Rethinking Channel dimensions for efficient model design에 대한 논문 리뷰를 시작하겠습니다.

효율적인 인공신경망을 만드는 것은 어렵습니다. 이 논문에서 저자는 효율적인 인공신경망의 정확도가 Channel dimension configuration에 의해서 제한된다고 이야기합니다. 그리고 이 논문을 통해서 저자는 더욱더 효과적인 channel dimension configuration에 대해서 이야기를 하려고 합니다.

이 논문은 크게 4가지 이야기를 중점적으로 합니다. 첫번쨰로는 현제 사용되고 있는 channel dimension configuration에 대해서 이야기를 합니다. 두번째로는 더욱더 효과적인 channel dimension configuration을 만들기 위한 single layer의 표현력에 대해서 이야기합니다. 세번째로는 channel dimension configuration의 탐색에 대한 이야기를 합니다. 마지막으로는 탐색을 통해서 얻은 인공신경망을 실험을 통해서 얼마나 좋은 결과를 가지는 지에 대한 실험을 하였습니다.

효율적인 인공신경망을 만드는 것은 연구자와 실무자 모두에게 중요한 일입니다. 유명한 인공신경망들은 ImageNet classification을 위하여 만들어졌습니다. 그렇기에 대부분 비슷한 구조를 가지고 있습니다. Low-dimensional input channel을 증폭시켜 마지막 layer에서는 classification을 하는 class의 개수보다 더 많은 channel dimension을 가지도록 만듭니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

하지만 여기 표에서 보이는 lightweight model의 경우 마지막 building block까지의 크기가 그렇게 많이 증가하지 않습니다. ImageNet의 경우 1000개의 class가 있는 반면, 여기의 block의 output channel의 수를 보면, 대부분 200에서 300 정도로, 100개보다 훨씬 적습니다. 이를 해결하기 위해서 가장 마지막 레이어인 fully connected layer 전에 class의 개수보다 더 많은 channel을 가지도록 증폭시킵니다. 이는 FLOP efficiency를 줄이기위한 하나의 방법입니다.

이러한 설계 방식은 MobileNetV2가 처음 소개한 방식입니다. MobileNetV2에서 소개된 이후로 표에서 보이는 바와 같이 많은 인공신경망에서 사용되었습니다. 심지어 인공신경망 탐색을 통해서 찾은 신경망에서도 비슷한 방식으로 사용되었습니다.

이 논문은 이를 통해서 mobileNet의 설계를 따르는 것이 정확도를 제한한다고 생각하며, 이를 해결하기 위해서 각 레이어의 표현력을 증가시키는 방향으로 channel dimension configuration을 만듭니다.

가장 먼저 레이어 하나의 설계에 대한 이야기를 하려고 합니다. 레이어의 표현력을 증가시키는 식으로 인공신경망의 가장 뼈대가 됩니다. 레이어의 설계를 이야기 하기전에, 가장 먼저 표현력이 무엇인가에 대한 이야기부터 시작하여, 현제 사용되고 있는 신경망의 구조를 그 다음으로 설명할 것입니다.

이 다음에는, 레이어의 설계를 방식을 설명하기 위한 간단한 실험과, 그 실험 결과를 통해서 찾은 관찰 결과를 이야기할 것입니다.

표현력에 관한 설명은 “Breaking the softmax bottleneck: a highrank rnn language model”이라는 논문과 “Efficient and accurate approximations of nonlinear convolutional networks”라는 논문을 기반으로 설명합니다. 첫번째 논문에서, language modeling을 할 때 softmax 레이어가 Logit을 probability로 바꾸는 것에 어려움을 겪는다고 이야기합니다. 논문에서는 이 이유를 Rank deficiency라고 설명을 하고, 이는 softmax layer가 log probability 연산을 하는 중에 non-linearlity가 사라진다고 이야기합니다.

두번쨰 논문에서는 model 압축을 통해서 rank가 중요하다는 것을 설명합니다. 이때 사용하는 방식이 Singular Vector Decomposition을 통하여 Singular Value의 값을 이용한 model 압축을 하였습니다.

이 두가지 논문을 기반으로 저자는 행렬의 계수(Rank)가 표현력과 관련이 있다고 추측합니다.

