The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 20, No. 2, pp.165-170, Apr. 30, 2020. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2020.20.2.165
JIIBC 2020-2-22

ONNX기반 스파이킹 심층 신경망 변환 도구

Conversion Tools of Spiking Deep Neural Network based on ONNX

박상민*, 허준영**

Sangmin Park*, Junyoung Heo**

요 약 스파이킹 신경망은 기존 신경망과 다른 메커니즘으로 동작한다. 기존 신경망은 신경망을 구성하는 뉴런으로 들어오는 입력 값에 대해 생물학적 메커니즘을 고려하지 않은 활성화 함수를 거쳐 다음 뉴런으로 출력 값을 전달한다. 뿐만 아니라 VGGNet, ResNet, SSD, YOLO와 같은 심층 구조를 사용한 좋은 성과들이 있었다. 반면 스파이킹 신경망은 기존 활성화함수 보다 실제 뉴런의 생물학적 메커니즘과 유사하게 동작하는 방식이지만 스파이킹 뉴런을 사용한 심층 구조에 대한 연구는 기존 뉴런을 사용한 심층 신경망과 비교해 활발히 진행되지 않았다. 본 논문은 기존 뉴런으로 만들어진 심층 신경망 모델을 변환 툴에 로드하여 기존 뉴런을 스파이킹 뉴런으로 대체하여 스파이킹 심층 신경망으로 변환하는 방법에 대해 제안한다.

Abstract The spiking neural network operates in a different mechanism than the existing neural network. The existing neural network transfers the output value to the next neuron via an activation function that does not take into account the biological mechanism for the input value to the neuron that makes up the neural network. In addition, there have been good results using deep structures such as VGGNet, ResNet, SSD and YOLO. spiking neural networks, on the other hand, operate more like the biological mechanism of real neurons than the existing activation function, but studies of deep structures using spiking neurons have not been actively conducted compared to in-depth neural networks using conventional neurons. This paper proposes the method of loading an deep neural network model made from existing neurons into a conversion tool and converting it into a spiking deep neural network through the method of replacing an existing neuron with a spiking neuron.

Key Words : Deep neural network, ONNX, Spiking neural network

Ⅰ. 서 론

1. 연구 내용

기존 신경망과 스파이킹 신경망(Spiking nerual network)은 구조적으로 유사하지만 뉴런(Neuron)과 뉴런사이에 동작방식에 차이가 있다. 기존 신경망은 뉴런으로 들어온 입력 값을 보편적으로 사용하는Relu, Sigmoid, Tanh등 과 같은 활성화 함수(Activation function)를

*정회원, 에프에스솔루션
**정회원, 한성대학교 컴퓨터공학부(교신저자)
접수일자 2020년 2월 13일, 수정완료 2020년 3월 13일
게재확정일자 2020년 4월 3일

Received: 13 February, 2020 / Revised: 13 March, 2020 / Accepted: 3 April, 2020
*Corresponding Author: jyheo@hansung.ac.kr
Division of Computer Engineering, Hansung University, Korea

거쳐 출력 값을 다음 뉴런으로 전달한다. 이러한 활성화 함수는 생물학적 뉴런과 비교했을 때 매우 단순화한 방법이다.

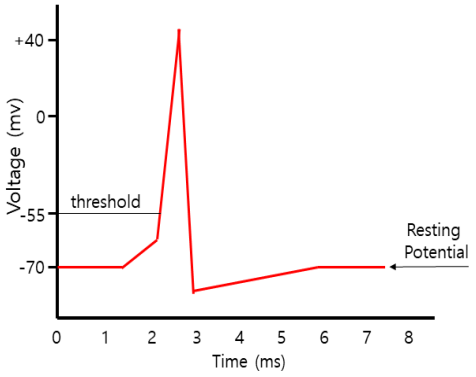

그림 1. Leaky Integrate-and-fire 모델 예시
Fig. 1. Leaky Integrate-and-fire model example

반면 생물학정 특성을 살린 뉴런모델로는 Leaky-integrate-and-fire(LIF), Izhikevich 등이 있으며 스파이킹 신경망에서는 그림 1과 같은 LIF모델이 보편적으로 사용된다.

$$T_{RC}\frac{dv(t)}{dt} = -v(t) + \mathcal{J}(t) \quad (1)$$

LIF모델의 방정식은 식 (1)과 같다. $v(t)$는 막 전위(Membrane voltage), $\mathcal{J}(t)$는 입력 전류, T_{RC}는 막 시간 상수(Membrane time constant)를 의미한다.

