Journal of Korea Robotics Society (2023) 18(1):088-092
https://doi.org/10.7746/jkros.2023.18.1.088 ISSN: 1975-6291 / eISSN: 2287-3961

로봇 임베디드 시스템에서 리튬이온 배터리 잔량 추정을 위한 신경망 프루닝 최적화 기법

Optimized Network Pruning Method for Li-ion Batteries State-of-charge Estimation on Robot Embedded System

박 동 현[1] · 장 희 덕[1] · 장 동 의†
Dong Hyun Park[1], Hee-deok Jang[1], Dong Eui Chang†

Abstract: Lithium-ion batteries are actively used in various industrial sites such as field robots, drones, and electric vehicles due to their high energy efficiency, light weight, long life span, and low self-discharge rate. When using a lithium-ion battery in a field, it is important to accurately estimate the SoC (State of Charge) of batteries to prevent damage. In recent years, SoC estimation using data-based artificial neural networks has been in the spotlight, but it has been difficult to deploy in the embedded board environment at the actual site because the computation is heavy and complex. To solve this problem, neural network lightening technologies such as network pruning have recently attracted attention. When pruning a neural network, the performance varies depending on which layer and how much pruning is performed. In this paper, we introduce an optimized pruning technique by improving the existing pruning method, and perform a comparative experiment to analyze the results.

Keywords: Battery Estimation, Network Pruning, Pruning Optimization

1. 연구 배경

리튬이온 배터리는 높은 에너지 효율과 가벼운 무게, 긴 수명, 낮은 자가 방전율 덕분에 필드 로봇, 드론, 전기자동차 등 다양한 산업현장에서 활발하게 사용되고 있다[1]. 이렇게 많은 장점이 있는 리튬이온 배터리도 사용자의 잘못된 판단이나 잘못된 사용법에 따라 배터리 성능 혹은 수명에 치명적인 손상을 일으킬 수 있다. 따라서 리튬이온 배터리에 배터리 관리 시스템(BMS: Battery Management System)을 배치하여 이러한 문제를 미연에 방지하고 주기적으로 배터리 고장 진단 및 제어할 필요가 있다.

BMS의 역할은 셀 모니터링, 셀 밸런싱, 배터리 상태 추정, 배터리 제어 등이 있다[1]. 여기서 배터리 상태 추정은 배터리 잔량(SoC : State of Charge), 배터리 퇴화도(SoH: State of Health) 등을 추정하는데 특히 SoC 추정은 배터리의 고장 진단 및 셀 밸런싱 제어 수행 등에 있어 매우 중요한 기능을 한다. 정확하고 정교한 SoC 추정은 필드 로봇, 드론 등에게 올바른 배터리 정보를 주고 배터리의 과충전 및 과방전을 방지하여 배터리 손상을 막을 수 있게 해준다[2]. 배터리의 정확한 SoC 추정을 위해 직접 측정 기법, 쿨롱 카운팅, 모델 기반 기법, 데이터 기반 기법 등 많은 연구가 진행되어 왔다[3]. 배터리 SoC 추정 계산은 기본적으로 비선형적이기 때문에 정확한 추정을 위해 최근에는 인공지능 신경망을 활용한 데이터 기반 기법이 각광 받고 있다[4]. 하지만 현장 로봇에 탑재된 저가형 임베디드 보드 환경의 경우에는 무겁고 복잡한 인공 신경망을 동작시키기엔 무리가 있다[1].

이러한 문제를 해결하기 위하여 최근에 클라우드 기반의 BMS가 연구되고 있다[1,5]. 클라우드 기반 BMS란, 배터리 정보 수집과 배터리 제어를 수행하는 slave와 정보를 계산하고 판단하여 slave에게 판단명령을 주는 master로 나누어 slave는 배터리 끝단에, master는 중앙 클라우드 서버에 배치하고 서로 통신하여 배터리를 관리하는 시스템이다. 이는 복잡한 인공 신경망으로도 배터리 상태를 추정할 수 있게 하고 다량의 배

Received : Oct. 28. 2022; Revised : Nov. 14. 2022; Accepted : Nov. 18. 2022
※ This work has been supported by the Unmanned Swarm CPS Research Laboratory Program of Defense Acquisition Program Administration and Agency for Defense Development (UD220005VD)
1. M.S. Student, Electrical Engineering, KAIST, Daejeon, Korea (pdh9284, jhd6844@kaist.ac.kr)
† Professor, Corresponding author: Electrical Engineering, KAIST, Daejeon, Korea (dechang@kaist.ac.kr)

터리 정보를 클라우드 서버에 저장하여 기록 보관을 가능하게 하였다.

