공학전문석사 학위 연구보고서

모바일 기기의 에너지 효율성을 고려한 신경망 아키텍처 탐색에 관한 연구

Neural Architecture Search considering energy efficiency of mobile device

2022년 2월

서울대학교 공학전문대학원
응용공학과 응용공학전공
김영윤

모바일 기기의 에너지 효율성을 고려한 신경망 아키텍처 탐색에 관한 연구

Neural Architecture Search considering energy efficiency of mobile device

지도교수 이영기

이 프로젝트 리포트를 공학전문석사 학위
연구보고서로 제출함
2022년 2월

서울대학교 공학전문대학원
응용공학과 응용공학전공
김영윤

김영윤의 공학전문석사 학위 연구보고서를 인준함
2022년 2월

위원장 오병수 (인)
위 원 이영기 (인)
위 원 김성우 (인)

초 록

모바일 기기 및 IoT 기기와 같은 임베디드 디바이스에서 사용 가능한 온 디바이스 AI 서비스에 대한 수요가 증가하고 있다. 온 디바이스 AI는 딥러닝 모델을 임베디드 디바이스에 내장하기 위한 기술로 클라우드 기반의 인공지능 기술 대비 저지연, 강화된 보안과 같은 장점이 있지만, 실행되는 하드웨어에 성능이 의존적이며 연산을 위해 프로세서, 메모리와 같은 많은 컴퓨팅 자원을 사용하여 과도한 전력을 소비한다. 이와 같은 이유로 경량 딥러닝 모델을 개발할 때 에너지 효율을 고려해야 한다.

본 연구에서는 모바일 기기의 에너지 효율을 고려한 에너지 효율적인 딥러닝 모델을 구성하는 방법으로 ELP-NAS를 제안한다. 딥러닝 모델이 모바일 기기에서 실행될 때 모델의 종단간 소비 전력과 지연 시간을 예측하고, 이 예측값들을 모델의 정확도와 함께 강화 학습 기반의 신경망 아키텍처 탐색을 통해 성능 좋은 모델을 탐색하고 학습한다.

CIFAR-10 데이터 세트에서 ELP-NAS의 정확도는 베이스라인 모델인 ENAS 대비 정확도는 0.35% 감소하여 1%미만으로 아주 작지만, 소비 전력과 지연 시간은 약 40% 개선된 것을 확인하였다.

주요어 : 신경망 아키텍처 탐색, 에너지 효율, 딥러닝

학번 : 2020-27656

목 차

그 림 목 차

표 목 차

제 1 장

서론

1.1 연구 배경 및 목적

스마트폰에 적용된 인공지능 기술은 사용자의 편의성을 높이는 방향으로 발전하고 있다. 사진을 업로드하면 이미지를 분석해서 유사한 상품을 검색하여 쇼핑을 할 수 있고, 카메라로 문서를 촬영하면 원하는 언어로 번역을 하거나 텍스트로 저장 할 수도 있다. 일례로 전 세계 10억 명 이상이 사용하고 있는 구글 포토(Google Photos)는 구글 드라이브에 업로드된 사진들을 딥러닝 기술을 이용하여 장소 및 사물을 인식하고, 사용자가 태그를 지정하지 않아도 알아서 사진을 분류하는 서비스를 제공한다. 이런 서비스는 초고속 이동통신 서비스와 무선 인터넷을 기반으로 스마트폰에서 수집한 정보를 중앙 클라우드 서버로 전송해 분석하고 다시 기기에 보내는 방식으로 제공되고 있다. 클라우드 기반의 인공지능 서비스는 모든 솔루션을 서버에서 제공하기 때문에 효율적으로 개발할 수 있고 기술 유지 보수가 편리한 장점이 있지만, 실제 네트워크 환경에서 폭증하는 트래픽으로 인한 성능 저하와 개인정보 유출에 대한 문제점을 갖고 있다.

이런 단점들을 보완하기 위해 모바일 및 IoT 기기와 같은 임베디드 환경에서 사용이 가능한 온 디바이스 인공지능(On-device AI) 서비스에 대한 수요가 증가하고 있다. 온 디바이스 인공지능은 클라우드 기반의 학습된 모델을 임베디드 디바이스에 내장하기 위한 필수 기술이다. 네트워크나 서버를 거치지 않고 스마트기기에서 AI 기능을 수행하여 저지연을

통한 빠른 작업이 가능함과 동시에 개인정보보호 등 보안을 강화할 수 있다. 또한 별도의 네트워크가 필요 없기 때문에 인터넷 연결이 어려운 환경에서도 실행이 가능하다는 장점이 있다.

모바일 기기에 온 디바이스 인공지능 기술을 적용할 때 고려해야 하는 조건이 있다. 우선 배터리 사용 시간이다. 스마트폰 배터리 사용시간을 늘리기 위해 배터리 용량을 늘리고 싶지만, 제품 크기 제한으로 인해 내장 배터리 용량을 증가시키는데 한계가 있다. 디바이스에서 딥러닝 연산을 하기 때문에 프로세서와 메모리 등 많은 컴퓨팅 자원을 사용하면서 소비 전력이 증가하게 된다. 이렇게 소비된 전력은 대부분 열에너지로 전환되어 전력 소모량에 비례하여 발열량이 증가하게 되고, 이로 인해 디바이스의 성능이 제한되거나 오작동의 원인이 된다. 그래서 한정된 용량의 배터리를 사용하는 임베디드 디바이스에서 전력 문제는 매우 중요한 이슈이다.

또한 스마트폰과 같은 임베디드 디바이스는 제한된 하드웨어 사양 및 컴퓨팅 능력의 한계로 인해 강력한 클라우드 기반의 컴퓨팅 자원을 바탕으로 제공하는 기존 서비스와 동일한 성능을 기대하기 어렵다. 예를 들어 PC와 스마트폰에서 Inception_V3 모델[1]의 이미지 추론 성능을 비교해보면, PC에서 사용하는 Nvidia Geforce GTX 1060 그래픽카드에서는 초당 140여 장의 이미지를 추론할 수 있다. 하지만 모바일 프로세서인 Snapdragon 855 에서는 초당 약 14장 정도 가능하며, Exynos 9810 에서는 1초에 약 5장 정도 추론이 가능하다[2]. 이와 같이 PC GPU(Graphics Processing Unit)와 스마트기기의 프로세서에서 딥러닝 모델을 처리하는 속도에 큰 차이가 있고, 스마트기기에 탑재되는 프로세서 종류에 따라 많은 성능 차이가 발생한다.

임베디드 환경에서 딥러닝 모델을 최적화하기 위해서는 기존의 학습

된 모델의 정확도를 유지하면서 모델의 크기를 줄이고 고속으로 추론할 수 있어야 한다. 또한 임베디드 하드웨어의 성능과 함께 에너지 효율성도 함께 고려되어야 한다.

1.2 연구 범위

기존에 딥러닝 모델의 전력 소모와 관련한 연구들[3, 4]이 진행되고 있지만, 에너지 소비를 평가하기 위해 구체적인 접근 방법이 부족하고, 텐서플로우 [5], Caffe2 [6], PyTorch [7] 와 같은 딥러닝 프레임워크에서 소비 전력 관련해서 제공하는 기능이 아직은 많이 부족하다 [8]. 또한 경량 딥러닝 알고리즘 개발[9, 10]을 위해 개발자가 합성곱 신경망(Convolutional Neural Network)을 구성할 수 있지만, 해당 분야의 전문지식이 많이 필요하다. 또한 학습에 사용하는 데이터들이 변경되거나 모델을 변경하게 되면 모델 최적화를 위해 기존에 진행한 작업을 계속 반복 수행해야 하고, 이를 위해 많은 시간과 컴퓨팅 자원이 필요하다. 이런 문제점들을 극복하기 위해 자동 기계 학습(Automated Machine Learning) 분야 중 신경망 모델 구조를 자동으로 탐색하는 신경망 아키텍처 탐색(Neural Architecture Search)을 적용하여 문제에 접근하고자 한다.

본 연구에서는 신경망 아키텍처 탐색을 이용하여 임베디드 환경에서 에너지 효율을 고려한 저전력 딥러닝 모델을 설계하는 방법을 제시한다. 이를 위해 딥러닝 모델의 요구사항을 맞추기 위해 딥러닝 모델의 에너지 소모량과 지연시간을 추정하는 알고리즘을 설계하고, 신경망 아키텍처 탐색에 적용하여 경량 딥러닝 모델을 설계하는 방법을 제공한다. 그리고 최종 설계된 모델에 대해 기존 연구들과 성능, 지연시간 뿐만 아니라 에너지 효율성과 딥러닝 설계 효율성에 대해 비교 평가하고 결과를 분석한다.

1.3 연구의 구성

본 연구 보고서는 총 5개의 장으로 구성되어 있으며 내용은 다음과 같다. 제1장에서는 본 연구를 수행하게 된 배경과 목적 그리고 연구 범위에 관해 서술한다. 제2장에서는 본 연구를 이해하기 위해 필요한 신경망 아키텍처 탐색에 대해 이론적인 내용에 대해 소개하고 이와 관련된 선행 연구를 살펴본다. 제3장에서는 본 연구에서 딥러닝 모델의 에너지 소모와 지연 시간에 대해 예측하는 방법과 해당 알고리즘을 적용하여 제안한 모델에 관해 설명하고 이를 평가하는 방법에 관해 설명한다. 제4장에서는 실험을 진행하기 위한 사용한 프레임워크에 대한 간략한 설명과 개발 환경 구성에 관해 설명하고, 다양한 실험 결과에 대한 비교 분석을 기술한다. 마지막으로, 제5장에서는 본 연구 보고서의 결론에 대해 작성하며 연구 결과 및 기술한 연구 내용을 향후 연구 과제와 함께 기술하며 끝맺는다.

제 2 장

관련 연구

본 장에서는 연구 보고서의 배경이 되는 신경망 아키텍처 탐색에 관한 기본적인 개요를 제공하고 선행 연구에 대해 소개한다. 먼저 제1절에서는 본 연구의 기본이 되는 신경망 아키텍처 탐색에 대해 전반적으로 설명한다. 이어서, 제2절에서는 신경망 아키텍처 탐색에 대한 선행 연구에 대해서 소개한다.

2.1 신경망 아키텍처 탐색

자동 기계 학습(AutoML)은 딥러닝 모델 개발에서 시간이 오래 걸리는 반복적인 작업을 자동화하는 프로세스이다. 기존의 복잡하고 반복적이던 작업을 자동으로 처리해주기 때문에 자동 기계 학습을 이용하여 학습 데이터를 입력으로 제공하면 쉽고 효율적으로 최적화된 모델을 출력으로 받을 수 있다.

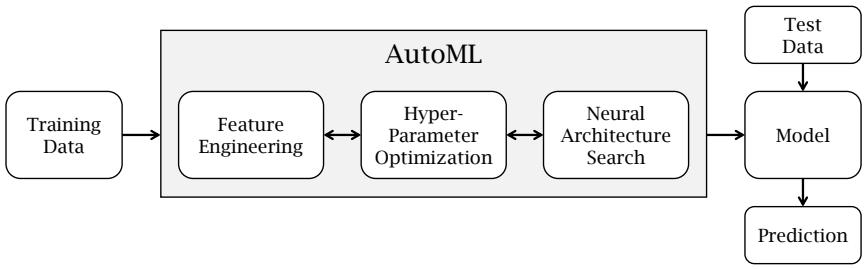


그림 1: 자동 기계학습 시스템 개요

자동 기계학습의 주요 프로세스는 그림 1 [11]과 같이 학습에 필요한 학습률이나 반복 횟수 등 하이퍼 파라미터를 최적화하여 모델을 학습시키는 하이퍼 파라미터 최적화(Hyper Parameter Optimization), 학습을 통해 유의미한 특징(Feature)을 추출해서 입력으로 사용하는 특성 추출(Feature Engineering), 그리고 사람이 직접 모델을 설정하던 것을 학습을 통해 최적의 아키텍처를 설계하는 신경망 아키텍처 탐색(Neural Architecture Search)으로 나눠볼 수 있다.

이 중에서 본 연구에서는 아키텍처 설계를 딥러닝으로 해결하기 위한 방법인 신경망 아키텍처 탐색에 대해 살펴보려고 한다. 즉 딥러닝으로 딥러닝 모델을 찾는 것이라고 할 수 있다.

신경망 아키텍처 탐색은 그림2[12]과 같이 크게 탐색 영역(Search Space) 설계, 탐색 전략(Search Strategy), 그리고 성능 평가 전략(Performance Estimation Strategy)로 구성되어 있으며, 각 구성에 대해 설명한다.

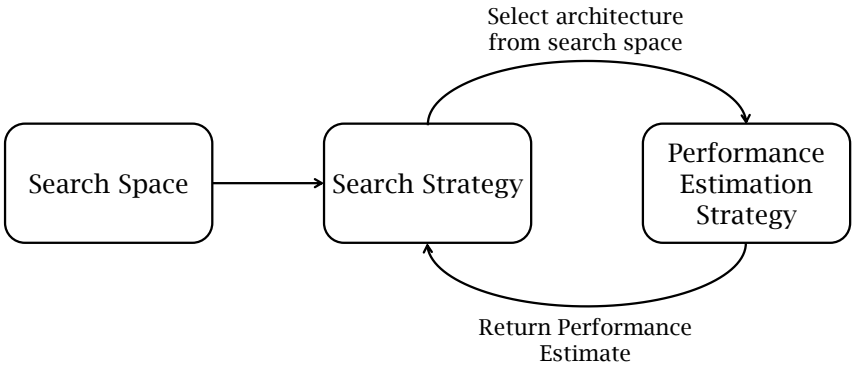


그림 2: 신경망 아키텍처 탐색 흐름도

2.1.1 탐색 영역 설계

탐색 영역(Search Space)는 표현할 수 있는 신경망 아키텍처(Neural Architecture)를 탐색할 수 있는 범위를 의미한다.

