공학석사학위논문

임베디드 시스템 상의 실시간 물체 인식을 위한 동적 네트워크 선택 기법

Real-time Object Detection Using Dynamic Network Selection on Embedded Systems

2018년 8월

서울대학교 대학원
컴퓨터공학부
강 정 현

요약

최근 딥러닝 기술이 비약적으로 발전함에 따라, 컴퓨터 비전 분야에 딥러닝 기술을 적용하여 큰 성공을 거두고 있다. 특히 이 중에서도 물체 인식의 경우 자율주행 자동차를 비롯한 다양한 임베디드 제품군에서 수요가 있어 각광을 받고 있다. 이에 따라 온-보드 물체 인식 기술은 최근 매우 중요한 이슈로 떠올랐다.

임베디드 시스템은 비용과 전력의 문제 때문에 컴퓨팅 파워가 적고, 메모리 크기가 작다는 제한점이 있다. 기존의 물체 인식 네트워크들의 경우 파라미터와 계산량이 많아 임베디드 시스템에 적합하지 않고, 따라서 이런 제약 조건 하에서 딥러닝을 이용한 물체 인식을 수행하기는 매우 어렵다.

본 논문은 딥러닝을 이용한 물체 인식 기술의 계산량을 줄이고, 임베디드 시스템 상에서 동작할 수 있도록 하는 동적 네트워크 선택 기법을 제안한다. 이미지와 영상 물체 인식 네트워크 각각에 대해 서로 다른 Q 학습 방법을 이용해 동적 네트워크 선택 기법을 학습시킴으로써 물체 인식 기술의 계산량을 줄였다. 그 결과 이미지, 영상 물체 인식의 두 가지 경우 각각 2.75, 2.5의 mAP 손실로 JETSON TX-2 상에서 2배의 속도 향상을 달성하였다.

주요어: 물체 인식, 임베디드 시스템, 강화학습

학번: 2016-29451

목차

표 목차

그림 목차

제 1장 서론

요즘 딥러닝 기술이 여러 분야에서 활용되고 있다. 그중에서도 컴퓨터 비전은 가장 활발하게 연구되고 있는 분야 중 하나로, 이미지 분류[1]와 이미지 물체 인식[2, 3], 영상 물체 인식[4, 5, 6, 7] 등의 문제에 딥러닝 기술을 적용하여 큰 성공을 거두었다.

물체 인식 기술(Object Detection)이란 이미지 내 물체의 종류와 위치를 분석하는 기술로, 이미지 정보를 픽셀로 받아 물체들의 위치를 경계 박스(Bounding Box)로 나타내는 기술이다. 딥러닝이 물체 인식 기술의 정확도를 높이는 데 큰 기여를 하였고, 이에 따라 물체 인식기술은 자율주행 자동차, 로봇 청소기를 비롯한 스마트홈 가전제품과 같이 다양한 임베디드 제품군에 적용될 수 있는 중요한 기술이 되었다. 이 중에서도 실시간으로 보드 상에서 물체 인식을 수행하는 기술의 개발이 최근 중요한 이슈이다. 왜냐하면 임베디드 제품들은 사용자와 직접 상호작용하므로 실시간 응답이 가능해야 하고, 사용자의 정보를 수집하고 가공하므로 프라이버시 문제가 있을 수 있기 때문이다. 그러나 딥러닝 어플리케이션을 보드 상에서 직접 실행하는 것은 아직 어려운 일이다.

임베디드 시스템에는 저전력 설계로 인한 몇 가지 제약 조건이 있다. 첫 번째는 컴퓨팅 파워이다. 대부분의 서버에서 딥러닝 어플리케이션을 수행할 때는 코어 수가 많은 GPU를 이용하여 가속하는데, 임베디드 시스템의 컴퓨팅 파워는 이에 비해 턱없이 모자라다. 따라서 딥러닝과 같은 계산 집약적인 어플리케이션의 경우 처리 속도가 현저히 떨어진다. 두 번째는 메모리이다. 임베디드 시스템의 경우 비교적 크기가 작은 메모리가 탑재된다. 또, 전력 소비를 줄이기 위해 메모리 사용을 최소화하도록 어플리케이션을 설계해야 한

다.

이런 제약 조건 아래에서 딥러닝을 이용한 물체 인식을 수행하기는 어렵다. 특히 고성능 물체 인식 네트워크들의 경우 메모리 사용량이 많아 별도의 전처리 과정 없이는 임베디드 시스템 위에서 동작하기 어려운 경우가 많고, 동작하더라도 속도가 느린 경우가 대부분이다.

따라서 본 논문은 동적 네트워크 선택 기법을 제안한다. 이 기술은 물체 인식 네트워크들의 속도-정확도 트레이드오프를 활용하여 딥러닝을 이용한 물체 인식 기술의 계산량을 줄이고, 임베디드 시스템 상에서 동작할 수 있도록 한다. 속도-정확도 트레이드오프를 활용하는 방법은 다음과 같다. 소형 모델의 빠른 속도를 최대한 유지하면서 대형 모델의 정확도 손실은 최소로 하는 것을 목표로 하여 속도가 빠르고 정확도는 낮은 소형 모델과, 반대로 속도가 느리고 정확도는 높은 대형 모델을 교차로 실행한다. 이때 실행할 네트워크를 결정하는 모듈을 네트워크 매니저라 하고, 이것을 강화학습을 통해 학습시킨다. 학습된 네트워크 매니저가 물체 인식 수행 시 동적으로 둘 중 어떤 네트워크를 동작시킬지 결정하게 된다.

본 논문에서는 이미지와 영상이라는 두 종류의 데이터에 적용되는 물체 인식 기술들에 대해 다룬다. 이미지 물체 인식 기술은 정지된 이미지 낱장 내에 존재하는 물체를 인식하는 기술로, 이미지 간 상관관계가 없다고 가정한다. 이에 반해 영상 물체 인식 기술은 영상 프레임 내에 존재하는 물체를 인식하는 기술로, 연속된 이미지 간 데이터 중복(data redundancy)이 있다. 실시간으로 수집한 카메라의 데이터를 이용한 물체 인식 기술이 여기에 해당한다. 움직이는 물체를 인식하기 때문에 이미지 물체 인식 기술과는 다른 어려움이 따른다.