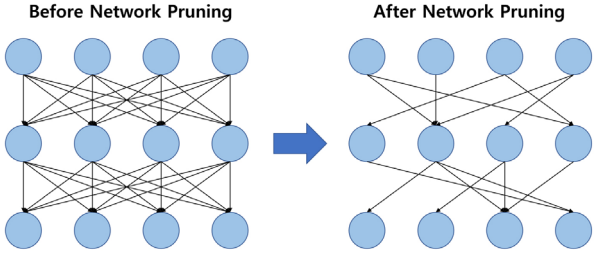
하지만 [1]의 저자는 클라우드 기반 BMS에서는 아직도 해결해야 할 문제들이 있다고 지적한다. 그중 하나는 바로 통신 비용이다. 산업현장에서 하나의 클라우드 서버에서 다량의 정교한 지역 정보를 수집하려면 고가의 통신 모뎀 비 혹은 넓은 통신 대역폭을 요구한다. 통신 비용을 아끼기 위해서는 수집 데이터의 퀄리티를 낮추어 데이터를 로깅 하는 방법이 있으나 이렇게 된다면 정교한 SoC를 추정하지 못하게 된다.

이러한 이유로 임베디드 엣지 단에서 센서를 통해 정교한 배터리 정보 수집과 동시에 신경망으로 SoC를 계산하여 서버로 보낼 수 있는 BMS에 대한 필요성이 현장에서 논의되고 있다. 임베디드 엣지 단에서 크고 복잡한 신경망을 배치하기 위해 신경망을 경량화하는 연구가 논의되고 있다. 그 방법 중 네트워크 프루닝[6]은 기존에 이미 학습된 네트워크의 필요 없는 웨이트를 [Fig. 1]과 같이 제거하여 모델을 경량화하는 기술이다. 신경망 경량화를 수행하면 모델이 가벼워지는 대신 성능이 저하 될 수 있다. 하지만 네트워크 프루닝의 경우 어떤 웨이트를 어떻게 제거하는지에 따라 신경망 성능 저하를 최소화할 수 있다[7]. 이 논문에서는 임베디드 엣지단에 SoC 추정을 위한 인공 신경망을 배치할 수 있도록 현재 개발된 SoC 추정 모델에 우리가 제안하는 최적화된 프루닝 기법을 수행하고 그 결과를 분석한다.

본 논문의 구성은 다음과 같다 2장에서는 네트워크 프루닝에 대한 관련 연구를 소개한다. 3장에서는 우리가 제안하는 프루닝 최적화 기법 알고리즘을 소개한다. 4장에서는 기존의 프루닝 기법과 우리가 제안한 프루닝 기법의 결과를 실험으로 확인하고 비교 분석해본다. 마지막 4장에서는 결론과 앞으로 진행할 연구를 소개한다.

2. 관련 연구

네트워크 프루닝은 잘 학습된 딥러닝 모델에서 쓸모없는



[Fig. 1] Schematic diagrams of networks before (left) and after (right) pruning. By pruning useless weights, the size of the neural networks can be reduced

웨이트를 제거하여 네트워크를 경량화하는 방법이다[6]. 웨이트를 제거하는 방법은 다양하지만 가장 널리 알려진 방식으로는 크기 기반의 방식이 있다. 크기 기반 방식이란 사전에 학습된 네트워크의 웨이트값 중 크기가 작은 순대로 제거하는 방식이다. 엔지니어가 원하는 sparsity를 선정하고 그 sparsity에 해당하는 만큼 작은 크기순으로 웨이트를 제거한다[6,7].

$$Sparsity = \frac{Number\ of\ nonezero\ parameters}{Number\ of\ total\ parameters} \quad (1)$$