스파이킹 뉴런 모델을 이용한 심층 신경망 학습 방법은 기존 심층 신경망 구조는 그대로 사용하면서 기존 뉴런의 활성화 함수 대신에 스파이킹 뉴런을 사용하는 방법과 입력 스파이크와 출력 스파이크의 상대적 시간차이를 이용한 학습 방법인 Spike timing dependent plasticity(STDP)방법 등 다양하다.

본 논문에서는 좋은 성능을 보여주었던 기존 심층 신경망구조를[1][2][3][4][5][6] 그대로 사용하면서 기존 뉴런 모델을 스파이킹 뉴런 모델로 변환하고자 한다. 변환과정에서 효율적으로 스파이킹 심층 신경망(Spiking deep neural network)을 표현하기 위해 ONNX-SNN을 제안하고 기존 심층 신경망모델을 ONNX-SNN로 변환 방법과 ONNX-SNN을 기반으로 Nengo 프레임워크에서 사용할 수 있는 스파이킹 심층 신경망 모델 변환 방법을 제안한다.

II. 관련연구

1. ONNX

ONNX(Open Neural Network Exchange)는 오픈 신경망교환포맷으로 페이스북(Facebook)과 마이크로소프트(Microsoft)의 합작으로 시작된 오픈소스 프로젝트다. ONNX는 인공지능 개발자들이 인공지능 모델 개발에 있어서 특정 한 프레임워크에서 개발한 인공지능 모델을 다양한 인공지능 개발 프레임워크에 쉽게 변환할 수 있도록 한다. 이를 통해 프로젝트에 적합한 프레임워크를 사용할 수 있도록 하는 것을 목표로 한다.

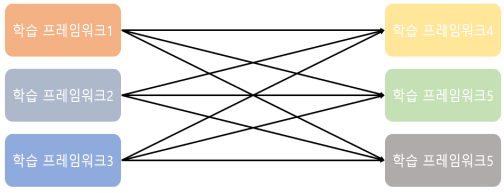

그림 2. 프레임워크 간 모델 변환
Fig. 2. Converting models between frameworks

인공지능 프레임워크마다 신경망그래프를 표현하는 방식이 다르다. 대표적으로 Pytouch는 동적 그래프를 사용하고 Tensorflow같은 경우 정적 그래프를 사용한다. 즉 프레임워크간의 다른 표현 방식 때문에 ONNX를 사용하지 않는 경우 그림 2과 같이 프레임워크마다 별도의 변환작업이 필요하다.

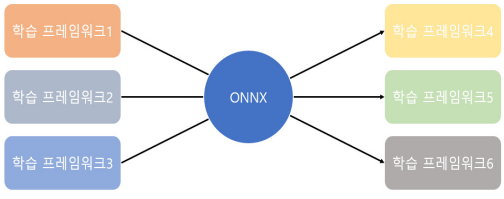

그림 3. ONNX를 사용한 프레임워크 간 모델 변환
Fig. 3. Converting Models Between Framework Using ONNX

그림 3와 같이 ONNX을 거쳐 다른 프레임워크에서 사용할 수 있는 모델로 변환한다면 프레임워크마다 별도의 변환 작업을 할 필요가 없다. ONNX는 데이터구조 지향적인 Protobuf 파일형식을 기반으로 하며 공통의 IR(Intermediate Repressentation)을 제공한다. 공통 IR을 제공하는 이유는 같은 신경망 그래프를 각각의 프레임워크마다 다른 표현 형식을 취한다. 이러한 문제를

인공지능 개발자는 공통 IR을 통해서 개발 프로젝트에 적합한 프레임워크를 쉽게 선택할 수 있게 된다.

ONNX 포맷은 신경망과 관련된 알고리즘을 지원하는 ONNX와 전통적인 머신러닝 알고리즘을 지원하는 ONNX-ML이 있다.

2. Nengo

Nengo[7]는 파이썬(Python)을 기반으로 하는 브레인(brain) 시뮬레이션 프레임워크다. Nengo 프레임워크는 뉴로모픽 하드웨어 환경에서 뿐만 아니라 뉴로모픽 하드웨어가 없는 환경에서도 스파이킹 뉴런 모델을 사용 할 수 있도록 시뮬레이션 환경을 제공한다.