본 논문에서는 단순 Q 학습을 이용한 동적 네트워크 선택 기법을 제안하고, 이를 통해 초소형/초저전력 이미지 물체 인식 기술을 구현한다. 또, 이 기술을 심화시킨 심층 Q 학습을 적용한 동적 네트워크 선택 기법을 제안하고, 이를 통해 임베디드 시스템에서 빠르게 실행됨과 동시에 움직이는 물체에 대해서도 비교적 고정확도를 유지하는 영상 물체 인식 기술을 구현한다.

임베디드 환경으로는 NVIDIA 사의 JETSON TX-2[8]를 선택하여 실험하였다. 결과적으로 이미지 물체 인식 기술의 경우 소형 모델의 계산량을 거의 그대로 유지하면서, mAP를 7.68 향상하였다. 이는 대형 모델 대비 2.75의 mAP 손실로 9.9배의 계산량을 감소시킨 것이다. 영상 물체 인식의 경우 2.5의 mAP 손실로 TX-2 상에서 대형 모델 대비 2배의 속도 향상을 달성하였다.

제 2장 관련 연구

2.1 CNN 계산 최적화

CNN(Convolutional Neural Network)의 동작 과정은 크게 학습(training)과 추론(inference)의 두 과정으로 나눌 수 있다. 학습은 이미 수집된 데이터와 정답을 이용하여 알맞은 동작을 하도록 네트워크를 가르치는 과정이고, 추론은 이렇게 학습된 네트워크를 이용하여 정답이 알려지지 않은 데이터에서 답을 도출하는 과정이다. 두 과정 모두 매우 많은 계산을 요구하므로, 계산량을 줄여야 할 필요가 있다. 그중에서도 추론 과정의 계산을 최적화하기 위한 다양한 연구들이 있다.

Park et al.(2015)은 이미지 분류에서 네트워크 선택 기법을 활용하여 추론 과정의 전력 소비를 최소화하였다[9]. 이 연구에서는 이미지 분류 시 간단한 네트워크로도 충분히 좋은 결과를 얻을 수 있는 경우가 대부분이고, 큰 네트워크가 필요한 경우는 소수에 불과하다는 점에 초점을 맞추었다. 대부분 계산량이 적은 네트워크(little)를 이용해 추론 과정을 진행하고, 몇몇 어려운 이미지의 경우에만 계산량이 많은 네트워크(big)를 사용함으로써 추론 시 소비 전력을 최소화하였다.

이때 가장 중요한 점은 어려운 이미지를 분류하는 기준이다. Park et al.은 분류기(classifier)의 분류별 출력 값 중 가장 높은 값과 두 번째로 높은 값의 차이를 스코어 마진이라고 하고, 이 스코어 마진이 클수록 분별력이 높다고 말한다. 모든 이미지의 추론 과정에 있어 계산량이 적은 네트워크를 먼저 실행하고, 스코어 마진이 특정 값을 넘으면 이 네트워크의 동작 결과가 신뢰할 만하다고 판단하였

다. 그러나 이런 스코어 마진 방식의 경우 이미지 분류에서는 잘 작동하지만, 물체 인식의 경우 물체의 분류 외에도 위치와 크기 등 고려해야 할 다른 점이 많아 적합하지 않다. 본 연구에서는 강화학습을 적용하여 이런 문제를 해결하였다.

Kim et al.(2016)은 기존에 학습된 네트워크를 압축하여 계산량을 줄이는 방법을 제시하였다[10]. 이 방법은 CNN의 커널 텐서에 Tucker 텐서 분해 기법[11]을 적용하는 것을 기본으로 한다. Tucker 텐서 분해 기법은 특이값 분해(Singular Value Decomposition)의 고차원 확장으로, 이 연구에서는 1x1 컨볼루션 층을 이용해 그 활용도를 높였다.

Kim et al.이 제시한 Tucker 텐서 분해 기법은 다음과 같다. 어떤 컨볼루션 층의 입력 채널이 256개이고, 3x3인 필터가 512개 있다고 가정하자. 이 경우 계산량은 256*512*3*3=1,179,648에 비례한다. 이때 랭크 R을 선택하여 텐서 분해 기법을 적용하면, 이 컨볼루션 층은 256*R*1*1, R*R*3*3, R*512*1*1의 세 개 층으로 나뉘게 된다. R=128일 때, 계산량은 33,024+ 147,456+ 65,536= 246,016에 비례하므로 약 4.8배 감소한다.

Tucker 텐서 분해 기법은 압축을 고려하지 않고 설계 및 학습된 CNN에도 적용이 가능하다는 장점이 있다. Kim et al.은 이 기법을 적용한 후 적은 양의 파인튜닝(finetuning) 만으로 정확도를 올릴 수 있음을 보였다. 그러나 랭크 선택에 따라 정확도가 회복되지 않는 경우가 있고, 최적의 랭크는 경험적으로 판단해야 한다는 단점이 있다. 본 논문에서는 이 기법을 적용하여 다양한 속도-정확도 조합을 가진 물체 인식 네트워크를 생성하였다.

2.2 기존 물체 인식 네트워크

고정밀 이미지 물체 인식 네트워크로 R-CNN 계열 네트워크들이 있다[12, 13, 2, 14]. 이 네트워크들은 크게 두 개의 단계에 걸쳐 물체를 인식한다. 첫 번째 단계에서는 이미지 상에 물체가 존재할만한 위치의 후보(Region of Interest)를 뽑는다. 두 번째 단계에서는 이렇게 선택된 위치 후보 각각에 대해 이미지 분류를 진행해 해당 위치에 어떤 분류의 물체가 있는지를 찾는다. 이런 방식의 네트워크들은 대부분 정확도는 높지만, 각각의 위치 후보들에 대해 분류 작업을 반복해야 하므로 느리다. R-FCN[14]의 경우 101층에 달하는 컨볼루션 층으로 구성되었으므로 파라미터 양 또한 매우 많고, 고성능 GPU에서도 1장의 이미지를 처리하는 데에 0.17초가 걸릴만큼 매우 느리다.

이런 R-CNN 계열 네트워크의 느린 실행 속도를 개선하기 위해 두 개의 단계를 한 번에 진행하는 네트워크들이 등장했다[3, 15]. YOLO[3]는 이미지를 그리드로 나누어, 추론 시 각 그리드 셀에 대해 물체 분류와 위치 추정을 동시에 진행한다. 물체 위치 추정을 위해 앵커 박스(anchor box)를 이용하는데, 앵커 박스란 물체를 쉽게 찾기 위해 대략적으로 물체 모양을 미리 정해둔 가이드라인을 의미한다. 하나의 그리드 셀에 대해 3~7개의 앵커 박스를 적용하여 물체를 찾는데, 이때 앵커 박스는 학습 데이터셋을 K-means 클러스터링으로 분석하여 미리 생성한다. 그리드를 기반으로 하므로 그리드 셀보다 작은 물체는 찾을 수 없다는 단점이 있다.