[Algorithm 1]은 [8]에서 기존의 크기 기반 프루닝 기법을 좀 더 발전시켜 제안한 프루닝 기법 알고리즘이다. [8]에서는 프루닝을 수행할 때 사전에 학습 완료된 모델을 바로 목표 sparsity까지 프루닝을 하지 않고 학습 완료된 모델에서 추가로 학습과 프루닝을 반복적으로 수행한다. 추가학습 단계에서는 학습을 진행하다가 정해진 시작 스텝부터 학습과 프루닝을 반복한다. 이때 각 스텝의 프루닝 단계에서는 [Algorithm 1]의 3번째 줄과 같이 낮은 초기 sparsity에서 목표 sparsity까지 점진적으로 프루닝을 진행한다. 이때 s_i는 초기 sparsity s_f는 끝 sparsity, n은 현재 스텝, n_{beign}는 시작 스텝 n_{end}는 학습의 마지막 스텝을 의미한다. 이 스케쥴대로 낮은 sparsity로 프루닝시키고 한 스텝 학습하여 성능을 회복시키고, sparsity를 조금 더 올려서 프루닝시키고 한 스텝 학습하여 성능을 회복시키고를 반복하면서 목표 sparsity에 도달하면 프루닝으로 인한 성능 저하를 줄이며 목표 sparsity에 도달할 수 있다.

한편 네트워크 프루닝의 범위에 따라 글로벌 프루닝과 로컬 프루닝으로 나뉠 수가 있다[7]. 글로벌 프루닝은 프루닝을

[Algorithm 1] Improved magnitude-based pruning algorithm

Input:
$\mathbf{W} \in R^{M\times N}$: Weight matrix of dimension $M \times N$,
$\mathbf{Msk} \in R^{M\times N}$:weight mask of dimension $M \times N$,
S_i: Initial sparsity ratio
S_f: Final sparsity ratio
n_{begin}: Initial step
n_{end}: end step

1: **while** step $n < n_{end}$ **do**
2: **if** $n > n_{begin}$ **then**
3: $S_t = S_f + (S_i - S_f)(1 - \frac{n-n_{begin}}{n_{end}-n_{begin}})^3$
4: $th = (S_t \times M \times N)$th largest element in $|\mathbf{W}|$
5: **for** $1 \le m \le M, 1 \le n \le N$ **do**
6: **if** $|W_{m,n}| < th$ **then**
7: $Msk_{m,n}$=0
8: **else**
9: $Msk_{m,n}$=1
10: **end if**
11: **end for**
12: $\mathbf{W} \leftarrow \mathbf{W} \otimes \mathbf{Msk}$ # $\otimes$: Matrix element multiplication
13: **end if**
14: Train $\mathbf{W}$
15: **end while**
16: **Return** $\mathbf{W}$

할 때 전체 네트워크의 웨이트를 기준으로 목표 sparsity로 프루닝 하고 로컬 프루닝은 각 레이어 마다 목표 sparsity를 다르게 하여 좀 더 세분화 된 프루닝을 수행한다. [9]에서는 CNN 기반 배터리 용량 추정 모델을 로컬 프루닝을 통해 경량화하였다. 이때 어떤 레이어를 얼마나 프루닝을 할지를 최적화 기법을 사용하지 않고 상대적으로 파라미터 수가 많은 레이어를 로컬 프루닝하여 모델 성능 저하를 개선하였다.

프루닝을 실제로 구현할 때는 제거하고자 하는 웨이트값을 0으로 만들어서 구현한다. 이렇게 되면 기존의 dense 네트워크에서 pruning 한 sparse 네트워크가 된다[6,7]. 배치하고자 하는 네트워크가 sparse 하다면 하드웨어 아키텍쳐에 따라 크게 연산속도와 모델 사이즈에서 큰 이득을 볼 수 있다. sparse 네트워크를 효율적으로 계산하는 로직을 하드웨어에 연산장치에 적용하여 연산속도에서 큰 효과를 볼 수 있고[10,11] Huffman coding과 같은 압축 기술로 sparse 네트워크를 효율적으로 저장하여 모델 사이즈에서도 큰 효과를 볼 수 있다[6].

3. 알고리즘

3장에서는 [Algorithm 1]을 사용하여 로컬 단위로 프루닝을 시행할 때 최적의 레이어 sparsity를 찾는 우리의 알고리즘을 제안한다. 인공 신경망을 프루닝할 때 프루닝 민감도는 각 레이어마다 상이하다[7]. 즉, 각 레이어에 똑같이 목표 sparsity를 높였음에도 불구하고 어떤 레이어는 성능에 큰 타격을 주는 반면 어떤 레이어는 큰 성능 저하를 유발하지 않는다. 따라서 프루닝 하고자 하는 인공 신경망 모델을 목표 전체 sparsity에 따라 각 레이어의 최적의 sparsity를 차등하게 두면 모델 경량화에 의한 성능 저하를 최소화할 수 있다.

[Algorithm 2]는 최적의 프루닝 레이어 sparsity를 찾기 위해서 우리는 제약 조건을 반영한 격자 탐색 기법을 활용하였다.