기존 심층 신경망분야에서 심층 신경망 구조에 대한 연구가 활발해 지면서 좋은 성과들이 있었다. Nengo 프레임워크에서는 기존 심층 신경망 구조를 스파이킹 뉴런에 적용 할 수 있도록 돕기 위해 NengoDL[8] 라이브러리를 제공한다. NengoDL은 미분 가능하도록 최적화된 뉴런 모델을 제공하며 Tensorflow기반으로 구현되어 있어 비 스파이킹 신경망에서 사용하던 기본적인 구조를 Nengo 프레임워크에서 사용할 수 있도록 돕는다.

3. 스파이킹 심층 신경망 학습 및 추론

본 논문에서 다루는 스파이킹 심층 신경망 학습방법은 기존 심층 신경망을 그대로 사용하면서 활성화 함수 대신 스파이킹 뉴런 모델을 사용하는 방법이다. 즉 기존 심층신경망에서 사용하던 신경망 학습방법을 그대로 사용하게 되며 신경망을 학습하는 과정에서 역전파(Backpropagation)가 사용된다.

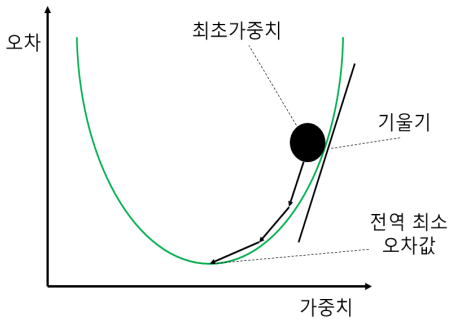

그림 4. 경사 하강법 예시
Fig. 4. Gradient descent example

경사 하강법은 비용함수(Cost function)의 최솟값을 찾기 위해 사용되며 역전파 과정에서 필요한 최적화 알고리즘이다. 그림 4과 같이 미분을 통해 구해진 기울기가 최소가 되는 가중치(Weight) 값을 찾아 계속 이동시키며 극값에 도달할 때까지 반복하는 방법이다. 경사 하강법의 종류로는 확률적 경사 하강법(Stochastic Gradient Descent, SGD), 모멘텀(Momemtum), 아담(Adam), 알엠에스프롭(RMSProp) 등이 있다.

Ⅲ. 스파이킹 신경망 변환 과정

기존 신경망 모델을 스파이킹 신경망 모델로 변환하기 위해 위해서 중간 변환과정이 필요하다. 변환과정을 설명함에 앞서 ONNX의 구조의 일부를 그림 5와 그림 6를 통해 알 수 있다.

GraphProto

```
1: message GraphProto {
2: repeated NodeProto node = 1;
3: optional string name = 2;
4: repeated TensorProto initializer = 5;
5: repeated SparseTensorProto sparse_initializer = 15;
6: optional string doc_string = 10;
7: repeated ValueInfoProto input = 11;
8: repeated ValueInfoProto output = 12;
9: repeated ValueInfoProto value_info = 13;
10: }
```

그림 5. ONNX의 그래프 proto code
Fig. 5. Graph code structure of ONNX

NodeProto

```
1: message NodeProto {
2: repeated string input = 1;
3: repeated string output = 2;
4: optional string name = 3;
5: optional string op_type = 4;
6: optional string domain = 7;
7: repeated AttributeProto attribute = 5;
8: optional string doc_string = 6;
9: }
```

그림 6. ONNX 노드 proto 코드
Fig. 6. Node proto code of ONNX

GraphProto는 ONNX의 컨포넌트들(Components) 중 하나로 신경망에 관련된 다양한 정보를 갖고 있으며 그림 5와 같은 구조를 갖고 있다. 그림 5의 line 2는 GraphProto가 NodeProto를 컨포넌트로 갖고 있는 것을 확인 할 수 있다.

NodeProto는 신경망을 구성하는 노드들을 표현할 때 사용되며 그림 6과 같은 구조로 되어 있다. 그림 6의 line 5의 op_type은 노드의 오퍼레이션 타입(Operation

type)을 나타내는 속성 값이다. 오퍼레이션 타입에는 각각의 노드의 연산(활성화 함수, 덧셈, 합성곱, Softmax 등등)정보가 포함되어 있다.

변환과정은 다음과 같다. 첫 번째 과정으로 기존 신경망 모델을 ONNX로 변환한다. ONNX로 변환방법은 기존 개발 프레임워크에서 변환툴을 지원하고 있다. 두 번째 과정은 변환된 ONNX에서 ONNX-SNN으로 변환한다. ONNX-SNN변환 방법에 대한 설명과 결과는 Ⅳ. 실험 및 결과의 2. ONNX-SNN 변환 결과에서 확인할 수 있다.

