https://doi.org/10.7840/kics.2025.50.2.234

Multi-Exit Faster R-CNN(MEF): 변동하는 자원에 적응적인 다중 출구 신경망

김 현 수•, 구 동 우•, 이 다 연*, 최 민 석°

Multi-Exit Faster R-CNN(MEF): An Adaptive Multi-Exit Neural Network for Time-Varying Computing Resources

Hyeonsu Kim•, Dongwoo Goo•, Dayeon Lee*, Minseok Choi°

요 약

이 논문은 변동하는 자원 환경에서 적응적으로 정확도와 지연시간을 조절할 수 있는 Faster R-CNN 모델과 출구 선택 알고리즘을 개발한다. 최근 컴퓨터 비전과 자연어 처리 분야의 급격한 발전과 함께 추론 지연시간이 중요한 요소로 작용하는 반면, 기존 연구들에서는 자원이 동적으로 변하는 환경에서 지연시간과 정확도 간의 최적의 트레이드-오프를 보장하는 모델과 알고리즘이 존재하지 않았다. 본 연구에서는 이러한 문제를 해결하기 위해 다중 출구(multi-exit) 신경망 기법과 Lyapunov 최적화를 활용한 Faster R-CNN 모델 및 출구 선택 알고리즘을 제안한다. 제안한 기술은 자원의 변화가 불확실한 상황에서도 시스템의 안정성을 유지하면서 장기적으로 최적의 성능을 보장하며, 자원 상황에 따라 추론을 위한 최적의 출구를 선택할 수 있다. 또한, 시뮬레이션을 통해 제안한 기술이 가용 자원에 따라 정확도와 지연시간 사이의 트레이드-오프를 조절할 수 있음을 검증하였다.

Key Words : Object detection, Multi-exit neural network, Lyapunov optimization

ABSTRACT

This paper develops a multi-exit Faster R-CNN model and an exit selection algorithm that can adaptively adjust accuracy and latency in a fluctuating resource environment. Despite the rapid advancements in computer vision and natural language processing, where inference latency is increasingly critical, existing studies lack models and algorithms that ensure an optimal trade-off between latency and accuracy in dynamically changing environments. This study proposes a multi-exit neural network technique and a Faster R-CNN model utilizing Lyapunov optimization to address these issues. The proposed approach guarantees long-term optimal performance while maintaining system stability under uncertain resource changes and allows for the selection of the optimal exit for inference based on resource conditions. Additionally, simulations have verified that the proposed technology can adjust the trade-off between accuracy and latency according to available resources.

※ 본 연구는 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-02201, 사용자 프라이버시를 보존하는 비디오 캐싱을 위한 연합 학습 시스템).
※ 본 연구는 과학기술정보통신부 및 정보통신기획평가원의(오픈랜 인력양성 프로그램(연세대))연구 결과로 수행되었음 (IITP-2025-RS-2024-00434743)
• First Author(Equal Contribution) : Kyung Hee University, Department of Electronics Engineering, nan111nan78@gmail.com; wendy1301@naver.com, 학생회원
° Corresponding Author : Kyung Hee University, Department of Electronics Engineering, choims@khu.ac.kr, 종신회원
* Kyung Hee University, Department of Electronics Engineering, rainbowgt7@khu.ac.kr
논문번호 : 202408-186-A-RE, Received August 24, 2024; Revised October 13, 2024; Accepted October 14, 2024

Ⅰ. 서 론

1.1 연구 배경

최근 컴퓨터 기술의 급속한 발전으로 인해 기존의 텍스트 위주의 사용자 환경에서 벗어나 이미지, 그래픽, 오디오 및 비디오 데이터 등을 제공하는 멀티미디어 사용자 환경으로 변화하고 있다. 이와 함께 딥러닝을 활용한 컴퓨터 비전은 자연어 처리와 함께 큰 발전을 이뤄내고 있다. 컴퓨터 비전에서는 성능을 판단하는 다양한 지표들이 존재한다. 특히 사용자와 실시간 상호작용이 중요한 환경에서 새로 개발하는 딥러닝 모델들의 주요 성능 지표들은 추론의 높은 정확성, 낮은 지연 시간(Latency) 등이 있다. 그렇지만 주로 사용하는 프로그램의 종류와 사용자의 취향에 따라 특정 지표가 다른 지표들에 비해 더 높은 우선순위를 가질 때도 있다. 예를 들어, 자율주행과 메타버스와 같은 증강현실 및 가상현실 기술에서는 즉각적인 반응으로 사고를 예방하기 위해 정확도보다는 지연시간의 중요도가 높다.[14]

이처럼 빠른 추론을 달성하기 위해 다중 출구(multi-exit) 신경망이 개발되어 테스트 샘플에 따라 이른 출구에서의 추론을 허용하였다. 그러나 기존 다중 출구 신경망에서는 주어진 컴퓨팅 자원이 변하는 상황은 고려하지 않았고, 순차적인 레이어로 구성된 단일 모델에서만 다중 출구 구조를 적용하였다[1]. 반면, 다중 프로세스 시스템에서 백그라운드 프로그램이 점점 많아지며 고정된 양의 컴퓨팅 자원(GPU, CPU 클럭 등)을 항상 사용하기 힘들고 가용 자원이 실시간으로 변화할 수 있다. 이런 환경에서도 일정 수준의 지연시간을 유지하기 위해서 분할 컴퓨팅(split computing) 기술이 등장했지만, 여전히 모든 모델 구조를 통과해야만 결과를 얻을 수 있다.[13] 따라서, 본 논문에서는 가용 자원이 외부 프로세스로 인해 실시간으로 변화하는 환경에서도 일정 수준의 지연시간을 유지하기 위해 다중 출구 신경망에서의 동적 출구 선택 알고리즘을 설계하였다. 또한, 이것을 다중 특징(feature) 활용을 위해 모델 블록을 분리시켜 객체 탐지 부문에서 SOTA 성능을 내는 Faster R-CNN에 적용하여 모든 레이어가 직렬 연결되지 않은 모델에서도 성능을 낼 수 있음을 보였다. 최종적으로, 주어진 자원이 동적으로 변하는 환경에서 일정 정확도는 보장하지만 동시에 지연 시간은는 최소화할 수 있는 MEF(Multi-Exit Faster R-CNN) 모델과 출구 선택 알고리즘을 개발하였다.

1.2 선행 연구 추세

컴퓨터 비전은 크게 주어진 입력 이미지에서 픽셀별로 어떤 물체인지 분류하는 의미적 분할 (Semantic Segmentation), 그리고 주어진 입력 이미지에서 하나의 객체와 객체의 위치를 알려주는 분류 및 위치 식별(Localization), 그리고 주어진 입력 이미지에서 여러 개의 객체와 객체들의 위치를 알려주는 객체 탐지(Object detection), 마지막으로 객체 탐지에서 경계선마다 분할을 나눠주는 인스턴스 분할(Instance Segmentation)으로 나뉜다.

