KIPS Trans. Softw. and Data Eng.
Vol.9, No.5 pp.169~176 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2020.9.5.169

Parameter-Efficient Neural Networks Using Template Reuse

Daeyeon Kim† · Woochul Kang††

ABSTRACT

Recently, deep neural networks (DNNs) have brought revolutions to many mobile and embedded devices by providing human-level machine intelligence for various applications. However, high inference accuracy of such DNNs comes at high computational costs, and, hence, there have been significant efforts to reduce computational overheads of DNNs either by compressing off-the-shelf models or by designing a new small footprint DNN architecture tailored to resource constrained devices. One notable recent paradigm in designing small footprint DNN models is sharing parameters in several layers. However, in previous approaches, the parameter-sharing techniques have been applied to large deep networks, such as ResNet, that are known to have high redundancy. In this paper, we propose a parameter-sharing method for already parameter-efficient small networks such as ShuffleNetV2. In our approach, small templates are combined with small layer-specific parameters to generate weights. Our experiment results on ImageNet and CIFAR100 datasets show that our approach can reduce the size of parameters by 15%-35% of ShuffleNetV2 while achieving smaller drops in accuracies compared to previous parameter-sharing and pruning approaches. We further show that the proposed approach is efficient in terms of latency and energy consumption on modern embedded devices.

Keywords : Neural Network, Parameter Sharing, Layer Reuse, Parameter Efficiency

템플릿 재사용을 통한 패러미터 효율적 신경망 네트워크

김 대 연† · 강 우 철††

요 약

최근 심층 신경망 (Deep Neural Networks, DNNs)는 모바일 및 임베디드 디바이스에 인간과 유사한 수준의 인공지능을 제공해 많은 응용에서 혁명을 가져왔다. 하지만, 이러한 DNN의 높은 추론 정확도는 큰 연산량을 요구하며, 따라서 기존의 사용되던 모델을 압축하거나 리소스가 제한적인 디바이스를 위해 작은 풋프린트를 가진 새로운 DNN 구조를 만드는 방법으로 DNN의 연산 오버헤드를 줄이기 위한 많은 노력들이 있어왔다. 이들 중 최근 작은 메모리 풋프린트를 갖는 모델 설계에서 주목받는 기법중 하나는 레이어 간에 패러미터를 공유하는 것이다. 하지만, 기존의 패러미터 공유 기법들은 ResNet과 같이 패러미터에 중복(redundancy)이 높은 것으로 알려진 깊은 심층 신경망에 적용되어왔다. 본 논문은 ShuffleNetV2와 같이 이미 패러미터 사용에 효율적인 구조를 갖는 소형 신경망에 적용할 수 있는 패러미터 공유 방법을 제안한다. 본 논문에서 제안하는 방법은 작은 크기의 템플릿과 레이어에 고유한 작은 패러미터를 결합하여 가중치를 생성한다. ImageNet과 CIFAR-100 데이터셋에 대한 우리의 실험 결과는 ShuffleNetV2의 패러미터를 15%-35% 감소시키면서도 기존의 패러미터 공유 방법과 pruning 방법에 대비 작은 정확도 감소만이 발생한다. 또한 우리는 제안된 방법이 최근의 임베디드 디바이스상에서 응답속도 및 에너지 소모량 측면에서 효율적임을 보여준다.

키워드 : 신경망, 패러미터 공유, 레이어 재사용, 패러미터 효율

1. 서 론

최근 들어 심층 신경망(Deep Neural Networks, DNNs)은 많은 분야에서 사용되고 있으며, 높은 정확도에 대한 요구가 증가하고 있다. 특히, 현재 주로 사용되고 있는 심층 신경망인 컨볼루션 신경망(Convolutional Neural Networks, CNNs)의 구조는 정확도의 증가를 위해서 더 깊고 더 복잡한 연산을 요구하는 형태로 발전하였다. 하지만, 메모리와 연산 능력 등의 리소스가 제한된 임베디드 환경에서는 리소스를 적게 사용하면서도 높은 정확성을 유지할 수 있는 신경망 구조에 대한 필요성이 제기되고 있다. 신경망 모델들의 메모리 사용량을 줄이기 위해서는 이들의 패러미터 요구량을 줄여야 한다. 그러나, 지금까지의 방법들은 몇몇 단점을 가진다. 예를 들어 pruning[1-4]의 경우에는 어떤 패러미터가 신경망에서 제거되는가에 따라 큰 정확도 손실을 보이며, 심지어

※ 이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1F1A1060959).
† 준 회 원 : 인천대학교 임베디드시스템공학과 석사과정
†† 정 회 원 : 인천대학교 임베디드시스템공학과 교수
Manuscript Received : February 4, 2020
Accepted : March 30, 2020
* Corresponding Author : Woochul Kang(wchkang@inu.ac.kr)

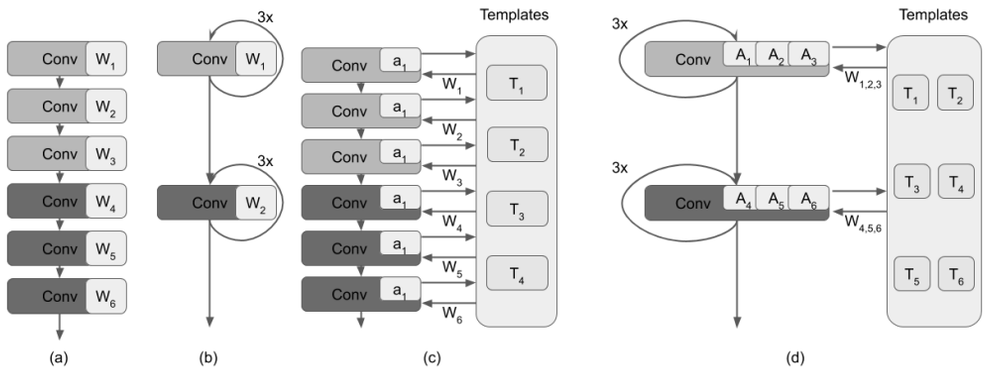

Fig. 1. Parameter Saving Approaches in Convolution Layers.

