엣지 인공지능 프로세서 성능 향상을 위한 학습 데이터셋 구성 방식 비교 분석

이채빈(발표자), 김재성, 이다영, 안성모, 박정민, *이승은
서울과학기술대학교 전자공학과
e-mail:{leechaebin, kimjaeseong, leedayoung, ahnseongmo, parkjoungmin, *seung.lee}@seoultech.ac.kr

Analysis of Training Dataset Construction Methods for Improving the Performance of the Edge AI Processor

Chaebin Lee, Jaeseong Kim, Dayoung Lee, Seongmo An, Joungmin Park, and *Seung Eun Lee
Dept. of Electronic Engineering, Seoul National University of Science and Technology

요 약

본 연구는 *k*-NN 기반 엣지 AI 프로세서의 분류 성능 향상을 위해 Random Sampling, *k*-Means Clustering, Margin Sampling, CNN 기반 특징 추출 등 다양한 학습 데이터셋 구성 방식을 비교하였다. 실험은 MNIST 데이터셋을 활용하였으며 CNN을 이용한 방식이 최고 성능(98.21%)을 보였다. *k*-Means Clustering과 Margin Sampling을 혼합한 방식은 CNN 대비 낮은 연산 복잡도에도 불구하고 비교적 우수한 정확도(83.22%)를 나타냈다.

키워드 : *k*-NN, Edge AI, *k*-Means Clustering, Margin Sampling, CNN, MNIST

I. 서론

k-최근접 이웃(*k*-Nearest Neighbor, *k*-NN) 알고리즘은 단순하고 직관적인 분류 방식이다. 이 알고리즘은 훈련 단계에서 모델 파라미터 학습을 수행하지 않고 데이터를 그대로 저장한 후, 입력 벡터와 저장된 벡터 간의 거리를 계산하여 가장 가까운 *k*개의 이웃에 기반해 분류를 수행한다. 이러한 특성은 연산량이 많은 전통적인 Fully Connected(FC) 계층 기반 분류기와 구별된다. *k*-NN은 모델 파라미터가 없고, 단순한 거리 계산만으로 추론이 가능하므로 메모리 사용량과 연산 복잡도가 낮아, 메모리와 연산 자원이 제한된 임베디드 환경에 특히 적합하다. Intellino는 이러한 *k*-NN 알고리즘의 장점을 하드웨어 수준에서 구현한 경량화된 엣지 AI 프로세서로, 기존의 CNN(Convolution Neural Network) 분류기 구조에서 FC 계층을 대체하거나, 단독으로 동작 인식 등의 분류 문제를 처리할 수 있도록 설계되었다 [1, 2].

하지만 이와 같은 *k*-NN 기반 AI 프로세서의 성능은 단순히 하드웨어 구조나 알고리즘만으로 결정되지 않는다. *k*-NN은 훈련 데이터를 그대로 저장하고 이를 참조해 분류를 수행하기 때문에, 중복되거나 분포가 불균형한 데이터가 포함되면 불필요한 연산이 증가하고 분류 성능이 저하될 수 있다. 특히 입력 벡터의 구성, 데이터의 분포, 거리 계산 방식에 따라 형성되는 분류 경계는 학습 데이터셋 구성 방식에 의해 좌우되며, 이는 전체 정확도와 처리 효율에 직접적인 영향을 미친다.

또한, 엣지 환경에서는 제한된 메모리 자원 내에서 최적의 성능을 달성해야 하므로, 선별 없이 많은 데이터를 저장하는 방식은 한계를 가진다. 따라서 제한된 메모리 내에서 가장 분류 효과가 좋은 학습 데이터를 선별·구성하는 전략이 필요하며, 이는 곧 학습 데이터셋 구성 방식이 AI 프로세서의 전체 성능을 좌우하는 핵심 요소임을 의미한다.

이에 본 연구는 *k*-NN 기반 엣지 AI 프로세서의 성능 향상을 위한 네 가지 학습 데이터셋 구성 방식을 제안하며, 제안하는 방식에 따른 분류 정확도와 연산 자원 소모 간의 상호 관계를 실험적으로 분석하는 것을 목표로 한다.

II. 제안방식

2.1 Random Sampling

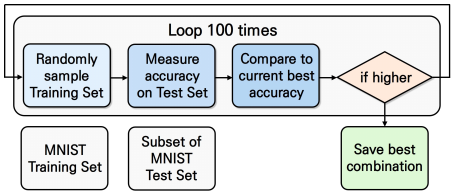

그림 1. Random Sampling 방식 개념도

그림 1은 Random Sampling 방식의 개념도를 나타낸다. 전체 학습 데이터셋에서 무작위로 샘플 데이터셋을 선정한 후, 추론 데이터셋의 일부에 대해 분류 정확도를 산출한다. 이 과정을 여러 번 반복하여 가장 높은 정확도를 기록한 샘플 데이터셋을 선정하여 저장한다. 최종적으로 선정된 샘플 학습 데이터셋으로 추론 데이터셋 전체에 대해 평가하여 최종 정확도를 산출한다.