이 중 객체 탐지(Object Detection) 부문의 대표적인 모델들은 크게 1-단계(1-Stage) 모델들과 2-단계(2-Stage) 모델들로 두 갈래로 분류할 수 있다. 먼저 2-단계 모델들은 객체들이 있을 만한 영역을 찾는 과정인 객체 검출(Region Proposal)을 먼저 진행한 뒤 찾은 객체 후보 이미지들에 대해 분류를 순차적으로 실행하여 이미지에서 객체들을 탐지하고 분류한다. 반면에 1-단계 모델들은 객체 검출(Regional Proposal)과 분류를 동시에 진행하여 객체 탐지를 수행한다는 차이점이 있다.

2-단계 모델의 대표적인 모델들로는 RCNN (Regions with CNN)[10], Fast RCNN[11], Faster RCNN[12]이 있고 1-단계 모델의 대표적인 모델들로는 YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), RefineDet (Refinement Neural Network for Object Detection)이 있다.

먼저 처음으로 제안된 2-단계 객체 탐지 모델인 RCNN은 선택적 탐색(Selective Search)라는 알고리즘을 통해 객체 검출(Region Proposal)을 진행한 뒤 객체가 있을 만한 영역 후보군 2000개를 원본 이미지에서 잘라내고 줄이거나 늘려 후보군 2000개 각각에 대해 CNN 블록에 넣는 과정을 2000번 반복해서 이미지 분류를 진행하는 모델이다.[10] 이러한 RCNN은 분류 정확도 측면에서 다른 1-단계 객체 탐지 모델들에 비해 우수했지만 다른 1-단계 모델들에 비해 느리다는 것이 단점이다.

이후 제안된 Fast RCNN에서는 RCNN과 마찬가지로 선택적 탐색이라는 알고리즘을 통해 객체 검출을 진행한 뒤, 객체가 있을 만한 영역 2000개를 원본 이미지가 아닌 특징 맵(Featrue Map) 단계에서 관심 영역(Region of Interest, RoI)을 추출하여 CNN 블록에서 단 한 번의 이미지 분류를 통해 객체 탐지를 수행할 수 있게 만들어 기존 RCNN에 비해 속도를 향상시켰다.[11]

마지막으로 Faster RCNN에서는 선택적 탐색 알고리즘이 CPU 연산을 요구함에 따라 객체 검출(Region Proposal) 부문을 GPU 영역으로 끌어오기 위해 객체

검출(Region Proposal) 기능을 수행하는 딥러닝 인공지능 신경망인 RPN(Region Proposal Network을 제작하여 기존 선택적 탐색 알고리즘을 사용했을 때와 비교해 속도 향상을 이루었다.[12]

다중 출구의 개념은 흔히 "Multi-Exit", "Anytime Predictions"[2] 그리고 "BranchyNet"[1] 같이 여러 단어로 언급되며 연구 되어왔다. 이러한 모델들은 모델이 최종 출력 레이어 이전에도 추론할 수 있도록 보조 분류기를 앞쪽 레이어에 배치하여 이른 출구(early exit)에서 추론 결과를 뽑을 수 있도록 설계되었다. "BranchyNet"은 일정 수준의 엔트로피 값을 보이면 이른 출구로 출구를 선택하여 전체적인 추론 시간을 줄일 수 있음을 보였다.

랴프노프 최적화(Lyapunov Optimization) 이론은 시스템이 확률적 불확실성을 내포하고 있어도 장기적으로 최적의 성능을 보장하는 수학적 기법이다. 고려하려는 변수를 반영하는 가상의 큐를 선언한 다음 큐의 길이를 발산하지 않도록 안정화하여 큐의 길이가 동적으로 변하더라도 큐의 안정성은 보장함으로 장기적으로 최적의 성능을 보장한다.[3]

현재 딥러닝 분야에서 랴프노프 최적화는 네트워크 버퍼나 엣지 컴퓨팅 등 다양한 분야에서 최적화에 활용된다.[4-9] 지금까지 다양한 분야에서 다양한 지표들을 변수로 삼아 랴프노프 최적화를 통해 최적화를 시도하였지만 주로 엣지 컴퓨팅에서 각 서버들에게 태스크 오프로딩(Task Offloading) 문제에서 에너지 그리고 서버와의 통신 지연 시간 변수들을 최적화하는 경우다. 이뿐만이 아니라 컴퓨팅 자원과 지연 시간 사이의 관계를 연구한 논문도 존재한다.[5] 하지만 이는 주어진 자원도 통제할 변수로 삼아 우리의 관심사인 변동하는 자원에 따른 지연 시간과 정확도를 최적화하는 우리의 연구의 목적과 맞지 않았다. 그렇기에 지금까지의 연구에서는 우리의 목적을 달성하는 모델과 알고리즘은 없었다. 그렇기에 주어진 자원이 계속 변동하는 상황에서 지연 시간과 정확도 사이의 최적화가 되는 출구 지점을 찾기 위해 본 연구를 진행하였다.

1.3 연구 목적

본 논문에서는 다중 프로세스 시스템에서 가용 컴퓨팅 자원이 고정적이지 않을 때 딥러닝 모델의 추론 정확도와 지연시간 사이의 트레이드오프를 조절할 수 있는 다중 출구 신경망 구조와 학습 방법을 설계하고, 추론 시에 동적으로 예측 값을 얻어낼 출구를 선택하는 알고리즘을 제시한다. 이를 통해 제안한 모델 단 하나를 사용함에도 주로 추론되는 출구의 경향을 조정할 수 있어 한 번의 학습으로 작은 모델과 큰 모델을 취향에 맞게 선택하거나 이 두 모델의 타협점을 선택하여, 자원 상황에 따라 최적의 정확도와 지연 속도를 만족시키는 출구를 찾아준다. 따라서, 우리는 SOTA 성능을 내는 객체 탐지 모델 중에서도 가장 추론 속도가 빠른 Faster R-CNN에서도 제안한 다중 출구 모델 및 동적 출구 선택 방법이 효과가 있는지 검증하고자 한다. 또한, 대부분의 기존 다중 출구 신경망 활용 추론 속도 향상 기술들은 순차적인 레이어로 구성된 모델에만 적용했지만, 본 논문에서는 Faster R-CNN과 같이 다중 특징 추출을 위해 일부 모델 블록이 분리되어 병렬적으로 구동되는 모델에서도 다중 출구 신경망 구조가 효용성이 있는지도 검증한다.