이미지 물체 인식은 이미 높은 수준의 정확도에 도달했으나, 실시간으로 동작하기에 여전히 느리고, 소형 물체의 경우 정확도가 떨어지는 등 위에서 언급된 문제점들 때문에 여전히 활발히 연구되는 분야이다. 특성(feature) 피라미드를 이용해 다양한 물체 크기에 대해 정확도를 올리거나[16], 파라미터 양을 획기적으로 줄이는 등[17]의 다양한 연구가 진행 중이다.

영상 물체 인식의 경우 이미지 물체 인식과는 다른 특성이 있다. 물체가 움직이는 경우 영상 프레임이 흐릿하게 표현되는 블러 현상이 나타난다. 이와 같은 문제 때문에 이미지 물체 인식 기술만을 사용해 영상 물체 인식을 수행하기는 어렵다. 따라서 여러 연구에서 프레임 간 데이터 중복을 활용한 방법을 더해 정확도를 높였다.

트래킹(tracking)은 데이터 중복을 이용한 대표적인 기법이다. 이 기법은 연속된 프레임 간의 관계를 이용하여 어떤 물체가 영상 내에서 움직이는 궤적을 추적한다. 느린 속도가 가장 큰 문제점이었으나, 뉴럴 네트워크의 적용으로 속도가 많이 향상되었다[4]. 그러나 여전히 영상 중간에 새롭게 등장하는 물체를 찾지 못한다는 점에서 문제가 있다. 이미지 물체 인식 기술에 트래킹의 특성을 결합해 둘의 장점을 모두 취한 연구도 있다. Kang et al.(2017)은 이미지 물체 인식 기술인 CNN 네트워크에 데이터의 연속성을 활용할 수 있는 LSTM(Long Short-Term Memory)을 더하여 정밀한 결과를 낼 수 있도록 하였다[5].

옵티컬 플로우(optical flow)를 활용한 연구도 많이 찾아볼 수 있다. 옵티컬 플로우[18]는 화면 내 픽셀들의 변화를 추적하여 벡터로 표현한 것이다. Hetang et al.(2016)과 Zhu et al.(2017)의 연구에서는 정해진 키 프레임에서만 CNN을 실행하고, 나머지 프레임에서는 키 프레임과의 옵티컬 플로우만 계산해 그에 따라 결과를 조절하는 식으로 영상 물체 인식의 효율성을 높였다[6, 7]. 하지만 이런 연구들의 경우 키 프레임을 정적으로 정하므로, 새로운 물체가 등장하거나 물체가 빠르게 움직이는 경우 등의 상황을 판단할 수 없다는 아쉬움이 있다.

2.3 Q 학습

강화학습은 에이전트(agent)가 환경(environment)과 상호작용을 하면서, 환경으로부터 최대 보상을 얻는 방법을 학습하는 과정을 말한다. 매 반복마다 에이전트는 환경으로부터 현재 상태 S를 받고 이에 따라 최대 보상을 얻기 위한 행동 A을 취한다. 이때, 어떤 행동을 취할 것인지를 결정하는 기준을 정책(policy)이라고 한다. 에이전트가 정책에 따라 행동을 취하고 나면 다시 환경으로부터 바뀐 상태 S'과 행동에 따른 보상 R을 받는다.

Q 학습은 값-기반(value-based) 강화학습의 한 종류이다. Q 학습에서는 Q 함수를 학습한다. Q 함수는 현재 상태에서 어떤 행동을 취했을 때 받을 수 있는 미래의 보상을 예측하는 함수이다. Q 러닝 에이전트는 학습 과정 동안 탐험(exploration)과 유지(exploitation)를 반복하며 각각의 (S, A) 쌍에 따라 최적의 Q 값을 출력하는 Q 함수를 학습한다. 최적의 Q 함수를 구한다면, 에이전트는 매번 Q 값이 최대가 되는 행동을 선택하는 것으로 최선의 정책을 얻을 수 있다.

Q 함수는 (S, A) 쌍에 따라 Q 값을 출력하므로 테이블을 이용해 근사할 수 있다. 본 논문에서는 테이블 기반 Q 학습을 단순 Q 학습이라 지칭하기로 한다. 한편, 테이블 대신 뉴럴 네트워크로 Q 함수를 근사하는 방법이 있다. 이를 심층 Q 학습(Deep Q Learning)이라고 한다. 이때, 뉴럴 네트워크로는 MLP(multi-layer perceptron), CNN, LSTM 등 다양한 네트워크가 사용되는데, Mnih et al.(2013)은 CNN을 사용한 심층 Q 학습으로 아타리 게임을 별도의 다른 정보 없이 시각 데이터, 즉 화면의 픽셀 값만 이용하여 자동 플레이하는 데에 성공했다[19]. Mnih et al.에 따르면 보통의 뉴럴 네트워크가 비용 함수의 값이 감소하는 것을 통해 학습 진행도를 판단할 수 있는 데에 반해, 심층 Q 학습의 경우 평균 Q

값이 증가하는 것을 통해 학습 진행도를 대략적으로 판단할 수 있다고 한다.

제 3 장 초소형 물체 인식 네트워크 생성

실험 결과 기존의 물체 인식 네트워크 중 TX-2 상에서 가장 빠른 추론 속도를 내는 네트워크는 Tiny YOLO(Fast YOLO)였다[20, 3]. Tiny YOLO는 YOLOv2와 비슷한 구성이지만, 그보다 더 적은 컨볼루션 층을 사용해 구성된 네트워크이다. 그러나 이 또한 Pascal VOC 데이터셋 기준 24FPS로 실시간 물체 인식에는 못 미치는 속도이므로, 더 빠른 물체 인식 네트워크를 생성해야 했다. 또, 속도-정확도의 트레이드오프를 활용한 실험을 진행하기 위해서는 다양한 속도-정확도 조합의 네트워크들이 필요하다.