이번에는 인공신경망의 구조에 대하여 살펴봅시다. ResNet기반의 인공신경망은 Bottleneck 레이어를 사용하고, channel의 크기가 두배씩 증가합니다. 다른 효율적인 인공신경망은 inverted bottleneck 구조를 채택하고 있으며, channel의 크기는 천천히 증가합니다, 이는 MobileNet에서 소개된 구조를 기반으로 만들어진것입니다.

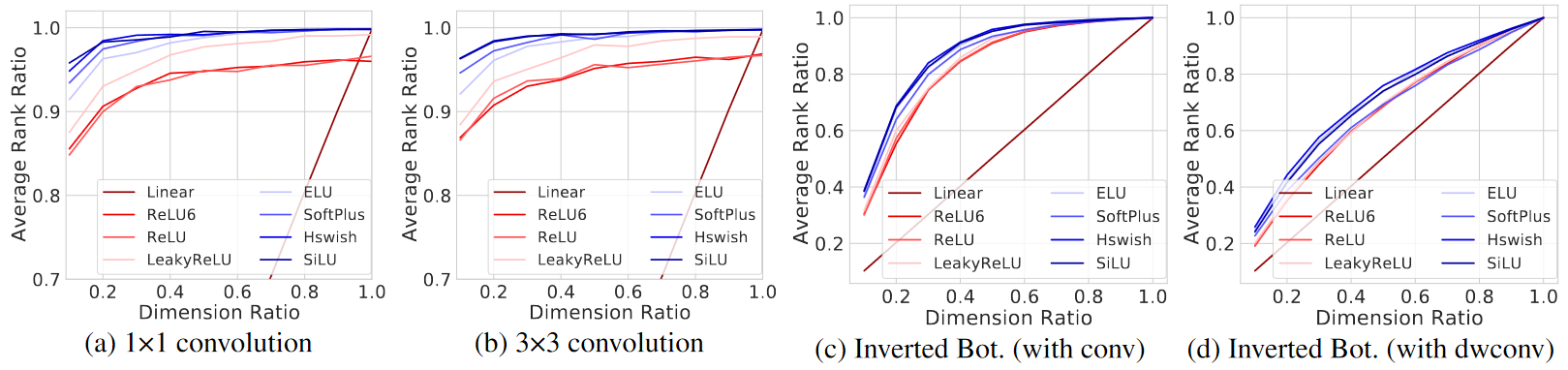
이때 사용되는 bottleneck과 inverted bottleneck 구조 모두 expansion을 사용합니다. 그리고 expansion ratio은 대체적으로 6이나 4를 많이 사용합니다.

저자는 하나의 질문을 제기합니다. 논문에서 나오는 ResNet기반 인공신경망들과 효율적인 신경망의 레이어가 정확하게 설계되었고 이를 조합하여 인공신경망을 만드는 것이 좋은 것인가?

저자는 이 질문에 대한 답을 찾기 위해서 하나의 간단한 실험을 하였습니다.

이 실험의 목적은 효율적인 신경망에서 사용하는 expansion layer에 관한 설계에 대한 설명을 합니다. 여기서 출력 행렬의 계수를 구하는 것으로 레이어의 표현력을 확인합니다. 이때 행렬의 계수는 입력 값에 의해서 제한되어집니다. 하지만 non-linear 함수를 사용함으로 non-linear function의 효과를 살펴보고, 언제 사용해야 효과가 가장 큰지에 관한 실험입니다.

이 실험에서 살펴볼 신경망 구조는 총 4가지입니다. 1by1 convolution, 3by3 convolution, Inverse bottleneck, inverse bottleneck with depthwise separable convolution입니다. 이러한 신경망들의 rank ratio와 dimension ratio의 관계를 살펴볼 것입니다. Rank Ratio는 출력의 Rank 나누기 출력의 channel 크기입니다. Dimension ratio는 입력 channel의 크기 나누기 출력 channel의 크기입니다. 이때 Inverted Bottleneck의 경우, 1by1convolution의 입력 값을 입력 값으로, skip connection과의 덧셈이후에 나오는 출력 값을 출력 값으로 계산한 값입니다.



실험 결과를 확인해보면, 3가지를 확인할 수 있습니다.

첫번째로 Dimension Ratio가 작으면, Rank ratio가 작은 것을 확인할 수 있습니다. 이는 일반적인 convolution 연산을 물론, Inverted Bottleneck 모두에게 나타나는 현상입니다.