(a) Typical Convolution Layers Without Parameter Sharing (b) **Layer Reuse** Method [5, 6]: A Weight W_i is used Repeatedly in Successive Convolution Layers, (c) **Templated Layer** Method [7]: Each Layer's Weight W_i is Generated by Multiplying Coefficients a_i and Common Templates T, (d) **Templated Layer Reuse** (Our Proposed Method): Layer-specific Coefficient Matrix A_i is Combined with Light-weight Templates T to Generate Weight W_i, and a Layer is Repeatedly used.

pruning된 것과 동일한 구조의 신경망을 기존 학습된 결과 없이 초기화하여 처음부터 학습하는 것이 더 높은 정확도를 가지는 문제점이 지적되기도 하였다. 최근에 주목받는 방법들 중 하나는 계층들간에 패러미터를 공유하는 방법으로, 레이어 재사용[5, 6]과 템플릿 레이어[7]가 그 예이다. 이와 같은 방법들은 동일한 패러미터를 반복적으로 재사용해 패러미터의 수를 줄이면서 연산량과 성능은 유지하는 패러미터 공유 (parameter sharing) 방법들이다.

본 논문에서는 기존의 방법들을 개선한 새로운 패러미터 공유 방법을 제안한다. Fig. 1은 제안된 방법과 기존 방법들을 비교한다. 가장 중요한 차이점은 기존 방법은 큰 패러미터 크기를 가지는 공유 패러미터를 사용하며 공유되는 패러미터를 사용할 때마다 비교적 작은 변화만을 주지만, 제안된 방법은 작은 패러미터 크기를 가지는 다수의 공유 패러미터를 결합하여 공유되는 패러미터를 사용할 때 마다 비교적 큰 변화를 준다는 것이다. 이를 통해 본 연구에서는 전체적으로 더 적은 패러미터를 사용하면서도 더 높은 정확도를 달성하고자 하였다. 본 논문에서 제안하는 방법은 ShuffleNetV2 1.0x [10] 모델에서 기존의 방법인 pruning과 대비해 유사한 수의 패러미터를 사용하면서도 약 2.5% 더 높은 top-1 정확도를 달성하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련된 연구를 소개하고, 3장에서는 본 연구의 기초가 되는 템플릿 레이어(Templated Layer)와 레이어 재사용(Layer Reuse)에 대해 설명하고, 이를 개선하여 템플릿과 레이어 재사용을 결합한 템플릿 레이어 재사용(Templated Layer Reuse)을 제안한다. 4장에서는 본 연구에서 제안한 패러미터의 재사용의 정확도와 성능에 대한 실험 결과를 비교 분석한다. 5장에서는 본 연구를 요약하고 결론과 향후연구를 제시한다.

2. 관련 연구

현재 기존보다 적은 수의 패러미터를 사용하여 신경망의 효율성을 높이는 주제로 여러 연구가 활발히 진행중이며, 그 대표적인 예로는 pruning[1-3]이 있다. Pruning의 경우, 가중치(weight)를 신경망에서 제거하고도 성능을 유지하는 방법이다. 하지만 pruning은 밀집행렬(dense matrix)을 희소행렬(sparse matrix)로 바꾸어서 전용의 하드웨어가 없다면 실제 처리시간과 에너지 소모량이 오히려 증가할 수 있는 문제점이 있다[2, 3]. 이것을 해결한 방법이 필터 pruning으로, 컨볼루션 신경망에서 필터를 제거하고 성능을 유지하는 방법이다[4]. 이러한 필터 pruning을 임베디드 환경에서 활용하는 연구가 활발하게 이루어지고 있다[11]. 하지만 필터 pruning의 경우에는 어떤 필터가 제거되는가에 따라 신경망의 원래 성능을 복구하지 못하거나, 심지어 정확도 손실이 매우 큰 경우가 존재한다. 또 최근 연구에 의하면, 학습된 네트워크를 pruning 하는 것보다 대상이 되는 신경망의 패러미터를 줄여 처음부터 학습한 신경망이 더 정확도가 높을 수 있다는 한계점도 있다[12].

연산량만을 줄이는 예로는 네트워크에 게이트 구조를 추가하여 특정 레이어의 연산을 건너뛰는 방법이 있다. 이 대표적인 예로는 SkipNet[13]과 M. Figurnov의 연구[14] 등이 있다. 이러한 방법들은 연산량을 줄이는 데에는 효과적이나, 게이트 구조 추가로 인한 패러미터와 처리시간의 증가가 발생하는 문제점이 있으며, 연산을 감소시키면서 원본의 정확도를 유지하기 위해서는 추가적인 하이퍼패러미터 최적화 과정이 필요하다.

최근의 패러미터 감소 방법으로는 적은 수의 패러미터만으로 높은 성능을 내도록 최적화된 신경망 구조를 만들어내는 방법이 있다. 이러한 방법은 일반적으로는 패러미터와 연산량을 감소시키는 것이 주요한 목적으로, 대표적인 예로는 group convolution을 사용하는 ResNext[8], depthwise separable convolution을 도입한 Xception[9]이 있으며, 이 두 방법을 활용해 패러미터와 연산량을 크게 낮추며 정확도를 유지한 ShuffleNet[10, 15]과 MobileNet[16]이 대표적인 최근의 신경망들이다. 이 신경망 모델들은 임베디드 환경에서도 사용 가능할 수준으로 적은 패러미터를 사용해 더 적은 메모리와 연산량을 요구하면서도 기존의 신경망과 유사

한 정확도 성능을 낸다는 특징을 가지고 있다.