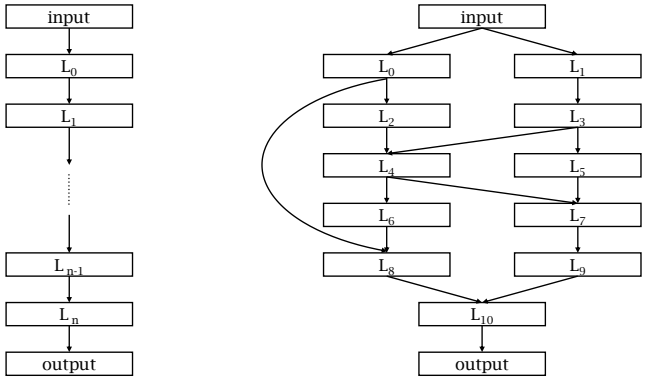


그림 3: 체인 구조 탐색 영역 예시

가장 기본적인 탐색 영역은 그림3[12]의 좌측 그림과 같이 체인 구조 탐색 영역(chain-structured search space)이다. 체인 구조 신경망은 N개의 레이어(Layer)로 구성되어 있으며 i 번째 레이어는 $(i-1)$번째로부터 입력을 받고 $(i+1)$번째 레이어에 출력을 전달하는 구조이다. 탐색 영역에서 설정할 수 있는 파라미터들은 레이어의 최대 개수, 레이어에서 실행하는 오퍼레이션의 종류, 오퍼레이션에서 설정할 수 있는 하이퍼 파라미터 설정(필터 개수, 크기 등)등을 설정할 수 있다[13, 14].

또한 그림3의 우측 그림과 같이 단순한 체인 구조를 보다 복잡하게 설정하기 위해 i 번째 레이어의 출력을 j 번째 레이어의 입력으로 설정하는 스킵 연결(Skip connection)이나, 여러 개의 브랜치를 만들어 다중 입출력 연결을 설정하는 멀티 브랜치(Multi branch)와 같은 구조도 적용할 수 있다 [1, 10].

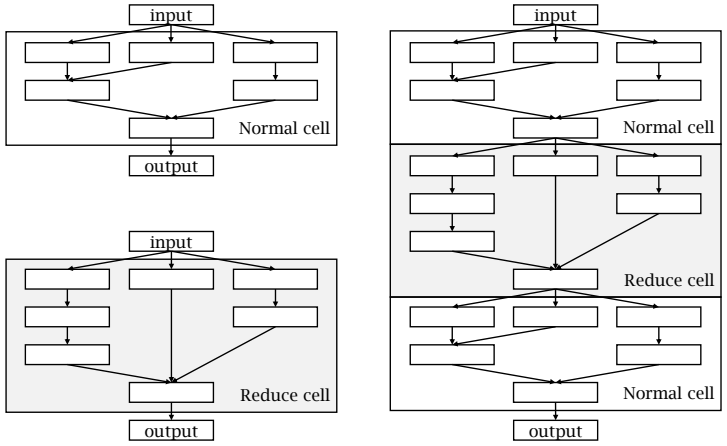


그림 4: 셀 기반 탐색 영역

셀 기반 탐색 영역(cell-based search space)은 그림4[12]과 같이 입력의 차원을 유지하는 Normal Cell과 차원을 감소시키는 Reduce Cell을 찾고, 찾은 셀들을 정해진 방법으로 쌓은 신경망 구조를 의미한다[15]. 셀 기반의 탐색은 Normal cell과 reduce cell을 탐색하기 때문에 전체 아키텍처를 찾는 것보다 탐색 영역의 크기가 크게 감소하게 되고, 목표로 하는 데이터 세트에 따라 셀의 개수나 쌓는 방법을 변경하여 적용이 가능하다는 장점이 있다.

이렇게 탐색 영역에서 기존 연구를 통해 알고 있는 사전 지식(Prior Knowledge)을 반영하면 검색 공간의 크기를 줄이고 검색을 단순화할 수 있는 장점이 있다. 하지만 이것이 인간의 편견(Human bias)으로 적용되면 오히려 새로운 아키텍처를 찾는 것에 방해가 되는 문제가 발생할 수 있다.

2.1.2 탐색 전략

탐색 전략(Search strategy)은 탐색 영역을 탐색하는 방법으로 다양한 방법 중에서 대표적으로 강화학습 기반의 탐색 [13, 16, 15, 17, 13]과 진화 알고리즘 기반의 탐색 알고리즘 [18, 19]이 있다.

강화 학습(Reinforcement Learning)은 기계학습의 한 영역으로, 정해진 환경(Environment) 안에서 정의된 주체(Agent)가 현재 상태(State)를 인식하여 보상(Reward)을 최대화하는 행동(Action)을 선택하는 방법이다.

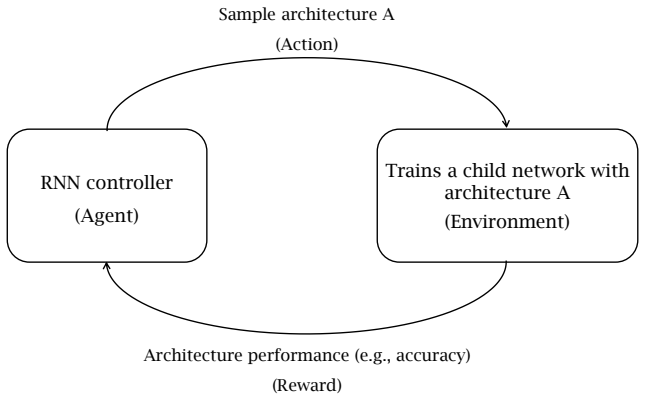


그림 5: 강화 학습 기반 탐색

강화 학습 기법을 신경망 아키텍처 탐색 문제에 적용하면 그림5과 같이 행동, 환경, 보상의 요소들을 탐색, 탐색 영역, 아키텍처 성능으로 각각 표현할 수 있다. 순환 신경망(RNN: Recurrent Neural Network)을 기반으로 하는 제어기(Controller)가 뉴럴 아키텍처의 파라미터들을 결정하고, 결정된 파라미터를 기반으로 생성된 여러 개의 신경망 네트워크를 학습하여 정확성(Accuracy)을 보상(Reward)으로 사용하여 신경망의 성능이 최대로 하는 네트워크를 탐색한다.

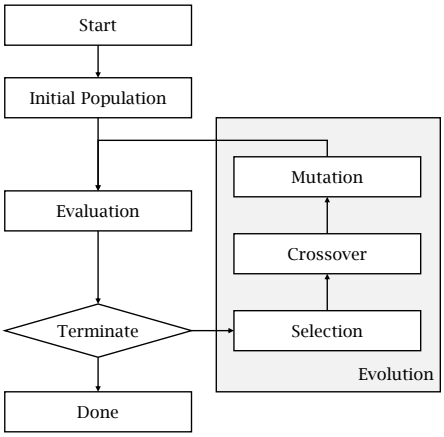


그림 6: 진화 알고리즘 기반 탐색

다음으로 그림6와 같이 생물학적 진화로부터 영감을 얻은 진화 알고리즘 (Evolutionary Methods)은 임의의 후보해들의 집합(Population)을 생성하고 '선택-교차-변이-평가'의 과정을 반복하여 해당 집합을 업데이트하고 가장 만족스러운 후보해를 탐색하는 방법이다.

이런 진화 알고리즘을 신경망 아키텍처 탐색문제에 적용하면 선택(Selection)은 신경망 구조에서 후보군을 선택하고, 교차(Crossover)는 후보군 중에서 임의의 후보를 뽑아 자손(offspring)을 생성하고, 자손의 돌연변이를 만드는 과정인 변이(Mutation)은 레이어 추가나 삭제, 하이퍼파라미터 변경 등을 통해 다시 평가하여 새로운 후보군을 전달하는 과정으로 표현할 수 있다.

2.1.3 성능 평가 전략

성능 평가 전략(Performance estimation strategy)은 탐색 전략으로부터 탐색한 모델에 대한 성능을 평가하는 방법을 말한다. 새로운 데이터

(Unseen data)나 검증 데이터 세트(Validation dataset)에 대해서 가장 좋은 성능을 보이는 모델을 탐색하기 위해 탐색한 모델을 적절하게 평가하는 것이 중요하다. 일반적으로 학습 데이터 세트(Train dataset)로 학습하고 검증 데이터 세트로 성능을 평가하는 것이 가장 간단한 방법이지만, 데이터가 많은 경우 모델을 학습하기 위해 많은 연산량과 시간이 소요된다. 모델의 학습 시간을 줄이기 위해 학습 횟수나 학습 데이터 수를 줄이거나, 데이터를 분할하여 학습하는 방법을 적용할 수 있고, 적은 횟수로 학습하여 모델의 최종 성능을 추정하는 방법을 적용할 수 있다. 또한 탐색한 모델을 매번 처음부터 학습하는 것이 아니라 기존의 가중치를 상속하거나, 처음에 큰 모델을 만들고 그 모델에서 아키텍처를 탐색하여 학습된 모델의 가중치를 공유하는 방법도 있다. 이렇게 학습 시간을 단축시켜 성능 평가 속도를 높일 수 있다.

2.2 선행 연구

2.2.1 NAS

NAS[16]는 구글의 딥 러닝 인공지능을 연구하는 팀인 구글 브레인(Google Brain)에서 2017년 ICLR(International Conference on Learning Representations)에 발표한 내용으로 신경망 아키텍처 탐색(Neural Architecture Search)이라는 방법을 처음으로 제안하였다.

탐색 영역은 체인 구조를 기반으로 하여 순환 신경망 제어기로 부터 여러개의 신경망 아키텍처를 탐색한다. 그리고 강화 학습 기반으로 생성된 네트워크(child network)를 학습하고 해당 모델의 정확성을 강화 학습의 보상으로 사용하여 성능이 최대로 하는 네트워크를 탐색한다. 또한 탐색 영역을 넓히기 위해 스킵 연결과 멀티 브랜치도 적용하였다.

CIFAR-10 데이터 세트를 대상으로 3.65%의 테스트 에러율(test error rate)을 달성하여, 당시 유사한 아키텍처를 사용한 모델인 DenseNet[20] 보다 성능이 0.09% 좋았다. 하지만 신경망 아키텍처 탐색을 위해 800대의 GPUs (Nvidia K40 GPUs)를 활용하여 28일의 시간이 소요되어 22,400 GPU Days의 연산량이 필요로 한다. 1 GPU Days는 1대의 GPU로 1일의 시간이 소요되는 것을 가리키는데, 이를 환산하여 GPU 1대로 탐색을 한다면 약 61.4년의 시간이 필요하게 된다.

2.2.2 NASNet

NASNet[15]은 NAS[16] 의 후속 연구로 이미지 분류 모델에 대한 탐색 복잡도를 개선하기 위해 탐색 영역에 제약조건을 두어 네트워크를 탐색하는 방법을 제안하였다. 셀 기반 탐색 영역을 적용하여 전체 네트워크 대신 최적의 합성곱 셀(Convolutional Cell)을 탐색하고 해당 셀을 반복적으로 쌓아서 네트워크를 구성한다.

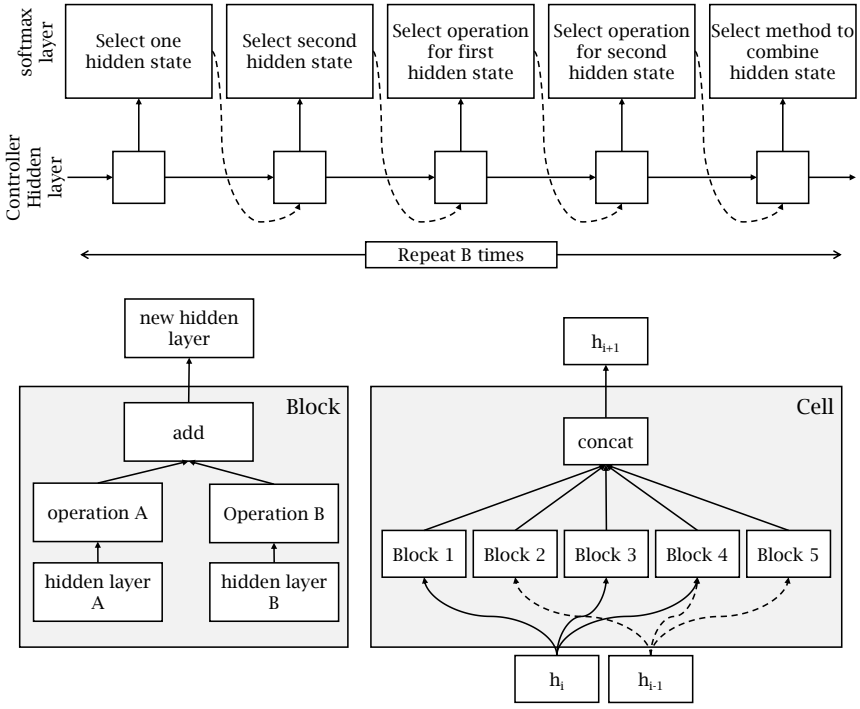


그림 7: NASNet 탐색 영역

그림7의 상단 그림과 같이 신경순환망 제어기로 부터 두개의 입력값, 두개의 연산자, 한개의 병합 연산자, 이렇게 다섯 가지의 파라미터가 결정되도록 학습한다. 이때 사용되는 연산자는 다음과 같다.

- Identity
- 1x1 convolution
- 3x3 convolution
- 3x3 max pooling
- 5x5 max pooling
- 7x7 max pooling
- 3x3 average pooling
- 1x3 then 3x1 convolution
- 1x7 then 7x1 convolution
- 3x3 diltated convlolution
- 3x3 depthwise-separable conv
- 5x5 depthwise-separable conv
- 7x7 depthwise-separable conv

이 다섯 가지의 파라미터는 합성곱 셀의 신경망을 구성하는 단위 구조인 블록(Block)을 구성하는 요소가 되는데, 그림7의 하단 좌측 그림과 같이 두개의 입력을 받아 연산을 처리하는 노드와 이 두 노드의 결과를 합하여 결과를 출력하는 노드로 구성된다. 그리고 이 블록들을 일정한 개수로 연결하여 하나의 레어어로 구성하면 그림7의 하단 우측 그림과 같이 셀(Cell)이 되고 앞서 설명한 내용과 같이 셀 기반 탐색 영역에서 입력과 출력의 차원이 같은 Normal Cell과 출력의 차원이 입력의 절반이 되는 Reduction Cell를 탐색한다.