두번째는 Non-linearlity를 사용하면, rank ratio가 커지는 것을 확인할 수 있습니다. 실험한 4가지의 조건에서 모두 동일하게 나타났습니다. 또한 ELU, softplus, Hswish, SiLU등 복잡한 non-linear 함수를 사용할 경우, 더욱더 rank ratio가 커집니다. 또한 Dimension ratio가 작을 때 더 커지는 것을 확인할 수 있습니다.

마지막으로 Convolution연산에서 non-linearlity가 더 중요합니다. Convolution에 non-linear함수를 사용한 경우 Inverted bottleneck보다 효과가 뛰어난 것을 확인할 수 있습니다.

이 세가지 관측 결과를 중심으로 배운 점이 있습니다.

첫번째로 Inverted bottleneck의 1by1 convolution을 사용할 때, expansion ratio를 6이하의 숫자로 하는 것이 좋습니다,

두번째로는 Inverted bottleneck with depthwise를 사용할 때는 higher channel dimension ratio를 사용해야합니다.

마지막으로, 복잡한 activation 함수를 사용하는 것은 convolution 연산 뒤에 사용해야합니다. Depthwise convolution 뒤에 사용할 경우 별로 효과를 보지 못합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위에서 배운 점을 활용해서 실제로 적용해보았습니다. 인공신경망은 가장 간단한 구조로 Inverted bottleneck을 두개만 활용하였습니다. BaseLine의 경우, 1by1convolution의 expansion ratio를 20으로 설정하였습니다. 다음으로는 expansion ratio를 20에서 6으로 줄였습니다. 여기서 Inverted bottleneck 전체의 dimension ratio또한 0.22에서 0.8로 증가시켰습니다. 마지막으로 ReLU6을 SiLU로 바꾼 것입니다.

위에서 배운 것들을 하나 하나씩 적용할수록, 정확도가 향상하는 것을 확인할 수 있습니다. 이를 통해서 위에서 확인한 실험결과가 사실인 것을 알 수 있습니다.

Channel Configuration을 찾기 전에 어떠한 방식으로 찾을 것인가에 대한 설정을 해야 합니다. 이 논문에서는 여기 표현된 수식을 기반으로 channel configuration을 찾을 것입니다.

텍스트이(가) 표시된 사진

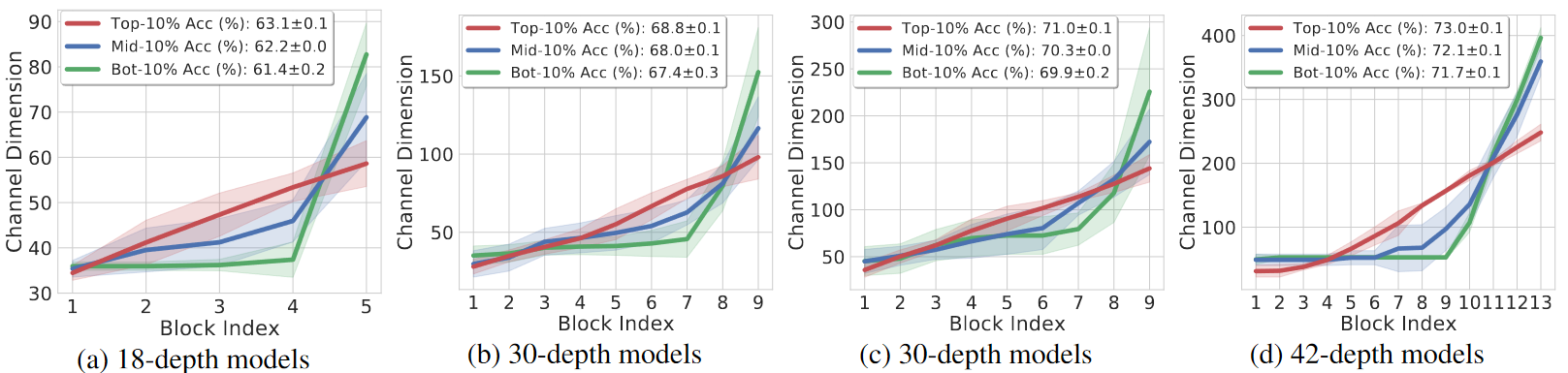
자동 생성된 설명

여기서 함수 Acc는 top-1 정확도를 의미합니다. Ci는 i번쨰 레이어의 channel dimension의 크기를 의미합니다. 이때 c i는 점진적으로 증가하는데, 이는 conventional channel configuration 또한 포함할 수 있게 만들기 위함입니다. 또한 여기서 latency가 아닌 FLOPs를 기반으로 탐색을 하는 이유는 FLOPs가 더 범용성이 강하기 때문입니다.