이와 같은 필요성에 따라 베이스 모델을 YOLOv2로 정하였다[3]. YOLOv2의 경우 완전 컨볼루션 네트워크로, 완전 연결층이 존재하지 않기 때문에 하나의 학습된 웨이트를 가지고 재학습 없이 다양한 크기의 입력을 처리할 수 있다. 즉, 한 번의 학습으로 다른 입력 크기를 처리하는 여러 개의 네트워크를 얻는 효과를 낼 수 있다. 또 완전 컨볼루션 네트워크이기 때문에 모든 층에 Tucker 텐서 분해 기법을 적용할 수 있다.

이미지 물체 인식 기술의 경우 Pascal VOC 데이터셋을 기반으로 학습된 Tiny YOLO를 대형 모델로 하였고, 소형 네트워크 생성을 위해 이 대형 모델에 Tucker 텐서 분해 기법을 적용하였다. 그림 1-1, 1-2는 각각 Tiny YOLO의 conv3(입력에서 3번째 컨볼루션 층), conv14(입력에서 14번째, 출력에서 2번째 컨볼루션 층)에 Tucker 텐서 분해 기법을 적용한 결과이다. 가로축은 계산량의 감소율을, 세로축은 그에 따른 mAP의 변화를 보여준다. conv3의 경우 계산량이 감소함에 따라 mAP가 급격하게 줄어들지만, conv14의 경우 계산량이 50배 이상 감소해도 mAP 변화가 크지 않다. 이를 통해 입력과 가까운 층의 계산은 최대한 보존해야 하고, 출력과

가까운 층일수록 분해를 통해 계산량을 많이 줄일 수 있음을 알 수 있다.

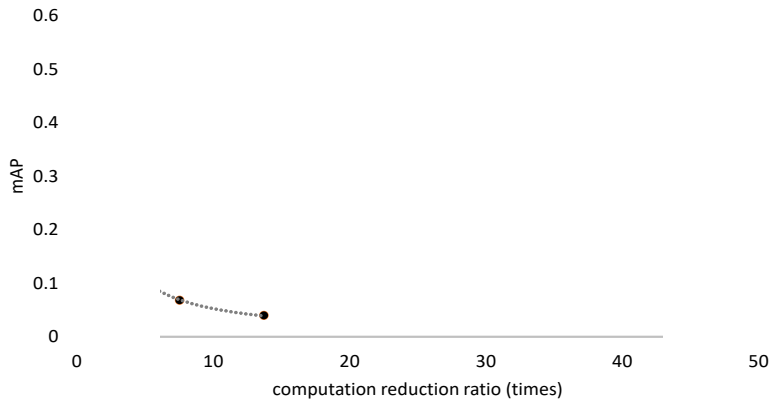

그림 1-1 Tiny YOLO의 conv3에 Tucker 텐서 분해를 적용한 그래프

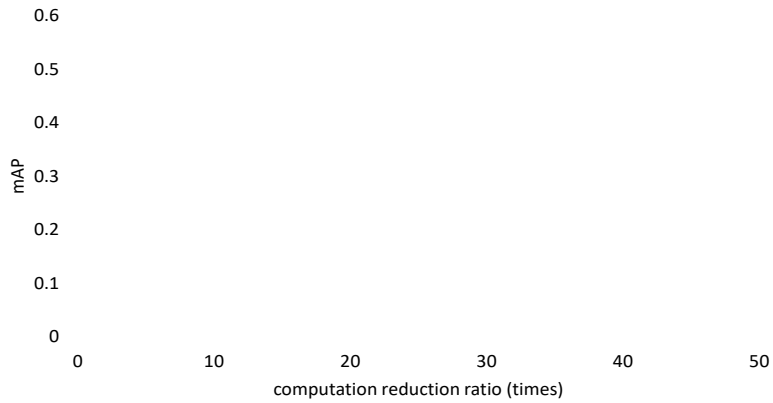

그림 1-2 Tiny YOLO의 conv14에 Tucker 텐서 분해를 적용한 그래프

영상 물체 인식 기술의 경우에는 YOLOv2 기반의 네트워크들을 ImageNet VID 데이터셋에 맞게 수정하고, 재학습시켰다. 그러나 Pascal VOC 데이터셋의 경우와는 다르게 ImageNet VID 데이터셋의 경우 Tiny YOLO의 정확도가 매우 낮았다. 마찬가지로 Tiny YOLO나 YOLO v2에 Tucker 텐서 분해를 적용하고 파인튜닝했을 때에도 정확도가 거의 회복되지 않았다. 실용성 있는 기술 개발을 위해서 정확도가 어느 정도 이상 보장되어야 하므로, Tucker 텐서 분해를 적용하는 대신 기존의 Tiny YOLO와 YOLOv2에 다양한 크기의 입력을 주어 사용하였다.

제 4장 강화 학습 기반 동적 네트워크 선택 기법

4.1 단순 Q 학습을 이용한 동적 네트워크 선택 기법

본 논문에서는 초소형/초저전력 이미지 물체 인식 기술을 위해 단순 Q 학습을 이용하였다. 적은 양의 정확도 손실로 최대한 계산량을 줄이는 것이 목표이므로, 계산 오버헤드가 적은 단순 Q 학습을 이용해 네트워크 매니저를 구성하였다. 단순 Q 학습 에이전트의 경우 한번 학습이 완료된 후에는 테이블 엔트리에 접근하는 단순한 동작만 하므로 뉴럴 네트워크 대비 계산 오버헤드가 거의 없다고 볼 수 있다.

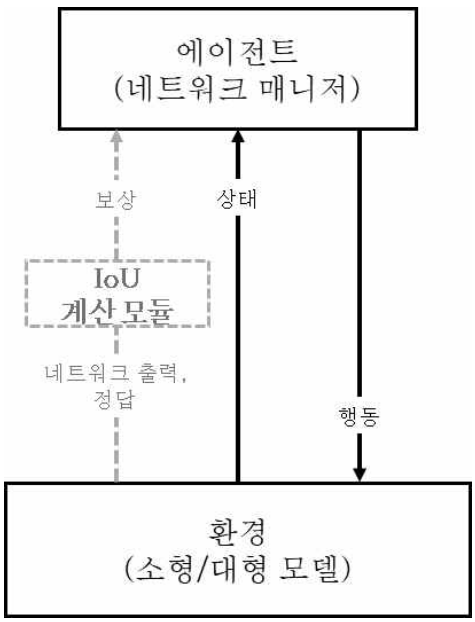

그림 2 단순 Q 학습의 구조

그림 2는 단순 Q 학습의 구성을 나타낸다. 환경은 이미지 물체 인식을 수행할 때 매번 소형 모델을 실행하고, 그 결과를 상태로 변환하여 네트워크 매니저에 전달한다. 이 상태를 보고 네트워크 매니저는 대형 모델을 수행할지 말지를 결정한다. 환경은 그 행동의 결과에 따라 보상을 주고, 네트워크 매니저는 이를 통해 Q 함수를 학습한다.