Ⅱ. 다중 출구 신경망 기반의 Faster R-CNN 설계

2.1 Faster R-CNN

Faster R-CNN 구조는 크게 백본(backbone)과 RPN, 검출기(Detector), 3가지로 구성된 객체 탐지 및 분류 인공지능 신경망이다.[12]. Faster R-CNN은 Detector와 별개의 신경망인 RPN을 도입하여 관심 영역(ROI)를 따로 찾아내어, 기존 Fast R-CNN 내 선택적 탐지 방식에 요구되는 긴 소요시간을 크게 단축하였다. 백본으로 사용되는 모델은 기본적인 특징들을 표현해주기 위해 미리 학습된된 가중치를 활용한다. 다만 미리 학습된 모델의 구조에서 RPN의 입력으로는 원본의 1/64 크기인 특징 맵이 사용되며 이미지의 특정 부분이 객체일지에 대한 여부를 예측한다. RPN의 결과로 나온 앵커(Anchor)는 검출기에서 어느 객체인지에 대한 예측이 진행된다. 전체 구조는 그림 1과 같다.

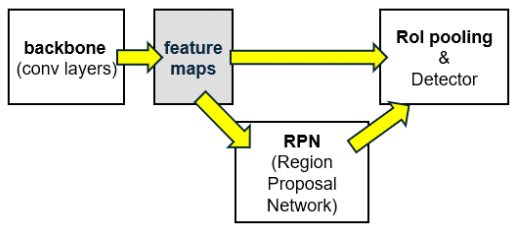


그림 1. 전체적인 Faster R-CNN의 구조
Fig. 1. Overall structure of Faster R-CNN

2.2 MEF: 다중 출구 Faster R-CNN

본 연구에서 설계하고자 하는 다중 출구 Faster R-CNN는 연산량의 대부분을 차지하는 백본에 분기점

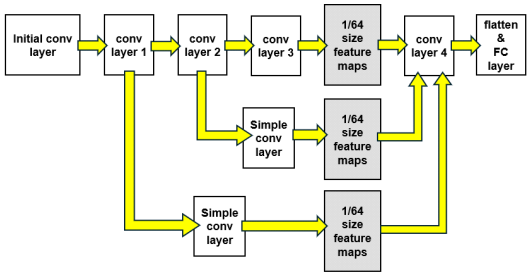


그림 2. K=3일 때 Multi-Exit Faster R-CNN(MEF) 백본의 구조
Fig. 2. Backbone structure of Multi-Exit Faster R-CNN(MEF) with K=3

을 두어 이른 출구(Early Exit)를 배치하고자 한다. 하지만 Faster R-CNN의 특성상 RPN의 입력으로는 원본의 1/64크기의 특징 맵이 들어가야 하며, 백본에서의 이른 출구에서 바로 추론 결과를 내는 것이 아니라 RPN의 출력과 함께 검출기를 통과하여 최종 출력값을 추론해야 한다. 이러한 구조로 인해 기존의 다중 출구 신경망의 구조와 같이 각 분기의 출구로 나온 특징 맵을 그대로 1차원화(Flatten) 한 뒤 검출기의 입력으로 넣기에는 문제가 발생한다. 따라서 본 연구에서는 각 분기마다 결과의 특징 맵을 검출기와 RPN에 들어가기에 알맞은 크기의 특징 맵으로 변화시키기 위해 연산량이 매우 간단한 보조 CNN 블록을 둔다.

이 추가적인 CNN 블록의 출력은 RPN의 입력에 맞추어 원본의 1/64크기의 특징 맵이 됨과 동시에 그 특징 맵이 분기가 아닌 원래의 CNN 블록을 다 지나왔을 때의 다음 CNN 블록 입력으로 들어가더라도 높은 정확도가 나오는 상황이 구현되어야 한다. 미리 학습된 백본 네트워크에 분기를 K개 만든다고 가정하였을 때, 그림 2는 K=3일 때의 다중 출구 구조 기반의 백본 네트워크를 묘사한다. 그림 2에서 initial conv layer, conv layer 1, conv layer 2, conv layer 3, conv layer 4와 flatten & FC layer는 미리 학습해야 할 백본 네트워크의 블록들이다. 이때, conv layer 3에서 conv layer 4로 전달되는 특징 맵은 1/64 크기를 가지므로, conv layer 1과 conv layer 2에서 이른 출구를 설정하고자 하면 그 뒤에 간단한 CNN 블록을 이용해 특징 맵의 크기를 맞춘다. RPN과 결합하여 최종적으로 본 연구에서 사용되는 다중 출구 Faster R-CNN 모델의 구조는 그림 3과 같이 구성된다.

2.3 백본 학습

기존 Faster R-CNN 모델에는 ImageNet-1K 데이터셋을 바탕으로 학습이 완료된 백본이 사용된다. 그러나 본 연구에서는 백본 네트워크에 여러 개의 출구를 만들고 각 출구의 출력 값인 특징 맵이 모두 검출기와 RPN에게 있어 의미 있는 입력 값이 되도록 해야 하므로, 다중 출구 구조 기반의 백본 네트워크의 학습도 새로 해 주어야 한다.

다중 출구를 가진 백본을 학습시키는 과정은 기존 다중 출구 신경망을 학습시키는 과정과 유사하지만, 본 연구에서는 백본 네트워크의 출구들이 최종 출력 값을 내리지는 않는다. 따라서 C개의 종류를 구분하는 C-Class 분류 문제에 대해서 백본에 K=3개의 분기를 추가할 때, 이른 출구인 처음 2개의 출구에는 간단한 CNN블록에 더불어 보조 분류기까지 배치하여 학습을 진행한다. 학습 후에 그림 3처럼 다중 출구 기반의 Faster R-CNN을 구성할 때에는 보조 분류기는 제거한 채 사용하도록 한다.

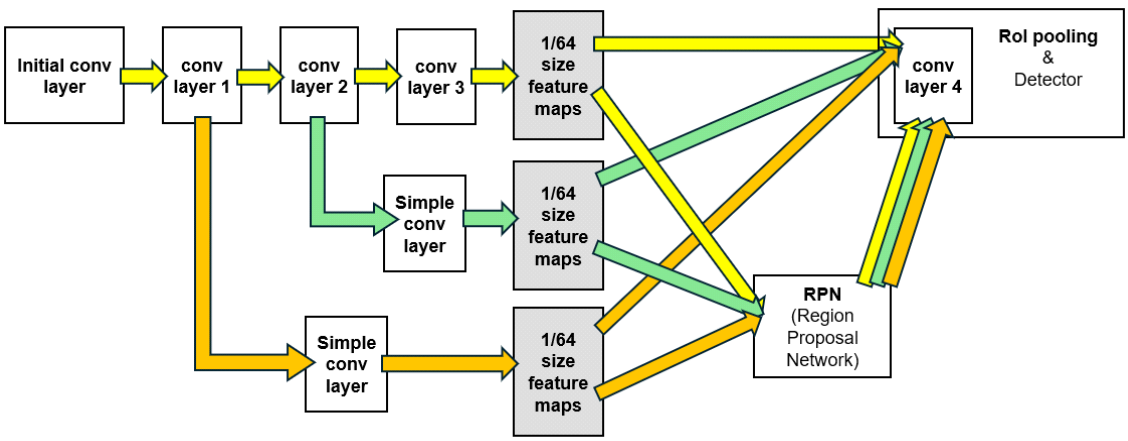


그림 3. K=3일 때 Multi-Exit Faster R-CNN(MEF) 구조
Fig. 3. Overall structure of Multi-Exit Faster R-CNN(MEF) with K=3