2.2 *k*-Means Clustering

k-Means Clustering 알고리즘은 그림 2와 같이 비슷한 특성을 가지는 데이터들을 *k*개의 군집으로 묶는 알고리즘이다. 초기에 *k*개의 중심점을 설정하고 중심점으로부터 가까운 데이터들끼리 군집을 형성하여 모든 데이터를 군집에 배정한다. 군집 형성이 끝나면 각 군집 내의 데이터를 기준으로 중심점을 다시 계산하고, 이를 기준으로 모든 점에 대해 다시 군집을 형성한다. 해당 과정을 중심점의 이동이 없을 때까지 반복하여 최종 군집을 완성한다. *k* 값, 즉 군집 수가 많을수록 더 작은 군집에 더 많은 세부 정보가 포함되며, *k* 값이 작을수록 더 큰 군집에 더 적은 세부 정보가 포함된다. 따라서 *k* 값에 따라 이미지가 가지는 대표성과 다양성의 정도가 달라질 수 있다. 이러한 *k*-Means Clustering 알고리즘을 통해 각 군집 별 평균 이미지를 학습하면, 데이터의 분포를 효과적으로 반영하면서도 다양한 특징을 포함한 학습용 데이터셋을 구성할 수 있다.

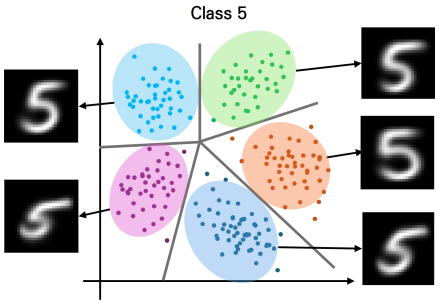

그림 2. *k*-Means Clustering 알고리즘 개념도

2.3 Margin Sampling

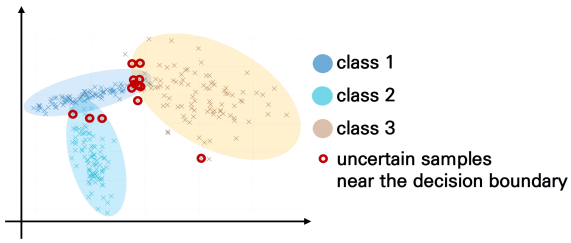

그림 3. Margin Sampling 알고리즘 개념도

Margin Sampling은 예측 결과의 상위 두 클래스 간 확률 차이를 기반으로, 불확실성이 높은 샘플을 선별하는 알고리즘이다. 이는 클래스 내에서 모델이 가장 확신하지 못하는, 즉 결정 경계에 가까운 샘플을 선택함으로써 모델이 취약한 영역에 대해 더 민감하게 학습할 수 있도록 한다. 본 연구에서는 Margin Sampling과 *k*-Means Clustering을 함께 활용하여, 각 클래스별 대표 이미지와 경계 근처의 이미지를 적절히 혼합한 학습 데이터셋을 구성한 후, 이를 기반으로 최종 정확도를 산출한다. 그림 3은 이를 설명하기 위한 도식으로, Margin Sampling에 의해 선택된 경계 부근 샘플들이 시각적으로 강조되어 있다.

2.4 CNN 하이브리드

그림 4는 제안하는 CNN 기반 하이브리드 분류 구조의 전체 흐름을 개념적으로 나타낸 것이다. CNN은 이미지와 같은 격자 형태의 데이터로부터 공간적 패턴과 특징을 추출할 수 있는 딥러닝 모델이다. 모델 내부의 합성곱 계층(convolution layer)과 풀링 계층(pooling layer)을 통해 입력 데이터를 저차원의 의미 있는 특징 벡터로 변환할 수 있다. 본 연구에서는 두 개의 합성곱 계층으로 이루어진 소형 CNN 모델을 이용하여 이미지로부터 특징 벡터를 추출한 후, 클래스 별로 *k*-Means Clustering을 이용하여 대표 벡터들을 선정한다. 이렇게 추출한 대표 벡터들을 기반으로 학습 데이터셋을 구성한 후, 해당 데이터셋을 이용하여 최종 정확도를 산출한다.

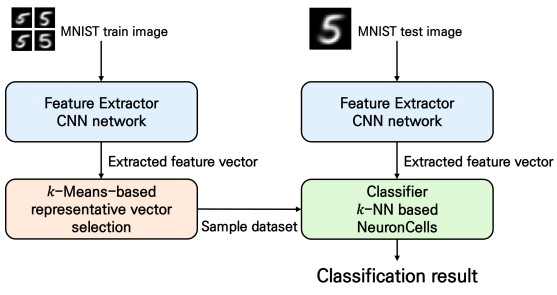

그림 4. CNN 하이브리드 구조 개념도

III. 실험 및 결과

실험은 Intel Core i7-10700 @ 2.90GHz CPU를 사용하는 로컬 PC에서 원격 서버에 접속하여 수행되었으며, 학습은 NVIDIA RTX A4000 GPU에서 PyTorch 2.5.1 (CUDA 12.1) 환경으로 진행되었다. 데이터셋으로는 MNIST를 사용하였다[3]. 하드웨어 설계 시의 제한된 메모리 면적을 고려하여 이미지 한 장당 벡터의 크기는 256으로 총 64장의 이미지를 학습에 사용하였으며, MNIST의 클래스 중 가장 분류가 어려운 '4', '5', '8', '9'에 대해 다른 클래스 보다 이미지를 한 장씩 더 학습에 사용하였다[4]. CNN 기반 모델은 Adam 옵티마이저 (learning rate=0.001), 배치 크기 512, 총 20 epoch으로 학습하였고, 손실 함수는 CrossEntropyLoss를 사용하였다. 학습 완료 후에는 모델의 특징 벡터를 추출하여 클래스별로 벡터를 저장하였다.