물체 인식 네트워크의 출력은 경계 박스들과 그 박스들의 분류기 결과값이다. 분류기 결과값이 가장 높은 분류를 해당 경계 박스 안에 들어있는 물체의 분류로 판단하고, 이때의 분류기 결과값을 스코어라 하자. 이때 스코어는 정수가 아니므로 Q 테이블의 엔트리 값으로 쓰기에 적절하지 않다. 이 때문에 경계 박스들 중 상위 스코어 5개의 평균을 오프셋 단위로 묶어(binning) 정수로 만든다. 이 값에, 이전 N개 이미지 중 대형 모델을 실행한 이미지의 개수를 짝지어 두 정수의 쌍으로 상태 값을 결정한다.

행동의 경우 대형 모델을 수행할지(1), 하지 않을지(0)가 되며, 보상의 경우 매번 선택한 행동을 통해 얻은 경계 박스와 정답의 경계 박스의 IoU(Intersection over Union)를 계산해 얻는다. 이때, 대부분의 경우 대형 모델의 보상이 크다. 하지만 대형 모델은 계산량이 많기 때문에 보상에서 그 점이 반영되어야 한다. 이에 따라 페널티 값(<1)을 정하고, 대형 모델의 경우 얻는 보상에 이 페널티 값를 곱해주었다. 오직 학습 시에만 보상이 중요하므로, 학습이 끝난 이후에는 보상을 계산할 필요가 없다. 따라서 네트워크 매니저는 정답을 모르는 데이터에 대해서도 동적으로 행동을 선택할 수 있게 된다. 이때는 테이블에 저장된 Q 값을 이용해 행동을 선택하기만 하고, 그 값을 갱신하지는 않는다.

4.2 심층 Q 학습을 이용한 동적 네트워크 선택 기법

임베디드 시스템에서 물체 인식 기술을 필요로 하는 경우는 대체로 자율 주행 자동차와 같이 시스템 내 카메라로 수집한 정보를 실시간으로 분석하기 위함이다. 이런 기술은 이미지 인식보다는 영상 인식으로 분류할 수 있다. 그러나 영상의 경우 움직이는 물체가 흐릿하게 잡히는 블러 현상 때문에 물체 인식이 더 어렵다.

4.1의 경우 초저전력 시스템을 구축하기에는 효율적이지만, 베이스 모델을 고정밀 모델로 하면 학습이 되지 않는 문제가 있었다. 따라서 영상 물체 인식과 같이 고정밀을 요구하는 환경에 적합하지 않다. 이를 위해서는 데이터 중복의 특성을 활용할 수 있는 기법이 필요하다. 심층 Q 학습의 경우 계산량 오버헤드는 늘지만, 4.1의 경우보다 훨씬 정교하게 Q 함수를 근사할 수 있어 알맞은 방법이다. 심층 Q 학습의 Q 함수의 경우 3개의 은닉층으로 구성된 MLP를 이용하여 근사하였다.

Q 함수의 근사 방법을 제외한 나머지는 4.1과 유사하게 설정하였으나, 심층 Q 학습의 경우 4.1보다 더 복잡한 상태를 다루기 충분하므로 이를 수정하였다. 상태에 물체의 위치 및 크기 정보를 반영할 수 있도록 보강하였다. 그림 3과 같이 이미지를 5개의 구역으로 나누어, 각각의 구역에서 가장 스코어가 높은 박스의 정보를 상태 정보에 넣었다. 그리고 4.1과 유사하게 현재 영상에서 몇 번이나 대형 모델을 실행했는지 또한 상태 정보에 넣었다. 이를 통해 구성된 상태는 그림 4와 같다.

행동의 경우 4.1과 큰 차이가 있다. 심층 Q 학습의 행동은 현재 프레임의 물체 인식 결과를 통해 다음 프레임에 실행할 네트워크를 정하는 것이다. 이는 매번 소형 모델을 실행하는 계산 오버헤드를 줄이기 위해서다. 데이터 중복의 특성에 따라 어떤 프레임에서 소형 모델로 찾아낸 결과가 비교적 정확했다면, 그에 인접한 프레임에서

도 그럴 것이라는 가정이 반영된 것이다.

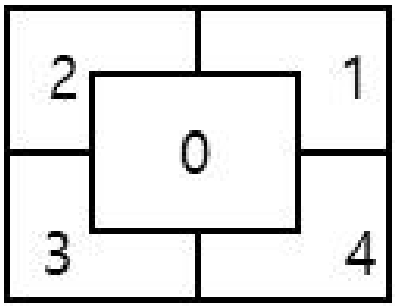

그림 3 이미지 구역

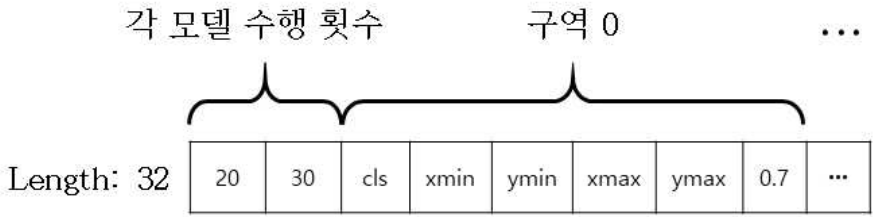

그림 4 심층 Q 학습의 상태 구성

보상은 4.1과 유사하게 페널티 값을 이용하였다. 그러나 심층 Q 학습에서는 추가 보상을 주었다. 마지막 프레임에서 영상 내 실행된 소형 모델과 대형 모델의 비율을 계산해, 만약 원하는 비율을 만족하면 양의 보상을, 만족하지 못하면 음의 보상을 추가로 주었다. 이를 통해 소형 모델과 대형 모델의 비율을 조절하고자 하였다.

