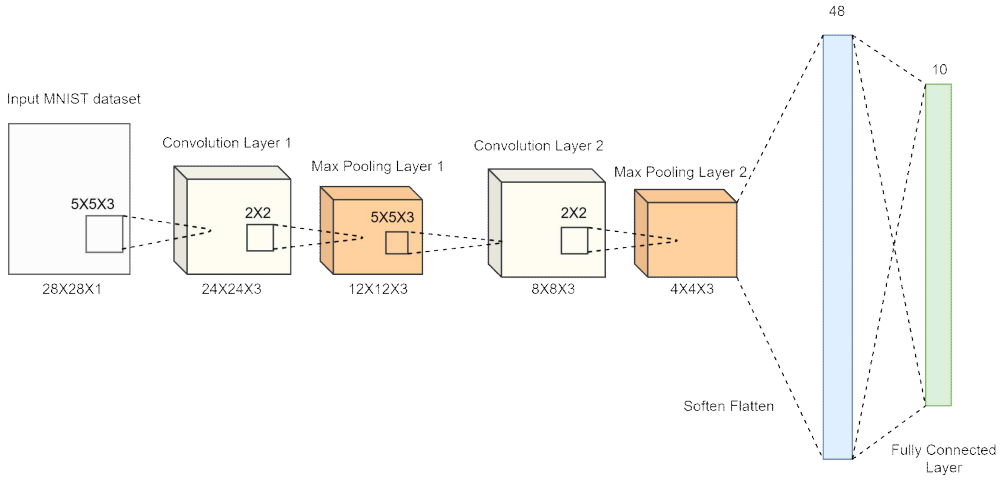


Fig. 5. 2-Layer CNN Structure for MNIST Dataset

Table. 3. Parameter Information

Layer	Filter	Stride	Pooling	Activation Function	Input Shape	Output Shape
Convolution Layer1	(5, 5, 3)	1	X	ReLU	(28, 28, 1)	(24, 24, 3)
Max Pooling Layer1	X	X	(2, 2)	X	(24, 24, 3)	(12, 12, 3)
Convolution Layer2	(5, 5, 3)	1	X	ReLU	(12, 12, 3)	(8, 8, 3)
Max Pooling Layer2	X	X	(2, 2)	X	(8, 8, 3)	(4, 4, 3)
Flatten	X	X	X	X	(4, 4, 3)	(48, 1)
Fully Connected Layer	X	X	X	X	(48, 1)	(10, 1)

파라미터는 아래의 표와 같이 설정하였다.

Table. 4. Parameter Setting for MNIST CNN

Batch Size	64
Training Epoch	10
Learning Rate	0.01
Optimizer	Stochastical Gradient Descent, Momentum = 0.5
Activation Function	ReLU

학습 결과 96.29%의 적중률을 보였다.

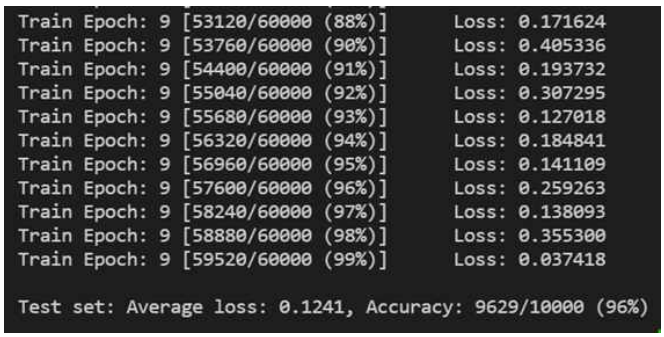


Fig. 6. Test Result of 2-Layer CNN for MNIST Dataset

해당 CNN model을 이용하여 RTL-level Verilog 설계를 위해 필요한 가중치(weight)와 편향(bias) 그리고 각 Layer의 Output Data를 추출하여 텍스트 파일로 저장한다. PyTorch에서 torchvision 패키지를 이용해 다운받은 MNIST dataset의 binary file을 big-endien 포맷으로 읽어 각 이미지파일을 bitmap file로 저장한다. bmp file을 읽어 28X28 크기로 reshape한 후 해당 이미지에 대한 학습을 진행한다. 각 Convolution Layer에서의 weight와 bias에 128(2^7)를 곱하고, 2의 보수를 취한 int 형으로 각각 텍스트 파일에 저장한다. 아래의 그림은 CNN결과 추출된 첫 번째 Convolution Layer에 대한 3개 채널의 weight와 bias이다.

