진화 계산 기법을 활용한 딥러닝 하이퍼파라미터 최적화

이상욱*
목원대학교 게임소프트웨어공학과 교수

Deep Learning Hyperparameter Optimization Using Evolutionary Computation Techniques

Sangwook Lee*
Professor, Department of Game Software Engineering, Mokwon University

요 약 본 딥러닝은 컴퓨터 비전, 자연어 처리, 음성 인식 등 다양한 인공지능 응용 분야에서 뛰어난 성과를 보여주고 있다. 그러나 모델의 성능은 하이퍼파라미터 설정에 크게 좌우되며, 적절하지 않은 하이퍼파라미터는 수렴 속도를 저해하거나 과적합 문제를 유발할 수 있다. 본 논문에서는 하이퍼파라미터 설정을 최적화 하기 위한 방법으로 유전 알고리즘, 입자 군집 최적화, 차분 진화 알고리즘의 세 가지 진화 계산 기법을 제시한다. 세 가지 진화 계산 기법의 성능을 확인하기 위해, SimpleCNN 기반 CIFAR-10 이미지 분류 문제에 사용되는 6개의 하이퍼파라미터를 최적화 하는데 적용하였다. 실험 결과, 동일한 계산량으로 기본적인 그리드 탐색에 비해 진화 계산 기법이 우수한 성능을 보이며, 특히 입자 군집 최적화가 가장 우수한 성능을 보임을 확인하였다. 또한, 안정적인 학습을 위해서는 딥러닝 하이퍼파라미터 중 학습률을 낮추는 것이 효율적이라는 것을 확인할 수 있었다.

주제어 : 딥러닝 하이퍼파라미터, 진화 계산, 유전 알고리즘, 입자 군집 최적화, 차분 진화 알고리즘

Abstract Deep learning has demonstrated outstanding performance in various AI applications, including computer vision, natural language processing, and speech recognition. However, model performance is highly dependent on hyperparameter settings, and inappropriate hyperparameters can slow convergence or lead to overfitting. In this paper, we propose three evolutionary computation techniques—genetic algorithms, particle swarm optimization, and differential evolution algorithms—to optimize hyperparameter settings. To demonstrate the performance of these three evolutionary computation techniques, we apply them to optimize six hyperparameters for the SimpleCNN-based CIFAR-10 image classification problem. Experimental results show that evolutionary computational techniques outperform basic grid search with the same amount of computation, and particle swarm optimization in particular shows the best performance. Additionally, we were able to confirm that lowering the learning rate among deep learning hyperparameters is effective for stable learning.

Key Words : Deep Learning Hyperparameter, Evolutionary Computation, Genetic Algorithms, Particle Swarm Optimization, Differential Evolution

본 논문은 2023년도 목원대학교 연구년 지원에 의하여 연구되었음.
*교신저자 : 이상욱(slee@mokwon.ac.kr)
접수일 2025년 09월 09일 수정일 2025년 09월 30일 심사완료일 2025년 10월 10일

1. 서론

딥러닝(Deep Learning, DL)은 심층 신경망을 활용하여 데이터로부터 표현을 학습하는 기술로, AlexNet 이후 급격히 발전하며 다양한 응용 영역에서 성공을 거두었다. 하지만 딥러닝 모델은 수많은 하이퍼파라미터(hyperparameter)를 가지며, 이들의 조정이 모델 성능에 중대한 영향을 미친다. 이러한 딥러닝 하이퍼파라미터에는 다음과 같은 특성이 존재한다.

- 고차원성 (High Dimensionality): 대규모 모델에서는 수십에서 수백 개의 하이퍼파라미터가 존재할 수 있음
- 비선형성 (Nonlinearity): 하이퍼파라미터와 검증 성능 간 관계는 복잡하고 다중 모드(multi-modal) 구조를 가짐
- 고비용성 (Expensive Evaluation): CPU 및 GPU 병렬 처리를 활용해도, 하나의 조합을 평가하기 위해 수 시간에서 수 일이 소요될 수 있음
- 동적 효과 (Dynamic Behavior): 학습 초기와 후반에 최적의 학습률이 달라지는 등 시간 종속적 특성이 존재함

주요 하이퍼파라미터로는 학습률(learning rate), 배치 크기(batch size), 가중치 초기화(weight initialization), 드롭아웃 비율(dropout rate) 등이 있으며, 이러한 하이퍼파라미터는 학습의 안정성과 정확도를 좌우하는 핵심 요인이라 할 수 있다. 따라서 하이퍼파라미터 최적화(Hyperparameter Optimization, HPO)는 딥러닝 연구와 실무 모두에서 중요한 과제가 되었다.

하이퍼파라미터 최적화를 위한 전통적인 탐색 기법으로는 그리드 탐색(Grid Search, GS)과 무작위 탐색(Random Search, RS)이 있다[1]. 그리드 탐색은 모든 하이퍼파라미터 조합을 격자 형태로 탐색하는 방법이다. 단순하지만 계산 비용이 지수적으로 증가한다. 무작위 탐색은 하이퍼파라미터 조합을 무작위로 탐색하는 방법이다. 이러한 무작위 탐색은 그리드 탐색보다 효율적이라는 사실이 확인되었으며, 특히 일부 하이퍼파라미터가 성능에 민감하게 작용하는 경우 더 좋은 성능을 보임이 확인되었다.