NASNet은 CIFAR-10 데이터셋을 기반으로 2.4%의 테스트 에러율을 달성하여 기존 NAS [16] 과 비교했을때 네트워크의 성능이 더 좋은 것을 확인하였다. 또한 탐색 시간도 2,000 GPU days로 기존 NAS [16] 보다 탐색 시간이 많이 개선되었지만, 500대의 GPUs (Nvidia P100s GPUs)를 활용하여 4일의 시간이 필요하여 여전히 신경망 아키텍처 탐색에 많은 자원이 필요로 하는 것을 알 수 있다.

2.2.3 MnasNet

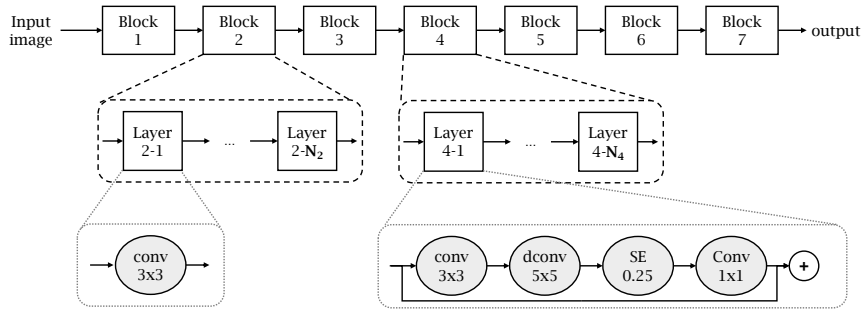


그림 8: MnasNet 탐색 영역

MnasNet[21]은 모바일 기기에서 정확도가 높고 지연 시간이 짧은 CNN(Convolutional Neural Network) 모델을 찾기 위해 제안되었다. 앞선 연구와 동일하게 강화 학습 기반으로 네트워크를 탐색하지만 실제 모바일 기기인 Google Pixel 스마트폰에서 모델을 직접 실행하여 측정한 실행 시간(latency)를 모델의 정확도(Accuracy)와 함께 에이전트의 보상함수에 반영하여 탐색의 정확도를 개선하였다. 기존 연구에서는 몇 개의 복잡한 셀을 검색한 다음 동일한 셀을 반복적으로 연결하는 구조였다면, MnasNet은 그림8[21]와 같이 탐색 성능을 개선하기 위해 합성곱 신경망 모델을 고유한 블록으로 나누고 블록을 개별적으로 검색하여 서로 다른 블록에서 서로 다른 아키텍처를 구성한다. 각 블록을 구성하는 연산자 종류, 커널 크기, 계층 개수 등과 같은 파라미터들을 탐색할 있도록 선택하여 탐색 영역의 크기를 축소시켰다. ImageNet[22] 데이터 세트 기준으로 MnasNet은 MobileNet V2(1.4x)[9]보다 0.5% 더 높은 정확도로 Pixel 스마트폰에서 1.8배 빠르게 실행되었다. 그리고 자동 탐색된 CNN 모델과 비교하면 NASNet-A 보다 1.2% 정확도가 높고 2.3배 더 빠르게 실행되었다.

2.2.4 ENAS

앞서 기술한 NAS[16], NASNet[15], MnasNet[21]과 같은 강화 학습 기반의 신경망 아키텍처 탐색 기법의 경우 많은 컴퓨팅 비용과 시간이 소요된다. 그 이유는 신경망 제어기로 부터 샘플링된 차일드 모델(Child Model)들이 학습된 가중치를 모두 버리고 처음부터 재학습하여 정확도를 측정하기 때문이다. ENAS[23]는 이런 탐색 비효율성을 극복하기 위해 모든 차일드 모델 간의 가중치를 공유하여 신규 탐색된 모델이 처음부터 재학습하지 않도록 하여 탐색 효율성을 개선하였다. 탐색할 수 있는 모든 모델을 하나의 DAG(Directed Acyclic Graph)로 표현하고 샘플링된 차일드 모델은 서브 그래프로 정의하고, 각 노드에서 연산이 동일하면 파라미터를 공유하여 사용한다. DAG은 방향성 비순환 그래프는 개별 노드들이 방향성을 가지면서 서로 순환하지 않는 구조로 짜여진 그래프를 말한다. 이렇게 DAG 구조를 사용하여 합성곱 신경망 전체 구조를 찾는 것을 macro search라고 하고, 셀 기반의 탐색 영역과 같이 합성곱 셀을 탐색하는 방식을 micro search 라고 한다. 두 방식을 비교해보면 micro search의 성능이 더 좋다고 알려져 있으며, micro search를 예를 들면 그림9과 같다.

예를 들어 노드가 4개인 경우를 가정하면, 그림9의 하단 좌측 그림과 같이 DAG으로 표현하면 4개의 노드에서 실선으로 연결된 노드가 활성화된 연결을 의미한다. 상단 그림에서 노드1과 노드2는 셀의 입력이 되고 컨트롤러는 노드3과 노드4를 찾으면 된다. 노드3과 노드4의 출력을 살펴보면 각각 4개의 출력으로 표현되는데 첫번째와 두번째 출력은 각 오퍼레이션의 입력 노드를 의미하고, 세번째와 네번째 출력은 각 오퍼레이션의 종류를 의미한다. 이렇게 상단 그림에서 찾은 결과를 합성곱 셀로 표현하면 하단 우측 그림과 같이 나타낼 수 있다.

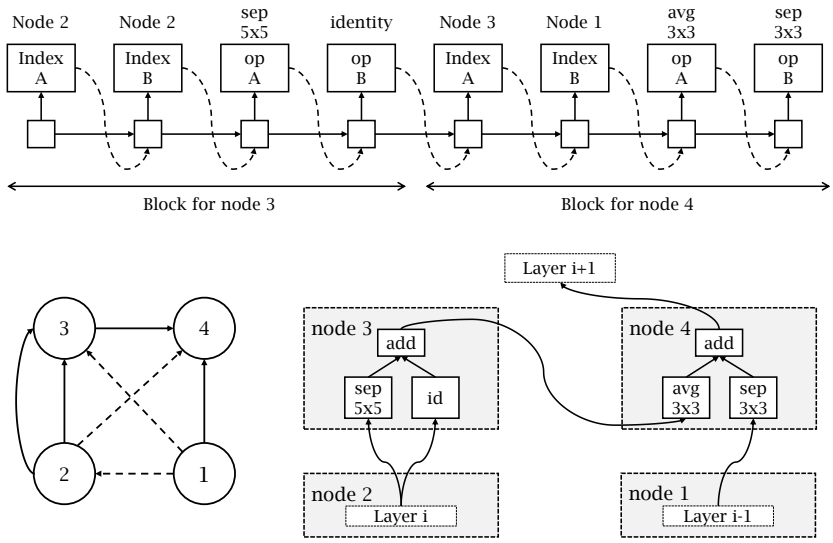


그림 9: 마이크로 탐색 기반 ENAS 탐색 영역

ENAS는 CIFAR-10 데이터셋을 기반으로 2.89%의 테스트 에러율을 달성하여 NASNet에 근접한 테스트 에러율를 달성하였지만, 합성곱 신경망 구조를 0.45 GPU days 만에 탐색하여 기존 NAS 대비 50,000배, NASNet 대비 4400배의 탐색 속도를 개선하였다.

제 3 장

연구 방법

본 장에서는 문제를 정의하고 이를 해결하기 위한 방법으로 딥러닝 모델의 정확도, 지연시간 및 소비 전력을 고려한 다목적 최적 설계에 관해 설명한다. 그리고 다목적 함수에 필요한 딥러닝 모델의 종단간(End to End) 소비 전력 및 지연시간을 예측하는 알고리즘을 구현하고 이를 신경망 아키텍처 탐색 모델에 적용하여 저전력 딥러닝 모델을 탐색하는 방법에 대해 연구한다. 마지막으로 딥러닝 모델의 성능 평가 방법에 관해 기술한다.

3.1 문제 정의

모바일 기기에 탑재되는 딥러닝 모델을 최적화하기 위해서는 저지연, 고성능 그리고 저전력과 같은 요구 사항들을 고려해야 한다. 정확도를 높이기 위해 딥러닝 모델의 신경망 층을 많이 쌓으면 지연 시간이 길어지고 에너지 소비가 증가하게 된다. 반대로 지연 시간을 줄이기 위해 신경망 층을 적게 쌓으면 에너지 소비는 줄이게 되지만 정확도가 떨어질 수 있다[24].

이렇게 한 개 이상의 목적 함수를 동시에 고려하는 것을 다목적 최적 설계(Multi-Objective optimization)라고 한다. 목적 함수의 상충하는 특성으로 인해 하나의 최적해가 존재하는 것이 아니라, 가장 최적의 집합을 찾는데 이를 파레토 최적해(Pareto optimal solution)[25]라고 한다.

본 연구에서는 파레토 최적해를 찾는 다양한 방법 중 선행 연구[21]에서 검증된 가중치 곱 방법(the weighted product method)을 확장하여 적용한다. 다목적 함수는 모델의 높은 정확도, 짧은 지연 시간과 낮은 소비 전력의 조건에 충족하는 최적의 모델을 찾는 것이 목표이며 이를 수식 (3.1)과 같이 정의한다. 주어진 모델을 m이라고 하면, $ACC(m)$는 해당 모델의 정확도를 의미한다. $LAT_{est}(m)$와 $ENERGY_{est}(m)$는 모바일 기기에서 딥러닝 모델을 실행할 때 지연 시간과 소비 전력을 예측한 값을 의미한다. 그리고 T는 지연 시간 목표 값, E는 소비 전력 목표 값을 의미한다.

$$\underset{m}{\text{maximize}} \quad ACC(m) \times \left[\frac{LAT_{est}(m)}{T}\right]^{w} \times \left[\frac{ENERGY_{est}(m)}{E}\right]^{w} \tag{3.1}$$

w는 가중치 계수(weight factor)로 정의하며 다음과 같다.

$$w = \begin{cases} \alpha, & \text{if } LAT_{est}(m) \leq T,\ ENERGY_{est}(m) \leq E \\ \beta, & \text{otherwise} \end{cases} \tag{3.2}$$

α 와 β 는 가중치 계수로 사용할 상수로 파레토 최적해가 서로 다른 정확도, 지연 시간과 소비 전력의 트레이드오프(TRADE-OFF) 관계에서 유사한 보상을 하기 위해 설정하는 값이다. 그래서 가중치 계수 상수(α, β) 설정에 따라 보상값의 차이가 발생한다. 가중치 계수 상수는 선행 연구[21] 결과를 바탕으로, 목표치를 넘는 경우 보상 값을 급격하게 낮추는 강성 제약(Hard constraint)과 목표치를 넘어도 값을 부드럽게 조정하여 보상 값을 낮추는 연성 제약(Soft constraint)으로 나눠 설정하였다.

예를 들어 그림10과 같이 모델의 정확도가 0.8이고 목표 지연 시간이 80ms, 목표 소비 전력이 80uAh 인 경우를 가정하여 시뮬레이션을 해보면 가중치 계수 상수의 설정에 따라 보상값에 차이를 확인할 수 있다.

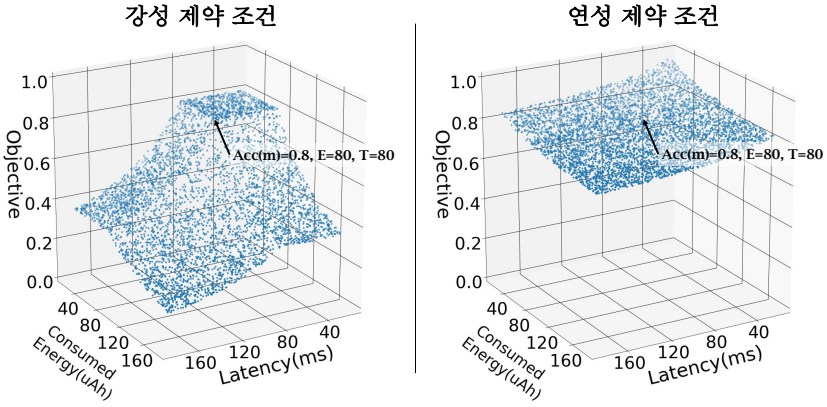


그림 10: 제약 조건에 따른 목적 함수 결과

강성 제약 조건($\alpha = 0, \beta = -1$)에서는 그림10의 좌측 그래프와 같이 목표치보다 값이 작거나 같은 경우에는 모델 정확도를 목표 값으로 사용하고 목표치보다 값이 큰 경우 급격하게 감소시켜 목표 값으로 설정한다. 연성 제약 조건($\alpha = -0.07, \beta = -0.07$)은 우측 그래프과 같이 목표치보다 값이 작은 경우에는 목표 값을 약간 증가시키고 목표치보다 큰 경우에는 목적 값을 약간 감소시켜서 전체적으로 값을 부드럽게 설정한다. 본 연구에서는 강성 제약 조건과 연성 제약 조건을 모두 살펴보고 어떤 제약 조건이 학습에 더 유리한지 살펴본다.

3.2 ELP 알고리즘

본 연구에서 정의한 다목적 함수에서 $LAT_{est}(m)$과 $ENERGY_{est}(m)$는 모바일 기기에서 딥러닝 모델이 실행할 때의 지연 시간과 소비 전력을 예측한 값이다. 선행 연구[21]와 같이 직접 스마트폰에서 소비 전력과 지연 시간과 측정하면 정확한 값을 얻을 수 있지만, 매번 탐색 영역에서 샘플링한 모델을 스마트폰에서 실행하기 위해 변환하고 실험하는데 많은 시간과 자원이 소모된다. 이런 문제점을 해결하기 위해 딥러닝 모델의 종단간 소비 전력과 지연 시간을 예측하는 알고리즘(Energy and Latency Predictor)을 제안한다.