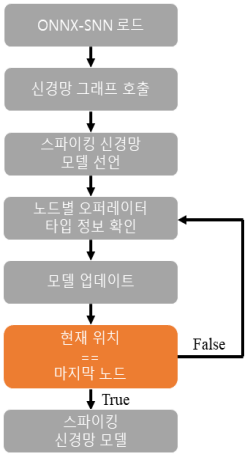

그림 7. 스파이킹 신경망 모델 생성과정
Fig. 7. Spiking Neural Network Model Generation Process

세 번째 과정으로 ONNX-SNN을 통해 Nengo 프레임워크에서 학습 및 추론 할 수 있는 모델로 변환한다. 변환과정은 그림 7을 통해 설명한다.

가장 먼저 변환하고자 하는 ONNX-SNN을 로드한 후 ONNX-SNN에서 그래프 컨포넌트를 호출한다. 그리고 변환하고자 하는 프레임워크(본 논문에서는 Nengo를 사용했다.)상에 아무것도 정의되어있지 않은 스파이킹 모델을 선언한다. 이 과정이 완료되면 호출한 신경망 그래프를 참조하여 그래프에 포함되어 있는 노드들을 순회하는 과정이 필요하다. 노드를 순회하면서 획득한 오퍼레이션 타입을 기반으로 프레임워크에 맞는 API를 사용하여 모델을 업데이트하게 된다. 이 과정이 그래프의 마지막 노드까지 완료되면 프레임워크에 학습 및 추론이 가능한 스파이킹 신경망 모델이 완성된다.

Ⅳ. 실험 및 결과

1. 실험 환경

실험에서 사용된 라이브러리는 표 1과 같다. 그래픽카드를 사용하여 학습 및 추론하기 위해서는 cudatoolkit과 cudnn을 설치해야한다.

표 1. 실험 python 라이브러리
Table 1. Experiment Python Library

라이브러리	버전
tensorflow	1.10.0
nengo	2.8.0
nengodl	2.2.0
onnx	1.6.0
numpy	1.14.5
protobuf	3.6.0
cudatoolkit	9.0
cudnn	7.6.4

본 실험에서는 MNIST데이터를 사용하여 기존 심층 신경망과 스파이킹 심층 신경망으로 변환 후 추론 결과를 비교한다. 사용하는 모델은 CNN구조의 시초가 되는 LeNet[]모델을 사용했으며 그림 8에서 볼 수 있듯이 LeNet구조 중에서 LeNet-1, LeNet-4, LeNet-5 모델을 사용한다.

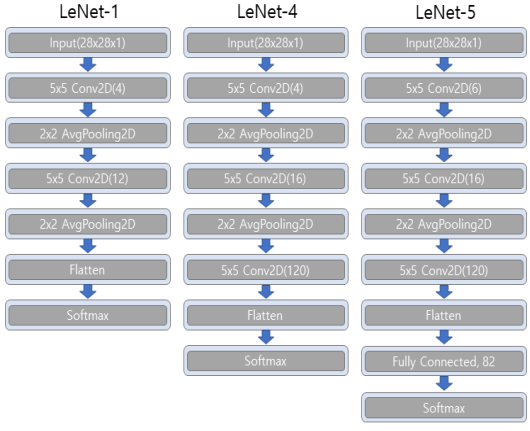

그림 8. 실험 심층 신경망 모델 구조(LeNet-1, LeNet-4, LeNet-5)
Fig. 8. Experiment Deep Neural Network Model Structures(LeNet-1, LeNet-4, LeNet-5)

그림 8의 3가지 LeNet모델의 차이점은 합성곱(Convolution) 레이어 개수와 완전연결 레이어가 모델에 포함되어 있는지의 유무이며 합성곱 연산에서 커널(Kernel) 또는 필터(Filter)의 크기가 다르다는 점이다.

2. ONNX-SNN 변환 결과

ONNX-SNN은 ONNX에서 오퍼레이션 타입 중 활성화 함수를 스파이킹 뉴런 모델(LIF, SoftLIFRate, Izhikevich 등)을 다루는 신경망 포맷이다.