하이퍼파라미터 최적화를 위해 베이지안 최적화(Bayesian Optimization, BO) 기법도 활용되었다[2]. 베이지안 최적화는 성능 함수의 사후 분포를 모델링하여 탐색과 활용 간 균형을 맞춰 준다. 이때, 가우시안 프로세스(Gaussian Process, GP) 또는 Tree-structured Parzen Estimators (TPE) 기반의 방법이 많이 사용된다. 베이지안 최적화는 고비용 함수 평가를 줄이는 데 효과적이나, 고차원 공간에서는 성능이 저하되는 한계가 있음이 확인되었다. 근래에는 강화학습 기반의 신경망 구조 탐색 방법이 제안되었고[3] 현재 신경망 구조 최적화를 위한 주된 방법으로 사용되고 있다.

그러나, 기존에 제안된 방법들은 지역 최적해를 벗어나기 어렵다는 단점이 있다. 본 논문에서는 전역 최적해를 찾는데 특화된 메타 휴리스틱 기법인 유전 알고리즘(Genetic Algorithms, GA), 입자 군집 최적화(Particle Swarm Optimization, PSO) 및 차분 진화 알고리즘 (Differential Evolution, DE)의 세 가지 진화 계산 (Evolutionary Computation, EC) 기법을 제안한다. 그리고, 성능 비교를 통해 딥러닝 하이퍼파라미터 최적화에서 우수한 성능을 보여줄 가능성이 높은 진화 계산 방법을 찾고자 한다.

2. 딥러닝 하이퍼파라미터

딥러닝은 컴퓨터 비전, 자연어 처리, 음성 인식 등 다양한 분야에서 탁월한 성능을 보여주고 있다. 그러나 모델의 성능은 하이퍼파라미터 선택에 크게 좌우된다.

하이퍼파라미터는 학습률, 네트워크 깊이, 뉴런 수, 드롭아웃 비율, 배치 크기 등 모델 학습 과정에 직접적으로 영향을 미치는 변수이다. 이들의 최적 조합을 탐색하는 것은 고성능 딥러닝 모델 개발의 핵심이다.

이번 장에서는 일반적으로 정의되는 딥러닝 하이퍼파라미터의 유형에 대해 살펴본 후, 하이퍼파라미터 최적화를 위해 본 연구에서 활용할 데이터 셋 (CIFAR-10), 해당 데이터 셋에 적용할 인공지능 모델(SimpleCNN)을 설명한다. 마지막으로, SimpleCNN 기반 CIFAR-10 분류 문제 해결에 사용되는 하이퍼파라미터에 대해 설명한다.

2.1 딥러닝 하이퍼파라미터 유형

딥러닝 하이퍼파라미터는 모델 구조, 학습 과정, 데이터/훈련 전략 등과 관련하여 다음과 같이 다양한 유형으로 정의되어 있다.

2.1.1 모델 구조 관련

- 네트워크 깊이 (Depth): 레이어 수 (예: ResNet-50 vs ResNet-101)
- 폭 (Width): 레이어 내 채널 수 또는 뉴런 수
- 활성화 함수 (Activation Function): ReLU, GELU, Swish 등
- 정규화 기법 (Normalization): BatchNorm, LayerNorm, GroupNorm

2.1.2 학습 과정 관련

- 학습률 (Learning Rate): 최적화에서 가장 중요한 하이퍼파라미터
- Optimizer: SGD, Adam, AdamW, RMSProp 등
- 배치 크기 (Batch Size): 일반화 성능과 학습 안정성에 영향
- 정규화 기법 (Regularization): Dropout 비율, Weight Decay 등

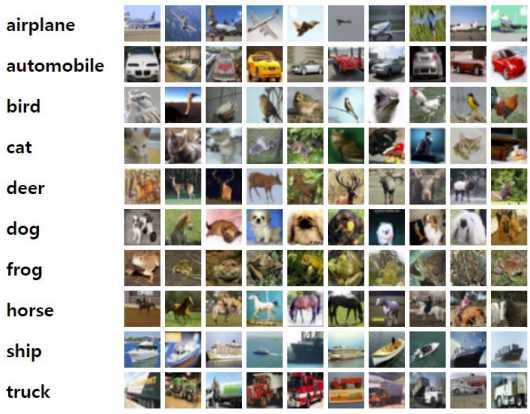
2.1.3 데이터/훈련 전략 관련

- 데이터 증강 (Data Augmentation): RandomCrop, CutMix, MixUp 등
- Early Stopping/Epochs: 과적합 방지를 위한 조기 종료
- Scheduler: Cosine Annealing, Step Decay, Warmup 등

딥러닝 하이퍼파라미터는 해결하고자 하는 문제, 적용한 딥러닝 모델에 따라 달라진다. 본 연구에서는 딥러닝 및 머신러닝 분야에서 가장 널리 사용되는 벤치마크 중에 하나인 CIFAR-10 데이터 셋을 활용하였다.