또 다른 최근의 연구로는 패러미터 재사용이 있으며, 대표적인 예로는 레이어 재사용 및 템플릿 레이어가 있다[5, 7]. 레이어 재사용의 경우에는 동일한 가중치를 가진 레이어를 재사용하는 것이며, 템플릿 레이어의 경우에는 공유되는 가중치 템플릿에 레이어마다 상이한 계수를 곱하여 레이어를 구성하는 방식이다. 이와 같은 두 방식을 적용하게 되면 충분히 높은 정확도에 동일한 연산량을 유지하면서 더 적은 패러미터를 사용한다는 장점이 있다.

3. 기존의 패러미터 공유 방법 설명 및 새로운 패러미터 공유 방법의 제안

본 논문은 템플릿 레이어[7]와 레이어 재사용[5, 6]을 동시에 사용해 기존 방법과 유사한 숫자의 패러미터로 높은 정확도를 달성하는 것을 목표를 가진 새로운 패러미터 공유 방법을 제안한다. 제안할 패러미터 공유방법의 이해를 돕기 위해 기존의 방법들인 템플릿 레이어와 레이어 재사용을 우선 소개하고, 제안하는 템플릿 레이어 재사용을 통한 패러미터 효율적 컨볼루션 신경망에 대하여 설명한다.

3.1 템플릿 레이어(Templated Layer)

템플릿 레이어[7]는 동일한 차원의 가중치를 가진 레이어간에 공유되는 다수의 템플릿이 존재하며, 레이어들은 4차원의 컨볼루션 가중치 대신 이 템플릿들의 총 개수만큼의 크기를 가진 계수 벡터를 가지게 된다. Fig. 2는 템플릿 레이어의 방식을 표현한다. 벡터 계수 값들은 이 템플릿의 계수 역할을 하며, 신경망의 순방향 진행에서 각각의 계수는 템플릿에 곱해져 전체를 합한 값이 컨볼루션 레이어의 가중치가 된다. 역방향 전파에서도 똑같이 가중치가 계산되는 경로로 역전파가 가능해 정상적으로 템플릿과 계수의 그레이디언트(gradient) 갱신이 가능하므로 기존의 컨볼루션 레이어를 대체할 수 있다.

$$U_i(X_i) = W_i * X_i = \sum_{j=1}^{k} a_i^j T^j * X_i \quad (1)$$

Equation (1)은 템플릿 레이어 방식에서의 컨볼루션 레이어의 연산이다. 여기서 $U_i(X_i)$는 i번째 컨볼루션 레이어의 출력, X_i는 i번째 컨볼루션 레이어의 입력이다. 컨볼루션 레이어의 가중치 W_i는 가중치 템플릿 T_j와 계수 a_i^j의 선형결합이다. 이 식에서 볼 수 있듯, 템플릿은 여러 레이어에서 공유되고 각 레이어는 원래의 가중치보다 작은 계수 벡터만을 가지기 때문에 원본 신경망이 많은 레이어를 가지고 있을수록 더 큰 패러미터 감소가 가능함을 알 수 있다. 이와 반대로, 만약 원본 네트워크가 매우 적은 레이어를 가지고 있다면 템플릿 레이어를 적용하는 것을 통해 패러미터 상의 이득을 크게 얻지 못하거나 오히려 더 많은 패러미터를 사용하는 결과를 가져올 수 있다.

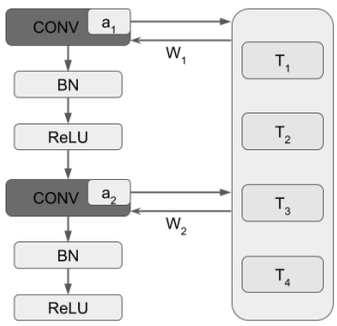

Fig. 2. Parameter Sharing Through Templated Layers

이와 같은 템플릿 레이어 방식은 패러미터 수가 기존의 신경망 구조에 비해 적기 때문에 학습 속도에 있어서 기존 방식보다 우수하다. 이는 일반적인 컨볼루션 신경망은 패러미터의 중복이 크게 일어나며 이로 인해 패러미터를 과도하게 사용하는 경향이 있음을 보여준다.

그러나 ShuffleNet[10] 또는 MobileNet[16]과 같은 신경망의 경우에는 연산량을 줄이기 위해 패러미터 효율성을 염두에 두고 만들어진 신경망이며, 실제로도 기존 신경망에 비해 훨씬 적은 패러미터를 가지고 유사한 정확도를 보여준다. 따라서, 템플릿 레이어 방식은 이와 같은 신경망 구조에서는 이전과 같은 효과를 내지 못할 가능성이 존재한다.

Table 1은 ShuffleNetV2 1.0x 모델에 템플릿 레이어 방식을 적용해 패러미터를 공유한 결과이다. 결과에서 볼 수 있듯, 템플릿을 사용하는 경우 단순히 원본 신경망의 패러미터 수를 감소시켜 처음부터 학습한 축소 모델(Reduced)과 정확도에서 큰 차이를 보이지 않는다. 이는 즉 패러미터 수는 비슷하더라도 템플릿을 사용하는 경우에는 추가적인 연산이 발생한다는 점에서 템플릿 레이어 방식의 한계를 보여준다. 또한, 템플릿을 사용한 방법은 원본 네트워크에 대비하여 패러미터가 약 8.7% 정도만 감소하였기 때문에, 패러미터 감소 효과 역시 제한적임을 알 수 있다.

