Regular paper | Journal of the KIECS. pp. 417-426, vol. 19, no. 2, Apr. 30. 2024, t. 124, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2024.19.2.417

유전 알고리즘을 이용한 임베디드 프로세서 기반의 머신러닝 알고리즘에 관한 연구

이소행* · 석경휴**

A Study on Machine Learning Algorithms based on Embedded Processors Using Genetic Algorithm

So-Haeng Lee* · Gyeong-Hyu Seok**

요 약

일반적으로 머신러닝을 수행하기 위해서는 딥러닝 모델에 대한 사전 지식과 경험이 필요하고, 데이터를 연산하기 위해 고성능 하드웨어와 많은 시간이 필요하게 된다. 이러한 이유로 머신러닝은 임베디드 프로세서에서 실행하기에는 많은 제약이 있다.본 논문에서는 이러한 문제를 해결하기 위해 머신러닝의 과정 중 콘볼루션 연산(Convolution operation)에 유전 알고리즘을 적용하여 선택적 콘볼루션 연산(Selective convolution operation)과 학습 방법을 제안한다. 선택적 콘볼루션 연산에서는 유전 알고리즘에 의해 추출된 픽셀에 대해서만 콘볼루션을 수행하는 방식이다. 이 방식은 유전 알고리즘에서 지정한 비율만큼 픽셀을 선택하여 연산하는 방식으로 연산량을 지정된 비율만큼 줄일 수 있다. 본 논문에서는 유전 알고리즘을 적용한 머신러닝 연산의 심화학습을 진행하여 해당 세대의 적합도가 목표치에 도달하는지 확인하고 기존 방식의 연산량과 비교한다. 적합도가 충분히 수렴할 수 있도록 세대를 반복하여 학습하고, 적합도가 높은 모델을 유전 알고리즘의 교배와 돌연변이를 통해 다음 세대의 연산에 활용한다.

ABSTRACT

In general, the implementation of machine learning requires prior knowledge and experience with deep learning models, and substantial computational resources and time are necessary for data processing. As a result, machine learning encounters several limitations when deployed on embedded processors. To address these challenges, this paper introduces a novel approach where a genetic algorithm is applied to the convolution operation within the machine learning process, specifically for performing a selective convolution operation.In the selective convolution operation, the convolution is executed exclusively on pixels identified by a genetic algorithm. This method selects and computes pixels based on a ratio determined by the genetic algorithm, effectively reducing the computational workload by the specified ratio. The paper thoroughly explores the integration of genetic algorithms into machine learning computations, monitoring the fitness of each generation to ascertain if it reaches the target value. This approach is then compared with the computational requirements of existing methods.The learning process involves iteratively training generations to ensure that the fitness adequately converges.

키워드
Genetic Algorithm, Machine Learning, Embedded Processor, Cortex
유전 알고리즘, 머신러닝, 임베디드 프로세서, Cortex

* 순천대학교 SW중심대학사업단
**순천대학교 SW중심대학사업단(khseok@scnu.ac.kr)
· 접 수 일 : 2024. 02. 03
· 수정완료일 : 2024. 03. 08
· 게재확정일 : 2024. 04. 12
· Received : Feb. 03, 2024, Revised : Mar. 08, 2024, Accepted : Feb. 12, 2024
· Corresponding Author : So-Haeng Lee
Suncheon National University SW Centered University Project Group
Email : imac@scnu.ac.kr

Ⅰ. 서 론

대부분의 딥러닝(Deep Learning)은 인공 신경망(ANN : Artificial Neural Network), 심층 신경망(DNN : Deep Neural Network), 콘볼루션 신경망(CNN : Convolution Neural Network), 순환 신경망(RNN : Recurrent Neural Network) 등을 기반으로 설계되었고, 대량의 데이터를 통한 학습은 신경망의 신경을 구성하는 것과 유사하다. 학습이 완료되면 새로운 데이터를 처리하기 위한 모델이 완성되고, 완성된 모델도 끊임없이 계속되는 누적 학습을 통해 연산의 깊이와 정밀도가 향상된다. 이러한 인공지능 기술을 다양한 분야에 응용하기 위해서는 고속 연산에 특화된 전용 GPU(: Graphic Processing Unit)를 장착한 그래픽카드와 대용량의 컴퓨터 메모리가 필요하고, 학습을 위한 연산시간과 전력 소비를 줄이는 것이 관건이다. 특히 모바일, 사물 인터넷(Internet Of Things) 및 웨어러블(Wearable) 장치 등과 같은 임베디드 시스템(Embedded System)에 머신러닝을 실행할 경우, 임베디드 프로세서의 전력 소비, 메모리 및 계산 기능 측면에서 자원이 한정되어 있기 때문에 연산 횟수를 효과적으로 줄일 수 있는 효율적인 머신러닝 알고리즘의 설계가 필수적이다.

일반적으로 머신러닝 알고리즘의 효율을 개선하기 위한 접근 방식 중의 하나로 유전 알고리즘(Genetic Algorithm)을 적용하는 방법이 있다. 유전 알고리즘은 최적의 경로를 찾기 위해 자연 진화 과정을 모방한 검색 알고리즘으로, 머신러닝 알고리즘의 성능 최적화 뿐만 아니라 다양한 산업 분야에서 최적화 문제에 적용하기 위해 응용되고 있다.