백본 모델 파라미터를 θ, 실제 라벨 값 y, 출구 k의 softmax 예측을 $\hat{\boldsymbol{y}}_{\boldsymbol{k}}=[y_{k,1},y_{k,2},\cdots,y_{k,C}]$ 라 할 때, 출구 k의 손실함수($L_k(\hat{\boldsymbol{y}}_{\boldsymbol{k}},y;\theta)$)는

$$L_k(\hat{\boldsymbol{y}}_{\boldsymbol{k}},y;\theta)=-\frac{1}{C}\sum_{c=1}^{C}y\log\hat{\boldsymbol{y}}_{\boldsymbol{k,c}} \tag{1}$$

로 정의된다. 본 연구에서는 모든 출구에서의 특징 맵이 검출기와 RPN에 의미 있는 입력이 되도록 해야 하므로, 전체 손실함수는 다음과 같이 각 출구의 손실함수의 가중 평균으로 정의될 수 있다:

$$L(\hat{\boldsymbol{y}},\boldsymbol{y};\theta)=\quad-\frac{1}{N}\sum_{k=1}^{K}\gamma_kL_k(\hat{\boldsymbol{y}}_{\boldsymbol{k}},y;\theta) \tag{2}$$

식 (2)에서 γ_k은 출구 k의 손실함수에 대한 가중치로, $0\le\gamma_k\le1$과 $\sum_{k=1}^{K}\gamma_k=1$을 만족한다. 출구 별로 정확도 성능이 모두 같지 않으므로, 가중치 γ_k를 조절하여 출구 별 정확도의 차이를 일부러 만들거나 좁힐 수도 있다. 이와 같이, 손실함수 (2)를 최소화하는 경사 하강법을 통해 다중 출구 Faster R-CNN에 사용될 백본 네트워크를 학습시킬 수 있다.

2.4 다중 출구 Faster R-CNN의 학습

Faster R-CNN의 학습 과정에서는 학습이 완료된 다중 출구 백본을 사용한다. 그 과정에서 보조 분류기를 제외한 CNN 블록의 가중치만을 활용한다. 또한 미리 학습하는 과정으로 만들어진 다중 출구를 가진 백본에서 원본 이미지의 1/64 크기의 특징 맵을 만드는 간단한 CNN 블록까지의 가중치는 학습이 되지 않도록 막으며 RPN과 검출기의 입력으로 사용된다. 이후 1차원화의 신경망 이전까지의 CNN 블록은 검출기로 활용된다.

객체를 예측하는 검출기를 학습함과 동시에 Faster R-CNN의 핵심인 RPN을 함께 학습하기에 있어, 다중 출구 신경망의 개념을 도입시키는 방법에는 크게 2가지 방법이 사용될 수 있다. 첫 번째 방법은 각 출구에 따른 RPN과 검출기를 병렬적으로 구성하는 방법이며 두 번째 방법은 각 출구에 있어 공유하는 하나의 RPN과 검출기를 구성하는 방법이다.

먼저 출구마다 RPN과 검출기를 병렬적으로 구성하기 위해 출구마다 따로 $RPN_k(\hat{\boldsymbol{x}})$과 $Detector_k(\hat{\boldsymbol{x}})$를 만든다. 이후 단순히 K개의 RPN과 검출기를 K번의 별개의 학습 과정을 통해 따로 학습시킨다.

먼저 $RPN_k(\hat{\boldsymbol{x}})$의 결과로 나온 각 앵커에 대해 i번째 앵커가 객체일 확률 $p_{i,k}$은 실제 객체를 포함하고 있는지에 대한 label p_i^* 과 함께 손실함수 L_{ds_k}을 구성하게 된다. 또한 앵커를 포함하더라도 세부적인 바운딩 박스를 표현하기 위해 실제 바운딩 박스 좌표 t_i^*와 예측한 바운딩 박스의 좌표 $t_{i,k}$ 를 이용하여 손실함수 L_{reg_k}을 구성한다. 최종적으로 k번째 출구에 대한 $RPN_k(\hat{\boldsymbol{x}})$ 학습에 사용되는 손실함수는

$$\begin{aligned}L_k(\{p_{i,k}\},\{t_{i,k}\})&=\frac{1}{N_{cls}}\sum_{i=1}^{C}L_{ds_k}(p_{i,k},p_i^*)\\&+\lambda\frac{1}{N_{reg}}\sum_{i=1}^{C}p_i^*L_{ds_k}(t_{i,k},t_i^*)\end{aligned} \tag{3}$$

와 같이 구성된다. N_{ds}와 N_{reg}는 미니배치에 대한 정규화를 위한 값으로 미니배치의 크기이며 객체 여부와 위치에 대한 수식의 균형을 맞추는 λ는 기본 10으로 설정한다.

두 번째 방법의 경우에는 백본 네트워크에서 같은 형태의 특징 맵을 내도록 하는 모든 출구에 대해 공유하는 $RPN(\hat{\boldsymbol{x}})$과 $Detector(\hat{\boldsymbol{x}})$를 각각 구성한다. 이 경우에는 학습 과정에서 매번 출구의 개수만큼의 추론을 진행하며 각 예측의 결과 $p_{i,k}$와 정답 p_i^*으로 손실함수 L_{ds_k}를 구성한다. 또한 실제 바운딩 박스 좌표 t_i^*와 예측한 바운딩 박스의 좌표 $t_{i,k}$를 이용하여 손실함수 L_{reg_k}를 구성한다. 최종적으로 각 출구에 대한 손실함수는 수식 (3)와 동일하게 만들어지며 공유되는 모델 $RPN(\hat{\boldsymbol{x}})$의 학습에 사용되는 최종적인 손실함수는

$$L(\widehat{\boldsymbol{L}_{\boldsymbol{n}}})=\frac{1}{K}\sum_{k=1}^{K}L_k(\{p_{i,k}\},\{t_{i,k}\}) \tag{4}$$

와 같이 구성된다.

Ⅲ. 동적 출구 선택 알고리즘

3.1 추론 지연 시간 모델

그림 3은 출구가 3개인 다중 출구 Faster R-CNN (MEF)의 구조를 보여준다. 우리는 앞으로 단위 시간 $\triangle t$(sec)의 지속 시간을 가진 $t\in\{0,1,\cdots,T-1\}$로 구성된 이산적인 시간의 시스템(Discrete-time system)을 가정하고, 매 시각 t마다 하나의 테스트 이미지가 입력

된다고 가정한다.