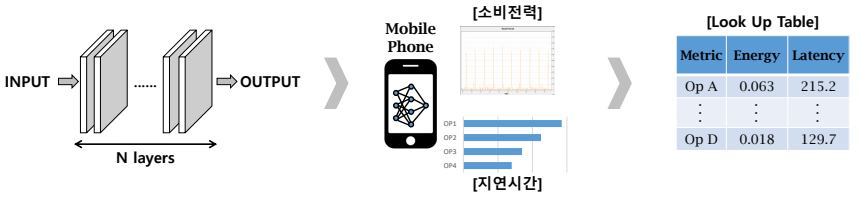


그림 11: 오퍼레이션 모델 측정 및 LUT 생성 방법

딥러닝 모델은 다양한 오퍼레이션(Operation)들이 서로 연결된 구조이다. 그림11과 같이 동일한 오퍼레이션을 반복적으로 N개를 연결하여 오퍼레이션 모델을 생성한다. 이 모델을 모바일 기기에서 실행하여 소비 전력과 지연 시간을 측정한다. 그리고 측정 결과를 오퍼레이션 개수로 나눈 평균 값을 LUT(Look Up Table)로 생성한다. 이 방법으로 탐색 영역에서 사용하는 모든 오퍼레이션의 소비 전력과 지연 시간 테이블을 생성하여 테이블로 생성한다. 본 연구에서는 LUT를 기반으로 탐색된 모델의 아키텍처를 분석하여 소비 전력과 지연시간을 예측하는 알고리즘을 구현하였다.

3.3 ELP-NAS 시스템

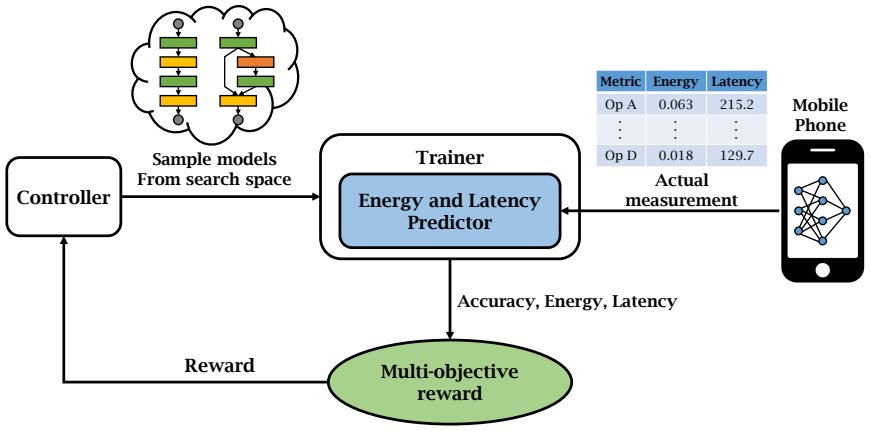


그림 12: ELP-NAS 시스템 개요

본 연구에서 모바일 환경에서 에너지 효율적인 모델 탐색 방법으로 딥러닝 모델의 소비 전력과 지연 시간을 예측하는 ELP 알고리즘(Energy and Latency Predictor)를 적용한 ELP-NAS 시스템을 제안한다.

아키텍처 탐색을 위한 시스템 개발을 위하여 컴퓨팅 자원을 효율적으로 사용하고 빠르게 학습할 수 있는 베이스라인 모델을 선정하였다. 그래서 본 연구에서는 강화 학습 기반의 선행 연구들 중에서 마이크로 탐색 영역 기반의 ENAS[23]를 베이스라인 모델로 선정하였다. 베이스라인 모델은 학습시간이 GPU 1대로 반나절정도 소요되고 기존 연구들과 동등한 수준의 성능을 보여준다. 그리고 탐색 영역이 한정되어 있으며, 셀 기반의 확장 가능한 아키텍처라는 장점을 갖고 있다.

베이스라인 모델의 기본적인 구조는 그림13과 같이 블록(Block) 단위이며, 두개의 오퍼레이션 연산으로 구성되어 있다. 그리고 N개의 블록이 모여 셀(Cell)을 구성하고, 셀이 반복적으로 연결되어 전체 네트워크를

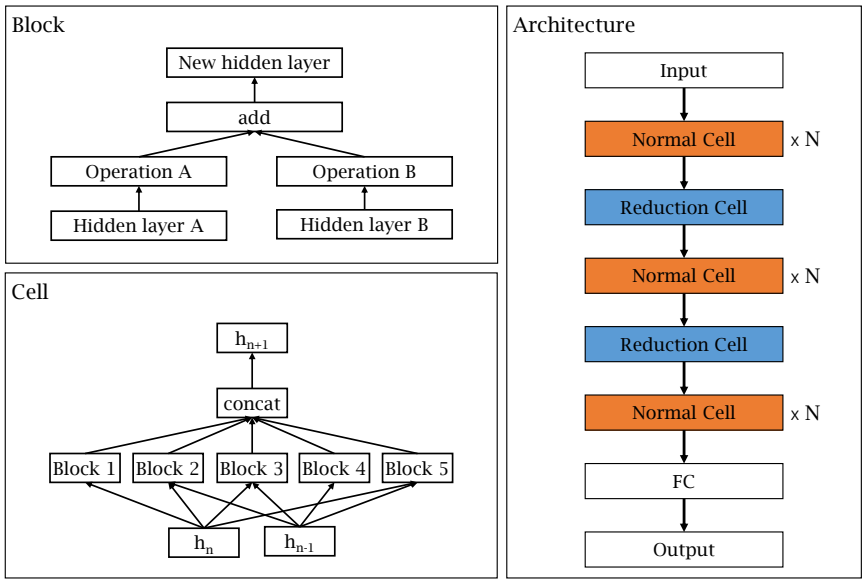


그림 13: ELP-NAS 탐색 영역

구성한다. 셀은 입력 차원을 감소시키는 일반 셀(Normal Cell)과 차원을 감소시키는 축소 셀(Reduction Cell)이 있이며 베이스라인 모델은 셀 단위로 탐색한다. 베이스라인 모델에서 사용할 수 있는 오퍼레이션 종류는 총 5가지로 3x3 separable convolution, 5x5 separable convolution, average pooling, max pooling, identity 를 사용할 수 있고, 필터크기 및 개수는 32x32x36, 16x16x72, 8x8x144 를 사용하여 탐색 영역이 한정되어 있다. 또한 아키텍처를 구성할 때 블록 개수와 셀 개수를 변경할 수 있어서 확장이 가능하다.

본 연구에서 제안한 ELP-NAS는 베이스라인 모델에서 가장 성능이 좋은 최종 모델을 기반으로 아키텍처를 구성하여, 5개의 블록이 하나의 셀을 구성하고 15개의 일반 셀로 구성하여 5개의 일반 셀마다 하나의 축소 셀이 순차적으로 연결되는 아키텍처 구조를 갖는다.

이와 같이 ELP-NAS는 탐색 영역이 한정되어 있고 전체 아키텍처를 구성하는 셀과 블록의 개수가 정해져 있기 때문에 ELP 알고리즘 적용이 가능하다. ELP 알고리즘의 기반이 되는 오퍼레이션 LUT를 생성하기 위해 각 오퍼레이션별로 레이어 $N = 100$인 모델을 생성하여 소비 전력과 실행 시간을 측정하였다. 5개의 오퍼레이션 중에서 identity는 입력과 모양이 같은 출력을 반환하여 실행 시간 및 소비 되는 전력이 없기 때문에 측정에서 제외하였다. 그래서 총 12개의 오퍼레이션 모델을 생성하여 LG G8 스마트폰에서 측정하였으며, 각 모델별로 총 100회씩 측정하고 평균 값을 기준으로 LUT를 생성하였다. 오퍼레이션별로 소비 전력과 지연 시간 측정 결과는 그림14과 그림15과 같다. 측정 결과 기반으로 각 오퍼레이션의 LUT를 생성하고, ELP 알고리즘은 컨트롤러에서 탐색한 모델의 구조를 분석하여 LUT 기반으로 소비 전력과 지연 시간을 예측한다. 그리고 예측한 소비 전력과 지연 시간을 다음 모델을 탐색하기 위한 입력 값을 사용한다.

측정 결과를 살펴보면, 오퍼레이션 기준으로 비교해보면 5x5 separable convolution 연산이 3x3 separable convolution 연산 보다 소비 전력과 지연 시간이 크고, Max Pooling 연산이 Average Pooling 연산보다 소비 전력과 지연 시간이 더 크게 측정되었다. 그리고 필터 사이즈 기준으로 살펴보면 32x32x36, 16x16x72, 8x8x144 순으로 소비 전력이 많고 지연 시간이 길게 측정되었다. 각 오퍼레이션의 32x32x36 의 측정 결과가 높게 측정되어 다른 연산에 비해 상대적으로 더 많은 CPU 자원으로 쓴 것으로 볼 수 있다.

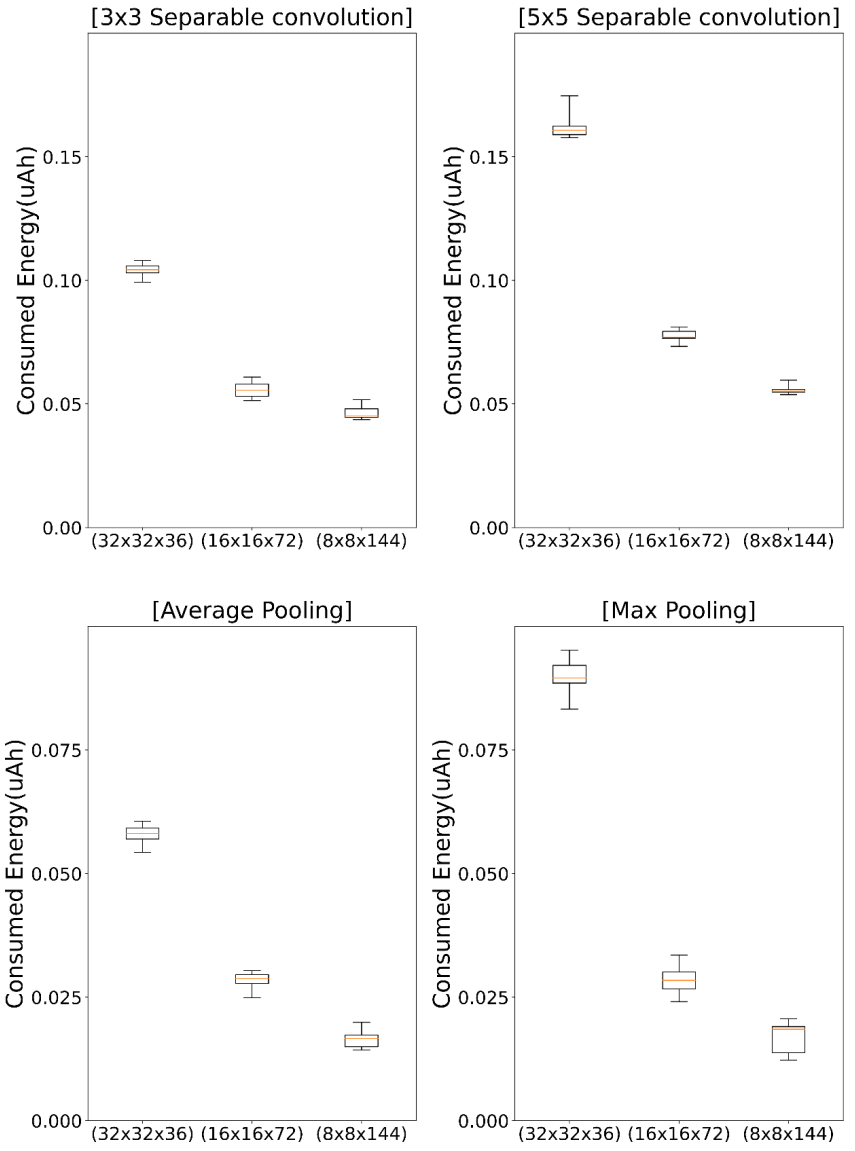


그림 14: 오퍼레이션 소비 전력 측정 결과

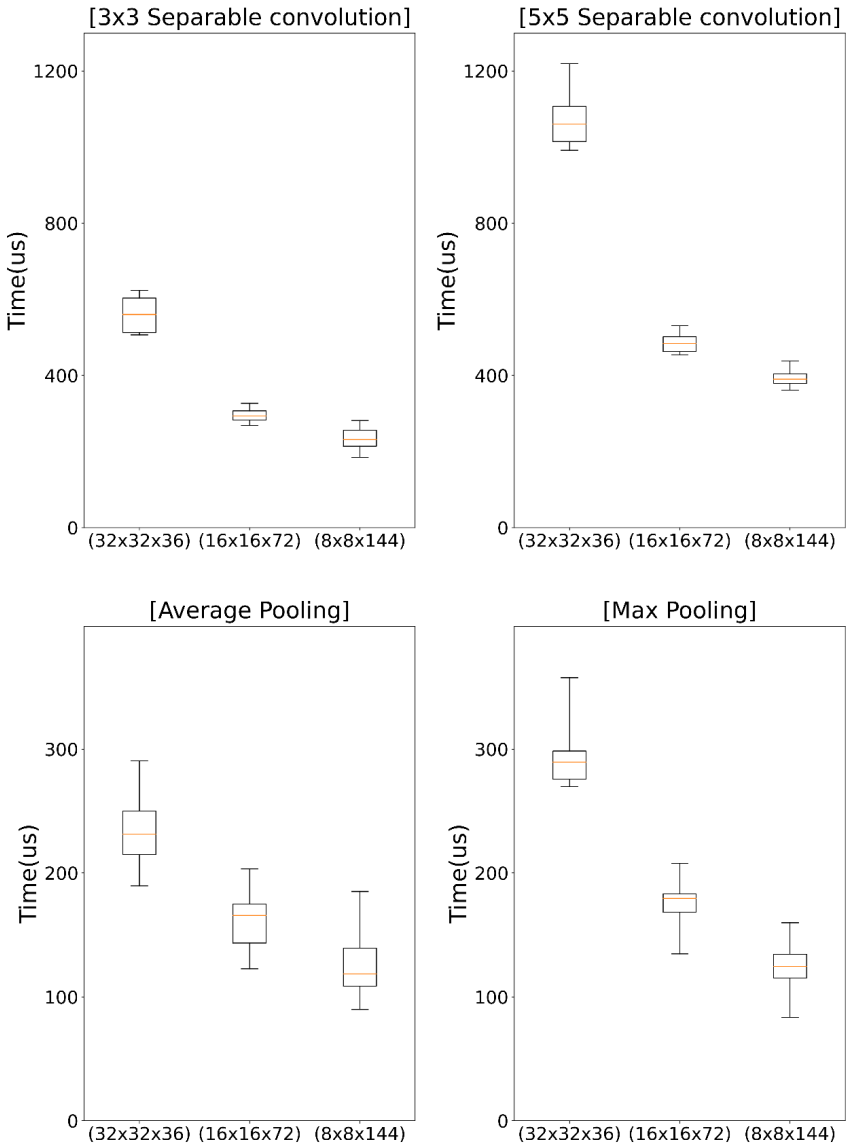


그림 15: 오퍼레이션 지연 시간 측정 결과

3.4 성능 평가 방법

일반적으로 딥러닝 모델 설계의 효율성을 평가하는 기준으로 가장 많이 사용하는 방법은 그래프를 통한 해석 방법이다. 예를 들어 전체 학습 파라미터 수, FLOPS(FLoating point Operations Per Second), 추론 시간 그리고 지연시간과 같은 항목의 값에 대비하여 정확도를 그래프로 표현하고 각 항목에서 같은 값을 기준으로 더 높은 정확도를 보이는 경우를 더 좋은 성능을 보인다고 평가할 수 있다.