유전 알고리즘은 정확한 답을 찾는 것이 아니라 답에 가까운 결과를 찾기 위해 만들어진 알고리즘이다. 연산량을 획기적으로 줄이기 위해 유전의 과정을 일련의 알고리즘으로 해석해서 문제에 적용하는 방식으로, 반복 횟수를 늘릴수록 답에 가까운 결과를 얻거나 정확한 답을 얻을 수 있어서 효율 측면에서는 효용성이 비교적 낮은 편이다. 그러나 제한된 자원을 사용하여 답에 가까운 결과를 빠르게 얻어 낼 수 있으므로 적절한 알고리즘을 사용한다면, 임베디드 프로세서에도 유용하게 적용할 수 있다.

본 논문에서는 제한된 연산환경에서 머신러닝 알고리즘의 성능을 최적화하기 위해 유전 알고리즘을 적용한 임베디드 프로세서 기반의 머신러닝 알고리즘을 제안하고 구현하였다. 그리고 제안한 알고리즘을 기존 머신러닝 알고리즘과 연산시간, 정확도, 전력 소모 등과 같은 측면에서 비교 평가하였다.

제안한 방식의 검증을 위해 본 논문에서는 ARM사의 Cortex Core 기반 임베디드 프로세서 모델인 M4를 사용하였다. M4 프로세서는 부동소수점 계산을 위한 하드웨어 FPU(: Floating Point Unit)를 가지고 있는데, 이는 딥러닝 연산 시 부동소수점 계산 시간에 많은 영향을 주기 때문에 반드시 필요로 한다. 그리고 검증을 위해 60,000개의 트레이닝 세트와 10,000개의 테스트 세트로 이루어진 MNIST 데이터베이스(Modified National Institute Of Standards And Technology Database)를 사용하였다. 이중 임베디드 프로세서의 내부 메모리를 고려하여 제한된 갯수의 트레이닝 세트와 테스트 세트, 그리고 직접 손으로 쓴 숫자 세트 100개를 수집하여 테스트를 진행하였다.

또한 본 논문에서는 딥러닝 과정 중 콘볼루션 연산에 유전 알고리즘을 적용하여 선택적 콘볼루션 연산(Selective Convolution Operation)과 학습 방법을 제안하였다. 본 논문에서 제안한 콘볼루션 연산 과정은 전체 픽셀에 연산을 수행하는 것이 아니라 유전 알고리즘을 적용하여 지정된 비율의 픽셀에만 연산을 취하는 방식으로 연산을 수행하게 하였다. 그리고 테스트 세트에 제안된 알고리즘을 적용하여 만들어진 학습모델의 정확도를 평가하였다.

Ⅱ. 기술 개발 내용

2.1 유전 알고리즘

유전 알고리즘은 생물학적 진화 과정에 기초한 계산 모델로 John Holland에 의해서 1975년에 개발된 전역 최적화 기법으로, 최적화 문제를 해결하는 진화 연산의 대표적인 기법이다[1]. 이 알고리즘은 실제 진화의 과정에서 많은 부분을 차용 하였으며, 변이(돌연변이), 교배 연산 등이 존재한다[2].

유전 알고리즘에서는 잠재적 최적 모집단 데이터가 생성되고 각 데이터는 일련의 매개변수 또는 유전자로 표시된다. 그런 다음 당면한 문제를 얼마나 잘 해

결하는지 측정하는 적합도 함수를 기반으로 데이터를 평가한다. 그리고 적합도가 가장 높은 데이터를 선택하여 유전적 변이 및 교차 과정을 통해 생성되는 새로운 데이터를 번식(breed)이라 하고 이를 생성하는 과정이다. 이러한 선택, 유전적 변이 및 교차 과정은 여러 세대에 걸쳐 반복되며 각 세대는 잠재적 데이터의 새로운 개체군을 생성하고, 시간이 지남에 따라 유전 알고리즘은 기능을 최적화하고 그림 1에서와 같이 문제를 해결하는 데이터로 수렴하게 된다.

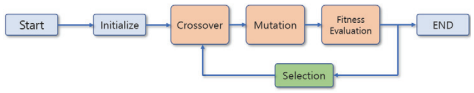

그림 1. 유전 알고리즘 플로우차트
Fig. 1 Flowchart of genetic algorithm

2.2 오차 역전파

화각 및 해상도 차이에 따른 scale 비율 설정하고, RGB 및 depth 센서에서 같은 해상도(1280x720)로 촬영 시 화각 차이가 있고, 역전파의 이해를 위해 사용할 인공 신경망은 입력층(x_1, x_2), 은닉층(h_1, h_2), 출력층(o_1, o_2)으로 나뉜다. 역전파는 입력 데이터를 통해 신경망의 출력값을 계산하는 과정이므로 순전파(forward propagation), 손실 함수(loss function) 계산, 역전파, 가중치 업데이트 과정을 거친다. 이를 통해서 신경망의 오차를 최소화하는 방향으로 가중치를 조정하고, 이 과정을 반복적으로 수행하면 신경망의 가중치가 최적화되고 모델의 성능이 향상된다. 그림 3-9에서는 2개의 입력, 2개의 은닉층 연결, 2개의 출력층 연결을 사용한다. 그림 2에서와 같이 은닉층(hidden layer)과 출력층에서는 시그모이드 함수를 활성화 함수로 사용 한다.