먼저 다중 출구 Faster R-CNN에서 출구의 개수가 K일 때, $k-1$번째 출구와 k번째 출구 사이의 모델 블록을 θ_k라 하자. 이때, C_k는 추론 시 블록 θ_k가 순전파를 하는데 필요한 부동소수점 연산이라 하면, 처음부터 k번째 출구까지의 부동소수점 연산량의 총합은 $S_k = \sum_{l=1}^{k} C_l$로 정의한다. 또한, RPN을 처리하는 데 소요되는 부동소수점 연산량은 C_{RD}라 하면, 하나의 테스트 이미지를 k번째 백본 출구에서 추론할 때 소요되는 지연시간인 $\tau_k(t)$는 식 (5)과 같이 나타낼 수 있다:

$$\tau_k(t) = \frac{S_k + C_K + C_{RD}}{u_d(t)} + \alpha \tag{5}$$

여기에서, $u_d(t)$는 시각 t에서의 가용 GPU 자원의 단위 부동소수점 연산당 소요되는 주기 (Hz)이고, α는 알고리즘을 통해 최적의 출구를 구하는 데 걸린 시간이다. 이 과정은 챕터 2.3.2에서 자세히 소개되겠지만, $O(K)$의 연산으로 α는 매우 작은 값이다. 또한, 마지막 검출기는 무조건 통과해야 하므로, C_K는 상수로서 필요 부동소수점 연산량에 포함되었다.

3.2 문제 정의

제안하는 동적 출구 선택 알고리즘의 목표는 지연시간을 최소화하면서 장기적으로는 기준 임계값 이상의 정확도는 보장하는 것이다. 먼저 총 지연 시간을 최소화하기 위해서는 모든 t에 대해 각 $\tau_{k(t)}(t)$의 기댓값들의 장기 평균을 최소로 만드는 $\boldsymbol{k} = [k(1), k(2), \cdots k(T)]$를 찾으면 된다. 여기에서, $k(t)$는 시각 t에 입력된 테스트 이미지 추론을 위해 선택한 출구 번호이다. 이와 마찬가지로 장기적으로 기준 임계값 이상으로 모델의 정확도를 보장하기 위해서는 매 추론의 정확도 $P_{exit}(k(t))$의 기댓값들의 장기 평균이 기준 임계값보다 높도록 설정할 것이다.

여기서 $k(t)$번째 출구에서의 실제 추론 정확도인 $P_{exit}(k(t))$는 실제로 $k(t)$번째 출구까지 모델을 실행시켜야만 알 수 있지만, 우리는 테스트셋과 학습 데이터셋의 분포가 비슷하다고 가정하여 실제 추론에서도 각 출구에서의 $P_{exit}(k(t))$는 테스트 때와 동일하다고 가정하고 $P_{exit}(k(t))$는 미리 구한 테스트 데이터셋의 mAP로 대체한다.

식으로 정리하면 다음과 같다.

$$\mathbf{min}_{\boldsymbol{k}} \ \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\tau_{k(t)}(t)] \tag{6}$$

$$(\mathrm{S \cdot T}) \ \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[P_{exit}(k(t))] \geq P_{th} \tag{7}$$

3.3 알고리즘 도출

식 (6)와 식 (7)은 최종 출구의 순번인 $k(t)$이 증가하거나 감소할 때 서로 상충된 결과를 보여준다. 예를 들어 실제로 식 (6)은 $k(t)$가 감소할수록 전체 모델 중 뒷 순번의 블록을 거치치 않고 바로 추론하기 때문에 $\tau_{k(t)}(t)$가 감소한다. 반대로 식 (7)은 $k(t)$가 증가할수록 증가하는데, 그 이유는 일반적으로 더 깊은 모델을 통해 나온 분류 결과가 좋기 때문이다. 따라서, 우리는 식 (7)을 만족시키는 출구 중에서 식 (6)을 최소화하는 절충된 출구 순번인 $k(t)$를 찾아야 한다. 식 (6)과 같이 장기적인 시간의 평균 목적함수를 최적화하기 위해 본 논문에서는 라프노프 최적화 이론을 문제에 적용했다.

먼저 식 (7)인 제한 사항을 만족시키기 위해 시간 t까지 정확도 보장이 잘되었는지 판단할 수 있는 가상의 큐 $Y(t)$(단, $Y(0) < \infty$)를 정의한다. 이를 수식으로 작성하면 가상의 큐 $Y(t)$는 다음 식을 만족해야 한다:

$$Y(t+1) = \mathbf{max}(Y(t) + P_{th} - P_{exit}(k(t)),\ 0). \tag{8}$$

라프노프 최적화 이론에 따르면, 큐 $Y(t)$가 충분한 안정성(strong stability)을 만족하면 식 (7)이 성립함을 증명할 수 있다.[3] 큐 $Y(t)$의 충분한 안정성은 P_{th}의 장기 평균을 $P_{exit}(k(t))$의 장기 평균보다 작게 만들어서 다음과 같은 식을 만족시키면 된다: $\lim_{T\to\infty} \mathbb{E}\{Y(t)\}/T = 0$. 큐 $Y(t)$의 안정성을 만족시키기 위해 라프노프 함수 $L(t)$와 $L(t)$의 드리프트, $\triangle L(t)$를 식 (9)와 식 (10)과 같이 정의한다.

$$L(t) = Y(t)^2 \tag{9}$$

$$\triangle L(t) = \mathbb{E}[L(t+1) - L(t)] \tag{10}$$

라프노프 드리프트(Lyapunov Drift)와 문제 상황에서 패널티인 지연 시간의 가중합인 라프노프 드리프트 플러스 패널티(Lyapunov Drift-Plus-Penalty, DPP)는 식 (11)과 같이 작성될 수 있다:

$$\triangle L(t) + V\mathbb{E}[\tau(t) | Y(t)] \tag{11}$$

식 (11)에서 V는 음수가 아닌 상수로 가상 큐 $Y(t)$를 안정화하거나 지연 시간을 줄이는 트레이드 오프를 결정하는 매개변수이다. V가 작다면 지연 시간을 줄이는 것보다는 정확도가 잘 지켜지는지를 의미하는 큐 $Y(t)$의 값을 감소시키는데 더 집중할 것이고 V가 크다면 정확도를 보장하는 것보다는 지연 시간을 줄이는 것에 더 집중할 것이다.