3.2 레이어 재사용 (Layer Reuse)

레이어 재사용[5]은 여러 컨볼루션 레이어가 동일한 가중치를 반복하여 사용하는 방식이다. 여러 레이어에 각기 다른 패

Table 1. Results for Templated Layers Experiments on CIFAR-100, ShuffleNetv2 1.0x

		Top-1 Accuracy	Top-5 Accuracy	Parameters	Parameters (%)
ShuffleNet v2 1.0x	Original	73.01%	93.05%	1.356M	100.00%
	Reduced	70.51%	92.25%	1.223M	90.15%
	Templated Layer	70.76%	92.06%	1.238M	91.32%

Table 2. Results for Layer Reuse Experiments on CIFAR-100, ShuffleNetv2 1.0x

		Top-1 Accuracy	Top-5 Accuracy	Parameters	Parameters (%)
ShuffleNet v2 1.0x	Original	73.01%	93.05%	1.356M	100.00%
	Reduced	70.15%	91.70%	0.948M	69.88%
	Layer Reuse	71.53%	92.10%	0.954M	70.38%

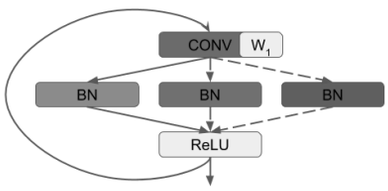

Fig. 3. Parameter Sharing Through Layer Reuse

러미터를 추가하는 대신 공유되는 단 하나 또는 소수의 패러미터를 재사용하므로 패러미터의 크기가 줄어들어 메모리 접근 비용이 감소하고, 또 레이어 재사용이 발생하는 레이어는 네트워크의 여러 부분으로부터 그레이디언트 업데이트를 받게 된다는 장점이 있다. Fig. 3은 레이어 재사용의 방식을 표현한다.

$$\begin{aligned} U_{i+1}(X_{i+1}) &= U_{i+1}(ReLU(BN_i(U_i(X_i)))) \\ &= U_i(ReLU(BN_i(U_i(X_i)))) \\ &= W_i * ReLU(BN_i(W_i * X_i)) \end{aligned} \quad (2)$$

Equation (2)는 레이어 재사용 방식에서의 컨볼루션 레이어 연산이다. 여기서 $U_i(X_i)$는 i번째 컨볼루션 레이어의 출력, X_i는 i번째 컨볼루션 레이어의 입력, W_i는 컨볼루션 레이어 i의 가중치, BN_i는 i번째 배치 정규화(batch normalization) 레이어를 의미한다. 위 식에서 입력값 X_i에 대하여 동일한 컨볼루션 연산 U과 가중치 W가 반복적으로 사용됨을 알 수 있다. 따라서 레이어 재사용 방법의 경우에는 템플릿 레이어 방식보다 더 많은 패러미터의 감소가 가능하다. 레이어 재사용 방식에서는 매 반복마다 동일한 가중치를 사용하기 때문에 매 루프마다 각기 다른 배치 정규화(batch normalization) 레이어가 추가적으로 필요하다.

이와 같은 레이어 재사용은 ResNet과 같은 깊은 구조를 가진 네트워크에서는 큰 효과를 얻을 수 있지만 ShuffleNet이나 MobileNet과 같은 패러미터 효율적인 네트워크의 경우는 큰 정확도 손실이 발생할 가능성이 있다.

Table 2는 ShuffleNetV2 1.0x 모델에 레이어 재사용 방식을 적용해 패러미터를 공유한 결과이다. 이 결과에서 볼 수 있듯이, 레이어 재사용 방식은 템플릿 레이어 방식과는 다르게 패러미터를 기존 대비 약 70%로 감소시켰으나 top-1 정확도 기준 1.5%의 정확도 감소가 발생한 것을 볼 수 있다. 이 결과를 원본 모델의 패러미터 개수를 축소시킨 모델(Reduced)과 비교해 볼 때, 레이어 재사용을 함으로써 패러미터상의 이득을 얻을 수 있는 것은 맞지만, 원본에 대비해 정확도가 손실이 불가피하다는 것을 알 수 있다.

따라서 이와 같은 최근의 신경망 모델에 대해서는 레이어의 재사용이 발생하는 경우, 재사용이 가능한 패러미터 자체가 적을뿐더러 패러미터 재사용을 통한 패러미터 개수 감소 대비 큰 정확도 손실이 발생하기에 한계점을 해결하여 정확도가 손실되는 정도를 최대한으로 줄여야 함을 알 수 있다.

3.3 템플릿 레이어의 재사용(Templated Layer Reuse)

본 논문에서 제안하는 템플릿 레이어의 재사용은 앞서 소개한 두 방법의 장점을 하나로 합치는 것으로, 이미 패러미터 효율성이 높은 신경망에서도 패러미터를 효과적으로 줄여 더욱 효율성을 높일 수 있다.

앞에서 언급한 문제점들에 더불어, 템플릿 레이어는 레이어의 가중치와 동일한 차원을 가진 템플릿이 residual block 그룹마다 다수 존재해야 한다는 문제점 역시 존재한다[7]. 하지만 최근의 신경망 모델들 중 패러미터 효율성을 염두에 둔 모델들은 많은 레이어를 사용하지 않는 경우가 있기 때문에 템플릿을 다수 사용할 경우 패러미터 감소라는 측면에서의 이득이 적을 수 있다[10, 16]. 본 연구에서는 이 문제를 해결하기 위해, 기존 연구에서 사용한 가중치 전체와 동일한 크기가 아닌 가중치보다 작은 크기의 템플릿을 만드는 방법을 사용하였다[7]. Fig. 4는 제안하는 방법을 표현한다.

제안된 템플릿 레이어 재사용의 경우, 기존의 템플릿 레이어와 동일한 방법으로 레이어의 가중치를 계산하지만 레이어의 가중치 전체가 아닌 일부분마다 계산하여 선형결합을 통해 가중치를 만들어낸다는 차이점이 존재한다. 템플릿 레이어에서 사용되는 가중치를 대체하던 벡터 계수들은 제안된 방식에서는 2차원의 계수가 되며, 템플릿들과 계수들이 곱해진 값의 합으로 얻어진 컨볼루션 레이어 가중치의 일부분들로 레이어의 가중치 전체를 구성한다.