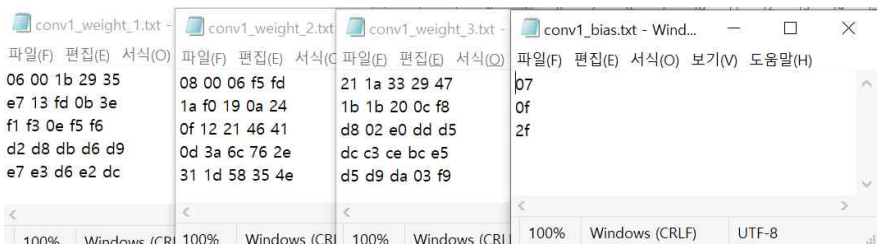


Fig. 7. Extracted weight & bias value from 1st Layer of CNN

또한, 후에 설계된 CNN을 검증하기 위해 각 Layer를 통과시킨 후 출력 데이터의 Feature Map 또한 위와 유사한 방법으로 calibration한 후 아래와 같이 텍스트 파일로 저장한다.

```
np.savetxt('out_conv1_value_1.txt', model.conv1_out_np[0][0]*128, fmt='%1.5d',delimiter = " ")
np.savetxt('out_conv1_value_2.txt', model.conv1_out_np[0][1]*128, fmt='%1.5d',delimiter = " ")
np.savetxt('out_conv1_value_3.txt', model.conv1_out_np[0][2]*128, fmt='%1.5d',delimiter = " ")
```

Fig. 8. Save Output Data of 1st Convolution Layer

3) Verilog Block Diagram

CNN의 Top Block Diagram은 다음과 같다. Convolution 연산을 하는 모듈, Max Pooling 모듈, ReLU 모듈, Fully-Connected 모듈의 sub-block으로 구성되어 있다.

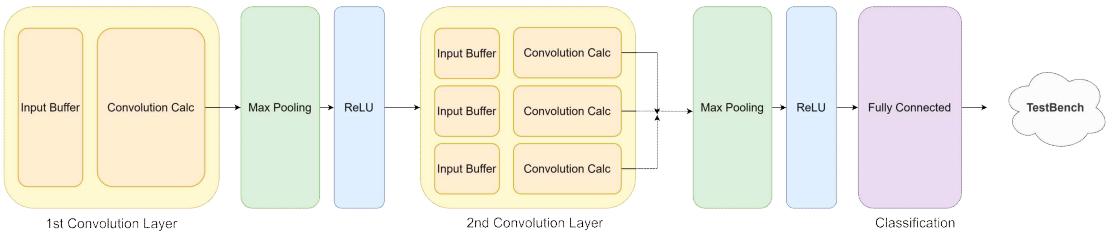


Fig. 9. CNN Block diagram

A. Top Testbench

- System의 clock을 initial block을 이용하여 구현하고, timescale을 선언한다.
- 각종 파라미터를 선언한다.
- 각 sub module을 instantiation하고 각각의 input/output 변수를 적절히 연결한다.
- 학습 파라미터(가중치, 편향)는 텍스트 파일를 읽어 값을 저장한다.
- Output Data value를 Buffer에 담고, comparator를 통해 학습의 결과가 적중하였는지 판단한다.

B. Convolution Calculation Module

- Input Buffer를 통해 유입된 입력 데이터에 대해, PyTorch 학습 결과로 얻은 가중치값(weight)과 편향(bias)을 이용하여 합성곱 연산을 수행하여 output data로 출력한다.
- 2nd layer에서는 채널이 3개이므로 Input Buffer와 Convolution module이 세 개씩 선언되어야 한다.