DPP의 상한선(Upper-Bound)을 구하기 전 먼저 드리프트의 상한선을 구하기 위해 식 (10)을 전개 후 정리하면 식 (12)과 같이 상한선를 정리할 수 있다.

$$\begin{aligned}&\triangle L(t) = \mathbb{E}[L(t+1) - L(t)] \\ &= \mathbb{E}[Y(t+1)^2 - Y(t)^2] \\ &\leq \mathbb{E}[(Y(t) + P_{th} - P_{exit}(k(t)))^2 - Y(t)^2] \\ &\leq 2Y(t)(P_{th} - P_{exit}(k(t))) + (P_{th} - P_{exit}(k(t)))^2\}\end{aligned} \tag{12}$$

식 (12)에서 구한 드리프트의 상한선을 통해 드리프트와 지연 시간의 가중합인 DPP의 상한선은 식 (13)과 같이 정리된다.

$$\begin{aligned}&\triangle L(t) + V\mathbb{E}[\tau_{k(t)}(t) | Y(t)] \\ &\leq 2Y(t)(P_{th} - P_{exit}(k(t))) + (P_{th} - P_{exit}(k(t)))^2 \\ &+ V\left(\frac{S_{k(t)} + E_{k(t)} + C_K + C_{RD}}{u_d(t)} + \alpha\right)\end{aligned} \tag{13}$$

위에서 작성한 라프노프 최적화는 조절 가능한 변수 V, α 그리고 $P_{exit}(k(t))$를 통해 DPP의 최소 상한선 값을 찾을 수 있는 방법이다. 즉, 가상의 큐를 안정화하면서 동시에 트레이드 오프 관계에 있는 지연 시간을 줄일 수 있다. 이는 비록 근시안적으로 시간 t에서의 DPP 함수를 최적화할 뿐이지만, 장기적으로 식(13)의 우변을 최소화하는 과정으로 식 (6)의 최소화와 식 (7)을 동시에 만족하는 것으로 알려졌다.[3]

따라서 매 시각 t마다 입력되는 테스트 이미지에 대해 모든 출구에 대해서 식 (9), 즉 DPP의 상한선을 구하고, 그중에서 최소 값을 생성하는 출구 $k^*(t)$를 선택하는 것으로 간략화 할 수 있게 된다. 테스트 이미지마다 출구의 개수인 K개의 상한선 값을 계산하므로, 연산량은 $O(K)$가 된다. 실제 환경에서 출구 개수 K는 큰 값이 될 수 없으므로 큰 계산량이나 지연시간 없이 동적으로 최적의 출구를 선택할 수 있다.

Ⅳ. 시뮬레이션 결과

4.1 실험 환경

시뮬레이션은 Ryzen 7600X CPU와 4070ti GPU 환경에서 진행되었다. GPU 클럭 자원 상황은 총 3가지로 구성하였다. 외부 프로세스에서 GPU 자원을 사용하지 않아 추론에 전체 GPU 자원을 사용할 수 있는 풍족한 상황, 외부 프로세스에서 GPU 자원을 많이 사용하여 추론에 전체 GPU의 10% 자원만 사용할 수 있는 결핍 상황, 자원이 전체 GPU 자원 중간 정도 있는 적당한 상황으로 엔비디아 API를 사용하여 GPU 클럭을 제한하였다. 알고리즘을 적용하기 전 K=4인 미리 학습된 다중 출구 Faster R-CNN 모델에서 Init layer와 Layer1에서 이른 출구를 만들었을 때 테스트 mAP가 40.3과 45.2인 매우 낮은 결과 값이 나와 이른 출구를 Layer2 이후에 추가한 K=2인 이른 출구 Faster R-CNN 모델로 실험하였다. 테스트는 PASCAL VOC 2007 (15+5) 테스트 데이터셋 이미지 5000개로 진행하였다. 새로 실험할 데이터 셋의 분산은 사전에 테스트한 데이터 셋과 같다고 가정한다. 그러므로 알고리즘에서 사용할 각 이미지마다의 정확도 기댓값은 사전에 구한 각 출구 별 mAP에 랜덤하게 10% 오차를 적용한 값을 사용하였다. 사전에 구한 테스트 mAP는 다음과 같다. (출구 0: 63.81, 출구 1: 70.07)

표 1. Multi-exit Resnet101 backbone의 출구별 학습 결과. 아래는 비교를 위한 torchvision에서 제공하는 pretrained Resnet101의 mAP.
Table 1. mAP results from different exits of multi-exit ResNet1010 backbone and pretrained ResNet101 provided by torchvision

Exit	Dateset	mAP(ACC@5)
Exit 0	ImageNet 1k	**92.188%**
Exit 1	ImageNet 1k	**95.312%**
pretrained Resnet101.V1	ImageNet 1k	93.546%
pretrained Resnet101.V2	ImageNet 1k	95.78%

4.2 백본 학습 결과

Faster R-CNN에서 백본은 미리 학습(pretrain)을 시켜놓은 이후에 백본과 RPN, 검출기를 포함한 Faster R-CNN의 전체 모델을 PASCAL VOC 2007 (15+5) 데이터셋으로 학습을 시킨다. PASCAL VOC 2007(15+5) 데이터셋은 일상생활에서 접할 수 있는 대표적인 객체들을 탐지 및 분류하는 과제를 위한 데이터

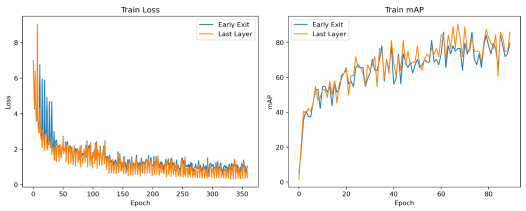


그림 4. Multi-exit Resnet101 백본 학습과정에서의 손실함수 결과와 mAP.
Fig. 4. Loss and mAP values of Multi-exit ResNet101 after training its backbone

셋으로, 따라서 백본 역시 일상생활에서 접할 수 있는 객체를 분류하는 과제 중에서도 더욱 다양한 종류를 분류하도록 ImageNet-1K 데이터셋으로 미리 학습한다. 학습 결과는 그림 4와 같다. 학습이 진행됨에 따라 출구별 손실함수의 결과는 줄어들었으며 정확도는 상승하여 수렴하였다. 또한 수치는 연산량이 비교적 적은 앞쪽의 분기에서 평균 정확도(mAP)는 92.188%로 뒤쪽의 분기의 mAP인 95.312%에 비해 작은 것을 확인할 수 있다. 또한 미리 학습시킨 백본이 사용함에 있어 실용성이 있는지에 대한 지표로 “torchvision”에서 제공하는 pretrained Resnet101 모델의 결과와 비교해 보았다. 이는 BranchyNet을 적용하지 않은 Faster R-CNN에서 백본으로 사용되는 모델로, 비교 결과 출구1의 결과는 버전1(V1)의 mAP 93.546%보다 높은 95.312%mAP를 보여주었다.

4.3 다중 출구 Faster R-CNN, MEF 결과

그림 5를 살펴보면 출구가 오직 앞쪽(Exit0)만 있는 모델과 뒤쪽(Exit1)만 있는 기존 모델의 정확도와 지연시간을 제안한 MEF 모델에서 P_{th} 값을 조절하며 모두 달성할 수 있는 것을 확인할 수 있다. 이뿐만이 아니라 정확도와 지연시간 사이의 중요도를 사용자의 취향에 따라 P_{th} 값을 조절하여 다양한 성능을 달성할 수 있다. 이때, P_{th} 값을 증가시킬수록 높은 정확도를 달성하기 위해 지연시간이 증가하고, P_{th} 값을 낮추면 약간의 정확도 성능 손실을 감수하면서 지연시간을 줄인다. Exit1 대신 Exit0에서 추론할 경우 mAP는 10%가 감소하지만 지연시간은 50% 이상이 줄어드는 것을 볼 수 있다.