관찰은 매번 임의의 연속된 영상 20 프레임을 골라 진행하였고, 이때 관찰된 결과를 재생 메모리에 저장하였다. 그 후 학습 시 재생 메모리에 저장된 프레임 중 연속된 프레임 쌍을 임의로 배치 크기만큼 골라 학습하였다. 그림 5는 이렇게 구성된 심층 Q 학습의 전체 구조를 나타낸다.

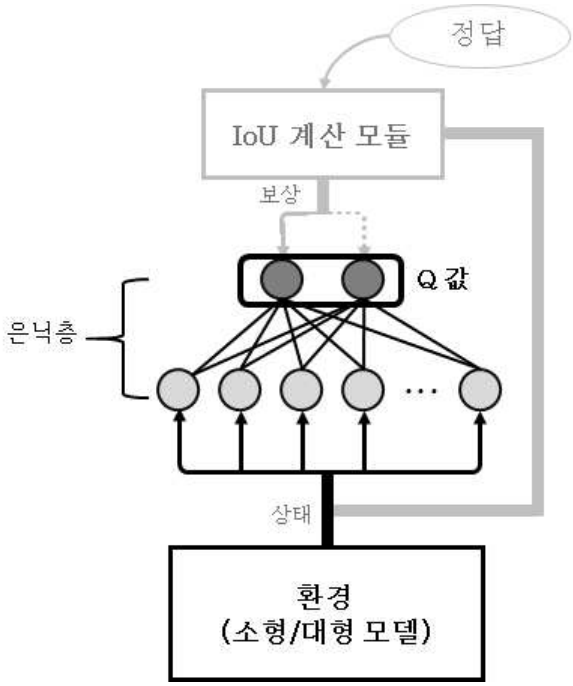

그림 5 심층 Q 학습의 구조

제 5 장 실험 및 결과

실험 과정 중 CNN 학습은 4개의 GTX 1080 TI를 이용하여 진행하였고, MLP 학습은 1개의 GTX 1070 TI를 이용하였다. FPS 측정은 GTX Titan X 1개와 JETSON TX-2에서 진행하였다.

5.1 물체 인식 네트워크 학습 결과

표 1은 Pascal VOC 데이터셋 기반 Tiny YOLO에 Tucker 텐서 분해를 적용한 결과이다. 번호가 큰 네트워크일수록 입력에 가까운 컨볼루션 층까지 분해한 네트워크이고, 12번 네트워크의 경우 conv14부터 conv4까지 전부 Tucker 텐서 분해가 적용되었다. 이 네트워크를 최종적으로 4.1의 단순 Q 학습에서 소형 모델로 사용하였다.

표 2는 ImageNet VID 데이터셋 기반 네트워크 학습의 결과이다. 기존 YOLOv2는 ImageNet VID 데이터셋 기반으로 학습된 결과가 공개되어 있지 않으므로, 앵커 박스를 구하기 위해 직접 K-means 클러스터링을 적용하였다. 그림 6은 이렇게 구한 앵커 박스를 나타낸다. 학습은 Zhu et al.(2017)의 방법을 활용해 매 미니배치마다 ImageNet VID 데이터와 ImageNet DET데이터가 2:1의 비율로 구성되도록 하였다[7]. 그림 7은 YOLO v2의 ImageNet VID 학습 로그이다.

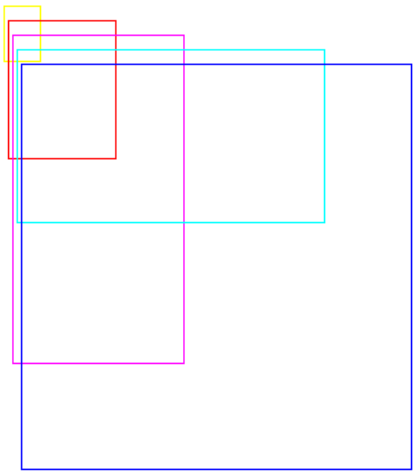

그림 6 ImageNet VID 데이터의
앵커 박스

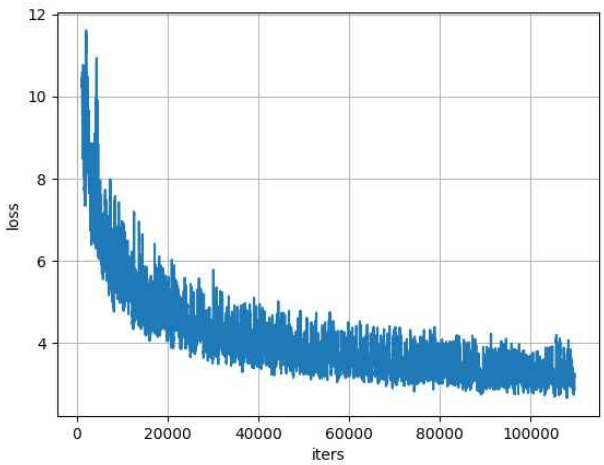

그림 7 YOLO v2의 학습 로그

번호	계산량(비율)	mAP	mAP 변화량
0	1	54.06	+ 0.00
1	0.316	55.45	+ 1.39
2	0.193	52.02	-2.04
3	0.139	49.81	-4.25
4	0.129	49.30	-4.76
5	0.127	48.44	-5.62
6	0.124	48.15	-5.91
7	0.121	46.73	-7.33
8	0.111	45.83	-8.23
9	0.107	45.67	-8.39
10	0.102	45.66	-8.40
11	0.100	43.67	-10.39
12	**0.098814**	**43.63**	-10.43

표 1 Pascal VOC 데이터셋 기반 Tiny YOLO Tucker 텐서 분해 적용 결과

	입력	계산량 (비율)	파라미터양 (MB)	FPS(titan X)	FPS(TX2)	mAP
YOLOv2	**544**	**1**	**193**	**44.17**	**4.10**	**67.9**
YOLOv2	416	0.584	193	69.18	8.02	65
Tiny YOLO	416	0.219	60.7	191.29	23.75	46.2
Tiny YOLO	**320**	**0.129**	**60.7**	**240.18**	**31.53**	**42**
Tiny YOLO	288	0.105	60.7	256.71	33.05	38.5
Tiny YOLO -tucker (2 layers)	416	0.060	7.15	218.81	35.24	30.2
Tiny YOLO -tucker (1 layer)	416	0.088	8.39	252.21	37.17	26.9

표 2 ImageNet VID 데이터셋 기반 네트워크 생성 결과

5.2 단순 Q 학습 실험 결과

단순 Q 학습 시에 중요한 부분은 오프셋, N값, 페널티를 정하는 것이다. 이는 휴리스틱을 이용하여 정하였다. 먼저 오프셋과 N값의 경우, 값이 바뀌면 Q 테이블의 엔트리 개수가 변화한다. 이를 통해 각각의 값을 바꾸며 학습을 반복하면서, Q 값이 비교적 고르게 분포하는 오프셋 값을 선택하였다. 페널티 값을 정하는 것은 학습에서 가장 중요한 과정 중 하나였으나, 학습을 완료하기 전에는 적합한지 그 정도를 예측하기가 어렵다는 문제가 있었다.