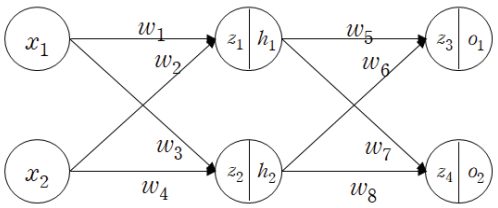

그림 2. 인공신경망의 예
Fig. 2 Example of ANN[3]

은닉층과 출력층의 모든 연결에서 변수 z가 존재하는데, 여기서 변수 z는 이전 층의 모든 입력이 각각의 가중치와 곱해진 값들이 모두 더해진 가중치의 합을 의미한다. 이 값은 연결에서 아직 시그모이드 함수를 거치지 않은 상태이다. z 우측의 "|" 를 지나서 존재하는 변수 h 또는 z가 시그모이드 함수를 지난 후의 값으로 각 연결의 출력값을 의미한다.

2.3 유전 알고리즘 기반 선택적 콘볼루션 방법의 제안

유전 알고리즘 기반 선택적 콘볼루션 방법은 이미지 처리 및 컴퓨터 비전 작업에서 콘볼루션 신경망의 성능을 최적화하는 것을 목표로 하는 새로운 접근 방식이다. 이 방법은 유전 알고리즘을 활용하여 주어진 작업에 가장 적합한 콘볼루션 계층과 필터를 선택하고 조정하여 계산 복잡성을 줄이고 네트워크의 전반적인 효율성을 유지하기 위해 제안되었다. 머신러닝에서 이미지를 인식하는 과정은 앞서 설명한 것처럼 다양한 방법이 존재하지만, 본 논문에서는 LeNet5[4] 알고리즘에서 제시된 방법에 유전 알고리즘을 적용하여 연산 효율을 개선하는 방법을 제안하였다.

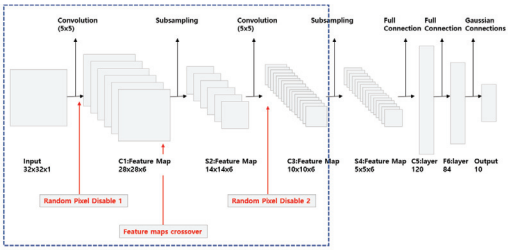

그림 3. 유전 알고리즘 기반 선택적 콘볼루션 방법의 제안
Fig. 3 Proposed genetic algorithm-based selective convolution method

그림 3에서 유전 알고리즘이 적용되는 부분은 첫 번째 콘볼루션 연산이 실행되는 위치, 특성맵이 생성되는 위치, 두 번째 콘볼루션 연산이 실행되는 위치이다. random pixel disable_1의 위치에서는 이미지에 연산 되지 않는 마스크 이미지를 생성해서 원본 이미지에 연산 되지 않을 위치를 표시하는 과정이고, 두 번째 feature maps crossover는 6장의 이미지를 특정

위치에서 교차시켜 새로운 특성맵을 생성하는 과정이다. 이 과정에서는 특성맵의 개수는 6개를 그대로 유지한다. 세 번째 random pixel disable_2에서는 14×14 크기로 줄어든 특성맵에 첫 번째와 같은 방법으로 연산하지 않을 픽셀을 표시한다. 표시된 픽셀은 콘볼루션 연산에서 제외하여 연산량을 줄이고자 하였다.

이 이미지에 콘볼루션 연산을 수행하기 위해 LeNet5 에서는 6개의 5×5 크기의 커널(또는 필터)을 생성하는데, 초기값은 가우시안 분포식[5]을 사용하여 무작위 숫자로 채워 넣는다. 가우시안 분포식은 식 (3-12)와 같고, 여기서 μ는 가우시안 분포의 평균, σ는 표준 편차이다[6]. 실험에서는 μ와 σ 모두 1로 설정하였다. 표준 편차가 클수록 더 넓은 범위에 골고루 값이 식 (1)과 같이 생성된다.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \cdots (1)$$

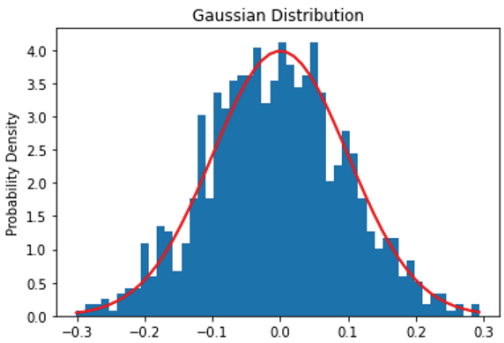

그림 4. 가우시안 분포와 식
Fig. 4 Gaussian distribution

여기서 가우시안 분포는 그림 4에 나타낸 바와 같이 평균값인 0을 중심으로 대칭 형태이고, 이 분포는 확률이 아니라 확률밀도(probability density function)를 의미한다. 즉 중심인 0에 가까울수록 발생 될 확률이 높아진다[7].

그림 5는 가우시안 평균 방법으로 선정한 랜덤 마스크(random mask)를 시각화한 예이다. 전체 픽셀 중 1로 마킹(marking)된 위치는 콘볼루션 연산에서 제외되는 것을 의미한다. 구현 단계에서는 연산에서 제외될 위치를 대조하여 대조된 위치는 연산하지 않고 0 값으로 정의하였다. 이 과정을 통해서 콘볼루션 전체 연산량을 줄일 수 있다. 이러한 방법은 이미지에 노이즈가 있는 것과 유사하여 연산의 결과에 약간의 영향을 미칠 수는 있으나 전체 픽셀의 일부만 노이즈로 인식하게 되므로 최종 결과에는 큰 영향이 없는 것을 확인하였다. 다만 무작위 위치를 선정하는 과정에서 가우시안 분포를 사용하면 표준 편차가 작을 때 이미지의 중앙에 더 많은 픽셀이 선택되는데, 이러한 경우에는 연산의 결과에 영향을 미칠 수 있게 된다. 그래서 연산에서 제외될 픽셀을 선택하는 방법은 가우시안 분포를 사용하되 표준 편차를 크게 하여 선택되는 픽셀이 넓게 분포되게 하거나 일반적인 랜덤 발생 방법을 사용하여 픽셀을 선택하는 방법을 사용하여 테스트하였는데 최종적으로는 가우시안 분포의 표준 편차를 크게 하는 방법을 선택하였다.