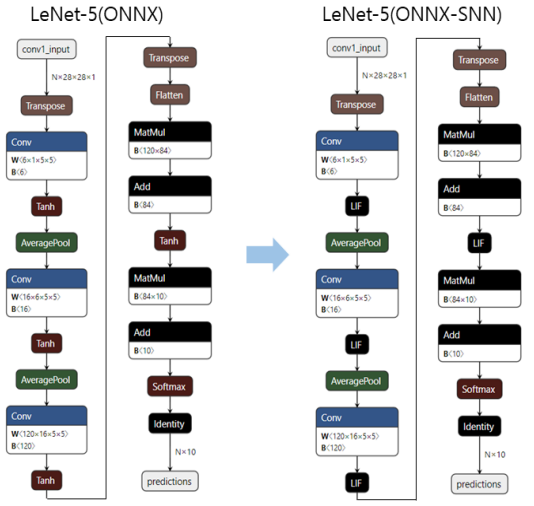

그림 9. LeNet-5모델 ONNX과 ONNX-SNN 시각화
Fig. 9. LeNet-5 model ONNX and ONNX-SNN visualization

그림 9는 신경망 모델 뷰어 응용프로그램 Netron을 사용하여 ONNX와 ONNX-SNN로 변환이 완료된 모델을 시각화한 모습이다. 그림 9의 좌측 ONNX는 기존 심층 신경망을 표현하고 있으며 LeNet-5의 활성화 함수로 Tanh가 적용되어 있는 모습을 볼 수 있다. 그림 9의 우측은 ONNX를 ONNX-SNN으로 변환하여 활성화 함수 Tanh를 스파이킹 뉴런 LIF로 변환된 것을 볼 수 있다.

변환 방법은 ONNX 신경망 그래프의 모든 노드를 순회하며 각각의 노드 오퍼레이션 타입이 활성화 함수인 경우 스파이킹 뉴런으로 변경하는 방식을 사용한다.

3. 활성화 함수별 신경망 추론결과 비교

학습 및 추론에 사용된 데이터는 MNIST이며 학습데이터로 6만개, 테스트데이터 1만개가 사용되었다. 하이퍼 파라미터(Hyper Parameter) 설정은 표 2와 같이 동일하게 진행되었다.

표 2. 하이퍼 파라미터 설정
Table 2. Hyper Parameter Settings

하이퍼 파라미터	설정 값
학습률	0.01
에폭	100
배치사이즈	200

표 3. 활성화 함수별 오차율
Table 3. Error rate by activation function

활성화 함수	LeNet-1	LeNet-4	LeNet-5
Tanh	0.06%	0.04%	0.34%
Sigmoid	0.18%	0.23%	0.34%
Relu	0.06%	0.06%	0.06%
LIF	3.58%	1.71%	3.32%
SoftLIFRate	1.07%	1.03%	1.09%

신경망 모델별로 기존 활성화 함수를 사용했을 때와 스파이킹 뉴런으로 변환하여 신경망 모델을 구성했을 때 오차율은 표 3와 같은 실험 결과물을 얻었다. 기존 활성화 함수로 대표적인 Tanh, Sigmoid, Relu를 사용했으며 스파이킹 뉴런 모델로는 LIF와 SoftLIFRate가 사용되었다. SoftLIFRate은 스파이크의 발화율을 기반으로 하는 모델로 추가적으로 미분가능 할 수 있도록 근사(Approximation)시킨 모델이다.

실험결과는 기존 활성화 함수를 사용했을 때 전반적으로 스파이킹 뉴런 모델을 사용했을 때보다 낮은 오차율을 보여주고 있다. 하지만 스파이킹 뉴런 모델들을 사용했을 때에도 상당히 낮은 오차율이라고 볼 수 있으며 다른 연구[9][10]에서의 실험결과와 비교했을 때 유사한 오차율을 확인 할 수 있다. 특히 SoftLIFRate를 사용했을 때 LIF보다 더 낮은 오차율을 보여주는데 이것은 Nengo 프레임워크의 특성 때문이다. Nengo는 LIF로 신경망을 구성해도 학습할 때에는 SoftLIFRate로 변환하여 학습을 진행한다. 반면 추론과정에서는 원래 LIF를 통해 추론 결과를 보여주기 때문에 SoftLIFRate로 설정했을 때보다 높은 오차율을 보여준다.