C. Max Pooling Module

- Pooling Layer는 Max Pooling을 사용하기 때문에 Max Pooling 필터 사이즈에 따라 입력 데이터의 영역을 나누고 해당 영역에서 data를 차례로 비교하며 max값을 찾아 출력 데이터로 저장하도록 한다.

D. ReLU Module

- Max Pooling 출력 데이터의 값이 음수인 경우 0을 출력한다.

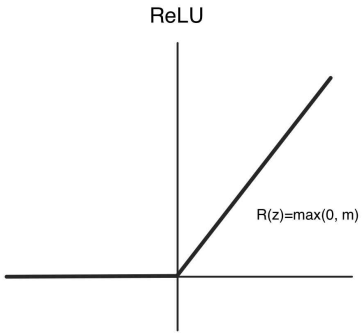


Fig. 10. ReLU Function

E. Fully Connected Module

- ReLU 함수의 출력 데이터를 Buffer에 담아 Flattening 하여 1차원 벡터로 저장한다.
- 10*48 = 480 개의 가중치를 input과 곱하여 10개의 output을 계산한다(softmax). 그 값중 가장 높은 확률값을 갖는 class가 해당 입력 이미지와 일치하는 class가 된다.

■ 결론 및 소감

현재 진행현황은 CNN 모델에 대한 내용 숙지/학습과 PyTorch 프레임워크에서 MNIST 데이터에 대한 CNN 구현과 학습, 데이터 추출, Verilog 개요 작성까지 마친 단계이다. CNN 동작 원리 및 사용되는 용어 등에 대한 학습을 먼저 진행하였고, PyTorch 프레임워크 환경을 설정하고 문법을 학습하였다. 다양한 예시를 통해 CNN 모델을 디자인해보고, 각 Layer마다 필터를 거쳐 나오는 Output Data의 폭/높이/채널 등을 계산하며 예측하고 직접 출력해보며 검증을 하였다. 그 결과 Layer가 무조건 깊다고 적중률이 올라가는 것은 아니라는 사실을 알 수 있었다. 그로부터 적중률 94% 이상이라는 조건을 만족하는 최적화된 MNIST 데이터셋 CNN 모델을 결정하였고, 해당 모델에 대한 파라미터를 추출하여 저장하였다.

향후 계획은 작성된 설계 개요에 따라 각 sub-block에 대한 Verilog Design을 진행하고 testbench를 설계하여 CNN을 검증할 것이다. 또한, waveform을 통해 인식률 및 인식에 소요되는 시간을 측정할 것이다.

■ 참고문헌

[1] Vishnu Subramanian, "Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch", Birmingham : Packt Publishing, 2018.

[2] He X, Xu S. Artificial neural networks[J]. Process Neural Networks: Theory andApplications, 2010: 20-42.

[3] Artificial neural networks[M]. Humana Press, 2015.

[4] Pradhan B, Lee S. Landslide susceptibility assessment and factor effect analysis:backpropagation artificial neural networks and their comparison with frequency ratioand bivariate logistic regression modelling[J]. Environmental Modelling & Software,2010, 25(6): 747-759.

[5] Wang G, Hao J, Ma J, et al. A new approach to intrusion detection using ArtificialNeural Networks and fuzzy clustering[J]. Expert systems with applications, 2010,37(9): 6225-6232.

[6] Choi, Jae-Seung. "Voiced-Unvoiced-Silence Detection Algorithm usingPerceptron Neural Network." The Journal of the Korea institute of electroniccommunication sciences 6.2 (2011): 237-242.

[7] Lee, Jae-Eun, Seo, Young-Ho, Kim, Dong-Wook "Convolutional Neural Network 기반의 워터마킹 프로세서의 설계", 2020년 한국방송, 미디어공학회 추계학술대회, November (2020), pp. 106-107

[8] Convolutional Neural Networks. Stanford.edu, [Internet] Available : https://cs231n.github.io/convolutional-networks/

[9] R. Xiao, J. Shi and C. Zhang, "FPGA Implementation of CNN for Handwritten Digit

Recognition," 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2020, pp. 1128-1133