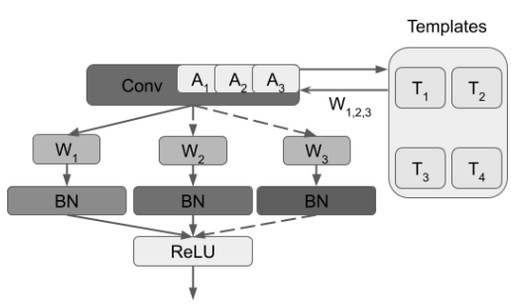

Fig. 4. Parameter Sharing Through Templated Layer Reuse

Fig. 5는 제안된 방식을 보여준다. 여기서 W_i는 컨볼루션 레이어 i의 가중치, T는 전체 템플릿, A_i는 i번째 레이어의

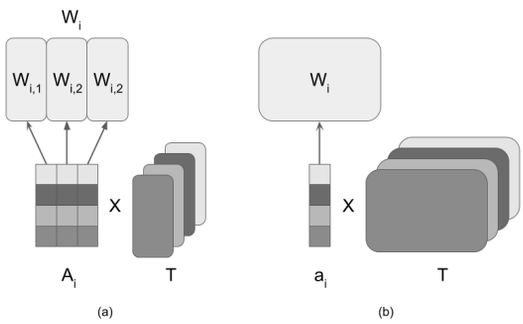

Fig. 5. (a) A Calculation of W_i by Proposed Method (b) A Calculation of W_i by Templated Layer

계수들이다. 제안된 방식은 기존의 템플릿 재사용 방식과 달리 템플릿 하나의 패러미터 크기는 레이어의 가중치의 패러미터보다 훨씬 작은 크기를 가지고 있으며, 계수는 기존의 방법보다 더 많은 차원을 가지고 있다. 즉, 템플릿에서 사용되는 패러미터는 줄이고, 계수에서 사용되는 패러미터는 늘리게 된다. 만약 템플릿 패러미터의 전체 크기가 원래의 컨볼루션 레이어의 가중치 패러미터 크기보다 작다면, 기존의 패러미터 공유 방법들보다도 더 많은 패러미터의 감소가 가능하다.

제안된 방법은 패러미터를 감소시키면서도 매 반복마다 컨볼루션 레이어의 가중치에 충분한 변화를 주기 때문에 다른 패러미터 공유 방법들보다 더 큰 네트워크의 용량을 확보할 수 있다는 장점을 가지고 있다.

$$
\begin{aligned}
&U_{i+1}(X_{i+1}) = U_{i+1}(ReLU(BN_i(U_i(X_i)))) \\
&= W_{i+1} * ReLU(BN_i(W_i * X_i)) \\
&= \sum_{j=1}^{k} A_{i+1}^{j}\, T^j * ReLU(BN_i(\sum_{j=1}^{k} A_i^j T^j * X_i))
\end{aligned} \tag{3}
$$

Equation (3)은 템플릿 레이어의 재사용 방식에서의 연속된 컨볼루션 레이어의 연산이다. 여기서 $U_i(X_i)$는 i번째 컨볼루션 레이어의 출력, X_i는 i번째 컨볼루션 레이어의 입력, W_i는 컨볼루션 레이어 i의 가중치, T^j는 j번째 템플릿, $A_{i,j}$는 i번째 레이어의 j번째 계수, BN_i는 i번째 배치 정규화(batch normalization) 레이어를 의미한다. Equation (3)에 의하면 본 논문에서 제안하는 방법과 템플릿 레이어 방법은 동일하게 보이나, 실제로는 템플릿과 계수의 크기와 차원의 차이가 존재한다. 또 다른 차이점으로는 제안된 방법은 일부 레이어를 변화 없이 재사용한다는 것이다. 제안된 방법에서는 신경망의 residual block 내 일부 컨볼루션 레이어만이 템플릿 레이어로 대체된다. 이를 통해 일부 레이어는 루프에 상관없이 동일한 가중치를 가지고, residual block 내에서 중요한 비중을 차지하는 레이어는 계수와 템플릿을 통해 매 반복마다 변화하는 가중치를 가지게 된다.

Table 3는 ShuffleNet V2 1.0x 모델에 제안된 템플릿 레이어 재사용 방식을 적용해 패러미터를 공유한 결과이다. 기존의 방식들과 다르게, 제안된 방식은 비슷한 패러미터를 가진 축소된 모델에 비해 약 2% 이상의 top-1 정확도를 증가시킴을 확인할 수 있다. 이러한 결과는 본 논문에서 제안한 템플릿 재사용 방식을 사용하게 된다면 기존의 패러미터 공유 방식들과 비슷하거나 적은 수의 패러미터를 사용하면서도 계수 및 템플릿을 학습할 수 있도록 충분한 신경망의 용량을 가지고, 레이어의 재사용이 이루어지므로 각 재사용되는 레이어는 네트워크의 여러 부분에서 업데이트가 발생하기 때문에 기존의 방법들보다 패러미터를 더욱 효율적으로 학습할 수 있으며, 작은 템플릿들과 계수를 조합해 전체 가중치를 만들어내도록 학습을 하는 과정에서 가중치들이 유사한 가중치로 수렴하는 현상이 감소해 이와 같은 효율적인 패러미터 사용이 가능한 것으로 추정된다.

4. 실험 결과 및 분석

본 논문에서 제안된 방식의 성능을 실험하기 위하여 가장 대표적인 데이터셋인 CIFAR-100, ILSVRC2012에 대하여 실험하였다. 4.1절과 4.2절에서는 각각 CIFAR-100과 ILSVRC2012 데이터셋을 사용할때 제안된 방식과 기존의 방식들인 레이어 재사용(Layer Reuse), 템플릿 레이어(Templated Layer), 필터 프루닝(Pruning)과의 패러미터 및 정확도에 대한 성능 비교를 한다.