V. 결 론

본 논문은 기존 심층 신경망[11][12][13]에서 사용하던 신경망 모델을 스파이킹 뉴런으로 대체하여 쉽게 변환할 수 있도록 ONNX-SNN을 제안했다. ONNX-SNN은 ONNX의 구조를 그대로 사용하며 ONNX에서 다루지

않는 스파이킹 뉴런 모델을 추가적으로 다룬다는 점에서 두 포맷의 차이가 거의 없지만 ONNX에서 표현하기 힘든 스파이킹 심층 신경망 구조 또는 알고리즘들이 추후에 추가될 수 있도록 별도의 포맷으로 정의했다. 또한 ONNX-SNN을 바탕으로 ONNX에서 ONNX-SNN으로의 변환 방법과 ONNX-SNN을 프레임워크 상에서 사용할 수 있는 변환 방법을 제안했으며 기존 심층 신경망을 스파이킹 심층 신경망으로 변환하고 학습 및 추론에 문제가 없음을 실험결과를 통해 확인할 수 있었다.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, & P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE pp. 2278-2324, Nov 1998.
DOI: https://doi.org/10.1109/5.726791

[2] K. Simonyan, & A. Zisserman, "Very deep convolutional networks for large-scale image recognition", arXiv preprint arXiv:1409.1556, 2014.

[3] K. He, X. Zhang, S. Ren, & J. Sun, "Deep residual learning for image recognition", In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778, 2016.
DOI :https://doi.org/10.1109/CVPR.2016.90

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, & A. C. Berg, "Ssd: Single shot multibox detector", European conference on computer vision. Springer, Cham, pp. 21-37, Oct 2016.
DOI: https://doi.org/10.1007/978-3-319-46448-0_2

[5] J. Redmon, S. Divvala, R. Girshick, & A. Farhadi, "You only look once: Unified, real-time object detection" Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779-788, 2016.
DOI : http://dx.doi.org/10.1109/CVPR.2016.91

[6] S. J. Bae, H. J. Choi, G. M. Jeong, "YOLO Model FPS Enhancement Method for Determining Human Facial Expression based on NVIDIA Jetson TX1", The Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 12, No. 5, pp. 467-474, Oct 2019.
DOI: http://dx.doi.org/10.17661/jkiiect.2019.12.5.467

[7] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, & C. Eliasmith, "Nengo: a Python tool for building large-scale functional brain models", Jan, 2014.
DOI: https://doi.org/10.3389/fninf.2013.00048

[8] D. Rasmussen, "NengoDL: Combining deep learning and neuromorphic modelling methods", Neuroinformatics, pp. 611-628, Apr 2019.
DOI: https://doi.org/10.1007/s12021-019-09424-z

[9] E. Hunsberger, & C. Eliasmith, "Spiking deep networks with LIF neurons", arXiv preprint arXiv:1510.08829, Oct 2015.

[10] E. Hunsberger, & C. Eliasmith, , "Training spiking deep networks for neuromorphic hardware", arXiv preprint arXiv:1611.05141, Nov 2016.
DOI: https://doi.org/10.13140/RG.2.2.10967.06566

[11] Joo, Young-Do. "Drone Image Classification based on Convolutional Neural Networks." The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 17, No. 5, pp. 97-102, 2017.
DOI: https://doi.org/10.7236/JIIBC.2017.17.5.97

[12] Kang, Byung-Jun, and Jongwon Kim. "Decision Support System of Obstacle Avoidance for Mobile Vehicles." Journal of the Korea Academia-Industrial cooperation Society, Vol. 19, No. 6, pp. 639-645, 2018.
DOI: https://doi.org/10.5762/KAIS.2018.19.6.639

[13] Kim, Juhyun, and Dongho Kim. "Neural network based real-time UAV detection and analysis by sound." Journal of Advanced Information Technology and Convergence, Vol. 8, No. 1, pp 43-52, 2018.
DOI: https://doi.org/10.14801/JAITC.2018.8.1.43

저 자 소 개

박 상 민(정회원)

- 2018년 : 한성대학교 IT응용시스템공학과 졸업(학사)
- 2020년 : 한성대학교 컴퓨터공학과 졸업(석사)
- 2020년 ~ 현재 : 에프에스솔루션
- 관심분야 : 인공지능, IoT

허 준 영(정회원)

- 1998년 : 서울대학교 컴퓨터공학과 졸업
- 2009년 : 서울대학교 컴퓨터 공학과 졸업(박사)
- 2009년 ~ 현재 : 한성대학교 컴퓨터공학부 부교수
- 관심분야 : 운영체제, 무선 센서 네트워크, 임베디드 시스템, 기계 학습

※ 본 연구는 한성대학교 교내학술연구비 지원과제 임