2.2 데이터 셋 (CIFAR-10)

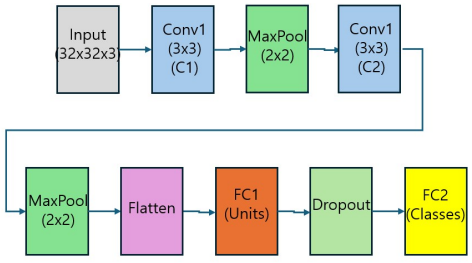
CIFAR-10 데이터 셋은 10개의 클래스(비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭)를 포함한 60,000장의 32×32 RGB 이미지로 구성되어 있다. 이 중 50,000개는 학습용, 10,000개는 테스트용으로 구성되어 있으며, 10개의 클래스별로 균등한 개수로 구성되어 있다. 작은 크기, 균형 잡힌 클래스 분포, 높은 난이도라는 특성 덕분에, 학계 및 산업계에서 새로운 모델의 성능을 평가하는데 표준적으로 널리 사용되고 있다. 그림 1은 CIFAR-10 데이터 셋의 10개 클래스별 이미지 샘플을 보여주고 있다[4].



[Fig. 1] CIFAR-10 Data Set

2.3 딥러닝 모델

하이퍼파라미터 최적화 연구를 위한 CIFAR-10 데이터 셋 이미지 분류 문제에 합성곱 신경망(Convolutional Neural Network, CNN) 아키텍처를 적용하였다[5],[6]. 본 모델은 합성곱 신경망(CNN)의 기본 구조를 단순화한 형태로 학습 효율성과 구조적 단순성을 동시에 고려하여 경량 설계되었으며, '단순 합성곱 신경망(SimpleCNN)'이라 명칭하였다. 본 연구에서 사용한 SimpleCNN의 아키텍처 구조를 블록 구조 네트워크 다이어그램으로 시각화 하면 그림 2와 같다.



[Fig. 2] SimpleCNN Architecture Block Structure Network Diagram

SimpleCNN은 2개의 합성곱 층(Conv Layer), 1개의 완전 연결 층 (Fully Connected Layer) 및 1개의 출력 층으로 구성되어 있으며, 세부적인 내용은 다음과 같다.

1) Conv1 Layer

- 입력: 3채널 RGB 이미지 (32×32×3)
- Conv2D: 필터 수 = conv1_channels
- 커널 크기: 3×3, padding=1
- 활성화 함수: ReLU

- Pooling: 2×2 MaxPooling
- 출력 크기 : (conv1_channels × 16 × 16)

2) Conv2 Layer
- 입력: Conv1의 출력
- Conv2D: 필터 수 = conv2_channels
- 커널 크기: 3×3, padding=1
- 활성화 함수: ReLU
- Pooling: 2×2 MaxPooling
- 출력 크기: (conv2_channels × 8 × 8)

3) Flatten Layer
- Conv2 출력을 벡터로 변환
- 크기: conv2_channels × 8 × 8

4) Fully Connected Layer (FC1)
- 입력 차원: conv2_channels × 8 × 8
- 출력 차원: fc1_units
- 활성화 함수: ReLU
- Dropout: p = dropout

5) Output Layer (FC2)
- 입력: fc1_units
- 출력: 10 (CIFAR-10 클래스 수)
- 활성화 없음 (CrossEntropyLoss에서 softmax 처리)

SimpleCNN은 CIFAR-10 데이터 셋을 대상으로 한 기본적인 CNN 아키텍처로, 복잡한 네트워크 대비 가볍고 직관적이며, 하이퍼파라미터 최적화를 통해 성능 개선의 여지가 크도록 설계되었다.

2.4 하이퍼파라미터

본 연구에서는 CIFAR-10 분류를 위한 SimpleCNN의 주요 하이퍼파라미터로 합성곱 채널 수 2가지(conv1, conv2), 완전연결 은닉 유닛 수(fc1), 학습률(learning rate), 드롭아웃(dropout), 배치 크기(batch size)의 6개로 설정하였다. 각 하이퍼파라미터에 대한 세부적인 내용 및 특징은 다음과 같다[7],[8].

2.4.1 Conv1 채널 수 (C_1 = conv1_channels)
- 첫 번째 합성곱 계층의 출력 채널 수를 의미함
- 에지·색상·텍스처 등의 저 수준의 특징 다양성을 결정함
- 파라미터 및 연산량에 있어서 선형 또는 준 선형적으로 영향을 미침
- 증가는 Conv2(입력 채널)와 FC1(간접적으로 경유)의 매개변수도 증대시켜 연쇄적 비용 증가를 유발
- CIFAR-10에서는 $C_1 \in [16, 64]$ 범위의 값으로 설정하는 것이 적절하다고 알려져 있음. 너무 작은 값은 표현력이 부족하며, 너무 큰 값은 과적합 및 연산 비용 증가의 위험이 있음

2.4.2 Conv2 채널 수 (C_2 = conv2_channels)
- 두 번째 합성곱 계층의 출력 채널 수를 의미함
- 물체 구성요소/패턴 표현력에 직접적인 영향을 미침
- 연산량에 있어서 선형 또는 준 선형적으로 영향을 미침
- FC1 입력 차원 $(8 \cdot 8 \cdot C_2)$ 크기를 결정하므로, 파라미터 및 메모리 폭증의 주원인이 됨
- C_2 증가는 FC1 파라미터를 $O(C_2 \cdot H)$로 확대하므로, 과적합 및 학습 불안정성 위험이 있음
- CIFAR-10에서는 $C_2 \in [64, 128]$ 범위의 값으로 설정하는 것이 적절하다고 알려져 있으며, FC1과 병행하여 조절할 필요가 있음