[Algorithm 2] Optimized layer sparsity searching algorithm

Input: T_s : Target sparsity(%) × Total Parameter

$L = [l_1\ l_2 \cdots l_n]$ *where* l_n is n_{th} layer paramter
$D = \{[s_1\ s_2 \cdots s_n] \mid s_n$ is Integer in [0,100]$\}$
F_s : Final Sparsity
K : Number of fold
$f(x,i)$: MAE of pruning model with layer sparsity x & i_{th} Fold

Let *Search space* $= \{x \mid x \in D,\ \frac{|x \cdot L - T_s|}{100} \le \varepsilon\}$
$MAE_{min} = 1$
for x **in** *Search space* **do**
  $MAE = 0$
  for i **in** K **do**
    $MAE = MAE + \frac{f(x,i)}{K}$
  end for
  if $MAE < MAE_{min}$ **then**
    $F_S = x$
  end if
end for
Return F_S

먼저 모델의 전체 목표 sparsity를 정하고 오차범위 내에서 그 sparsity를 보존할 수 있는 레이어 sparsity 조합을 찾는다. 본 논문에서는 허용 오차 범위(ε)를 0.1로 두었다. 이 조합으로 구성된 집합을 제약 조건이 반영된 탐색영역으로 정의한다. 탐색영역의 sparsity 조합 중 가장 좋은 조합은 사전에 정의된 손실함수 값이 가장 작은 조합으로 선정한다. 본 논문에서는 손실함수는 [Algorithm 2]의 5번째 줄에서 10번째 줄까지와 같이 k-fold cross validation 방식으로 구한 식 (2)의 MAE (Mean Absolute Error)의 평균값으로 선정하였다. 이 알고리즘대로 목표 전체 sparsity에 대한 최적의 레이어 sparsitiy 조합을 찾고 그 sparsity로 로컬 프루닝을 똑같이 수행하면 최종 모델이 완성된다.

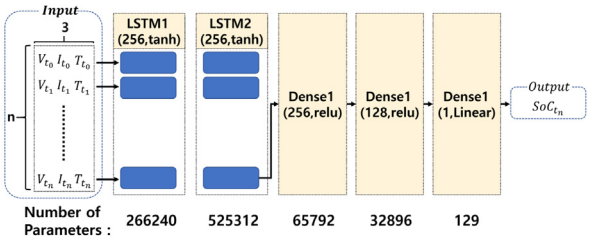
$$MAE = \frac{1}{T}\sum_{t=1}^{T} \mid SoC_{real,t} - SoC_{estimation,t} \mid \qquad (2)$$

4. 실험 및 결과

4.1 데이터 셋과 베이스라인 모델

이 논문에서는 LG 1865HG2 Li-ion Battery Dataset[12]을 이용하여 실험하였다. 이 데이터 셋에서는 6개의 다른 온도에서 LG 1865HG2 Li-ion Battery의 충, 방전 주기 동안 0.1초 간격으로 수집된 배터리 전압, 전류, 온도 시계열 데이터와 그에 따른 SoC 실측값으로 구성되어있다.

베이스라인 모델은 [2]를 참고하여 모델을 제작하였다. 베이스라인 모델은 [Fig. 2]와 같이 2개의 LSTM 레이어와 3개의 dense 레이어로 구성되어있고 Adam optimizer를 0.00001의 learning rate로 사용하였다. 배치 사이즈는 32로 하였다. 첫 번째 LSTM 레이어는 return sequence를 허용하고 두 번째 LSTM layer는 return sequence를 허용하지 않았다 손실함수는 huber를 사용하였다. 프루닝 과정에서 그레디언트 소실 혹은 폭주 문제를 막기 위해 LSTM의 activation function은 tanh로 수정하였고 나머지 layer는 ReLu함수를 그대로 사용하였다. 실험에서는 데이터 셋에서 0°C, 10°C, 25°C 데이터셋을 이용하였다.