이때 이른 출구 선택으로 감소하는 mAP 성능을 그림 6에서 확인할 수 있다. 그림 6은 학습에 사용된 PASCAL VOC 2007 (15+5) 데이터셋 중 임의 3장의 테스트 이미지의 객체 탐지 결과를 보여준다. 그림 5에서처럼 제안한 MEF의 이른 출구(Exit0)와 뒤쪽 출구(Exit1)에서의 mAP가 다중 출구를 사용하지 않은 기존 Faster R-CNN보다 낮은 값을 보이지만, 대부분 바운딩 박스의 크기 또는 위치의 차이가 있는 상황이 대부분이며 객체를 잘못 분류하는 경우는 그리 많지 않다. 잘못 판단한 바운딩 박스 역시 쇼파(Sofa)를 의자(Chair)로 판단하는 등 사람이 판단하기에 는 큰 차이를 보이지 않는다. 따라서, 이 정도의 성능 손실을 감수하면 50% 이상의 매우 낮은 연산량 및 지연시간을 제공할 수 있다.

또한, 실제로는 제안한 MEF가 P_{th} 값에 따라 Exit0과 Exit1을 동적으로 선택할 것이므로 그림 5와 같이 평균 mAP를 구하면 기존 Faster R-CNN 대비 성능 손실이 그리 크지 않을 것으로 기대된다. 반면 뒤쪽 출구(Exit1)의 경우에는 이른 출구(Exit0)에서 잘못 판단하는 사소한 요소들까지 높은 정확도로 판단을 내린다는 장점을 가진다. 따라서 사용자의 취향에 맞추어, 또는

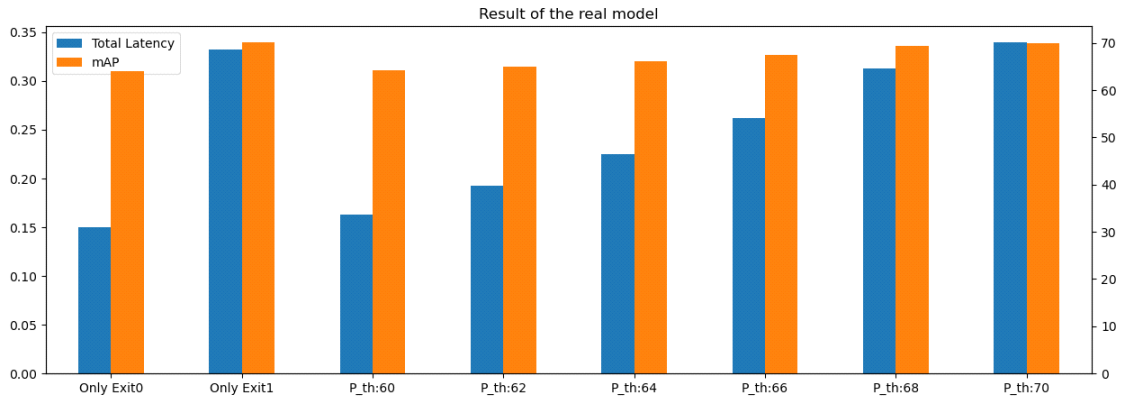


그림 5. MEF의 P_{th}를 68으로 설정할 때와 기존 Faster R-CNN에서 오직 앞쪽이나 뒤쪽 Exit에서 추론할 때와의 정확도와 Latency 비교
Fig. 5. mAP and latency comparisons with different P_{th} values and exits

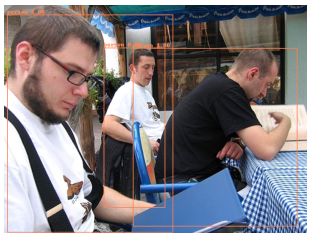
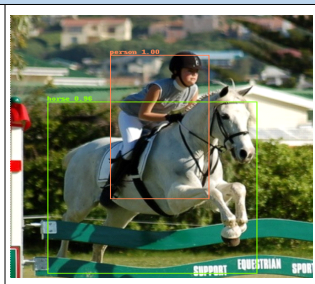
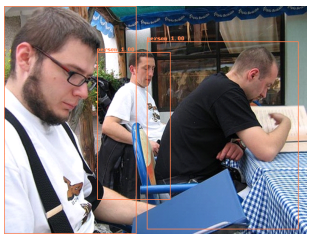
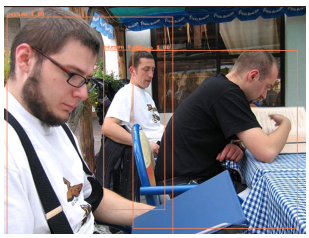
다중 출구 Faster R-CNN(MEF) Results				mAP
MEF (다중 출구 Faster R-CNN) Exit 0				63.81%
MEF (다중 출구 Faster R-CNN) Exit 1				70,07%
기존 Faster R-CNN				73.2%

그림 6. 최종적으로 학습이 끝난 다중 출구(Multi-Exit) Faster R-CNN, MEF의 다중 출구 별 PASCAL VOC 2007 (15+5) 데이터 셋 추론 결과 및 mAP. 아래는 기존의 다중 출구(Multi-Exit)를 사용하지 않은 Faster R-CNN의 추론 결과 및 mAP
Fig. 6. Inference and mAP results from different exits of Multi-exit Faster R-CNN

그림 7. MEF에서 실시간 GPU 클럭 자원 상황에 따른 각 이미지에서의 Exit와 Latency의 변화와 그에 따른 해당 이미지까지의 추론 Accuracy
Fig. 7. Selected exits, accuracy, and latency performances depending on the tiime-varying GPU clock resources

변동되는 연산 자원을 가진 상황에서 이러한 유동적인 출구 선택은 단일 모델을 가졌을 때에 비해 일정 수준 이상의 정확도를 보장하면서 연산량과 지연시간을 크게 낮출 수 있다.

마지막으로, 그림 7에서는 실제 GPU 클록 자원을 3단계로 변경했을 때 제안한 동적 출구 알고리즘의 작동 결과를 보여준다. 최대 GPU 자원을 사용할 때는 전체 모델의 연산량을 충분히 감당할 수 있어 지속적으로 깊은 출구(Exit1)에서 추론 결과를 얻어내고 있으며, 지연시간이 매우 안정적으로 낮고 평균 정확도도 높은 것을 볼 수 있다. 반면, GPU 클록 가용 자원이 10%로 변했을 때는, 단순히 이른 출구만을 선택하는 것이 아니라 테스트 이미지와 평균 mAP 성능에 따라서 이른 출구와 깊은 출구 중 하나를 동적으로 선택하는 것을 볼 수 있다. 평균 지연시간은 높아지고, 평균 정확도도 낮은 값들이 일부 찍히는 것을 볼 수 있다. GPU 클록 자원이 50% 이상 있는 경우 역시 이른 출구와 깊은 출구를 동적으로 선택하는데, 평균 정확도는 GPU 자원을 최대로 쓸 때와 크게 차이가 나지 않는 반면 평균 추론 시간은 크게 줄어든 것을 볼 수 있다.