2.4.3 FC1 은닉 유닛 수 (H= fc1_units)
- 최종 선형 분류기에 앞서 특징을 압축/변환하는 완전 연결 계층의 유닛 수를 의미함
- H 증가는 결정경계 복잡도를 높이며, CIFAR-10에서 매우 큰 H는 불필요한 메모리 용량과 과적합의 위험이 있음
- 파라미터 규모는 $(8 \cdot 8 \cdot C_2) \cdot H$로 선형 증가하며, 학습 및 메모리 병목 현상을 유발할 수 있음
- CIFAR-10에서 $H \in [128, 512]$ 범위의 값으로 설정하는 것이 적절하다고 알려져 있으며, C_2가 클수록 H를 낮추는 것을 권장하고 있음

2.4.4 학습률 (η = lr)
- 매 스텝마다 파라미터를 업데이트하는 스케일(Adam 사용)을 의미함
- 학습률이 크면 빠른 수렴이 가능하나 발산·진동의 위험이 있으며, 작으면 안정적이나 학습이 정체될 가능성이 있음
- Adam은 1차 모멘트(평균 기울기)와 2차 모멘트(제곱 기울기의 이동 평균)을 사용해 학습률을 조정하

여 파라미터마다 상대적인 학습률 조정을 적절히 수행하지만, 전역 스케일 기준점을 어디에 두느냐는 여전히 초기 학습률이 좌우하기에 초기 학습률은 성능 및 학습 속도에 매우 민감함

- CIFAR-10에서 Adam과 함께 사용할 시에서는 학습률을 $[10^{-4}, 10^{-2}]$의 로그 스케일 샘플링 범위의 값으로 설정하는 것이 적절하며, 초기 warmup 또는 cosine decay 등 스케줄러와 병행하면 견고성 향상을 기대할 수 있음

2.4.5 드롭 아웃 (δ= dropout)

- FC1의 뉴런을 학습 시 확률 δ(드롭 아웃)로 비활성화하는 정규화 기법(테스트 시 스케일 보정)에 활용됨
- co-adaptation을 억제하고 과적합을 완화하는 역할 담당함
- 과도한 드롭아웃은 신호 약화로 수렴 지연 및 성능 저하를 유발할 수 있음
- 소형 CNN+FC에서는 $\delta \in [0.3, 0.5]$ 범위의 값으로 설정하는 것이 안정적이라고 알려져 있으며, Conv에 드롭아웃을 추가하는 경우 비율을 더 낮게 설정하는 것을 권장하고 있음

2.4.6 배치 크기 (B= batch_size)

- 단일 업데이트에 사용되는 샘플 수를 의미함
- 작은 B는 노이즈를 증가시키고, 이는 암시적 정규화 효과와 일반화 향상을 가능하게 함. 큰 B는 업데이트 안정 및 가속화를 기대할 수 있으나, 일반화 악화를 유발할 수 있음
- 큰 B는 GPU 활용도 및 스루풋을 향상 시킬 수 있으나, 메모리 사용량은 B에 비례하여 증가함
- $B \in [32, 128]$ 범위의 값으로 설정하여 하드웨어 및 학습률과 같이 튜닝하는 것이 일반적임. 배치 크기를 늘릴 때 학습률도 같이 늘리면 훈련 안정성 및 수렴성이 증대됨

딥러닝에서 하이퍼파라미터는 모델 구조와 학습 과정을 규정하는 핵심 요소로, 최종 성능과 안정성에 큰 영향을 준다. CIFAR-10(32×32, 3채널)과 같은 소형 이미지 데이터 셋은 문제해결을 위해 용량(capacity)-정규화(regularization)-최적화(optimization)-자원(resource) 간의 균형이 필요하다. 본 연구에서 사용한 SimpleCNN은 (Conv-ReLU-Pool)×2 + FC + Dropout + Linear로 구성되어 있다. 여기서, 파라미터 수와 연산량은 주로 채널 수(conv1/conv2) 및 FC 유닛 수(fc1)에 의해 결정되며, 학습 안정성은 학습률과 배치 크기에 의해, 일반화는 드롭아웃과 배치 크기(간접적 정규화)에 의해 결정된다.

지금까지 딥러닝의 성능 향상을 위해 최적화 해야할 하이퍼파라미터에 대해 살펴보았다. 다음 장에서는 하이퍼파라미터 최적화를 위해 적용할 진화 계산 기법들에 대해 살펴본다.

3. 진화 계산

진화 계산이란 자연의 진화 현상에서 영감을 얻어 고안된 계산 기법으로 유전 알고리즘, 유전 프로그래밍 (Genetic Programming, GP), 진화 전략 (Evolutionary Strategy, ES) 등을 통칭하여 부른다[9]. 일반적으로 수학적 방법으로는 최적해를 구하기 어려워, 해집합의 모든 경우의 수를 조사해 봐야 하는 조합 최적화 문제(Combinatorial Optimization Problem)를 해결하기 위해 활용된다.

해결하고자 하는 문제를 진화 계산에 적용하기 위해서는 먼저 해 표현 방법을 설계해야 한다. 본 연구에서 하이퍼파라미터 최적화 문제를 진화 계산 기법에 적용하기 위해, 그림 3과 같이 6개의 하이퍼파라미터를 6개의 실수 변수에 대응시키도록 해 표현 방법을 설계하였다.

C_1	C_2	H	η	δ	B
18.6	55.9	123.3	0.002	0.035	1.52
[0]	[1]	[2]	[3]	[4]	[5]

[Fig. 3] Solution representation

해 표현에 포함된 6개 실수 변수의 범위와 해당 실수 변수 값을 6개의 하이퍼파라미터의 값으로 대응하는 것은 다음과 같은 로직으로 설계하였다.