공정한 평가를 위하여, 비교 대상이 되는 모든 신경망은 동일한 환경에서 학습되고 평가되었다. 실험은 PyTorch 1.3.0 환경에서 진행되었으며 4개의 GTX-1080Ti GPU, 128GB RAM 환경에서 학습과 평가가 진행되었다. CIFAR-100과 ILSVRC2012 학습 과정에서는 각각 다른 하이퍼패러미터가 사용되었다. CIFAR-100의 경우에는 G. Pereyra의 연구[17]와 동일한 cross entropy loss와 SGD optimizer, weight decay와 momentum은 각각 5e-4와 0.9로 설정되었다. learning rate는 초기 150 epoch에 대해서는 0.1, 이후 75 epoch에서는 0.01, 그리고 마지막 75 epoch에 대해서는 0.001로 총 300 epoch으로 학습되었다. 학습 과정에서 learning rate가 변경될 때마다 이전 epoch에서 제일 높은 정확도를 가진 가중치 값을 불러와 학습을 다시 진행하였다. ILSVRC2012 경우에서는 ShuffleNet[15] 학습에서 사용된 하이퍼패러미터와 learning rate scheduler를 사용하였다.

Table 3. Results for Template Layer Reuse Experiments on CIFAR-100, ShuffleNetv2 1.0x

		Top-1 Accuracy	Top-5 Accuracy	Parameters	Parameters (%)
ShuffleNet v2 1.0x	Original	73.01%	93.05%	1.356M	100.00%
	Reduced	70.15%	91.70%	0.948M	69.88%
	Templated Layer Reuse	72.33%	92.76%	0.964M	71.05%

Table 4. Results for Parameter Sharing Methods on CIFAR-100, ShuffleNet v2

		Top-1 accuracy	Top-5 accuracy	Parameters	Parameters (%)
ShuffleNetV2 0.5x	Original	69.13%	91.50%	0.444M	100.00%
	Templated Layer Reuse	**67.51%**	**90.61%**	0.379M	85.30%
	Layer Reuse	66.55%	90.44%	0.373M	83.90%
	Templated Layer	66.62%	90.33%	0.423M	95.10%
	Pruning	67.47%	90.38%	0.392M	88.14%
ShuffleNetV2 1.0x	Original	73.01%	93.05%	1.356M	100.00%
	Templated Layer Reuse	**72.33%**	**92.76%**	0.964M	71.05%
	Layer Reuse	71.53%	92.10%	0.954M	70.38%
	Templated Layer	70.76%	92.06%	1.238M	91.32%
	Pruning	69.96%	91.47%	0.971M	71.60%
ShuffleNetV2 1.5x	Original	74.00%	93.60%	2.581M	100.00%
	Templated Layer Reuse	**73.98%**	**93.14%**	1.667M	64.58%
	Layer Reuse	73.48%	92.72%	1.665M	64.52%
	Templated Layer	71.83%	92.67%	2.315M	92.01%
	Pruning	71.73%	92.04%	1.732M	67.09%
ShuffleNetV2 2.0x	Original	74.90%	93.24%	5.550M	100.00%
	Templated Layer Reuse	**74.35%**	**94.00%**	3.790M	68.29%
	Layer Reuse	73.82%	92.95%	3.799M	68.46%
	Templated Layer	73.16%	92.73%	5.045M	90.90%
	Pruning	72.75%	92.12%	4.039M	72.77%

Pruning은 residual block 내의 3x3 컨볼루션 레이어와 이후의 1x1 컨볼루션 레이어의 L1-norm 기준 필터 pruning[4]을 사용하였으며, 하이퍼패러미터의 경우와 동일한 epoch과 동일한 learning rate scheduler가 사용되었지만 learning rate는 0.01, 0.001, 0.0001로 조절되었으며 원본 신경망의 가중치가 초기값으로 사용되었다. 템플릿 레이어의 경우 첫 residual block 그룹에 대해서는 2개, 두 번째 그룹에 대해서는 3개, 세 번째 그룹에 대해서는 2개의 템플릿이 사용되었다.

4.1 CIFAR-100 데이터셋 실험 결과

CIFAR-100은 이미지당 600개의 이미지로 100개 클래스, 총 6,0000개의 32x32 3채널 이미지로 이루어진 데이터셋이다. 이중 5,0000개는 트레이닝 이미지, 1,0000개는 테스트 이미지다. 총 100개의 클래스는 20개씩 하나의 superclass로 분류될 수 있으며, 이는 즉 비슷한 클래스들이 존재함을 의미한다.

실험 결과는 Table 4와 같다. 이 결과에 의하면 기존의 레이어 재사용 방식은 최대 2.5% 정도의 정확도 감소를 보여주나, 본 논문에서 제안한 템플릿 레이어의 재사용 방법은 비슷한 수의 패러미터를 사용하면서 레이어 재사용 대비 최소 0.5%, 최대 약 1%의 더 높은 정확도를 가지는 것을 볼 수 있다. 특히 ShuffleNetV2 1.5x 모델의 경우처럼 원본 패러미터의 65%만을 사용하지만 거의 동일한 top-1 정확도를 가지는 경우나 ShuffleNetV2 2.0x 모델의 경우처럼 원본 신경망보다도 더 높은 top-5 정확도를 가지는 것에서 볼 수 있듯, 논문에서 제안하는 템플릿 레이어 재사용 방식은 기존의 패러미터 공유 방식에 비해 더 높은 패러미터 효율성을 가지는 것을 확인할 수 있다. 그러나 ShuffleNetV2 0.5x의 경우에는 다른 방법보다는 정확도 손실이 덜하지만 1.5% 정도의 top-1 정확도 손실이 발생하는 것을 볼 수 있는데, 이는 원본 신경망의 패러미터 수가 제한적이기 때문에 발생한 것으로 보인다.