그림 5. 가우시안 분포로 선택된 랜덤 마스크
Fig. 5 Random mask selected by the Gaussian mean method

그림 6에 나타낸 random pixel disable_1 과정에서는 random position map을 이용하여 입력 이미지에 pixel disable을 적용한다. pixel disable이 적용된 픽셀은 콘볼루션 연산에서 제외되고 연산 결과는 0으로 적용한다. 따라서 매 픽셀을 연산하기 전에 disable 되어 있는지 확인하는 과정이 필요하고, 이때 disable을 의미하는 값은 255로 설정하였다. 그리고 앞에서 설명한 random position map은 원본 이미지에서 연산 되지 않을 위치를 설정하는 맵이다. 이 비율을 각각 3%, 5%, 7%로 올리면서 연산을 수행하여 연산 결과를 측정하였다. 비율이 높을수록 연산에서 제외되는 픽셀의 개수가 늘어나 총 연산시간에 영향을 미치게

된다. 이러한 방법은 영상에 일부러 노이즈를 추가하여 연산을 수행하게 하는 것과 유사하나, 노이즈에 해당하는 픽셀을 연산에 그대로 사용하는 방법과는 다르게 노이즈에 해당하는 픽셀을 연산에서 제외하는 것으로 각 노이즈에 해당하는 픽셀의 콘볼루션 값을 0으로 연산하는 것이다.

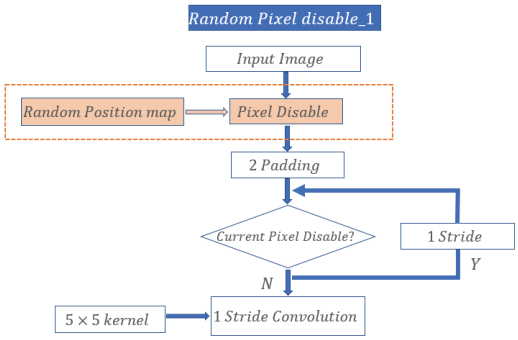

그림 6. 랜덤 픽셀 삭제를 위한 플로우차트
Fig. 6 Floatchart of random pixel disable_1

3가지 비율로 연산을 테스트한 것은 연산의 최종 결과가 인식률에 미치는 영향, 즉 시간과 정확도를 확인하기 위해서다. 여기서 비율을 더 크게 올리지 않는 이유는 실험에서 약 10% 이상 변형하였을 때 미치는 영향이 커서 적용하는데 부적절하기 때문이었다. 이 과정은 유전 알고리즘의 돌연변이와 유사하지만, 일반적인 유전 알고리즘의 돌연변이 확률에 비해 더 많은 확률로 변형을 시도하였고 방법 역시 비트 반전 연산과는 다른 방법인 pixel disable을 사용하였다.

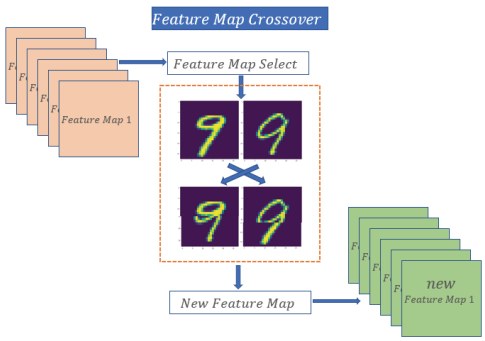

그림 7. 특성맵 교차연산을 위한 플로우차트
Fig. 7 Floatchart of feature map crossover

그리고 feature map crossover 연산은 만들어진 특성맵 중 무작위로 2개의 맵을 선택하여 교차 알고리즘을 적용한다. 특정한 위치를 기준으로 2개의 특성맵 값을 교차하여 바꾸는 연산이다. 그림 7에서 2개의 선택된 후보는 무작위로 선택된 위치에서 교차 연산을 수행한 뒤 새로운 특성맵으로 생성된다. 생성된 특성맵은 새로운 특성이 있으므로 기존의 학습 파라미터에도 영향을 미치고, 순전파 과정에서 새로운 특성맵의 특징이 파라미터에 적용된다. 그리고 역전파 되는 과정에서 특성맵의 파라미터는 좀 더 미세하게 수정되나 이는 전체 연산시간에는 적은 부분이다. 또한 특성맵의 교차 연산을 통해 더 많은 특성맵을 만들어 파리미터를 조정할 수도 있지만, 앞에서 정한 6개의 특성맵보다 더 많은 특성맵을 만들었을 경우 먼저 진행한 Random Pixel Disable_1 연산에서 줄인 시간을 유지할 수 없게 된다. 따라서 본 논문에서는 교차 연산으로 만들어 내는 특성맵은 2개의 맵만 교차 연산을 시행하여 적용하였다.