1) C_1 (Conv1 채널 수)

- [16, 64] 범위의 정수
- 해 표현 0 인덱스 값을 [16, 64] 범위의 실수로 설정
- 해 표현 0 인덱스의 값을 소수점 첫째 자리에서 반올림하여 정수로 변환하여 대응

2) C_2 (Conv2 채널 수)

- [32, 128] 범위의 정수

- 해 표현 1 인덱스 값을 [32, 128] 범위의 실수로 설정
- 해 표현 1 인덱스의 값을 소수점 첫째 자리에서 반올림하여 정수로 변환하여 대응함

3) H (FC1 은닉 유닛 수)
- [64, 512] 범위의 정수
- 해 표현 2 인덱스 값을 [64, 512] 범위의 실수로 설정
- 해 표현 2 인덱스의 값을 소수점 첫째 자리에서 반올림하여 정수로 변환하여 대응함

4) η (학습률)
- $[10^{-4}, 10^{-2}]$ 범위의 실수
- 해 표현 3 인덱스 값을 $[10^{-4}, 10^{-2}]$ 범위의 실수로 설정
- 해 표현 3 인덱스 값을 그대로 대응

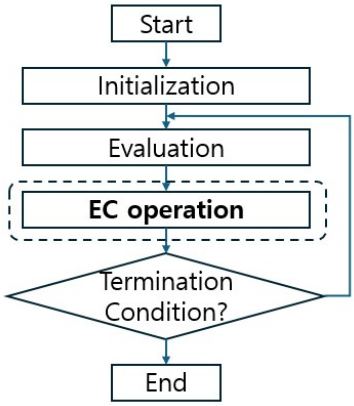
5) δ (드롭 아웃)
- [0.0, 0.6] 범위의 실수
- 해 표현 4 인덱스 값을 [0.0, 0.6] 범위의 실수로 설정
- 해 표현 4 인덱스 값을 그대로 대응

6) B (배치 크기)
- 32, 64, 128 의 3개 값 중 하나의 정수
- 해 표현 5 인덱스 값을 [0, 2] 범위의 실수로 설정
- 해 표현 5 인덱스의 값을 소수점 첫째 자리에서 반올림하여 정수로 변환하고, 0이면 32, 1이면 64, 2이면 128의 값에 대응함

CIFAR-10 분류 문제 해결을 위한 SimpleCNN 하이퍼파라미터 최적화를 위한 진화 계산 기법은 그림 4와 같이 로직을 구성하였다[10]. 먼저, 앞서 설계한 해 표현 방법을 활용하여 초기화(Initialization) 과정을 통해 임의로 해 집단을 생성한다. 다음 단계에서는 생성한 해 집단의 각 해들을 평가(Evaluation) 과정을 통해 적합도를 계산하고, 각 진화 연산 기법의 특성에 부합하는 진화 연산(EC operation)을 수행하며, 이러한 평가와 연산은 종료 조건을 만족할 때까지 반복된다.

여기서, 평가의 과정은 해의 인덱스 값들을 설계한 로직에 따라 6개의 하이퍼파라미터 값으로 변환(대응)하고 SimpleCNN에 적용한 후, CIFAR-10 데이터셋을 활용하여 학습 및 검증 과정을 거쳐 도출된 정확도(accuracy) 값을 해당 해의 적합도로 사용하였다.

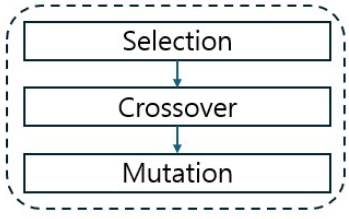
본 연구에서는 하이퍼파라미터 최적화를 위한 진화 계산 방법으로 유전 알고리즘, 입자 군집 최적화, 차분 진화 알고리즘의 3가지 방법을 사용하였다. 아래에는 3가지 진화 계산 기법들에 대한 간단히 소개와 적용된 진화 연산에 대해 설명한다.



[Fig. 4] Evolutionary Computation flow chart

3.1 유전 알고리즘

유전 알고리즘은 생물의 진화 과정을 모방한 최적화 기법으로 1975년 존 홀랜드(John Holland)에 의해 처음 소개되었다[11]. 후보 해(solution)를 개체(individual)로 정의하고 이를 집단(population) 단위로 관리하며, 알고리즘은 반복(iteration)마다 그림 5와 같은 선택(selection), 교차(crossover), 돌연변이(mutation) 연산의 진화 연산을 통해 자식을 생성하여 점진적으로 더 우수한 해로 진화시킨다. 본 연구에서는 다음과 같은 진화 연산이 사용되었다[12].