V. 결 론

제안한 다중 출구(Multi-exit) Faster R-CNN, MEF는 시간에 따라 컴퓨팅 자원이 변하는 환경에서 동적으로 추론 지연시간과 기대 정확도 간의 트레이드 오프를 조절할 수 있는 다중 출력 신경망 기반의 모델과 동적 출구 선택 알고리즘을 제안하였다. 따라서 모델의 전체를 뒤쪽 출구(Exit1)를 위해 학습된 기존의 모델(Multi-Exit을 사용하지 않은 모델)에 비해, 본 연구의 모델은 이른 출구에서도 비슷한 특징 맵의 결과를 내는 방향으로 학습이 되었다. 이로 인해 기존의 모델(Multi-Exit을 사용하지 않은 모델)에 비해 본 연구의 MEF 모델의 뒤쪽 출구 결과는 동일한 연산량임에도 불구하고 조금은 낮은 정확도를 보여주지만 이른 출구는 기존 모델에 비해 매우 낮은 연산량과 빠른 지연 속도를 보여주었다.(50%이상의 빠른 속도)

또한 바운딩 박스를 활용하는 객체 탐지 모델이라는 점에서 조금의 정확도 차이는 바운딩 박스의 크기 또는 위치의 차이로 발생할 수 있어 기존의 모델과 비교하여 조금의 낮은 정확도는 사람이 판단하기에 구분하기 힘든 정확도의 차이였다. 최종적으로 컴퓨팅 자원이 변화하는 환경에서 이를 통해 가용 컴퓨팅 자원이 변하는 환경에서도 일정 수준 이상의 정확도를 보장하면서 추론 시간을 최소화할 수 있음을 보였다.

References

[1] S. Teerapittayanon, B. McDanel, and H. T. Kung, “BranchyNet: Fast inference via early exiting from deep neural networks,” in *Proc. 23rd Int. Conf. Pattern Recognition*, 2016. (https://doi.org/10.1109/ICPR.2016.7900006)

[2] H. Hu, D. Dey, M. Hebert, and J. A. Bagnell, “Learning anytime predictions in neural networks via adaptive loss balancing,” *33rd AAAI Conf. Artificial Intell., AAAI 2019, 31st Innovative Appl. Artificial Intell. Conf., IAAI 2019 and the 9th AAAI Symp. Educational Advances in Artificial Intell., EAAI,* 2019. (https://doi.org/10.1609/aaai.v33i01.33013812)

[3] M. J. Neely, “Stochastic network optimization with application to communication and queueing systems,” *Synthesis Lectures on Commun. Netw.*, 2010. (https://doi.org/10.2200/S00271ED1V01Y201006CNT007)

[4] M. Choi, W. J. Yun, S. B. Seok, S. Park, and J. Kim, “Joint delay-sensitive and power-efficient quality control of dynamic video streaming using adaptive super-resolution,” *IEEE Trans. Green Commun. and Netw.,* vol. 8, no. 1, pp. 103-117, 2024. (https://doi.org/10.1109/TGCN.2023.3319437)

[5] J. Kwak, Y. Kim, J. Lee, and S. Chong, “DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems,” *IEEE J. Sel. Areas in Commun.*, vol. 33, no. 12, pp. 2510-2523, 2015. (https://doi.org/10.1109/JSAC.2015.2478718)

[6] Y. Kim, H. W. Lee, and S. Chong, “Mobile computation offloading for application throughput fairness and energy efficiency,” *IEEE Trans. Wireless Commun.,* vol. 18, no. 1, pp. 3-19, 2019. (https://doi.org/10.1109/TWC.2018.2868679)

[7] J. A. Lim and Y. Kim, “Real-time DNN model partitioning for QoE enhancement in mobile vision applications,” *Annual IEEE Commun. Soc. Conf. Sensor, Mesh and Ad*

Hoc Commun. and Netw. Wkshps. 2022, Sep. 2022. (https://doi.org/10.1109/SECON55815.2022.9918600)

[8] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132-4150, 2019. (https://doi.org/10.1109/TCOMM.2019.2898573)

[9] H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, W. Zhu, "JALAD: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution," in *Proc. 2018 IEEE 24th ICPADS*, Dec. 2018. (https://doi.org/10.1109/PADSW.2018.8645013)

[10] R Girshick, J. Donahue, T Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. 2014 IEEE Conf. CVPR*, 2014. (https://doi.org/10.1109/CVPR.2014.81)

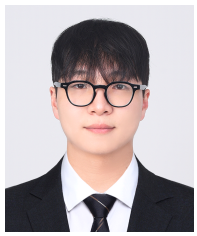
[11] R. Girshick, "Fast R-CNN," *2015 IEEE ICCV*, pp. 1440-1448, Santiago, Chile, 2015. (https://doi.org/10.1109/ICCV.2015.169)

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Analysis and Mach. Intell.*, vol. 39, no. 6, pp. 1137-1149, 2017. (https://doi.org/10.1109/TPAMI.2016.2577031)

[13] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Comput. Surv.* vol. 55, no. 5, pp. 1-30, May 2023. (https://doi.org/10.1145/3527155)

[14] H. Lee, K. Kim, and K. Lee, "Stochastic response time analysis for autonomous vehicle computing systems," *J. KIISE*, vol. 48, no. 5, pp. 486-492, May 2021.

김 현 수 (Hyeonsu Kim)



2019~현재 : 경희대학교 전자공학과 재학

<관심분야> 다출구 딥러닝 모델, 다중 에이전트 강화학습

[ORCID:0009-0007-6115-4459]

구 동 우 (Dongwoo Goo)

2019~현재 : 경희대학교 전자공학과 재학

<관심분야> 다출구 딥러닝 모델, 연합학습

[ORCID:0009-0004-5049-9652]

이 다 연 (Dayeon Lee)

2021~현재 : 경희대학교 전자공학과 재학

<관심분야> 컴퓨터 비전, 딥러닝

[ORCID:0009-0006-2197-4551]

최 민 석 (Minseok Choi)

2011 : B.S. degree, Korea Advanced Institute of Science and Technology (KAIST)

2013 : M.S. degree, KAIST

2018 : Ph.D. degree, KAIST

2019~2020 : Visiting Postdoc, University of Southern California.

2020~2022 : Assistant Professor, Jeju National University.

2022~Current : Assistant Professor with the Department of Electronics Engineering, Kyung Hee University, Yongin, South Korea.

<Research Interests> Wireless Caching Networks, Federated Learning, Reinforcement Learning, and Stochastic Network Optimization.

[ORCID:0000-0001-7027-1920]