4.2 ILSVRC2012 데이터셋 실험 결과

ILSVRC2012 데이터셋은 1000개의 클래스와 128,1167개의 이미지로 이루어진 데이터셋이다. CIFAR-100 데이터셋과는 다르게 이미지의 크기가 각각 다르며, 또한 클래스 별로 속해있는 이미지의 개수 역시 균등하지 않다.

CIFAR-100 데이터셋에서의 ShuffleNetV2와는 다르게, ILSVRC2012에서 학습된 네트워크는 레이어 재사용을 사용하여 줄일 수 있는 비율이 약 10% 정도 적다. 이는 ShuffleNetv2의 스테이지1, 스테이지5 부분은 단일의 컨볼루션 레이어와 batch normalization 레이어, ReLU 레이어로 이루어져 있기 때문에, 재사용과 같은 방법으로 패러미터를 줄일 수 없어서 발생하는 현상이다.

Table 5는 실험 결과를 보여준다. 이 실험에서 다른 방법들에 비해 제안된 방법은 다른 데이터셋에서도 더 높은 패러미터 효율성을 가지는 것을 알 수 있다. 이에 비해 pruning 방법과 재사용 방법의 경우에서는 큰 정확도 손실을 보인다. 특히 pruning 방법이 패러미터 감소량 대비 큰 정확도 손실을 보이는데, 이는 ShuffleNetV2 자체의 패러미터 효율성이 높다는 점에서 발생하는 현상으로 생각된다. ILSVRC2012는 CIFAR-100에 비해 더 크고 복잡한 데이터셋이므로, 이를 학습하기 위해서 신경망은 더 큰 용량을 요구하게 되는데

Table 5. Results for Parameter Dharing Methods on ILSVRC2012, ShuffleNetv2 1.0x

		Top-1 Accuracy	Top-5 Accuracy	Parameters	Parameters (%)
ShuffleNetV2 1.0x	Original	68.77%	88.56%	2.279M	100.00%
	Templated Layer Reuse	**65.30%**	**85.98%**	1.956M	85.86%
	Layer Reuse	64.31%	85.25%	1.877M	82.37%
	Pruning	63.36%	84.75%	1.894M	83.10%

Table 6. Inference Performance of Naseline CNN Models on Jetson TX2 Embedded Board (GPU Device)

Model		Latency (msec)	Energy per Inference (mJ/Image)	Parameters (millions)	Parameters (%)
CIFAR-100	Original	71.90	5.54	1.356	100.0
	Templated Layer Reuse	64.89	5.31	0.964	71.05
	Layer Reuse	66.75	5.56	0.954	70.38
ILSVRC2012	Original	1166.72	96.98	2.279	100.0
	Templated Layer Reuse	1120.69	95.07	1.956	85.86
	Layer Reuse	1109.23	93.59	1.877	82.37

pruning은 신경망의 용량을 지나치게 감소시키는 결과를 가져오는 것으로 추정된다. 이에 비해 레이어 재사용이나 제안된 방법과 같은 패러미터 재사용 방법은 패러미터를 감소시키면서도 신경망의 용량을 유지할 수 있는 것으로 보인다. 그러나 실험 결과에서 볼 수 있듯 레이어 재사용 방식은 약 4.5%의 정확도 손실을 보이는데, 이에 비해서 본 논문에서 제안하는 방법은 pruning에 비해 약 2%의 정확도 증가, 레이어 재사용 대비 약 1%의 정확도 증가를 보여준다. 즉, 제안된 템플릿 레이어 재사용 기법은 비슷한 수의 패러미터를 사용함에도 불구하고 정확도를 기존의 방법보다도 더 유지할 수 있다는 사실을 알 수 있다.

4.3 임베디드 환경에서의 실험 결과

본 논문에서 제안하는 방법의 실제 임베디드 디바이스에서의 효율성을 확인하기 위하여 Nvidia Jetson TX2 보드에서의 성능을 측정하였다. 신경망 네트워크는 원본 ShuffleNetV2 1.0x을 기준(Original)으로, 제안하는 템플릿 레이어 재사용(Templated Layer Reuse), 레이어 재사용(Layer Reuse)에 대해서 인식 지연시간(latency)과 에너지 소모량을 측정하였다. Jetson TX2의 GPU 디바이스에서 배치 사이즈는 128로 설정하였으며, 입력 이미지들이 신경망을 통해 처리되어 최종 인식 결과를 출력하기까지의 지연시간과 에너지 소모량을 측정하였다. 에너지는 Yokogawa WT-310E을 이용하여 인식시의 시스템 에너지 사용량을 측정하였다. 입력 이미지 크기의 영향을 실험하기 위하여, 각각의 비교 방법들은 CIFAR-100과 ILSVRC2012의 두가지 입력 이미지 크기에 대하여 성능을 측정하였다.

Table 6은 실험 결과를 보여준다. 실험 결과에 따르면, 레이어 재사용과 제안하는 템플릿 레이저 재사용 방식이 원본 ShuffleNetV2보다 다소 우수한 성능을 보여 주었으나, 지연시간과 에너지 소모에서 큰 차이를 보이지는 않았다. 비교된 세 가지 방식은 유사한 연산복잡도(FLOPs)를 갖으며, 제안된 템플릿 레이어 재사용 방법의 경우 컨볼루션 가중치를 만들기 위해 추가적인 연산이 요구되어, 성능 감소가 예상되었다. 그러나, 제안된 방법과 레이어 재사용 모두 원본 ShuffleNetV2보다 다소 우수한 성능을 보인다. 예를 들어, 제안한 템플릿 레이어 재사용 기법은 ILSVRC2012 입력에서 기본 ShuffleNetV2보다 지연시간과 에너지 소모량을 각각 7 milli seconds와 0.23 milli joule만큼 절감하게 된다. 이는 원본 신경망 대비 적은 패라미터 요구로 인한 MAC (Memory access cost)의 감소에 기인하는 것으로 추정할 수 있다. 제안된 템플릿 레이어 재사용과 레이어 재사용 방법 모두 원본 신경망보다 약 15~30% 적은 패라미터 크기를 가지므로, 상대적으로 적은 메모리 접근이 요구된다. 실험 결과는 패러미터 감소에서 비롯되는 MAC 감소가 처리시간과 에너지 소모를 동시에 감소시킬 수 있음을 보인다.