[Fig. 5] EC operation of Genetic Algorithms

1) 선택
- 토너먼트 선택(Tournament selection) 사용
- 무작위로 2개의 개체를 선택하고, 더 우수한 적합도를 가진 개체를 부모로 선택

2) 교차
- 균일 교차 (Uniform crossover) 적용
- 두 부모 개체의 각 유전자의 값을 일정 확률로 교환하여 새로운 자손 생성

3) 돌연변이
- 각 개체는 일정 확률로 돌연변이 수행
- 돌연변이가 발생하면 무작위로 선택된 하나의 유전자(하이퍼파라미터)를 범위 내의 새로운 값으로 변경함

3.2 입자 군집 최적화

입자 군집 최적화는 자연에서의 새 떼의 무리 이동(Swarm intelligence)에서 영감을 받은 확률적 최적화 기법으로 1995년 제임스 케네디(James Kennedy)와 러셀 에버하트(Russell Eberhart)에 의해 처음 소개되었다[13]. 유전 알고리즘의 초기화처럼 무작위로 후보해 집단을 생성하며 후보해를 입자(particle)라고 부른다. 각 입자는 위치(position, 본 연구에서는 하이퍼파라미터 벡터)와 속도(velocity)를 가지며, 매 반복(iteration)마다 그림 6과 같은 진화 연산을 통해 갱신된다. 본 연구에서는 입자의 위치 및 속도 갱신을 위해 다음과 같은 수식이 사용되었다[14].

1) 지역 및 전역 최적해 갱신
- i 번째 입자의 개인 최적해 (p_i^{best}) 갱신
- 전체 입자 집단의 전역 최적해 (g^{best}) 갱신

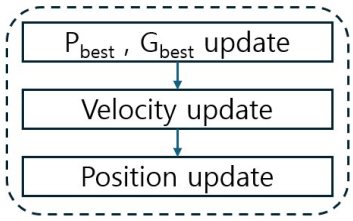
2) 속도 갱신
- i 번째 입자의 속도($v_i^{(t)}$)를 다음 수식에 따라 갱신

$$v_i^{(t+1)} = w \cdot v_i^{(t)} + c_1 \cdot r_1 \cdot \left(p_i^{best} - x_i^{(t)}\right) + c_2 \cdot r_2 \cdot \left(g^{best} - x_i^{(t)}\right)$$

3) 위치 갱신
- i 번째 입자의 위치($x_i^{(t)}$)를 다음 수식에 따라 갱신

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$$

여기서, w는 관성 계수, c_1 및 c_2는 학습 계수, r_1 및 r_2는 [0, 1] 범위의 난수이다.



[Fig. 6] EC operation of Particle Swarm Optimization

3.3 차분 진화 알고리즘

효과적으로 다룰 수 있는 진화 계산 알고리즘으로, 1997년 Storn과 Price에 의해 처음 제안되었다[15]. 이 알고리즘은 단순한 구조에도 불구하고 전역 탐색 성능이 우수하여 다양한 최적화 문제에 적용되고 있다. 앞에서 설명한 2가지 기법(유전 알고리즘, 입자 군집 최적화)과 같이 무작위로 해 집단을 생성하는 초기화 과정과 생성된 후보해를 평가하는 과정은 동일하게 수행된다. 그 다음은 반복(iteration)마다 그림 7과 같은 선택(selection), 돌연변이(mutation), 교차(crossover), 연산의 진화 연산을 통해 시험 벡터(Trial vector)를 생성하여 점진적으로 더 우수한 해로 진화시킨다. 본 연구에서는 다음과 같은 진화 연산이 사용되었다.

1) 선택
- 다음 수식에 의해 시험 벡터 u_i와 기존 개체 x_i 중 적합도가 더 우수한 해가 다음 세대에 포함됨

$$x_i^{(t+1)} = \begin{cases} u_i \text{ , if } f(u_i) \le f(x_i) \\ x_i \text{ , otherwise} \end{cases}$$

2) 돌연변이
- 다음 수식에 의해 돌연변이 수행하여 탐색 공간 확장

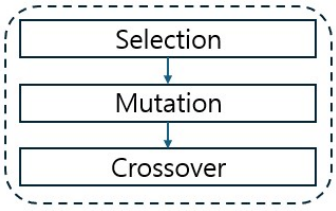
$$v_i = a + F \cdot (b - c)$$

여기서, a, b, c는 서로 다른 세 개체이며, $F \in (0, 2)$는 스케일링 팩터임

3) 교차
- 다음 수식을 활용하여 균일 교차 (Uniform crossover) 적용

$$u_{i,j} = \begin{cases} v_{i,j} \text{ , if } rand_j \le CR \text{ or } j = j_{rand} \\ x_{i,j} \text{ , otherwise} \end{cases}$$

• 즉, 돌연변이 된 벡터의 각 성분이 확률 CR(Crossover Rate)로 시험 벡터에 반영됨



[Fig. 7] EC operation of Differential Evolution

4. 실험

본 실험에서는 지금까지 살펴본 유전 알고리즘, 입자 군집 최적화 및 차분 진화 알고리즘의 3가지 진화 계산 기법을 SimpleCNN 기반 CIFAR-10 이미지 분류 문제의 딥러닝 하이퍼파라미터 최적화에 적용하여 정확도 향상 성능을 확인해 본다. 그리고, 딥러닝 하이퍼파라미터 최적화의 기본적인 알고리즘인 그리드 탐색 기법[11]과 성능을 비교 확인해 본다. 마지막으로는 SimpleCNN 기반 CIFAR-10 이미지 분류 문제의 딥러닝 하이퍼파라미터가 정확도 향상에 미치는 영향에 대해 분석해 본다.

4.1 실험 환경 및 알고리즘 변수 설정

1) 실험 환경

본 실험에 사용된 컴퓨팅 환경은 AMD Ryzen 9 7900 12-Core Processor와 RTX 4060 Ti GPU를 사용하여 CUDA 및 CPU 병렬처리를 하였고, 64GB Memory, window 10 64bit 운영체제이며, 개발 환경은 Anaconda3 2025.06-0(Python 3.11.13)의 jupyter notebook을 사용하였으며, 딥러닝 프레임워크는 PyTorch 2.5.1을 사용하였다.

