CUDA C 기반 SqueezeNet 을 이용한 영상 분할

전세윤*, 왕진영*, 이상환**
*한양대학교 융합기계공학과
**한양대학교 기계공학부
e-mail : jsyu3799@hanyang.ac.kr

Image Segmentation Using SqueezeNet based on CUDA C

Sae-Yun Jeon*, Jin-Yeong Wang*, Sang-Hwan Lee**
*Dept. of Mechanical Convergence Engineering, Hanyang University
**Dept. of Mechanical Engineering, Hanyang University

요 약

최근 영상처리 분야에서 딥러닝(Deep learning)을 이용한 기술이 좋은 성능을 보이면서 이에 대한 관심과 연구가 증가하고 있다. 본 연구에서는 최근 딥러닝 네트워크 중 적은 파라미터 수로 AlexNet 수준의 성능을 보인 SqueezeNet 을 영상 분할(Image segmentation)의 특징 추출(feature extraction) 영역으로 사용하고, CUDA C 기반으로 코드를 작성하여 정확도를 유지하면서 계산 속도 면에서도 좋은 성능을 얻을 수 있었다.

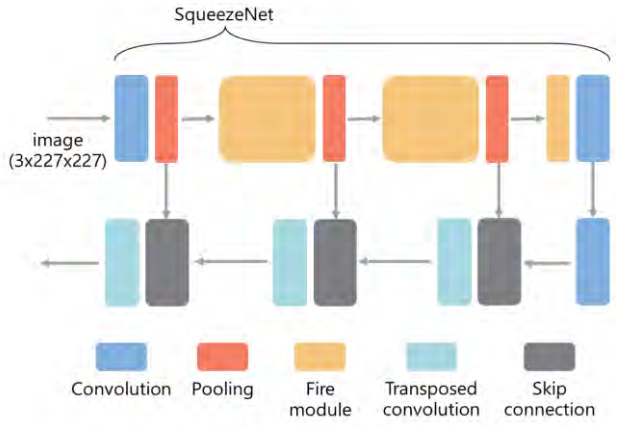
1. 서론

영상 분할은 영상 하나에 대해 분류하는 영상 분류(Image classification)와는 다르게 영상 내에 있는 모든 픽셀들을 몇 가지 클래스로 분류한다. 모든 픽셀들을 분류한다는 것은 네트워크를 통과하면서 작아진 공간적 특징 맵(spatial feature map)을 본래 영상 크기로 만드는 복호기(decoder) 과정이 들어간다는 뜻으로 그만큼의 메모리와 계산량이 요구된다.

이전까지 딥러닝을 이용한 영상 분할 모델들이 많이 개발되었다. 기본적으로 영상 분류문제에서 많이 사용되는 완전 연결 층(fully connected layer)을 없애고 대신 필터역할과 함께 임의의 크기의 영상들이 통과할 수 있도록 만들어진 FCN(Fully Convolutional Network)[1]과 암호기(encoder)와 복호기를 대칭적인 구조로 사용하고 암호기에서 사용된 풀링(pooling)의 위치 정보를 복호기에서 사용하여 적은 메모리를 이용하는 SegNet 이 있다[2]. 또한 기존의 합성곱 층(convolution layer)과는 다르게 같은 파라미터로 더 넓은 수용 영역(receptive field)을 가질 수 있는 atrous convolution 또는 dilated convolution 이 사용된 모델[3]들도 좋은 결과들을 보여주고 있다.

본 논문에서는 영상처리 분야에서 좋은 성능을 가진 AlexNet[4]보다 더 적은 메모리 사용으로 비슷한 성능을 낸 SqueezeNet[5]을 영상 분할 모델에 적용하였고 보다 다양한 프로그램과의 연결성과 함께 속도 면에서도 장점을 발휘할 수 있는 CUDA C 코드를 사용하였다. 2 장에서는 본 논문에서 사용된 네트워크 구조를 설명하고, 3 장에서는 계산결과를 표와 함께 설명한다. 마지막으로 4 장에서는 논문의 내용을 요약하여 결론을 맺는다.

2. 네트워크 구조



(그림 1) 네트워크 구조

본 연구에서 사용된 네트워크 구조는 위 그림과 같다 (그림 1). 네트워크의 암호기 또는 down-sampling 부분은 SqueezeNet 모델을 이용하였고 복호기 또는 up-sampling 부분은 transposed convolution 과 skip connection[1]을 이용하였다. 네트워크에 들어가는 이미지 한 장의 크기는 RGB 채널 3 개와 가로 세로의 길이 227 이다. 이미지 한 장이 들어갔을 때 변화되는 차원과 네트워크의 각 층에 사용되는 파라미터의 수는 아래 표에서 확인할 수 있다<표 1>.

암호기에 있는 fire module 은 SqueezeNet 에 들어가 있는 모듈과 같이 squeeze 부분과 expand 부분으로 나뉜다. squeeze 부분은 필터크기가 1 로 채널의 크기에 영향을 주고 expand 부분은 각각 다른 필터크기(1x1, 3x3)를 적용하고 끝에 이 둘은 서로 연결된다(concatenate).

풀링 층은 이미지의 크기를 줄이고 이를 두 가지 방향으로 전달한다. 한 방향은 down-sampling 방향이고 나머지 하나는 up-sampling 방향이다. up-sampling 쪽으로 가는 이미지는 합성곱 층을 통해 채널을 21로 변경한 후 skip connection 부분에서 이전의 up-sampling 되는 이미지와 더해진 후 다음 층으로 넘어가게 된다.