기존의 성능 평가 방법 이외에도 학습 파라미터 수 대비 정확도의 비율을 측정하는 정보 밀집도(Information Density)와 학습 파라미터 수 및 추론 시간 동안의 연산수 대비 정확도를 측정하는 NetScore[26]가 있다.

정보 밀집도 $D(N)$와 NetScore $\Omega(N)$는 아래와 같이 정의한다.

$$D(N) = \frac{a(N)}{p(N)} \tag{3.3}$$

$$\Omega(N) = 20\, log\left(\frac{a(N)^{\alpha}}{p(N)^{\beta}\, m(N)^{\gamma}}\right) \tag{3.4}$$

$a(N)$은 정확도, $p(N)$은 학습 파라미터 수 그리고 $m(N)$은 추론 시간 동안의 연산수 MAC(multiply-accumulate)이며, NetScore의 상수는 $\alpha = 2, \beta = 0.5, \gamma = 0.5$ 이다.

본 연구에서 성능 평가 방법은 지연시간 대비 정확도, 소비 전력 대비 정확도를 그래프로 표현하여 평가하고, 정보 밀집도와 NetScore에 대해서도 평가한다.

제 4 장

실험 및 결과

4.1 실험 개요

실험을 설계하기 앞서 실험을 위한 환경 설정이 필요하다. 딥러닝 프레임워크 개발 환경을 설정하고 학습에 필요한 데이터 세트를 선정한다. 본 연구에서 제안한 ELP-NAS 모델 기반으로 기본 모델의 소비 전력와 지연보다 개선된 목표 값을 설정하여 모델을 학습한다. 학습하는 과정에서 다목적 함수의 가중치 값인 α 와 β 파라미터에 대한 설정값에 따라 학습 결과에 어떤 영향을 미치는 확인한다. 그리고 가장 좋은 성능의 모델을 기준으로 기존 연구된 모델과 성능, 소비전력 그리고 지연시간을 비교 평가한다.

4.2 실험 환경 설정

4.2.1 개발 프레임워크

본 실험에서 딥러닝 모델 개발은 텐서플로(TensorFlow)[5]를 기반으로 하였다. 텐서플로는 구글에서 개발한 딥러닝 라이브러리로 딥러닝 모델 개발을 위한 API(Application Programming Interface)와 사전학습된 모델(pre-trained model)을 제공하고 있다.

텐서플로 기반으로 학습한 모델을 모바일 기기에서 실행하기 위해 사용한 변환 도구는 퀄컴(Qualcomm)에서 제공하는 SNPE SDK(Snapdragon

Neural Processing Engine Software Development Kit)[27]를 사용하였다. SNPE SDK는 구글의 텐서플로(TensorFlow)[5], 텐서플로 라이트(TensorFlow Lite)[5], 마이크로소프트의 ONNX(Open Neural Network Exchange)와 페이스북의 카페2(Caffe2)[6]와 같은 다양한 프레임워크를 지원한다. 다양한 딥러닝 프레임워크로 개발된 모델 파일을 SDK에서 제공하는 변환 도구를 이용하여 DLC(Deep Learning Container) 파일로 변환하면 스마트폰에서 직접 실행하거나 모바일 애플리케이션에서 실행할 수 있다.

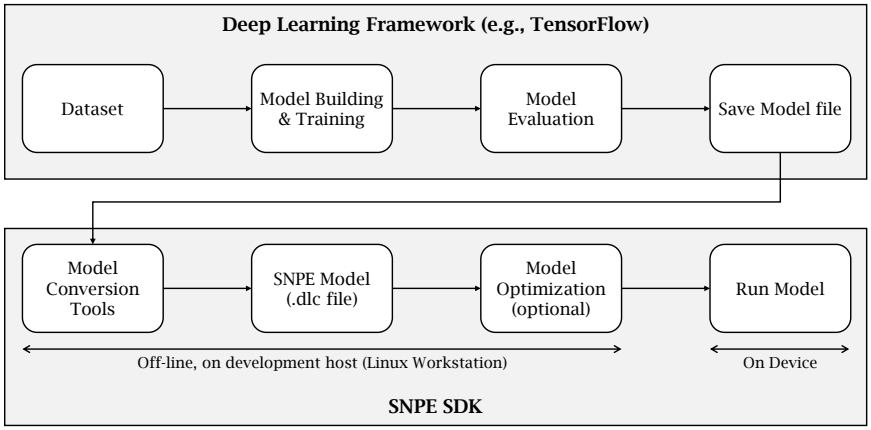


그림 16: 스냅드래곤 NPE SDK를 이용한 모델 개발 순서

스냅드래곤 NPE SDK는 개선 사항이 반영된 새로운 버전이 주기적으로 배포되고 있음으로 SDK를 실행하기 전에 각 버전의 시스템 요구사항 및 변경 사항 확인이 필요하다. 본 실험에서는 v1.49.0 버전을 사용하였으며, Ubuntu 18.04 리눅스 환경에서 아나콘다 플랫폼을 이용하여 가상환경을 구축하고 개발언어는 Python 3.6.10을 사용하였다. 아나콘다(Anaconda)는 Python 기반의 데이터 분석에 필요한 오픈소스를 제공하는 개발 플랫폼으로 가상환경 관리자와 패키지 관리자를 제공하여 개별 가상환경

을 구축하고, 패키지 간의 의존성을 확인하여 효율적인 패키지 설치를 할 수 있다. 실험 환경에 세부 정보는 표[1]과 같다.

표 1: 실험 환경 정보

세부 사양	
SNPE SDK	1.49.0
서버 운영 체제	Ubuntu 18.04
그래픽 카드	GeForce RTX 2080 Ti
병렬 컴퓨팅 플랫폼	CUDA 11.2
가상환경	Anaconda 4.10.1
개발 언어	Python 3.6.10
딥러닝 프레임워크	Tensorflow 1.15 Tensorflow 2.3.0

4.2.2 실험 데이터 세트

실험에 사용한 데이터 세트는 컴퓨터 비전 및 기계 학습 연구에 널리 사용되는 이미지 데이터 세트 중 하나인 CIFAR-10 (Canadian Institute For Advanced Research)[28]을 사용하였다.

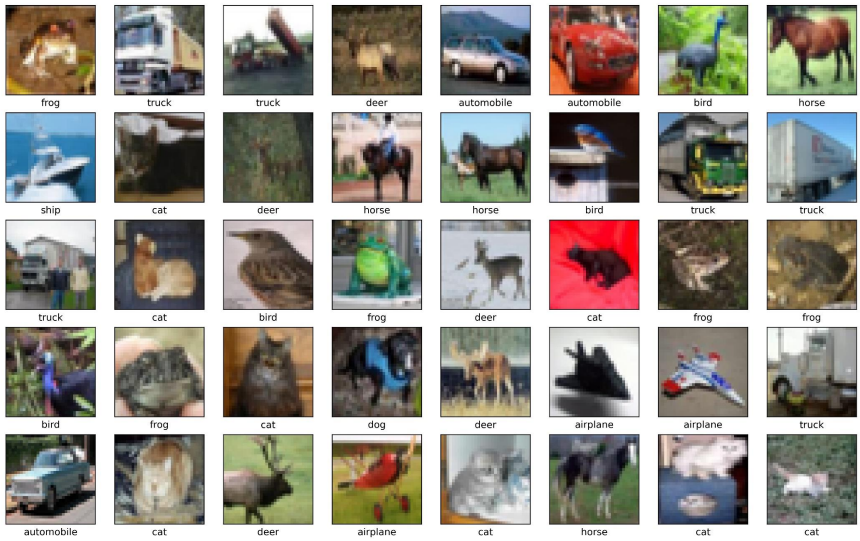


그림 17: CIFAR-10 학습 샘플 이미지

CIFAR-10은 32x32 픽셀의 60,000개 컬러 이미지로 구성된 데이터 세트이다. 10개의 클래스로 나눠져 있고, 각 클래스는 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배 및 트럭이 있으며 각 클래스마다 6,000개의 이미지로 구성되어 있다. 60,000개 이미지 중에서 50,000개는 학습용 데이터 세트로, 10,000개는 테스트 데이터 세트로 사용하였다.

4.2.3 실험 비교 모델 선정

본 연구에서 제안한 ELP-NAS 모델과 성능을 비교하기 위해 대표적으로 많이 사용하는 이미지 분류 모델 중에서 MobileNet V2[9], Inception_V3[1], Resnet V1 50[10], EfficientNet-lite0[29]를 선정하였다. 각 모델 별로 CIFAR-10 에 대한 정확도 평가 결과가 없어서 전이 학습을 진행하여 모델을 생성하고 평가를 진행하였다.

MobileNet V2, Inception_V3 그리고 Resnet V1 50 모델은 Tensorflow-Slim 라이브러리[30] 에서 제공하는 사전 학습된 모델을 기반으로 전이 학습을 하였다. 각 모델별로 동일하게 처음에는 CIFAR-10을 위해 새로 추가한 분류층만 10,000번 학습하고, 모델의 성능을 높이기 위해 기존 학습 결과를 기반으로 다시 10,000번 학습을 진행하였다. 그 중 MobileNet V2 학습을 위해 설정한 파라미터 정보는 표[2] 같다.

표 2: Mobilenet-V2 학습 파라미터 정보

Fine-Tune Step	only the new layers	All layers
dataset_name	cifar10	cifar10
model_name	mobilenet_v2	mobilenet_v2
checkpoint_exclude_scopes	MobilenetV2/Logits	
trainable_scopes	MobilenetV2/Logits	
max_number_of_steps	10000	10000
batch_size	32	32
learning_rate	0.01	0.001
optimizer	rmsprop	rmsprop
weight_decay	0.00004	0.00004

EfficientNet[29]은 이미지 분류 모델 중에서 기존보다 적은 파라미터수로 더욱 좋은 성능을 내서 State-Of-The-Art(SOTA)를 달성한 모델로 알려져 있으며, 모바일/IoT 기기를 위해 EfficientNet-Lite 모델을 제공하고 있다. EfficientNet-Lite 의 경우 입력 크기에 따라 Lite0(224x224)부터 Lite4(300x300) 까지 있는데, 본 실험에서는 입력 크기가 가장 작은 EfficientNet-Lite0를 대조군으로 선정하였다. EfficientNet-Lite0 모델 학습을 위해서 텐서플로 라이트 모델 메이커(TensorFlow Lite Model Maker) 라이브러리를 사용하였다. 모델 메이커 라이브러리를 이용하면 텐서플로 허브(TensorFlow Hub)에서 제공하는 모델의 기본 설정을 사용할 수 있으며, 사전의 훈련된 네트워크에서 전이 학습이 가능하다는 장점이 있다. 그리고 미세조정을 위해 텐서플로 허브에서 제공하는 네트워크와 새로 추가된 분류층을 함께 학습하였다. EfficientNet-Lite0 학습을 위한 설정한 파라미터 정보는 표[3]와 같다.

표 3: Efficientnet-Lite0 학습 파라미터 정보

Arguments	Value
model_spec	efficientnet_lite0
batch_size	64
epochs	15
train_whole_model	True
shuffle	False
use_augmentation	False
use_hub_library	True
do_train	True

4.2.4 실험 모바일 기기 선정

실험에 사용한 모바일 기기는 LG G8 ThinQ(LG G8 씽큐) 로 LG전자에서 2019년에 출시한 안드로이드 스마트폰이다. 주요 특징으로 전면 디스플레이의 노치 부분에 탑재된 ToF 센서를 이용하여 세계 최초로 정맥 인식 기능을 지원하며, ToF 센서로 물체의 거리를 미세하게 측정할 수 있기 때문에 3D 얼굴 인식 및 모션 인식 기능을 탑재했다. 스마트폰의 주요 사양은 아래와 같다.

표 4: LG G8 ThinQ 사양

구성	사양	
프로세서	퀄컴 스냅드래곤 855 모바일 플랫폼	
CPU	퀄컴 Kryo 485 프라임 싱글 코어 (2.84GHz) 퀄컴 Kryo 485 고성능 트리플 코어 (2.42GHz) 퀄컴 Kryo 485 저전력 쿼드 코어 (1.8GHz)	
GPU	퀄컴 Adreno 640 (585MHz)	
디스플레이	QHD+ (3120 x 1440)	
메모리	RAM	6G LPDDR4x
	내장메모리	128GB UFS2.1
카메라	전면카메라	일반카메라 800만 화소 Z 카메라 (ToF센서)
	후면카메라	일반카메라 1,200만 화소 망원카메라 1,200만 화소 초광각카메라 1,600만 화소
배터리	3,500mAh	
운영체제	안드로이드 9 Pie	

4.2.5 측정 방법

실험을 위해 딥러닝 모델의 지연 시간과 소비 전력을 측정해야 한다. 지연 시간 측정은 SNPE SDK에서 제공하는 snpe_bench(벤치마크 프로그램)을 이용하고, 소비 전력 측정은 몬순 솔루션(Monsoon Solution Inc) 사의 파워 모니터(Power Monitor)를 사용하였다.