5. 결 론

본 논문은 템플릿 레이어를 재사용하여 이미 패러미터 사용이 효율적인 컨볼루션 신경망 모델의 효율성을 더욱 높이는 방법을 제안하였다. 제안된 방법은 기존 네트워크에서 비중이 큰 가중치들을 몇 개의 템플릿과 작은 크기를 가진 다수의 계수로 대체하고, 그리고 여러 레이어의 가중치를 반복적으로 재사용해 추가적인 패러미터의 사용을 최대한으로 줄였다. 또한 본 논문은 기존의 패러미터 공유 방식의 패러미터를 과도하게 사용하는 문제점 및 패러미터의 효율성이 떨어지는 점을 지적하고, 이를 해결하기 위해 패러미터인 템플릿의 전체 패러미터를 줄이면서 학습은 더욱 효율적인 새 방법을 제안하였다.

제안된 방법은 보편적으로 사용되는 두 개의 테스트 데이터셋를 대상으로 평가되었으며, 실험 결과 기존 방법을 적용했을 때보다 유사한 수준의 패러미터를 사용하면서도 더 높은 정확도를 달성하여 제안한 방법이 데이터셋이나 원본 모델 크기의 무관하게 기존의 방법들보다 우수하다는 것을 보여주었다.

제안된 패러미터 공유 방법은 모바일/임베디드 시스템을 위해 설계되어 이미 패러미터 효율성이 높은 ShuffleNet과

같은 신경망 네트워크의 패러미터 효율성을 정확도 손실을 최소화하며 높일 수 있으므로, 리소스가 제한된 모바일/임베디드 환경에서 활용도가 높을 것으로 예상된다.

References

[1] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal Brain Surgeon," *Advances in Neural Information Processing Systems*, pp.164-171, 1993.

[2] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both Weights and Connections for Efficient Neural Networks," *Neural Information Processing Systems (NIPS)*, 2015.

[3] S. Anwar, K. Hwang, and W. Sung, "Structured Pruning of Deep Convolutional Neural Networks," *ACM Journal on Emerging Technologies in Computing Systems*, Vol.13, No.3, pp.1-18, 2015.

[4] H. Li and A. Kadav, "Pruning Filters for Efficient ConvNets," The International Conference on Learning Representations (ICLR), 2016.

[5] O. Köpüklü, M. Babaee, S. Hörmann, and G. Rigoll, "Convolutional Neural Networks with Layer Reuse," *2019 IEEE International Conference on Image Processing (ICIP)*, pp.345-349, 2019.

[6] Q. Guo, Z. Yu, Y. Wu, D. Liang, H. Qin, and J. Yan, "Dynamic Recursive Neural Network," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.5142-5151, 2019.

[7] P. Savarese and M. Maire, "Learning Implicitly Recurrent CNNs Through Parameter Sharing," *The International Conference on Learning Representations (ICLR)*, 2019.

[8] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.5987-5995, 2017.

[9] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1800-1807, 2017.

[10] N. Ma, X. Zhang, H. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," *The European Conference on Computer Vision (ECCV)*, 2018.

[11] W. Kang, D. Kim, and J. Park, "DMS: Dynamic Model Scaling for Quality-Aware Deep Learning Inference in Mobile and Embedded Devices," *IEEE Access*, Vol.7, pp.168048-168059, 2019.

[12] E. Crowley, J. Turner, A. Storkey, and M. O'Boyle, "Pruning neural networks: is it time to nip it in the bud?," *NIPS 2018 Workshop on Compact Deep Neural Networks with Industrial Applications*, 2018.

[13] X. Wang, F. Yu, Z. Dou, T. Darrell, and J. Gonzaelz, "SkipNet: Learning Dynamic Routing in Convolutional Networks," *The European Conference on Computer Vision (ECCV)*, 2018.

[14] M. Figurnov, M. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov, "Spatially Adaptive Computation Time for Residual Networks," *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1790-1799, 2017.

[15] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.6848-6856, 2017.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.4510-4520, 2017.

[17] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing Neural Networks by Penalizing Confident Output Distributions," *The International Conference on Learning Representations (ICLR)*, 2017.

[18] Y. He, J. Lin, Z. Liu, H. Wang, L. Li, and S. Han, "AMC: AutoML for Model Compression and Acceleration on Mobile Devices," ECCV, 2018.

김 대 연

https://orcid.org/0000-0001-9396-480X
e-mail : ssregibility2@inu.ac.kr
2018년 인천대학교 임베디드시스템공학과 (학사)
2018년 ~ 현 재 인천대학교 임베디드시스템공학과 석사과정
관심분야 : 딥러닝, 딥러닝 모델 압축

강 우 철

https://orcid.org/0000-0002-4757-8999
e-mail : wchkang@inu.ac.kr
1998년 경북대학교 컴퓨터공학과(학사)
2000년 한국과학기술원 전산학과(석사)
2009년 버지니아 주립대학교(UVa) 컴퓨터과학과(공학박사)
2000년 ~ 2012년 한국전자통신연구원 연구원/선임연구원
2012년 ~ 2013년 일리노이 주립대(UIUC) 포스닥 연구원
2013년 ~ 현 재 인천대학교 임베디드시스템공학과 조교수/부교수
관심분야 : 딥러닝 모델 압축, 실시간 임베디드 데이터베이스, 컴퓨팅 시스템 성능 제어 및 보장, 분산 미들웨어