이렇게 구성한 단순 Q 학습의 경우 매번 소형 모델을 무조건 한번은 실행해야 하므로 소형 모델의 계산량이 대형 모델 대비 아주 작을 때 효과적이다. 따라서 정확도 차이가 비교적 크더라도 대형 모델을 표 1의 0번 모델로, 소형 모델을 표 1의 12번 모델로 선택하였다.

학습한 네트워크 매니저를 이용해 Pascal VOC 2007 테스트셋에 대해 물체 인식을 진행하였다. 이 결과 대형 모델은 0.2%만 사용되었고, 계산량은 대형 모델 대비 9.9배 적었다. mAP는 51.31로 약 2.75 감소했다. 즉, 소형 모델 계산량의 2%를 추가 사용해 mAP는 소형 모델 대비 7.68 향상하였다. 이 경우 오프셋은 0.01, N은 4, 페널티 값은 0.45를 사용하였다.

5.3 심층 Q 학습 실험 결과

심층 Q 학습의 경우에도 학습에 영향을 미치는 인자들을 여러 개 관찰할 수 있었다. 따라서 이들 인자를 바꾸어가며 학습을 반복하였다. 이때, 대형 모델의 적절한 비율을 전체의 10%~50%로 설정하고 진행하였다.

첫 번째 인자는 할인율(discount factor)이다. 대형 모델의 비율을 적절하게 유지하면 마지막 영상 프레임에 부가 보상을 받게 되기 때문에, 미래의 보상(속도)을 얼마나 중요하게 여길지가 중요한 요소였다.

두 번째 인자는 페널티이다. 단순 Q 학습과는 다르게 페널티 값을 0.6~0.9 사이로 줬을 때 학습이 진행되었다. 페널티 값의 크기가 커짐에 따라 대형 모델의 비율이 증가하면, 이에 따라 mAP는 선형적으로 증가한다고 생각했으나, 실험 결과 그림 8과 같이 대형 모델 수행률과 mAP의 관계는 선형적이지 않음을 알 수 있었다. 이는 환경이 영상 내 마지막 프레임의 보상을 이용하여 대형 모델의 비율을 보정하는 과정의 영향 때문으로 추측할 수 있다.

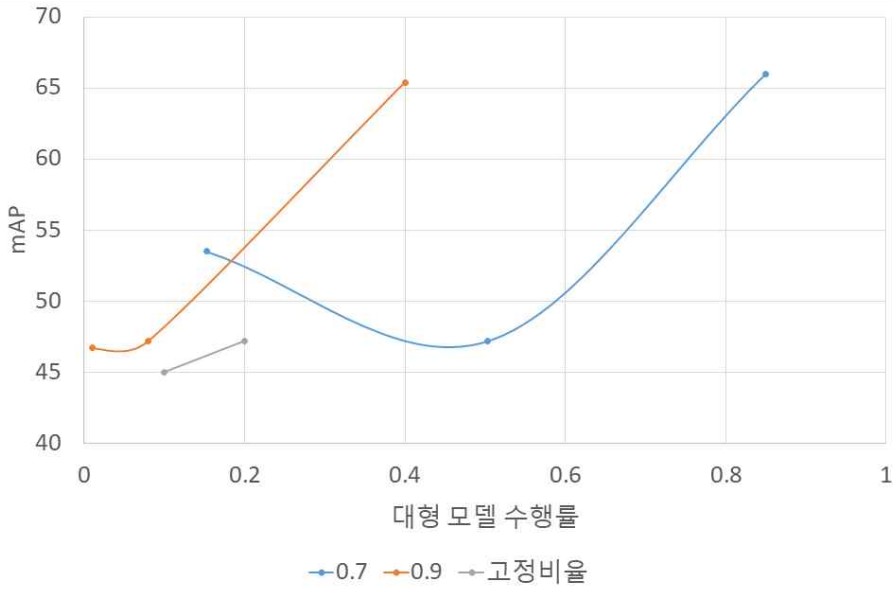

그림 8 대형 모델 수행률에 따른 mAP의 변화

세 번째 인자는 학습 반복 수와 학습률이다. 심층 Q 네트워크는 이 인자들에 민감하게 반응해서, 둘 중 하나만 바뀌어도 학습이 발산하였다. 그래서 적절한 값을 찾은 후 고정적으로 적용하였다. 이 때 반복 수는 16만번이고, 매 반복마다 1번의 관찰을 진행하고 크기 100의 미니배치를 학습하였다. 학습률은 10^{-5}을 사용하였다.

표 3은 심층 Q 학습의 주요 학습 결과를 정리한 도표이다. 할인율 0.7일 때, 페널티 값이 0.7에서 0.75로 증가하여도 mAP가 오히려 감소하는 모습을 볼 수 있다.

할인율	페널티값	mAP	FPS (TX-2)	FPS (Titan-X)	대형모델 수행률	계산량 (상대)
0.7	0.7	53.5	15.14	139.34	0.15	0.26
0.7	0.75	47.2	7.12	73.23	0.50	0.57
0.7	0.8	66	4.67	49.84	0.85	0.87
0.9	0.6	46.7	27.90	219.87	0.01	0.14
0.9	0.65	47.2	19.72	171.20	0.08	0.20
0.9	**0.7**	**65.4**	**8.43**	**85.09**	**0.40**	**0.48**
tiny 320 (소형 모델)		42	31.53	240.19		0.13
yolo 544 (대형 모델)		67.9	4.01	44.18		1.00

표 3 심층 Q 학습 결과

그림 9는 이 두 경우 학습에 따른 평균 Q 값의 변화를 그래프로 나타낸 것이다. 페널티 값이 0.75인 경우 평균 Q 값이 점차 감소하므로, 학습이 제대로 이루어지지 않았음을 알 수 있다.