파워 모니터는 파워툴(PowerTool) 소프트웨어를 이용하여 모바일 기기 및 임베디드 디바이스의 구동 전력을 모니터링하고 성능을 분석하여 설계를 최적화할 수 있다. 그림18 와 같이 실험에 사용한 스마트폰(LG G8 ThinQ)의 배터리를 분리하고 전원 단자와 파워 모니터를 연결하여 소비 전력을 측정하였다.

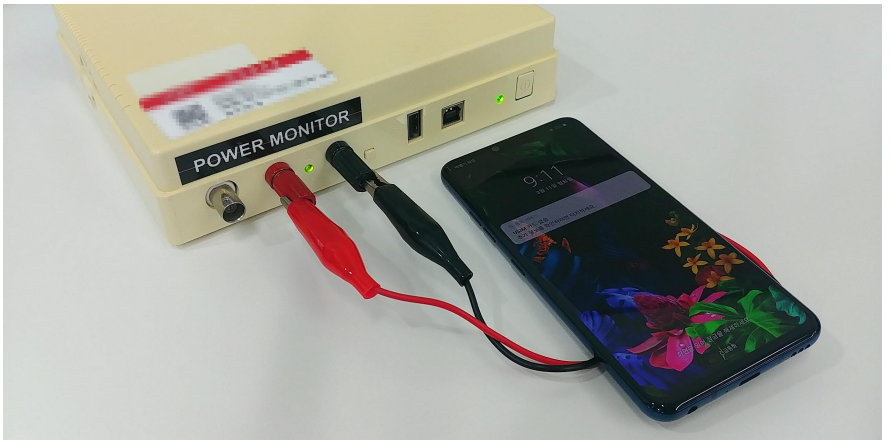


그림 18: 파워모니터를 이용한 소비 전력 측정 방법

실험에 적합한 전압값을 찾기 위해 입력 전압 대비 배터리 용량(State of Charge, SoC)을 확인하였다. 파워 모니터에서 설정 가능한 2.1 ∼ 4.5V 전압 구간별로 배터리 용량을 측정하였으며, WIFI, Bluetooth, GPS, NFC 와 같은 기능은 모두 끄고 측정하였다. 측정 결과는 그림19 과 같다.

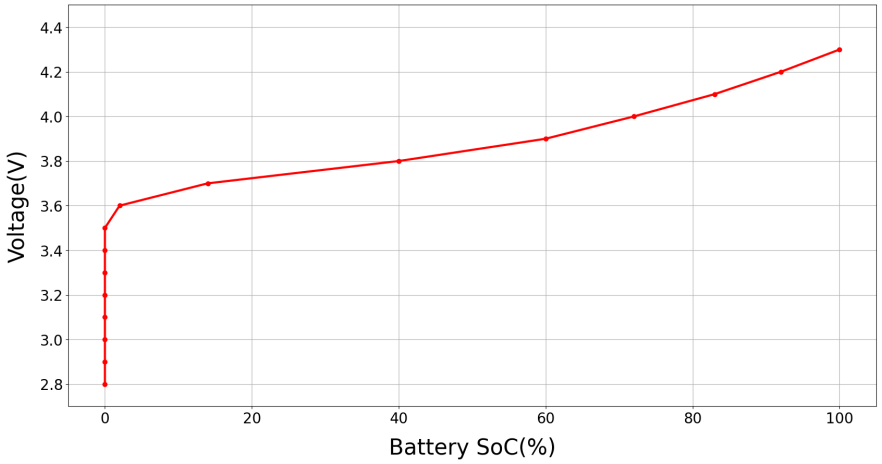


그림 19: 입력 전압 대비 배터리 용량

입력 전압에 따른 스마트폰 동작과 배터리 용량을 확인해보면, 입력 전압이 3.2V보다 낮은 구간에서는 전원이 켜지지 않거나 화면에 0%로 표시되며 충전기를 연결해달라는 아이콘이 표시되었다. 그 다음 3.5V보다 낮은 구간에는 스마트폰이 켜지다가 꺼지는 현상이 발생하였다. 입력 전압을 3.6V 로 설정했을때 정상적으로 전원이 켜졌지만, 배터리 용량이 2%로 표시되고 배터리 부족 알림과 함께 배터리 절전모드로 동작하여 화면 밝기가 자동으로 0%로 설정되었다. 그리고 입력 전압과 배터리 용량이 같이 증가하다가 4.3V에서 배터리 용량이 100%로 표시되었다. 실험 결과를 바탕으로 입력 전압은 배터리 용량이 약 70% 정도 되는 4V로 설정하여 실험을 진행하였다.

4.3 실험 결과 및 분석

본 연구에서 제안한 ELP-NAS의 소비 전력과 지연시간 목표값을 설정하기 위해 사전 실험을 진행한다. 그 결과를 기반으로 강성 제약 조건과 연성 제약 조건에 따라 실험을 진행하고 결과를 분석한다. 실험 결과 중 가장 성능이 좋은 아키텍처를 ELP-NAS 모델로 선정하고 베이스라인 모델의 아키텍처와 비교 분석한다.

4.3.1 목표값 설정 실험 결과

ELP-NAS의 소비 전력과 지연 시간 목표값을 찾기 위해 다목적 함수의 가중치 계수 값 α와 β를 0으로 설정하였다. 이 경우, 다목적 함수의 목적 값은 모델의 정확도가 되며 소비 전력과 지연 시간은 반영되지 않는다. 학습 결과로 탐색된 아키텍처는 그림20과 같다.

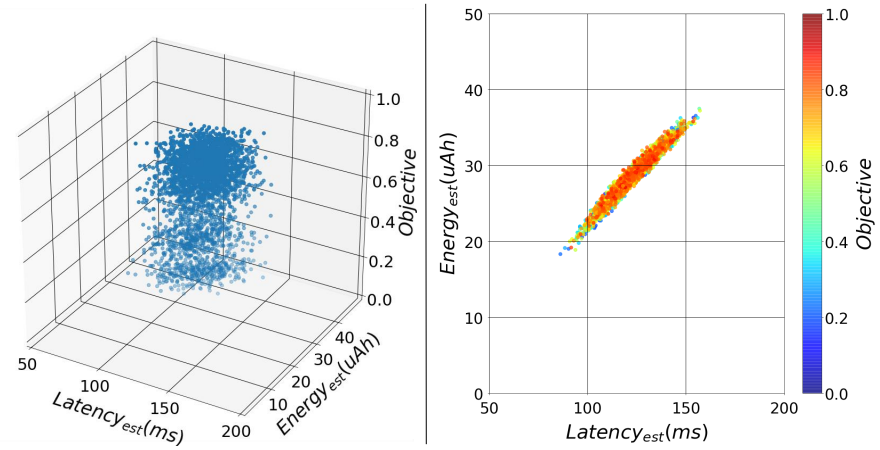


그림 20: 목표값 설정 실험 결과

탐색된 아키텍처를 ELP 알고리즘으로 예측한 소비 전력과 지연 시간 예측 값 그리고 목적 값에 대한 전체적인 분포를 3차원으로 표현하면 좌측 그래프와 같다. 이를 보다 시각적으로 분석하기 쉽게 우측 그래프와 같이 2차원으로 표현하였다. 탐색 된 아키텍처의 지연 시간 예측값(가로축)과 소비 전력 예측값(세로축) 기준으로 목적 값이 높을 수록 붉은 색으로 표현하였다. 3차원 분포를 2차원으로 표현하여 목적 값이 높은 아키텍처가 낮은 아키텍처의 상단에 표현된다.

실험 결과를 분석해보면, 붉은 색으로 표현된 목적 값이 높은 아키텍처들이 많이 탐색된 것을 확인할 수 있다. 그 중에서 예측한 지연 시간이 120 ~ 140ms 구간, 소비 전력은 25 ~ 35uAh 구간에서 성능이 좋은 아키텍처들이 탐색되는 것을 확인하였다. 실험 결과를 바탕으로 에너지 효율을 개션하기 위해 ELP-NAS의 목표치는 소비 전력은 20uAh와 10uAh, 지연 시간은 100ms와 50ms로 설정하여 강성 제약 조건과 연성 제약 조건에 따라 실험을 진행하였다.

4.3.2 강성 제약 조건 실험 결과

강성 제약 조건은 다목적 함수의 가중치 계수값을 $(\alpha = 0, \beta = -1)$로 설정하여 예측값이 목표치보다 큰 경우 목표값을 급격하게 감소시키고, 그렇지 않은 경우 모델의 정확도을 목적 값으로 사용한다.

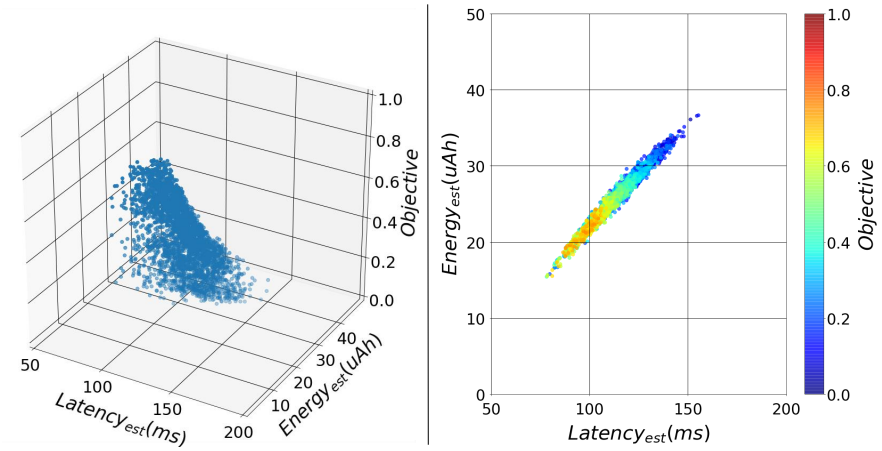


그림 21: 강성 제약 조건 실험1 결과

첫 번째 실험의 목표치는 지연 시간 100ms, 소비 전력 20uAh으로 설정하였으며 실험 결과는 그림21과 같다. 좌측 그래프를 보면 예측값이 목표치보다 큰 경우 목적 값이 급격하게 감소하고, 목표치와 같거나 작은 경우 모델의 정확도가 목적 값으로 적용되는 것을 확인하였다. 우측 그래프를 분석해보면 학습으로 탐색된 아키텍처들이 목표값으로 설정한 지연 시간 100ms, 소비 전력 20uAh 부근에서 목적 값이 높은 아키텍처들이 많이 탐색 되는 것을 볼 수 있는데, 목표치에 만족하는 모델이 탐색된 것을 확인할 수 있었다. 하지만 탐색된 아키텍처 목적 값의 평균 분포가 0.2 ~ 0.5 사이에 분포하고 최대 목적 값이 0.78로 탐색된 아키텍처의 성능이 낮은 것으로 분석되었다.

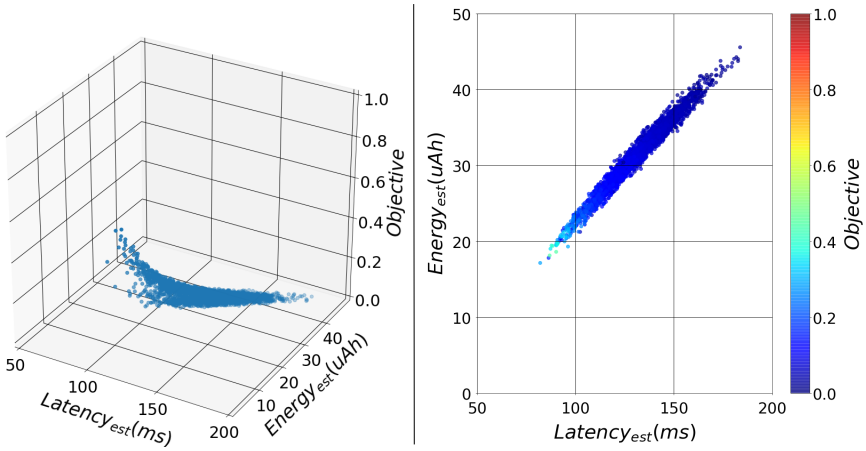


그림 22: 강성 제약 조건 실험2 결과

두 번째 조건 실험은 첫 번째 실험보다 더 낮은 조건으로 지연 시간 50ms, 소비 전력 10uAh 으로 설정하였으며 실험 결과는 그림22과 같다. 좌측 그래프를 보면 탐색된 아키텍처의 분포가 실험1 결과보다 더 낮게 분포되어 있는 것을 확인하였다. 우측 그래프를 분석해보면 탐색된 아키텍처들이 대부분 파란색으로 표시되었고, 목적 값이 평균적으로 0 ∼ 0.1 수준으로 매우 낮았다. 결과적으로 목표치에 만족하는 아키텍처를 탐색하지 못하여 정상적으로 학습되지 않은 것을 확인할 수 있었다.

4.3.3 연성 제약 조건 실험 결과

연성 제약 조건은 다목적 함수의 가중치 계수 값을 ($\alpha = -0.07, \beta = -0.07$)로 설정하여 예측값이 목표치보다 큰 경우 목적 값을 약간 감소시키고, 작은 경우 목적 값을 약간 증가시켜서 전체적인 값을 부드럽게 설정한다. 연성 제약 조건의 실험은 강성 제약 조건과 동일한 조건으로 진행하였다.