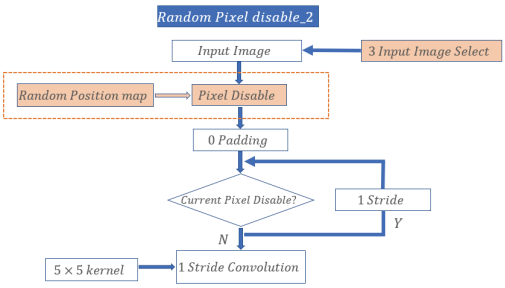

그림 8. 랜덤 픽셀 삭제를 위한 플로우차트
Fig. 8 Floatchart of random pixel disable_2

그림 8에 나타낸 random pixel disable_2 연산을 실행하기 전에 서브샘플링 방법 중 최대 풀링 방법으로 28×28 크기의 특성맵 이미지를 14×14 크기로 줄이는 과정을 진행하였다[8]. 이 과정에서 다시 3개의 특성맵만 골라 random pixel disable_1에 적용했던 방법과 동일하게 픽셀 disable 하였다.

그리고 6개의 특성맵을 모두 적용하지 않고 3개에만 적용한 것은, 이미지 크기가 14×14 크기로 줄어들어 모든 맵에 적용할 경우 연산의 정확도에 영향을 미칠 수 있기 때문이다. 그리고 Random pixel disable_1의 연산과 다른 점은 제로 패딩(zero

padding)으로, 28×28 크기를 유지하기 위해 패딩을 2를 사용했던 것과 달리 제로 패딩을 사용하는 경우 상하좌우에서 각각 2 픽셀씩 이미지의 크기가 줄어들게 된다. 14×14 크기의 특성맵은 위의 연산을 수행하고 나면 10×10 크기로 줄어들게 된다. 이 과정에서도 연산량이 감소하여 연산시간을 줄일 수 있다.또한 그림 3-14에서 제시한 방법은 S2 특성맵 까지만 유전 알고리즘 중 돌연변이와 교차 알고리즘을 적용하여 연산하였다.

S2 특성맵 이후 연산에 유전 알고리즘을 적용하지 않은 이유는 최대 풀링 연산을 거치면서 이미지의 크기가 급격히 줄어들고 1차원 평탄화 과정으로 인해 원래 이미지의 각 위치의 특성이 모두 사라지기 때문에 픽셀 하나의 영향력이 더욱 커지게 된다.

2차원 이미지를 1차원으로 변형하는 평탄화에서 다음 단계로 연결되는 weight, bias는 초기에는 모두 -1~1사이의 랜덤한 실수형 값을 갖는다. 이 값들은 역전파 과정에서 미세 조정이 이루어지는 과정에서 특성에 맞는 수정이 이루어진다.

사전 테스트 실험 결과, C3 특성맵과 S4 맵에도 random pixel disable 연산을 수행하였을 때 연산량의 급격한 변화로 인해 최종 정확도에 상당히 많은 영향이 있는 것으로 나타났다. 또, C4 특성맵에서 C5로 변환되는 전 연결과정에서는 기존의 특성맵의 성질은 모두 사라지게 된다. 그리고 C4단계에서 5×5×16＝400 개의 1차원 배열이 생성되고, 이것은 120개의 1차원 배열로 줄어드는 단계를 거치는데 여기서 전 연결을 통해 연산을 수행하게 되고, 다시 120개의 1차원 배열은 82개의 1차원 배열로 줄어들고 같은 방법으로 전 연결을 통해 연산을 수행하게 된다.

여기서 82개의 1차원 배열로 변형하는 이유는 ASC코드의 비트맵이 7×12의 크기[9]를 갖고 있기 때문에 이 비율을 유지하기 위해 생성된 1차원 배열이다.따라서 전 연결과정에 임의의 픽셀을 같은 방법으로 disable 하였을 경우, 연산을 수행한 뒤 역전파 단계에서 정상적인 역전파 보상이 이루어지지 않아 과정을 반복할수록 오차의 범위가 커지거나 과적합 또는 지역에 국한된 최적화 오류에 빠지게 된다[10].

따라서 본 논문에서는 2번의 무작위 픽셀 연산제외 방법인 돌연변이 연산과 1번의 교차 연산만을 사용하여 실험하여 결과를 도출하고자 하였다.

III. 시스템 개발 내용

3.1 유전 알고리즘 기반 콘볼루션 연산 실험 및 결과

그림 9는 본 논문에서 제안한 알고리즘을 임베디드 시스템에서 입증하고 평가하기 위한 실험 구성도이다. 실험에 사용된 마이크로프로세서는 그림 9에 나타낸 STMicroelectronis사의 STM32F446RE (Cortex-M4기반) 프로세서를 사용하였다. 이 프로세서는 512KB의 플래시 메모리, 128KB SRAM을 가지고 있다. 개발에 사용된 프로그래밍 언어는 C언어를 사용하였으며, 개발을 위한 IDE는 그림 9에 나타낸 STM32CubeIDE를 사용하였고, FPU 연산을 위해 CMSIS-DSP 라이브러리를 사용하였다.