표 1은 본 연구에서 사용된 각 방식의 계산 복잡도를 정리한 것이다. 이를 통해 시스템을 구성하는 주요 연산들의 상대적인 복잡도를 파악할 수 있다. 표 2는 네 가지 방식에 따른 분류 정확도를 비교한 결과를 나타낸다. Random Sampling 방식은 67.28%의 정확도를 보였으며, k-Means Clustering을 적용한 경우 74.22%로 향상되었다. k-Means Clustering + Margin Sampling 방식에서는 분류가 어려운 '4', '5', '8', '9' 클래스에서 각각 6장, 나머지 클래스에서는 5장씩 k-Means Clustering 알고리즘으로 추출하고, 모든 클래스에 대해 Margin Sampling을 통해 1장씩 추가하여 학습 데이터셋을 구성하였다. 해당 방식으로 학습한 결과, 정확도는 83.22%로 향상되었으며, 두 방식의 비율을 달리하여 구성한 여러 조합 중 본 조합이 가장 높은 정확도를 달성하였다. 마지막으로, CNN을 활용하여 특징을 추출한 후 k-Means Clustering를 적용한 방식은 98.21%로 가장 높은 분류 정확도를 기록하였다.

표 1. 방식별 연산 복잡도

Algorithm	Computational Complexity
CNN Train	$O(N \cdot E \cdot C_{CNN})$
CNN Inference	$O(N \cdot C_{CNN})$
***k*-Means Clustering**	$O(C \cdot N_C \cdot K \cdot I \cdot D)$
NeuronCells Training	$O(M \cdot D)$
Inference using NeuronCells	$O(N \cdot M \cdot D)$
Margin Sampling	$O(N^2 \cdot D)$
***k*-Means Clustering + Margin Sampling**	$O(N_C \cdot K \cdot I \cdot D) + O(N^2 \cdot D)$

C_{CNN} = FLOPs per forward pass through the CNN
N = Total number of input images
E = Number of training epochs
D = Dimensionality of the feature vector
M = Number of NeuronCells
K = Number of clusters in $k-$Means
I = Number of $k-$Means iterations
C = Number of classes
N_C = Number of samples per class

표 2. 방식별 분류 정확도

Method	Accuracy (%)
Random Sampling	**67.28**
***k*-Means Clustering**	**74.22**
***k*-Means Clustering + Margin Sampling**	**83.22**
CNN + *k*-Means Clustering	**98.21**

IV. 결론 및 분석

본 연구에서는 다양한 샘플링 및 특징 추출 방식이 분류 성능과 연산 자원 소모에 미치는 영향을 정량적으로 비교하였다. 실험 결과, CNN 기반 특징 추출 방식을 포함한 경우가 98.21%의 가장 높은 분류 정확도를 기록하였으나, 전처리 과정에서의 계산 복잡도는 가장 높은 수준을 요구하였다. 이는 고성능을 달성하기 위해 학습 단계에서 상당한 연산 자원이 수반됨을 의미한다. 반면, CNN을 사용하지 않고 k-Means Clustering 또는 Margin Sampling을 활용하여 학습 데이터를 구성한 경우에도 Random Sampling 대비 성능 향상이 관찰되었으며, 특히 k-Means Clustering과 Margin Sampling을 결합한 방식은 CNN 연산 대비 상대적으로 낮은 계산 복잡도 하에서 83.22%의 정확도를 보였다. 이는 고비용 모델 없이도 적절한 데이터 선택 전략을 통해 분류기의 성능을 향상시킬 수 있음을 시사한다. 특히, 제한된 연산 자원 환경에서 CNN을 포함하지 않는 데이터 기반 접근이 실용적인 대안이 될 수 있음을 실험적으로 확인하였다.

참고문헌

[1] Y. H. Yoon, D. H. Hwang, J. H. Yang, and S. E. Lee, "Intellino: Processor for Embedded Artificial Intelligence," Electronics, vol. 9, no. 7, pp. 1169, 2020.

[2] J. Park et al., "Accelerating Strawberry Ripeness Classification Using a Convolution-Based Feature Extractor along with an Edge AI Processor," Electronics, vol. 13, no. 2, pp. 344, 2024.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[4] P. Ceccon Ribeiro et al., "Visual Exploration of an Ensemble of Classifiers," Computers & Graphics, vol. 85, pp. 23-41, 2019.

※ 이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원-대학ICT연구센터(ITRC)의 지원을 받아 수행된 연구임(IITP-2025-RS-2022-00156295)