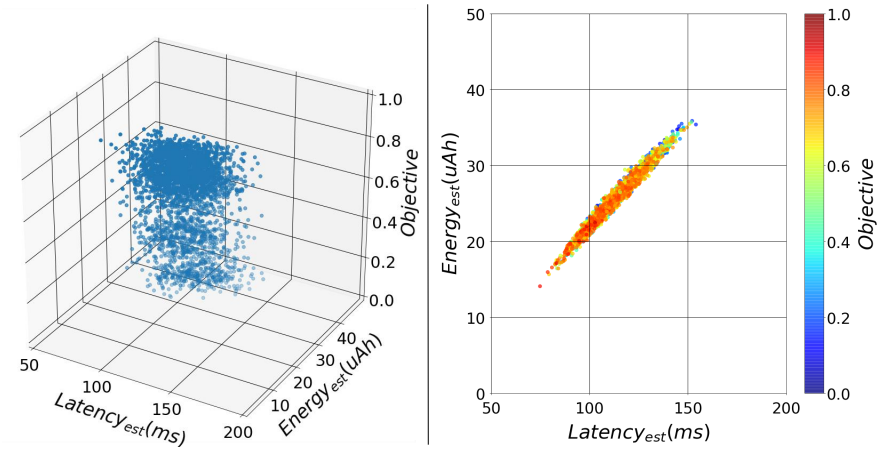


그림 23: 연성 제약 조건 실험1 결과

첫 번째 실험의 목표치는 지연 시간 100ms, 소비 전력 20uAh이며 실험 결과는 그림23과 같다. 좌측 그래프를 살펴보면 앞선 강성 제약 조건의 실험 결과와 달리 목표 값이 부드럽게 분포되어 있는 것을 볼 수 있다. 우측 그래프를 분석해보면 붉은 색으로 많이 표시되면서 탐색된 아키텍처의 목표 값이 높은 것을 확인하였다. 특히 목표치보다 더 높은 성능의 아키텍처들이 탐색되는 것을 확인할 수 있었다. 탐색된 아키텍처에 대한 목표 값의 평균 분포가 0.4 ~ 0.8 사이에 분포하여 강성 제약 조건보다 더 높은 구간에 분포하고 있고, 최대 목적 값이 0.93으로 높은 성능의 아키텍처가

탐색된 것을 확인하였다. 실험 결과를 보면 목표값 설정을 위한 사전 실험 결과와 유사해 보이지만, 아키텍처의 분포를 살펴보면 지연 시간 구간이 140ms에서 100ms으로, 소비 전력 구간이 30uAh 에서 20uAh으로 이동하여 저전력, 저지연 구간에서 높은 성능의 모델들이 분포된 것을 확인할 수 있었다.

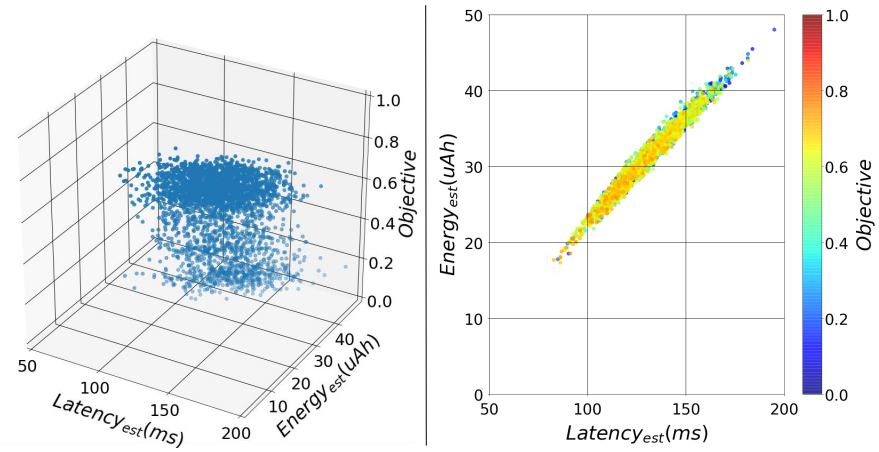
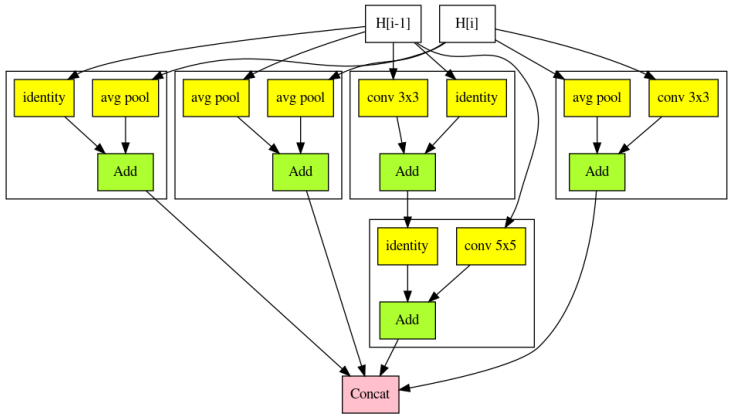


그림 24: 연성 제약 조건 실험2 결과

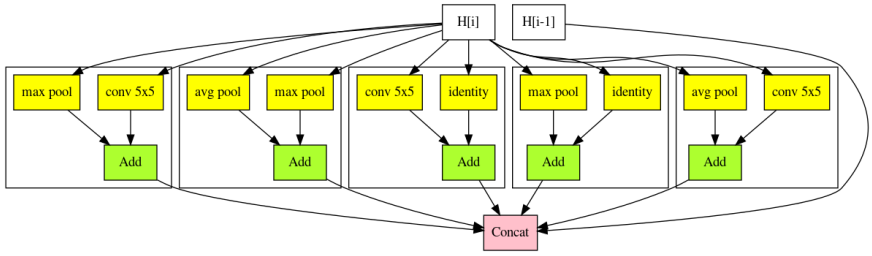
두 번째 조건 실험은 지연 시간 50ms, 소비 전력 10uAh으로 설정하였으며 실험 결과는 그림24과 같다. 좌측 그래프를 살펴보면 실험1 결과보다 탐색 된 아키텍처들이 더 넓고 낮게 분포되어 있는 것을 볼 수 있다. 좌측 그래프를 분석해보면 보다 확실하게 더 넓게 분포되어 있는 것을 확인할 수 있으며, 목표 값의 평균 분포가 0.3 ∼ 0.6 수준으로 탐색된 아키텍처들이 대부분 노란색으로 표시되었다. 그리고 강성 제약 조건의 실험2 결과와 동일하게 목표치에 만족하는 아키텍처를 탐색하지 못하여 정상적으로 학습되지 않는 것을 확인할 수 있었다.

4.3.4 실험 결과 분석

제약 조건과 목표치 설정에 따른 실험 결과를 분석해보면 연성 제약 조건에서 지연 시간 100ms, 소비 전력 20uAh 인 조건에서 성능이 가장 좋은 아키텍처가 탐색되었다. 해당 아키텍처를 최종 선정하여 ELP-NAS에 적용하였으며, 아키텍처 세부 구조는 그림25와 같다.

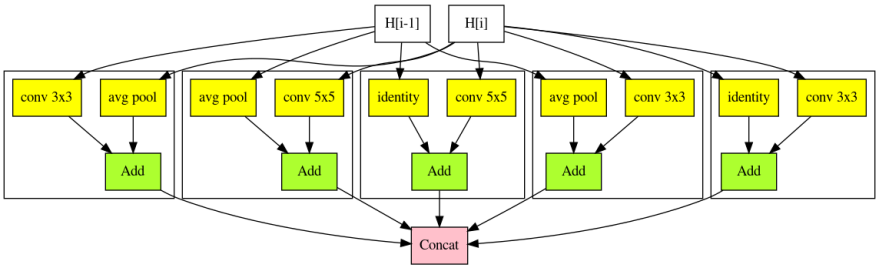


Normal Cell

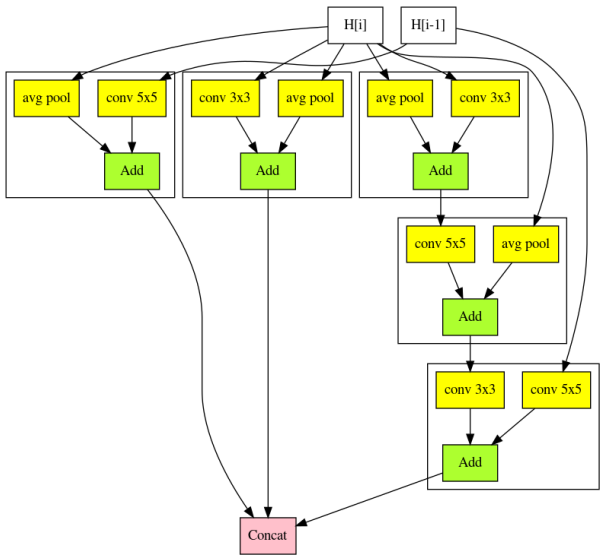


Reduction Cell

그림 25: ELP-NAS 아키텍처



Normal Cell



Reduction Cell

그림 26: 베이스라인 모델 아키텍처

표 5: 모델의 셀 단위 오퍼레이션 개수 비교

Model	Operation	Normal Cell (#Operations)	Reduction Cell (#Operations)
ELP-NAS (N=15)	sep conv 5x5	30	12
	sep conv 3x3	60	0
	avg pooling	60	4
	max pooling	0	6
	Identity	45	8
ENAS (N=15)	sep conv 5x5	60	6
	sep conv 3x3	90	6
	avg pooling	45	8
	max pooling	0	0
	Identity	30	0

ELP-NAS의 셀 구성은 그림25와 같고, 베이스라인 모델인 ENAS의 셀은 그림26와 같이 구성되어 있다. 이를 기반으로 각 모델별로 셀 단위의 오퍼레이션 개수를 비교하면 표5와 같다. 두 모델의 일반 셀의 개수는 15개이며 축소 셀은 2개이다. 그리고 아키텍처를 구성할 때 합성곱 연산은 2번씩 수행하는데, 이는 베이스라인 모델이 NASNet[15]을 기반으로 설계되어 동일하게 적용하였다. 전체 아키텍처 구성을 살펴보면 ELP-NAS는 ENAS 대비 합성곱 연산의 전체 개수는 약 37% 정도 감소하였고, 풀링 연산 개수는 약 32%, Identity 연산 개수는 77% 정도 증가하였다. 풀링 연산 개수가 증가하였지만, 상대적으로 소비 전력이 높은 합성곱 연산량이 감소하고 Identity 연산량이 증가하여 소모전류가 개선될 것으로 예상할 수 있다.

4.4 모델 성능 비교

ELP-NAS와 대조군으로 선정한 모델들의 지연 시간 대비 정확도와 소비 전력 대비 정확도는 그림 27과 같다.

본 연구에서 제안한 ELP-NAS 모델은 지연 시간은 약 50ms, 소비 전력은 약 9.5uAh 수준으로 베이스라인 모델인 ENAS보다 정확도는 약 0.35% 정도 소폭 감소하였지만 지연 시간과 소비 전력은 약 40% 개선된 것을 확인하였다.

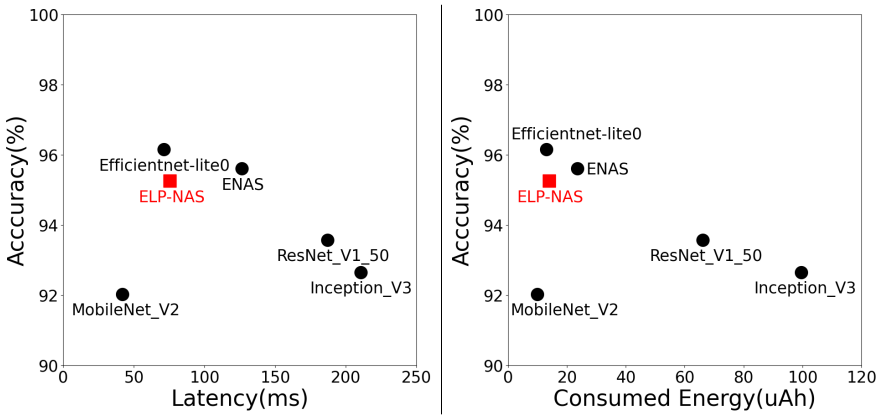


그림 27: 모델 성능 비교 결과

Model	Accuracy (%)	Latency (ms)	Energy (uAh)
ELP-NAS	**95.26**	**75.53**	**14.01**
ENAS	95.61	126.31	23.54
Efficientnet-lite0	96.16	71.41	13.02
Inception V3	92.66	210.60	99.54
MobileNet V2	92.03	41.88	9.88
ResNet V1 50	93.58	187.22	66.12

모델 성능을 비교해보면, 평가한 모델 중에서 EfficientNet-lite0 모델이 지연 시간과 소비 전력 대비 정확도에서 가장 좋은 성능을 보여주었다. ENAS는 ResNet_V1_50과 Inception_V3 모델보다 지연 시간이 짧고 에너지를 적게 소비하였다. MobileNet_V2 모델은 모델의 정확도는 가장 낮지만 지연 시간이 가장 짧고 가장 낮은 전력을 소비하였다.

ELP-NAS 모델의 셀 기반의 모델이기 때문에 추가적인 네트워크 탐색 없이 합성곱 셀의 개수를 변경하여 모델을 생성할 수 있다. 베이스 모델보다 에너지 효율을 높이기 위해 일반 셀 개수를 15개에서 9개와 3개로 변경하여 모델을 학습하고 성능 평가를 진행하였다. 셀 개수 변경에 따른 지연 시간 대비 정확도와 소비전력 대비 정확도는 그림 28과 같다.

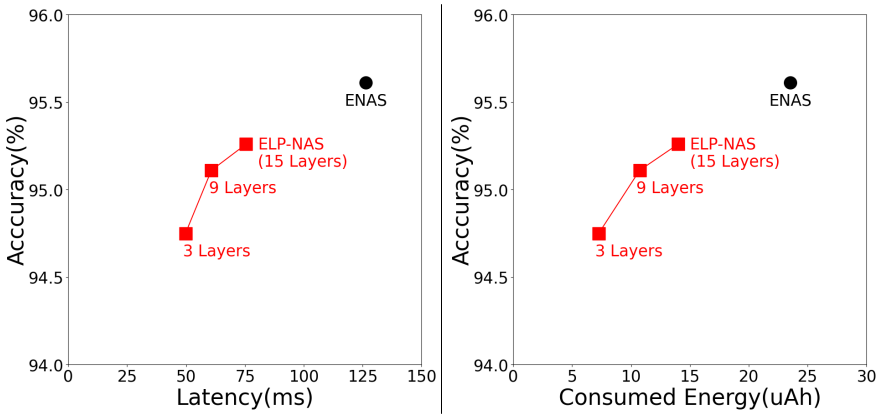


그림 28: 합성곱 셀 개수에 따른 성능 비교 결과

Model	Accuracy (%)	Latency (ms)	Energy (uAh)
ELP-NAS(15 Layers)	**95.26**	**75.53**	**14.01**
ELP-NAS(9 Layers)	95.11	60.79	10.73
ELP-NAS(3 Layers)	94.75	49.80	7.26

셀 개수를 변경한 모델 성능을 비교해보면 셀 개수가 9개인 경우 베이스 모델 대비 정확도는 0.15% 감소하지만, 소비 전력은 23.36% 개선되고, 지연 시간은 19.53% 정도 개선되었다. 또한 셀의 개수를 3개로 줄이면 정확도는 0.51% 감소하지만, 소비 전력은 48.15%, 지연 시간은 34.07% 개선되는 것을 확인하였다.