심층 Q 네트워크의 계산량 오버헤드는 대형 모델(YOLO v2 544) 대비 2.7%, 소형 모델(Tiny YOLO 320) 대비 8.3%였다. 표 3의

FPS 항목은 심층 Q 네트워크의 수행시간을 포함한 결과이다.

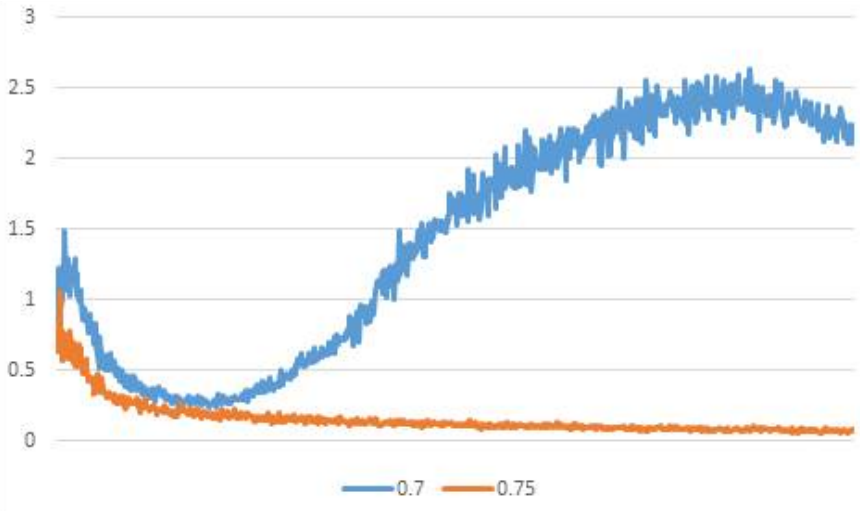

그림 9 학습에 따른 평균 Q 값의 변화

제 6장 결론

본 논문에서는 이미지 및 영상에 사용할 수 있는 임베디드 시스템용 물체 인식 기술을 제안하였다. 이 기술은 동적으로 네트워크를 선택하는 기술로, 매 이미지를 인식할 때에 어떤 네트워크를 사용하는 것이 전력 소비 및 속도 면에서 효율적인지를 선택하는 기술이다.

본 논문에서는 네트워크 선택 기법을 개발하기 위하여 다양한 물체 인식 네트워크를 생성하였다. 또한, 동적으로 네트워크를 선택할 수 있도록 강화학습을 통해 네트워크 매니저를 학습시켰다. 이 결과 이미지 물체 인식의 경우 대형 모델 대비 2.75의 mAP 손실로 계산량을 9.9배 줄일 수 있었다. 이는 소형 모델 대비 계산량을 2%만 추가로 사용하고, mAP를 7.68가량 개선한 것이다. 영상 물체 인식의 경우 대형 모델 대비 2.5의 mAP 손실로 계산량을 3.12배 줄이고, TX-2 상에서 속도를 2배 향상하였다.

본 연구는 다음과 같은 점에서 향후 개선될 여지가 있다. 이미지 물체 인식의 경우 Tucker 텐서 분해 기법을 적용하여서, 계산량은 적으나 네트워크의 깊이가 깊다. 적은 계산량의 컨볼루션 층이 깊게 쌓여있어 계산량 대비 속도 향상이 크지 않다는 문제가 있다.

영상 물체 인식의 경우 학습 과정의 자동화가 필요하다. 휴리스틱을 이용하여 학습 인자를 찾았으므로 모든 인자에 대해 탐색할 수 없었던 점도 개선점이다. 또, TX-2상에서 2배의 속도 향상이 있었지만 여전히 8 FPS에 그쳤으므로, 더 큰 속도 향상이 요구된다.

참고 문헌

[1] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25, pages 1106-1114, 2012.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[3] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In CVPR, 2017.

[4] David Held, Sebastian Thrun, and Silvio Savarese. Learning to Track at 100 FPS with Deep Regression Networks. In ECCV, 2016.

[5] Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object Detection in Videos with Tubelet Proposal Networks. In CVPR, 2017.

[6] Congrui Hetang, Hongwei Qin, Shaohui Liu, and Junjie Yan. Impression Network for Video Object Detection. arXiv preprint arXiv:1612.01051, 2016.

[7] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep Feature Flow for Video Recognition. In CVPR, 2017.

[8] https://www.nvidia.com/ko-kr/autonomous-machines/embedded-systems-dev-kits-modules/

[9] Eunhyeok Park , Dongyoung Kim , Soobeom Kim , Yong-Deok Kim , Gunhee Kim , Sungroh Yoon , Sungjoo Yoo, Big/little deep neural network for ultra low power inference, Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis, p.124-132, October 04-09, 2015, Amsterdam, The Netherlands

[10] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang and Dongjun Shin. Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications. In ICLR, 2016.

[11] Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. Psychometrika, 31(3): 279–311, 1966.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[13] Ross Girshick. Fast R-CNN. In ICCV, 2015.

[14] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In NIPS, 2016.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. CoRR, abs/1512.02325, 2015.

[16] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. arXiv preprint arXiv:1612.03144, 2016.

[17] Bichen Wu, Alvin Wan, Forrest Iandola, Peter H. Jin, and Kurt Keutzer. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. arXiv preprint arXiv:1612.01051, 2017.

[18] David J. Fleet and Yair Weiss. Optical Flow Estimation. Springer, 2005

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In NIPS Deep Learning Workshop, 2013.

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640, 2015.

Abstact

Real-time Object Detection Using Dynamic Network Selection on Embedded Systems

Jeonghyun Kang

Department of Computer Science and Engineering

The Graduate School

Seoul National University

Recently, the development of deep learning has made great success in computer vision tasks such as image classification and object detection. Especially, object detection is noticed because of its various applications on embedded systems.

Embedded systems have limited computing power and memory size due to cost and power issues. It is very difficult to perform object detection in real-time under these constraints. Conventional object detection networks are not suitable for

embedded systems because of its huge of parameters and computational complexity.

This paper present dynamic network selection techniques that reduce the computational complexity of object detection network and enable it to operate on embedded systems. By using various Q learning methods, dynamic network selection techniques can be applied for both still images and videos.

As a result, the proposed object detection method on images reduces the computation by 9.9 times compared to the baseline, with mAP loss of 2.75. In case of videos, this method doubles the speed compared to the baseline on JETSON TX-2, with a loss of mAP of 2.5.