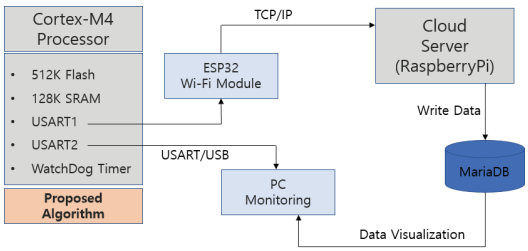

그림 9. 실험을 위한 시스템 구성
Fig. 9 Experimental system configuration diagram

사용된 Nucleo-F446RE 보드는 디버깅을 위해 ST-Link를 포함하고 있고, ST-Link는 외부 장치와 통신을 위해 USB를 통해 시리얼 통신을 한다. 실험에서는 115,200bps의 속도로 PC와 통신을 하며 데이터 모니터링을 위해 EBTerminal을 사용하였다. 또 데이터 수집을 위한 장치로 ESP32 device를 사용하여 WiFi를 통해 라즈베리파이 서버에 설치된 mariaDB에 데이터를 저장하였고, 저장된 데이터를 파이썬(python)으로 읽어 matplotlib를 사용하여 그래프로 표현하였고 표현한 데이터는 10개 평균을 계산하여 100회씩 출력하였다.

학습을 위해서 기술한 MNIST[11]의 데이터를 학습 데이터로 사용하였고, 평가에 사용된 데이터는 손

으로 쓴 숫자 세트 100개를 수집하여 28×28 크기로 변환한 뒤 테스트 데이터로 사용하였다. 그림 10은 손으로 쓴 평가용 데이터이다. 실험에 사용된 이미지의 특성은 MNIST의 데이터와 큰 차이가 없다. 원본은 200×200 픽셀 크기의 이미지를 28×28 크기로 줄인 뒤 MNIST와 같은 포맷으로 변형하여 사용하였다.

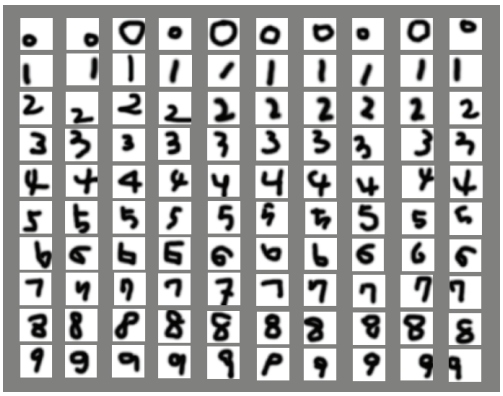

그림 10. 실험을 위한 실험 데이터
Fig. 10 Numeric data for evaluation

제안된 알고리즘이 적용되지 않은 기존 알고리즘(traditional algorithm)의 연산시간 및 제안된 알고리즘(proposed algorithm)이 적용된 연산시간은 필터 값, 각 픽셀 값에 따라 약간의 차이는 있으나 각각 최대 0.7%의 차이 내에서 연산시간이 측정되었다.

연산시간을 측정하기 위해서 Cortex-M4 프로세서의 타이머를 활용하였는데 시스템의 기본 클럭 틱(tick)인 ms(milisecond)를 사용하지 않고 μs(microsecond)를 사용하여 정밀하게 측정하였다[12-13]. 측정한 값의 연산은 소수점 이하 3자리까지만 획득하여 연산 결과에 반영하였다. 기존 알고리즘 연산의 경우 첫 번째 단계에서 평균 연산시간은 약 3.30ms ~ 3.35ms가 소요 되었고, 3.4절에서 제시한 알고리즘을 통해 전체 픽셀 중 7%를 제외하고 연산한 결과는 약 3.20ms ~3.25ms의 시간이 소요 되었다.

그림에서 보듯이 제외되는 픽셀의 비율이 낮을수록 연산시간은 기존 알고리즘에 비해 큰 차이를 보이지 않는 결과를 얻을 수 있었다. 다만 5%와 7%의 차이는 5%와 3%의 차이에 비해 연산 시간의 차이가 작았는데, 이는 MCU 특성상 중복되거나 반복되는 연산 중 동일한 연산은 다음 연산에서 자동으로 제외하는 최적화 특성으로 인해 생기는 문제이다. 그러나 연산의 결과는 유의미한 차이가 있으므로 그대로 테스트에 적용하여 사용하였다. 그림 11, 그림 12 및 그림 13은 random pixel disable_2에서 제시한 방법으로 전체 픽셀 중 7%, 5%, 3%를 각각 제외하고 연산한 실험 결과이다.

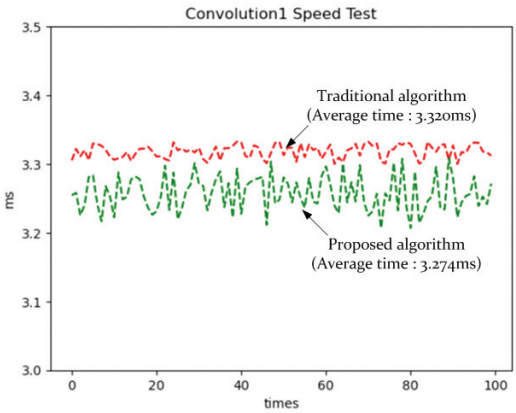

그림 11. 7% 픽셀 삭제를 적용하였을 때 연산
Fig. 11 Computation time when applying 7% pixel disable_1

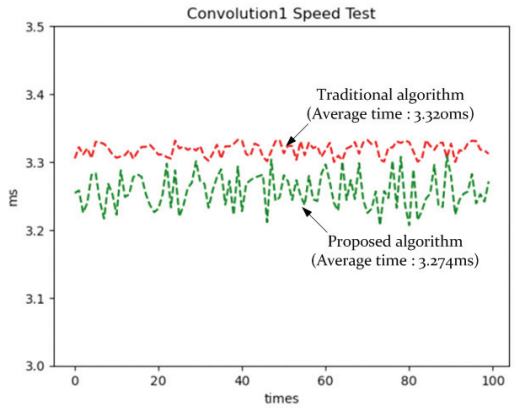

그림 12. 5% 픽셀 삭제를 적용하였을 때 연산
Fig. 12 Computation time when applying 5% pixel disable_1