Transposed convolution 부분은 FCN 에서 복호기로 사용되는 것과 동일하게 적용했다. 표에 나와있는 stride 는 필터가 움직이는 간격이 아닌 필터가 적용되는 영역에서의 간격이다.

<표 1> 네트워크 차원 및 파라미터 수

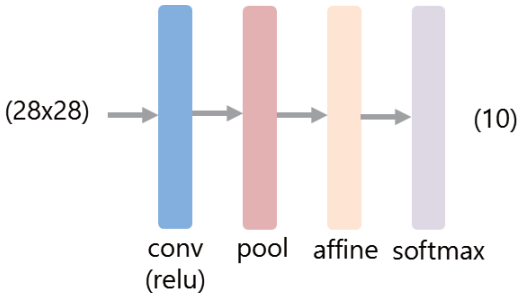
layer	output size	filter(window) size/stride	the number of parameters
conv1	111x111x3	7x7/2	14,208
pool1(max)	55x55x3	3x3/2	
fire2	55x55x128		11,920
fire3	55x55x128		12,432
fire4	55x55x256		45,344
pool4(max)	27x27x256	3x3/2	
fire5	27x27x256		49,440
fire6	27x27x384		104,880
fire7	27x27x384		111,024
fire8	27x27x512		188,992
pool8(max)	13x13x512	3x3/2	
fire9	13x13x512		197,184
conv10	13x13x1000	1x1/1	513,000
score	13x13x21	1x1/1	21,021
score(pool8)	13x13x21	1x1/1	10,773
conv11(trans)	27x27x21	3x3/2	3,969
score(pool4)	27x27x21	1x1/1	5,397
conv12(trans)	55x55x21	3x3/2	3,969
score(pool1)	55x55x21	1x1/1	2,037
conv13(trans)	227x227x21	11x11/4	53,361
		total	1,348,951

3. 계산결과

본 논문에서는 다른 딥러닝 프레임워크를 사용하지 않고 GPU 를 이용한 CUDA C 기반의 코드를 작성하여 사용하였다. 그렇기 때문에 먼저 기본적인 네트워크 층들에 대한 검증을 수행했다. 데이터셋은 이미지 분류 문제에서 자주 사용되는 MNIST 예제를 사용하였고, 네트워크는 각 층의 정확도와 성능을 보기위해 아래와 같이 구성했다(그림 2). 배치 크기는 100 으로 설정하고 추론(inference)을 1000 번 반복 수행했다. 결과는 아래 표에 나와있다<표 2>.

계산 결과 추론 시 올바른 값으로 분류를 해냈고 계산 속도면에서도 tensorflow 만큼의 속도를 얻을 수 있었다.

그 다음으로는 영상 분할을 올바르게 해내는지를



(그림 2) MNIST 를 이용한 네트워크 구조

<표 2> 배치크기 별 추론 속도 결과(ms)

batch size	tensorflow	cpu	gpu
1	0.002	0.001	0.001
10	0.003	0.001	0.001
100	0.004	0.003	0.002
1000	0.018	0.030	0.012
10000	0.130	0.274	0.104

평가해보았다. 이를 위해 같은 네트워크 구조를 Caffe 프레임워크로 작성하고 Nvida digits 을 이용해 학습(training)을 진행하였다. 학습에 사용된 optimizer 는 SGD 이다. 영상 분할을 위한 데이터셋은 Pascal VOC 데이터셋을 이용하였고 이미지를 227 크기로 조정 후 학습을 진행하였다. 학습을 통해 얻은 파라미터들을 CUDA C 코드로 가지고와 각 네트워크에 집어넣은 후 추론을 진행하였다. 추론된 결과와 파라미터가 올바르게 적용됐는지를 확인하기위해 digits 의 결과와 Caffe 의 결과를 같이 비교하였다(그림 3) <표 3>.

(그림 3) 영상 분할 추론 결과(이미지)

<표 3> 영상 분할 추론 결과(픽셀 평균값)

	digits-caffe	caffe	result
pixel mean	-132.768	-132.76796	-132.765

4. 결론

MNIST 를 이용한 영상 분류 문제와 Pascal VOC 를 이용한 영상 분할 문제에서 추론하는 코드를 CUDA C 기반으로 작성하여 검증 및 평가를 진행해보았다. MNIST 의 경우 배치 크기를 늘릴 경우 즉 병렬화가 커질수록 좋은 성능을 냈고, 영상 분할의 경우 파라

미터를 올바르게 적용시켰을 경우 각 클래스 별로 분할을 잘 하는 것을 볼 수 있었다. 이후에 추론 뿐만이 아니라 학습하는 부분도 CUDA C 기반의 코드로 작성한다면 계산 속도면에서 큰 장점이 될 것이다.

참고문헌

[1] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation.”, In Proceeding of the *IEEE* conference on computer vision and pattern recognition. 2015.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”, *arXiv preprint arXiv:1511.00561*, 2015.

[3] L.-C. Chen, G. Papandreou, I. Kokkions, K. Murphy, and A. L. Yuille, “Deeplap: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs”, *arXiv preprint arXiv:1606.00915*, 2016.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”, In *NIPS*, 2012.

[5] F. N. landola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size”, *arXiv preprint arXiv:1602.07360*, 2016.