지연 시간과 소비 전력 대비 정확도에 대한 평가 지표에 추가하여 딥러닝 설계 효율성을 평가하는 방법인 학습 파라미터수 대비 정확도의 비율을 측정하는 Information Density와 학습 파라미터수와 연산수 대비 정확도를 측정하는 NetScore가 있다. 이 두가지 방법으로 ELP-NAS 및 비교 모델 그리고 ELP-NAS 모델의 셀 개수를 변경한 모델까지 포함하여 평가하면 그림29와 같다. Information Density와 NetScore는 값이 크면 클수록 좋은 모델로 평가할 수 있다.

Information Density 평가 결과를 분석해 보면, 평가한 모델 중에서 본 연구에서 제안한 ELP-NAS의 셀 개수가 3개인 모델이 가장 높게 평가되었다. Information Density는 학습 파라미터수 대비 정확도를 기준으로 평가하기 때문에 ELP-NAS의 셀 개수가 감소하면서 전체 네트워크의 학습 파라미터 수가 ELP-NAS 베이스 모델대비 77% 정도 감소하여 매우 높은 점수로 평가되었다. 그 다음주으로 셀 개수가 9개인 ELP-NAS 모델, MobileNet_V2 순으로 평가되었다.

다음으로 NetScore 평가 결과를 분석해 보면 Information Density와 유사하게 셀 개수를 3개인 ELP-NAS 모델이 가장 높은 점수로 평가되었다. NetScore는 학습 파라미터 수와 연산수(Multiplay-ACcumulate, MAC)을 기준으로 정확도를 평가하기 때문에 ELP-NAS에서 셀 3개인 모델이 가장 높게 평가되었다.

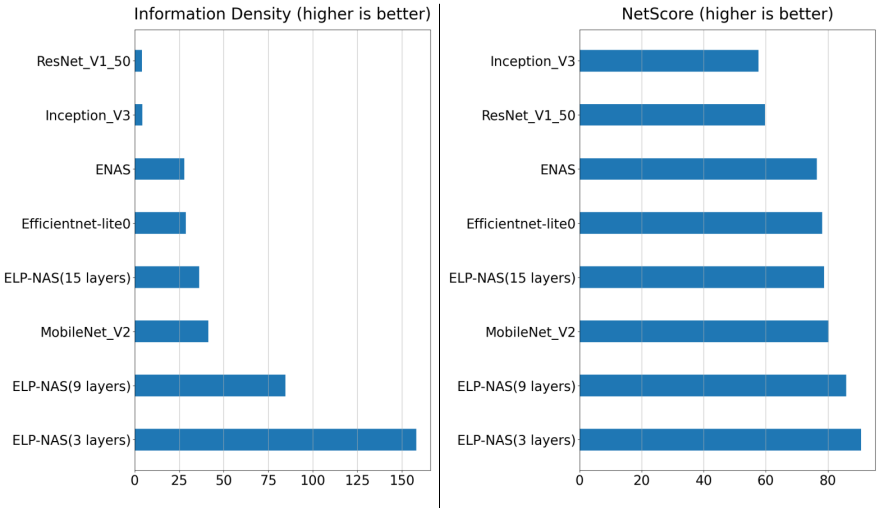


그림 29: Information Density 및 NetScore 평가 결과

Model	Information Density	NetScore
MobileNet V2	41.46	80.34
Efficientnet-lite0	28.60	78.21
ENAS	27.87	76.49
Inception V3	4.25	57.72
ResNet V1 50	3.98	59.72
ELP-NAS (15 Layers)	**36.30**	**78.81**
ELP-NAS (9 Layers)	**84.72**	**85.91**
ELP-NAS (3 Layers)	**158.23**	**90.76**

제 5 장

결론

본 연구에서 제안한 ELP-NAS 모델은 동일한 탐색 영역 조건의 베이스라인 모델 대비 정확도는 0.35% 감소하여 1%미만으로 아주 작지만, 소비 전력과 지연 시간은 약 40% 개선된 모델을 탐색한 것을 확인하였다. 또한 Information Density는 1.3배, NetScore는 1.03배 증가하였다. ELP-NAS는 에너지 효율 및 딥러닝 설계 효율성은 증가하였지만 정확도가 약간 감소하는 트레이드오프(TRADE-OFF) 관계를 확인하였다.

에너지 효율을 더 높이기 위해 기존 15개인 셀 개수를 변경하여 성능을 비교하였다. 셀 개수가 9개인 경우 소비 전력은 약 23%, 지연시간이 약 19% 정도, 3개인 경우 소비 전력은 약 48%, 지연시간이 약 34% 정도 개선되는 것을 확인하였다. 그리고 Information Density와 NetScore는 9개인 경우 2.3배와 1.1배, 3개인 경우 4.4배와 1.2배 증가하였다. 이런 실험 결과를 통해 ELP-NAS는 베이스라인 모델과 비교하여 보다 모바일기기에 최적화되고 에너지 효율적으로 탐색된 모델인 것을 확인하였다.

5.1 고찰

본 연구는 온 디바이스 AI 기술을 스마트폰과 같은 제품에 적용할 때, 에너지 효율적인 딥러닝 모델을 설계하기 위해 시작하였다. 모델 설계 문제를 딥러닝으로 해결하기 위해 신경망 아키텍처 탐색 방법을 적용하였고, 이를 기반으로 딥러닝 모델의 에너지를 예측하고 평가하는 방법에

대해 연구하였다. ELP-NAS의 목표치를 설정을 위해 베이스라인 모델에서 탐색 된 샘플 모델의 분포를 살펴보면 소비 전력과 지연 시간의 분포가 군집화되어 있고, 샘플 모델들의 지연 시간과 소비 전력의 분포가 선형의 구조를 보이고 있는 것을 알 수 있었다. 하지만 일반적인 선형 분포가 아닌 타원형으로 분포되어 동일한 지연 시간으로 예측된 샘플 모델들이 소비전력에 서로 차이가 있을 수 있고, 반대로 동일한 소비 전력으로 예측된 샘플 모델들이 지연 시간에 차이가 있다는 것을 확인할 수 있었다. 그래서 소비 전력과 지연 시간 중 하나의 조건만 고려하기 보다는 모두 고려하면 더욱 최적화된 모델을 학습할 수 있다는 것을 확인하였다. 또한 실험 결과를 통해 정확도와 에너지 효율이 서로 트레이드오프 관계를 확인하였지만, 정확도 감소 대비 에너지 효율성이 더 높게 평가 되었다. 그 이유를 살펴보면 본 실험에서 사용한 CIFAR-10 데이터 세트의 이미지 크기가 작고, 분류하고자 하는 클래스가 10개 정도로 실험군과 대조군으로 사용한 모델들의 성능 차이 보다 소비전력에 차이가 있는 것으로 보인다. ImageNet은 이미지가 1,000만개가 넘고 1,000개의 클래스로 구성되어 있다. CIFAR-10 대신 ImageNet과 같은 대규모 데이터셋을 적용한다면 모델의 정확도와 에너지 효율의 상관관계가 더 뚜렷하게 보일 것으로 기대된다.

5.2 연구 한계점

본 연구에서 제안한 ELP-NAS 모델은 소비 전력과 지연 시간의 목표 값을 설정하기 위한 실험이 필요하며, 실험 결과를 바탕으로 적절한 소비 전력과 지연 시간의 목표치를 설정해야 보다 성능이 좋은 모델을 탐색 할 수 있었다. 그 이유는 합성곱 셀의 개수를 15개로 고정하여 탐색 영역을 설계하였기 때문에 소비 전력과 지연 시간의 목표 값을 너무 작게 설정

하면 정상적으로 학습되지 않거나 목표한 모델을 탐색하지 못하는 것을 확인하였다. 특정 플랫폼을 기반으로 오퍼레이션들의 소비 전력과 지연 시간을 측정하였기 때문에, 플랫폼이 변경되거나 오퍼레이션이 변경 또는 추가되면 각 오퍼레이션의 측정 모델을 만들어서 소비 전력과 지연 시간을 측정하는 반복적인 작업이 필요하다. 본 연구의 베이스라인 모델인 ENAS가 NASNet[15]을 기반으로 한 모델이여서 마이크로 탐색 영역 기반으로 합성곱 셀을 구성할 때 합성곱 연산을 2번씩 수행한다. 그로 인해 합성곱 연산에서 소비 전력과 지연 시간이 2배씩 발생하게 되는데 이 조건은 베이스라인 아키텍처와 성능 평가를 위해 동일하게 설정하였다. 그리고 탐색 영역으로 설정한 오퍼레이션 외에 추가된 오퍼레이션 연산들은 바이어스로 추가하여 연산하였다.

5.3 향후 과제

본 연구에서는 스마트폰과 같은 모바일 기기의 에너지 효율성을 고려하여 딥러닝 모델을 신경망 아키텍처 탐색을 통해 더욱 성능이 좋은 딥러닝 모델을 탐색하고 학습이 가능하다는 것을 확인하였다. 향후 모바일 기기 뿐만 아니라 IoT 센서와 같이 더욱 성능이 제약된 하드웨어로 확장하여 본 연구 방법을 적용하면, 마이크로컨트롤러 기반에 매우 작은 용량의 메모리에서 실행되는 환경에서 보다 효율적인 딥러닝 모델 설계가 가능할 것으로 보인다. 또한 본 연구에서는 신경망 아키텍처 탐색과 관련해서 ENAS를 베이스라인 모델로 선정하였지만, 보다 학습시간이 짧고 성능이 좋은 탐색 영역과 전략을 가진 모델을 기반으로 소비 전력과 지연 시간을 고려하여 학습한다면 보다 성능이 개선된 모델을 탐색하는 방법에 대해 연구할 계획이다.

참고 문헌

[1] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[2] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, and L. Van Gool, "Ai benchmark: All about deep learning on smartphones in 2019," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 3617–3635, IEEE, 2019.

[3] E. Cai, D.-C. Juan, D. Stamoulis, and D. Marculescu, "Neuralpower: Predict and deploy energy-efficient convolutional neural networks," in *Asian Conference on Machine Learning*, pp. 622–637, PMLR, 2017.

[4] C. F. Rodrigues, G. Riley, and M. Luján, "Synergy: An energy measurement and prediction framework for convolutional neural networks on jetson tx1," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pp. 375–382, The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018.

[5] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678, 2014.

[7] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[8] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, 2019.

[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[10] 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*, pp. 770–778, 2016.

[11] Y. Moon, I. Shin, Y. Lee, and O. Min, "Recent research & development trends in automated machine learning," *Electronics and Telecommunications Trends*, vol. 34, no. 4, pp. 32–42, 2019.

[12] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[13] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[14] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in

Proceedings of the genetic and evolutionary computation conference, pp. 497–504, 2017.

[15] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

[16] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[17] Z. Zhong, J. Yan, and C.-L. Liu, "Practical network blocks design with q-learning," *arXiv preprint arXiv:1708.05552*, vol. 6, 2017.

[18] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, *et al.*, "Evolving deep neural networks," in *Artificial intelligence in the age of neural networks and brain computing*, pp. 293–312, Elsevier, 2019.

[19] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*, pp. 2902–2911, PMLR, 2017.

[20] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[21] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[23] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International Conference on Machine Learning*, pp. 4095–4104, PMLR, 2018.

[24] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018.

[25] K. Deb, "Multi-objective optimization," in *Search methodologies*, pp. 403–449, Springer, 2014.

[26] A. Wong, "Netscore: towards universal metrics for large-scale performance analysis of deep neural networks for practical on-device edge usage," in *International Conference on Image Analysis and Recognition*, pp. 15–26, Springer, 2019.

[27] "Snapdragon neural processing engine sdk reference guide." `https://developer.qualcomm.com/docs/snpe`.

[28] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[29] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, pp. 6105–6114, PMLR, 2019.

[30] Sergio Guadarrama, Nathan Silberman, "TensorFlow-Slim: A lightweight library for defining, training and evaluating complex models in tensorflow." `https://github.com/google-research/tf-slim`, 2016. [Online; accessed 29-June-2019].

Abstract

Neural Architecture Search considering energy efficiency of mobile device

Youngyun Kim
Graduate School of Practical Engineering
Seoul National University

The demand for on-device AI service-based image analysis technology that can be used in embedded devices such as mobile and IoT devices is increasing. On-device AI has advantages such as low latency and enhanced security, but AI performance is dependent on hardware performance and consumes excessive power by requiring a lot of computing resources such as processor and memory for AI operations. For this reason, there is a need to improve energy efficiency for on-device AI models.

In this study, we propose ELP-NAS as a method of constructing a deep learning model considering the energy efficiency of mobile devices. ELP-NAS trains deep learning models using neural network architecture search to design optimal architectures in automatic machine learning. By applying the algorithm to predict the end-to-end energy consumption and latency of the deep learning model, the predicted energy consumption and latency of

the discovered neural network architecture are used as a reward for reinforcement learning along with the accuracy of the model.

In the CIFAR-10 data set, the accuracy of the ELP-NAS was 95.26%, which is equivalent to the accuracy of the ENAS selected as the baseline, 95.61%, but it was confirmed that the power consumption and execution time were improved by about 40% compared to the baseline model.

Keywords : Neural Architecture Search, Energy Efficiency, Deep learning
Student Number : 2020-27656