2) 알고리즘 변수 설정

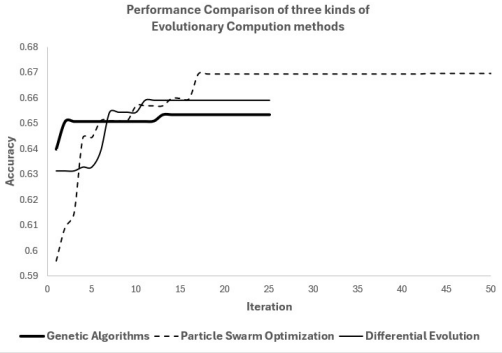
3가지의 진화 계산 기법에 대해 다음과 같이 알고리즘 변수를 설정하였다.

- 유전 알고리즘: 해집단 수 = 20, 반복 수 = 25, 교차 확률 = 0.5, 돌연변이 확률 = 0.2
- 입자 군집 최적화: 입자 수 = 10, 반복 수 = 50, w = 0.7, c1 = 1.5, c2 = 1.5
- 차분 진화 알고리즘: 해집단 수 = 20, 반복 수 = 25, F = 0.8, CR = 0.7

공평한 비교를 위해 적합도 계산 총 횟수는 [해집단 수(입자 수) × 반복 수]로 3가지 기법 모두 500회 계산하도록 설정하였다.

4.2 진화 계산 기법 성능 비교 분석

CIFAR-10 분류 데이터 셋 문제 해결을 위한 SimpleCNN의 하이퍼파라미터 최적화를 위해 3가지의 진화 계산 알고리즘(유전 알고리즘, 입자 군집 최적화 및 차분 진화 알고리즘)을 적용하였다. 그림 8은 3가지 알고리즘의 성능을 비교한 결과를 보여주고 있다. 차분 진화 알고리즘은 입자 수 10개로 다른 2개의 알고리즘에 비해 시작 지점의 정확도가 낮지만, 반복 수 5부터 다른 알고리즘의 성능에 근접하여 반복 수 10 이후부터는 추월하는 것을 확인할 수 있다. 본 실험을 통해 하이퍼파라미터 최적화에 있어서, 입자 군집 최적화, 차분 진화 알고리즘, 유전 알고리즘 순으로 성능이 우수함을 확인할 수 있었다.



[Fig. 8] Performance comparison of three ECs

4.3 그리드 탐색과의 비교 분석

딥러닝 하이퍼파라미터 최적화를 위한 진화 계산 기법을 기본적인 알고리즘과 성능을 비교하기 위해, 그리드 탐색 기법을 다음과 같이 적용해 보았다. 공평한 비교를 위해 각 하이퍼파라미터 경우의 수를 다음과 같이 [C_1 5개, C_2 5개, H 5개, η 2개, δ 2개, B 1개]로 설정하여 적합도 계산 총 횟수를 500(5×5×5×2×2×1)회로 맞추었다.

- C_1: [16, 32, 48, 64, 80]
- C_2: [64, 96, 128, 160, 192]
- H : [128, 192, 256, 320, 384]

- η : $[10^{-3},\ 5 \cdot 10^{-3}]$
- δ : [0.3, 0.5]
- B : [64]

그리고, 실험을 10번 반복하여 측정한 accuracy를 평균한 값을 사용하였으며, 매 실험마다 동일한 결과를 보여주도록 seed를 고정하여 실험하였다.

표 1은 3가지의 진화 계산 기법과 그리드 탐색을 통해 찾은 최적의 정확도(accuracy)에 대한 하이퍼파라미터 및 계산 시간을 나타내고 있다. 3가지 진화 계산 기법이 모두 그리드 탐색 기법에 비해 높은 정확도를 찾는데 좋은 성능을 보여줌을 확인하였으며, 그 중 입자 군집 최적화가 가장 좋은 성능을 보여주었다. 계산 시간 측면에서는 진화 연산에 대해 오버헤드가 없는 그리드 탐색이 가장 적은 시간을 소모하였으며, 입자 군집 최적화가 가장 많은 시간을 소모하였다. 이는 유전 알고리즘 및 차분 진화 알고리즘이 해집단 수를 20으로 설정한 반면에 입자 군집 최적화의 경우 입자 수를 10으로 설정함에 따라 평가(Evaluation) 단계의 CPU 병렬 처리에 있어서 모든 프로세스를 활용하지 못하여 발생한 결과로 보인다.

〈Table 1〉 Evolutionary Computation vs. Grid Search

		GA	PSO	DE	GS
hyper parameters	C_1	56	30	28	48
	C_2	97	101	128	192
	H	404	512	512	384
	η	0.00068	0.00104	0.00084	0.0010
	δ	0.03	0.05	0.05	0.3000
	B	32	32	32	64
Best Acc		0.6535	**0.6697**	0.6590	0.6389
time(sec)		5,559	6,239	6,145	**5,177**

4.4 하이퍼파라미터 민감도 분석

딥러닝 하이퍼파라미터가 성능에 어떤 영향을 미치는지 확인하기 위하여 정확도에 대한 하이퍼파라미터의 민감도를 분석하였다. 표 2.는 그 결과를 보여주고 있으며, 값이 1에 가까우면 해당 하이퍼파라미터는 정확도 성능과 비례 관계에 있고, -1에 가까우면 반비례 관계에 있음을 나타낸다.