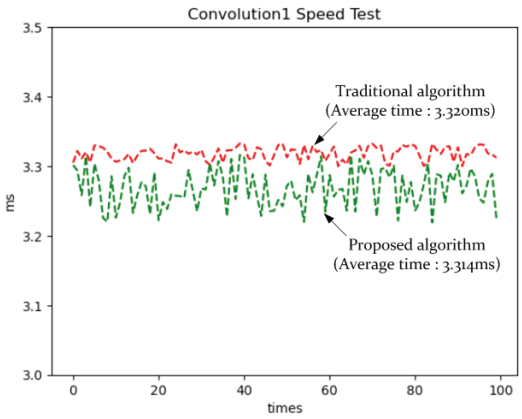

그림 13. 3% 픽셀 삭제를 적용하였을 때 연산
Fig. 13 Computation time when applying 3% pixel disable_1

실험 결과에서 알 수 있듯이, 첫 번째 단계의 콘볼루션에 비해 연산 되는 이미지의 크기가 작으므로, 연산시간의 차이가 첫 번째 단계에 비해 크지 않다는 것을 확인할 수 있다. 첫 번째 단계의 28×28 크기의 이미지가 14×14 크기의 이미지로 줄어들어 연산에서 제외되는 픽셀 수 역시 1/4 로 줄어든다. 이는 콘볼루션에서 제외되는 픽셀의 수가 첫 번째 단계에 비해 상당히 줄어 들었으나, 연산량이 크게 차이 나지 않아 실제 연산을 통해 시간을 측정하였을 경우 연산시간의 감소는 크지 않았다. 표 1은 convolotion1, convolution2에 제안된 알고리즘을 적용하여 연산한 결과를 정리한 평균 연산 시간이다. random pixel disable_1의 경우 3%, 5%, 7%를 적용하였을 때 시간 평균을 보면 기존 알고리즘에 비해 연산시간이 각각 0.2%, 1.4%, 2.7% 줄일수 있었다. 그리고 random pixel disable_2의 경우는 3%, 5%, 7%를 적용하였을 때 시간 평균을 보면 기존 알고리즘에 비해 연산시간이 각각 2.5%, 3.1%, 4.5% 줄일 수 있었다.

표 1. 제안된 연산 평균시간
Table 1. Average computation of proposed Algorithm

	Conv1_ 0%	Conv1_ 3%	Conv1_ 5%	Conv1_ 7%	Conv2_ 0%	Conv2_ 3%	Conv2_ 5%	Conv2_ 7%
1회	3.318	3.303	3.271	3.218	2.977	2.911	2.895	2.846
2회	3.331	3.314	3.289	3.222	2.975	2.899	2.892	2.831
3회	3.307	3.309	3.274	3.235	2.978	2.902	2.866	2.847
4회	3.320	3.322	3.269	3.241	2.973	2.903	2.875	2.845
5회	3.323	3.324	3.267	3.237	2.976	2.902	2.892	2.833
Total	3.320	3.314	3.274	3.231	2.976	2.903	2.884	2.840
%	100%	99.84%	98.62%	97.31%	100%	97.57%	96.92%	95.45%

표 2에 요약된 정확도 결과를 보면, 제안된 알고리즘은 기존 알고리즘 연산(accuracy_0%)에 비해 0.03%, 0.03, 0.02% 정도 정확도가 감소했음을 알 수 있다. 이는 제안된 알고리즘이 실제 연산 정확도에 미치는 영향은 미미한 수준으로, 연산의 결과를 판단하는 것에는 거의 영향이 없음을 알 수 있다. 그러나 추가 실험 결과 13% 이상의 연산에서는 연산 정확도가 5% 이상 급감하여 결과 판단에 영향이 꽤 큰 것으로 나타났다.

표 2. 픽셀 삭제에 따른 정확도 평균
Table 2. Accuracy average according to pixel disable percentages

	Accuracy_0%	Accuracy_3%	Accuracy_5%	Accuracy_7%
1회	95.725	95.655	95.657	95.658
2회	95.678	95.669	95.67	95.662
3회	95.694	95.695	95.696	95.701
4회	95.732	95.698	95.689	95.701
5회	95.711	95.702	95.699	95.703
Total	95.708	95.684	95.682	95.685
%	100%	99.97%	99.97%	99.98%

Ⅴ. 결론

본 논문에서는 임베디드 시스템등과 같은 제한된 연산환경에서 머신러닝 알고리즘의 성능을 개선하기 위해 유전 알고리즘을 적용한 머신러닝 알고리즘을 제안하였고, 성능을 평가하고 검증하기 위해 ARM사의 Cortex Core 기반 임베디드 프로세서 모델인 M4와 딥러닝모델 LeNet5를 적용하여 구현하였다. 그리고 검증을 위해 MNIST 데이터를 학습 데이터로 사용하였으며 손으로 쓴 숫자 세트 100개를 수집하여 테스트 데이터로 사용하였다. 또한 연산 과정 중 가장 많은 부분을 차지하는 콘볼루션 연산에 본 논문에서 제안한 유전 알고리즘을 적용하고, 이 알고리즘이 전체 연산 효율에 미치는 영향을 기존방식과 비교 평가하였다.