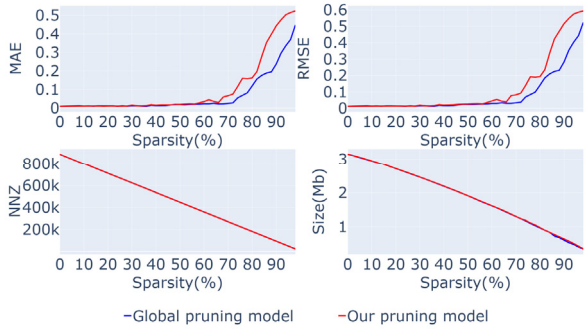
[Fig. 2] Baseline model architecture for Li-ion batteries state of charge estimation

4.2 실험 결과

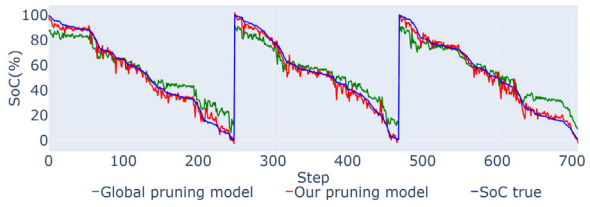
우리의 알고리즘 성능을 검증하기 위해 베이스라인 모델을 [Algorithm 1]의 방법으로 글로벌 단위로 수행한 모델과 우리가 찾아낸 최적의 레이어 sparsity 조합으로 로컬 프루닝한 모델을 비교하는 실험을 구성하였다. 베이스라인에 제안한 알고리즘을 적용할 때 [Fig. 2] 마지막 레이어를 제외한 4개의 레이어를 기준으로 최적의 sparsity를 찾았다. 두 모델을 평가하기 위한 평가 지표로는 식 (2)의 MAE와 추가로 식 (3)의 RMSE (Root Mean Square Error)를 사용하였고 경량화 정도를 확인하기 위해 신경망 모델의 0이 아닌 파라미터의 수(NNZ: Number of Nonzero parameters), Huffman coding을 통해 압축 및 저장한 모델 사이즈를 조사하였다.

[Table 1] Result of our experiment

Target Sparsity	Final layer sparsity	MAE	RMSE	NNZ	Size (Mb)
30%		0.0160	0.0188	624668	2.464
	[31,33,17,0]	**0.0125**	**0.0150**	**624675**	**2.460**
50%		0.0202	0.0260	447551	1.912
	[51,53,31,32]	**0.0194**	**0.0236**	**447579**	**1.909**
70%		0.0673	0.0824	270432	1.303
	[72,76,32,34]	**0.0260**	**0.0345**	**270455**	**1.297**



[Fig. 3] Comparison between global pruning model and our pruning model. It can be seen that the pruning performance of our models is improved then global pruning model



[Fig. 4] Results graph of comparison between global pruning and our local pruning

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(SoC_{real,t} - SoC_{estimation,t})^2} \quad (3)$$

두 모델 모두 Initial sparsity 0%에서 목표 sparsity에 도달하게 하는 프루닝 스케쥴과 함께 시작 스텝 0으로 2 epoch 추가 학습시켜 결과를 확인하였다. [Table 1]은 실험에 사용한 전체 테스트 데이터셋에 대하여 두 모델의 결과를 나타내는 표이다. 각 Target Sparsity의 첫 번째 줄은 글로벌 프루닝 결과이고 두 번째 줄은 제안한 최적화 기법을 적용한 우리의 모델 결과이다. 표에서 Final layer sparsity는 최적화를 통해 찾은 각 레이어 sparsity를 의미한다. 두 모델 모두 동일한 목표 sparsity를 가지고 프루닝을 수행하여 비슷한 NNZ 와 모델 사이즈를 가졌지만 프루닝 과정에 의한 모델 성능 저하 정도는 기존 모델보다 우리 모델이 더 적은 것을 확인할 수 있다.

[Fig. 3]은 두 모델의 목표 sparsity에 따른 성능변화를 실험에서 사용한 전체 테스트 데이터셋에 대하여 그래프로 나타낸 것이다. 목표 sparsity가 올라감에 따라 두 모델 모두 NNZ와 모델사이즈가 감소하는 것을 확인할 수 있다. 모든 sparsity에서 우리 모델이 글로벌 프루닝 모델보다 좋은 성능을 보였다. 특히 sparsity가 특정 지점부터 급격하게 성능이 저하되는 것을 확인할 수 있는데 그 sparsity 지점이 우리 모델은 72%로, 글로벌 프루닝 모델의 60% 지점보다 더 좋은 성능을 보여주는 것을 확인할 수 있었다.

[Fig. 4]는 우리의 실험 결과 중 25°C 환경의 테스트 데이터셋을 70% sparsity로 추정한 결과이다. 글로벌 프루닝 모델의 추정값의 경우 실제 SoC 값에서 많이 벗어난 모습을 볼 수 있다. 하지만 우리 모델의 추정값은 실제 SoC 값의 추세에서 벗어나지 않는 것을 확인할 수 있다.