4가지 알고리즘에 대해 공통적인 경향을 보이는 것은 다음과 같다. 학습률(η)은 정확도와 반비례 관계의 민감도가 높으므로, 안정적인 학습을 위해서는 낮은 학습률이 필요하다는 것을 확인할 수 있다. 드롭아웃(δ) 또한, 3가지 진화 계산에서는 정확도와 반비례 관계의 민감도가 일정 수준 이상으로 보이고 있으나, GS에서는 거의 관련성이 없는데, 이는 탐색 알고리즘의 탐색 경로 차이로 인해 발생하는 것으로 해석할 수 있다. 종합적으로 해석하면, 학습률 최적화가 가장 우선시되어야 하며, 그 다음은 FC1 은닉 유닛 수(H), Conv2 채널 수(C_2)의 조합이 성능 개선의 핵심 요소로 보임을 확인할 수 있다.

〈Table 2〉 Hyperparameter sensitivity analysis results for accuracy

		GA	PSO	DE	GS
hyper parameters	C_1	0.088405	-0.156010	-0.203875	-0.263853
	C_2	0.119462	0.375058	0.232001	0.095533
	H	-0.004934	0.581288	0.350405	0.018502
	η	-0.854365	-0.696153	-0.729963	-0.736188
	δ	-0.321188	-0.618395	-0.398496	-0.064839
	B	-0.102528	-0.204644	-0.058218	NaN

5. 결론

본 연구에서는 딥러닝 하이퍼파라미터 최적화를 위한 진화 계산 기법을 제시하고 실험을 통해 성능을 확인하였다. 이를 위해, 유전 알고리즘, 입자 군집 최적화 및 차분 진화 알고리즘의 3가지 진화 계산 기법을 SimpleCNN 기반 CIFAR-10 이미지 분류 문제의 딥러닝 하이퍼파라미터 최적화에 적용하여 정확도 향상에 기여하는 성능을 확인해 보았다. 실험 결과, 기본적인 그리드 탐색에 비해 진화 계산 기법이 하이퍼파라미터 최적화에 보다 좋은 성능을 보였으며, 특히, 입자 군집 최적화가 가장 좋은 성능을 보여줌을 확인하였다. 또한, SimpleCNN 기반 CIFAR-10 이미지 분류 문제의 딥러닝 하이퍼파라미터가 정확도에 미치는 영향을 확인해 보았으며, 안정적인 학습을 위해서는 낮은 학습률이 필요하다는 것을 확인할 수 있었다.

향후에는 CIFAR-10 외의 다른 데이터셋에 적용하는 것과, SimpleCNN 외에 보다 복잡한 네트워크에 적용하는 실험을 수행하여 딥러닝 하이퍼파라미터 최적화를 위한 진화계산 기법의 성능을 추가로 확인할 예정이다.

REFERENCES

[1] J.Bergstra and Y.Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, Vol.13,No. 1, pp.281-305, 2012.

[2] J.Snoek, H.Larochelle, and R.P.Adams, *Practical Bayesian Optimization of Machine Learning Algorithms*, Part of Advances in Neural Information Processing Systems 25 (NIPS 2012), 2012.

[3] B.Zoph and Q.V.Le. "Neural Architecture Search with Reinforcement Learning," *International Conference on Learning Representations*, 2017

[4] CIFAR-10 Data Set were created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton [Internet], https://www.cs.toronto.edu/~kriz/cifar.html.

[5] 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.

[6] A.Krizhevsky, I.Sutskever, and G.E.Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, Vol.60, No.6, pp.84-90, 2017.

[7] K.He, X.Zhang, S.Ren, and J.Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, pp.770-778, 2016.

[8] G.Huang, Z.Liu, L.Maaten, and K.Q.Weinberger, "Densely Connected Convolutional Networks," in Proceedings of the *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, pp.4700-4708, 2017.

[9] M.Mitchell and C.E.Taylor, "Evolutionary Computation: An Overview," *Annual Review of Ecology and Systematics*, Vol.30, pp.593-616, 1999.

[10] S.Lee, "Artificial Agent-based Bargaining Game considering the Cost incurred in the Bargaining Stage," *The Journal of the Korea Contents Association*, Vol.20, No.11, pp.292-300, 2020.

[11] J.Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[12] D.E.Goldberg, *Geneitc Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, 2008.

[13] J.Kennedy and R.Eberhart, "Particle Swarm Optimization," in Proceedings of The *IEEE International Conference of Neural Networks*, 4, pp.1942-1948, 1995.

[14] Wang, Dongshu, and L.Liu, "Particle swarm optimization algorithm: an overview," *Soft computing*, 22, pp.387-408, 2018.

[15] R.Strorn and K.Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, Vol.11, No.4, pp.341-359, 1997.

이 상 욱(Sangwook Lee) [정회원]

- 2007년 8월 : 광주과학기술원 정보기전공학부 (공학박사)
- 2007년 9월 ~ 2008년 9월 : 조지아공대 전산학과 박사후연구원
- 2008년 11월 ~ 2009년 2월 : 삼성전자 통신연구소 책임연구원
- 2009년 3월 ~ 현재 : 목원대학교 게임소프트웨어공학과 교수

〈관심분야〉
인공지능, 알고리즘, 사물인터넷, 정보통신, 게임