사전 테스트에서는 연산시간을 절약하기 위해 콘볼루션 연산량을 3%, 5%, 7%, 13%, 15% 등과 같은 비율로 각각 조정하여 실험하였고, 이것이 연산효율에 어떠한 영향이 있는지를 테스트하였다. 연산량을 줄이는 것은 일종의 노이즈 추가와 비슷한 방법으로 임의의 픽셀값들을 연산에서 제외시켜 연산하는 방식으로

진행하였고, 임의의 픽셀을 선택하는 방법으로는 가우시안 분포를 사용하거나 무작위 선택 방법을 사용하여 테스트하였다. 두 방법 모두 속도에 크게 영향을 주지는 않았지만, 무작위 선택 방법이 더 고르게 분포하여 실험에 사용하였다.

실험 결과, 기존 머신러닝 알고리즘과 비교하여 제안된 알고리즘을 적용한 콘볼루션 연산의 첫 번째 단계에서는 최대 2.7%, 콘볼루션 연산의 두 번째 단계에서는 4.5%의 연산시간을 줄일 수 있었고, 전체 연산시간도 0.42% ~ 3.42% 정도 줄일 수 있음을 확인하였다. 그리고 연산의 정확도는 최대 0.02% ~ 0.03% 정도 감소하는 것으로 나타나 제안된 알고리즘의 적용이 연산시간을 줄이면서도 연산 정확도에는 크게 영향을 미치지 않는다는 것을 입증하였다.

따라서 논문에서 제안한 유전 알고리즘을 적용한 연산량 감소 방법은 임베디드시스템에서 머신러닝을 구현하는 경우 연산량을 줄여 성능 향상과 에너지 효율성 증대에 기여할 것으로 기대되며, 각종 산업 및 연구 분야에서 다양한 응용이 가능할 것으로 전망된다. 그러나 머신러닝은 내부적으로 방대한 수학적 계산을 필요로하는 연산 방법이기 때문에 마이크로프로세서에서 유전 알고리즘을 적용한 머신러닝을 구현하기 위해서는 향후 다양한 연산 환경과 요구사항에 대응할 수 있는 유전 알고리즘의 변형과 최적화 기법이 연구되어야 할 것으로 생각된다.

본 연구는 2024년도 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업 지원을 받아수행되었음. (2023-0-00028)

References

[1] Graziano Crasta, and V. De Cicco, “A chain rule formula in BV and applications to conservation laws”, *HomeSIAM Journal on Mathematical Analysis* 3, 2010, vol 43, pp. 430-456.

[2] John H. Holland, “Genetic Algorithms,” Scientific American, 1992, vol. 267, no. 1, pp. 66-73.

[3] A. M. Saxe, J. L. McClelland, and S. Ganguli. “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks.“ Proceedings of the International Conference on Learning Represenatations 2014, pp. 89-112.

[4] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. “Overfeat: Integrated recognition, localization and detection using convolutional networks.” *International Conference on Learning Representations*, 2014, pp. 78-94.

[5] S. Lawrence, C. Lee Giles, A. C. Tsoi, and A. D. Back, “Face recognition : a convolutional neural network approach,” *IEEE Transactions on Neural Network*s, 1997, vol. 8, no. 1, pp. 98-113.

[6] Y. Bengio and P. Frasconi, “Input-Output HMMs for sequence processing,” IEEE Transactions on Neural Networks, 1996, vol. 7, no. 5, pp. 1231-1249.

[7] S.H. Kim, Y. G. Kim and W.J. Kim, “The Design of Method for Efficient Processing of Small Files in the Distributed System based on Hadoop Framework,” *J. of the Korea Institgrte of Electronic Communication Sciences*, vol. 10. no. 10,2015, pp.1115-1121.

[8] J. Deng, W. Dong, R. Socher, “ImageNet : A Large-scale hierarchical image database,” IEEE, 2009, pp. 245-255.

[9] H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,“ In Proceedings of the 26th Annual International Conference on Machine Learning of the ACM, 2009, pp. 609‐616.

10] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Communications of the ACM, 2017, pp. 84-90.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep residual learning for image recognition,“ IEEE, pp. 770-778, 2015

[12] Y. Bengio, P. Simard, and P. Frasconi., "Learning long-term dependentcies with gradient descent is difficult," IEEE Transactions on Neural Networks, 1994, vol. 5, no. 2, pp. 157 - 166.
[13] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. "Overfeat: Integrated recognition, localization and detection using convolutional networks." *International Conference on Learning Representations*, 2014, pp. 78-94

저자 소개

이소행(So-Haeng Lee)

1991년 광주대학교 전산학과 졸업(공학사)
1994년 조선대학교 대학원 산업공학과 졸업(공학석사)
2023년 순천대학교 대학원 전자공학과 졸업(공학박사)
1995년 ~ 2022년 청암대학교 컴퓨터공학과 교수
2023년 ~ 현재 국립순천대학교 SW중심사업단
※ 관심분야 : 인공지능, IoT, 마이크로프로세서

석경휴(Gyeong-Hyu Seok)

1995년 2월 호남대학교 전자공학과 졸업(공학사)
1997년 8월 조선대학교 대학원 컴퓨터학과 졸업(공학석사)
2005년 2월 조선대학교 대학원 컴퓨터학과 졸업(공학박사)
2004년 3월 ~ 2017년 12월 청암대학교
2018년 1월 ~ 2023년 9월 동강대학교
2023년 10월 ~ 현 국립순천대학교 SW중심대학사업단
한국직업능력개발원 통신분야 평가위원
한국의료정보협회 이사
※ 관심분야 : 데이터통신, 신경망, 전파법, 전파관리, 의료정보 등