5. 결 론

본 논문에서는 필드 로봇, 드론 등 다양한 산업현장에서 사용되는 리튬 이온 배터리의 잔량 추정 신경망 모델을 경량화하는 방법에 대하여 다루었다. 모델을 경량화할 때 네트워크 프루닝 기법에 우리가 제안하는 최적화 기법을 더하여 수행하였고 결과적으로 기존의 프루닝 기법보다 더 좋은 성능을 확인하였다. 본 논문의 결과를 바탕으로 임베디드 기반 BMS 장치에서 경량화된 SoC 추정 모델을 배치했을 때 실제 성능을 비교하는 연구가 추후 이루어질 것이다.

References

[1] M.-K. Tran, S. Panchal, T. D. Khang, K. Panchal, R. Fraser, and M. Fowler, “Concept review of a cloud-based smart battery

management system for lithium-ion batteries: feasibility, logistics, and functionality," *Batteries*, vol. 8, no. 2, pp. 19, Feb., 2022, DOI: 10.3390/batteries8020019.

[2] K. L. Wong, M. Bosello, R. Tse, C. Falcomer, C. Rossi, and G. Pau, "Li-ion batteries state-of-charge estimation using deep lstm at various battery specifications and discharge cycles," *Conference on Information Technology for Social Good*, Rome, Italy, pp. 85-90, 2021, DOI: 10.1145/3462203.3475878.

[3] M. U. Ali, A. Zafar, S. H. Nengroo, S. Hussain, M. J. Alvi, and H. J. Kim, "Towards a smarter battery management system for electric vehicle applications: a critical review of lithium-ion battery state of charge estimation," *Energies*, vol. 12, no. 3, pp. 446, Jan., 2019, DOI: 10.3390/en12030446.

[4] X. Yan, G. Zhou, W. Wang, P. Zhou, and Z. He, "A Hybrid Data-Driven Method for State-of-Charge Estimation of Lithium-Ion Batteries," *IEEE Sensors Journal*, vol. 22, no. 16, pp. 16263-16275, Aug., 2022, DOI: 10.1109/JSEN.2022.3188845.

[5] W. Li, M. Rentemeister, J. Badeda, D. Jöst, D. Schulte, and D. U. Sauer, "Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation," *J. Energy Storage*, vol. 30, Aug., 2020, DOI: 10.1016/j.est.2020.101557.

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations (ICLR)*, 2016, DOI: 10.48550/arXiv.1510.00149.

[7] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Neural and Evolutionary Computing*, 2015, DOI: 10.48550/arXiv.1506.02626.

[8] M. H. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *Machine Learning*, 2017, DOI: 10.48550/arXiv.1710.01878.

[9] Y. Li, K. Li, X. Liu, Y. Wang, and L. Zhang, "Lithium-ion battery capacity estimation–A pruned convolutional neural network approach assisted with transfer learning," *Applied Energy*, vol. 285, Dec., 2021, DOI: 10.1016/j.apenergy.2020.116410.

[10] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. G. H. Ong, Y. T. Liew, K. Srivatsan, D. Moss, S. Subhaschandra, and G. Boudoukh, "Can FPGAs beat GPUs in accelerating next generation deep neural networks?," *2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 5-14, 2017, DOI: 10.1145/3020078.3021740.

[11] X. Dai, H. Yin, and N. K. Jha, "Grow and prune compact, fast, and accurate LSTMs," *IEEE Transactions on Computers*, vol. 69, no. 3, pp. 441-452, 2019, DOI: 10.1109/TC.2019.2954495.

[12] P. Kollmeyer, "Panasonic 18650PF Li-ion Battery Data," *Mendeley Data*, 2018, DOI: 10.17632/wykht8y7tg.1.

박 동 현

2022 경북 대학교 전자공학부(학사)

2022~현재 KAIST 전기 및 전자공학부 석사과정

관심분야: 로보틱스, 배터리 상태 추정, 인공지능

장 희 덕

2021 경북 대학교 전자공학부(학사)

2021~현재 KAIST 전기 및 전자공학부 석사과정

관심분야: 로보틱스, 딥러닝

장 동 의

1994 서울대학교 제어계측 공학과(학사)

1997 서울대학교 전기공학부(석사)

2002 California Institute of Technology, Control & Dynamical Systems (박사)

현재 한국과학기술원 전기 및 전자공학부 교수

관심분야: 제어, 드론, 로보틱스, 딥러닝, 강화학습 등