여기서 Ci의 크기를 정하는 방법은, 입니다. 여기서 a와 b의 값은 탐색을 해야 하는 값이고, 함수 f(i)는 부분 선형 함수로 1에서 d의 값에서 선택된 부분 집합입니다. 이를 통해서 MobileNetV2에서 사용한 Channel Dimension Configuration 또한 포함할 수 있도록 한 것입니다.

탐색은 NASNet과 비슷하게 CIFAR-10과 CIFAR-100에서 탐색이 이루어졌습니다. Channel dimension의 탐색에 집중하기위해 다른 변수들은 고정시켰습니다. Expansion factor는 전에 expansion layer에서 실험한 것으로 6으로 고정했습니다.

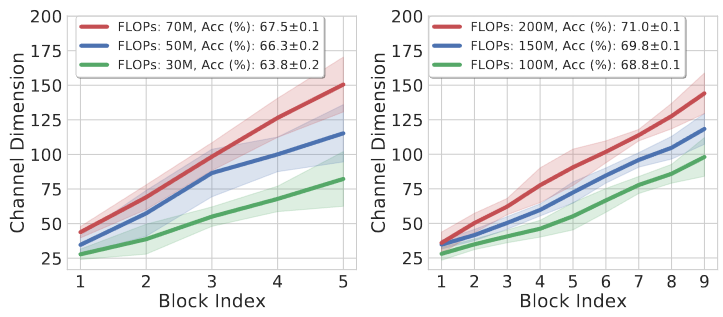
하나의 모델은 3번씩 30epoch씩 training 했습니다. 3번 훈련한 이유는 random initialization을 통해서 발생하는 정확도의 변화를 줄이기 위해서입니다.

테이블이(가) 표시된 사진

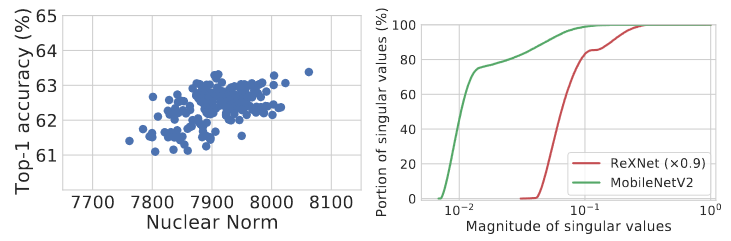
자동 생성된 설명

위의 그래프에서 보는 것처럼, 저자는 4가지 제한점을 준 상태에서 훈련을 하였습니다. 4가지 제한은 inverted bottleneck block의 개수, parameter 수, FLOPS 크기입니다. 여기 표에 자세한 설명이 되어있습니다. 이러한 제약을 가지고 탐색한 인공신경망을 정확도에 따라서 상위 10%, 중앙 10%(다른말로 상위 50%에서 상위 60%사이 값을) 하위 10%의 channel dimension configuration을 확인한 것입니다.

여기서 보이는 것처럼, 붉은 색 선이 상위 10%를 나타내는 결과 값입니다. 이 선에서 보는 것처럼, channel dimension이 linear 할 경우 비슷한 연산량을 가진 다른 네트워크보다 더욱더 높은 적확도를 가지는 것을 확인할 수 있습니다. 또한 파란색 선은 MobileNetV2에서 사용된 channel dimension configuration과 비슷한 것을 확인할 수 있습니다. 이를 통해서 이 논문에서 제시하는 channel dimension configuration을 사용하는 것이 관습적으로 사용해온 channel dimension configuration을 사용하는 것보다 더 높은 정확도를 가지고 있는 것을 알 수 있습니다.



이번에는 신경망의 깊이를 고정시킨 상태에서 FLOPs를 변화시켜 Channel dimension configuration을 확인한 결과입니다. 여기에 나타나 있는 모든 선들은 상위 10%의 결과의 평균 값을 모은 것입니다. Depth가 18일 때는 FLOPs를 30MFLOPs, 50MFLOPs, 70MFLOPs가 되도록 탐색했고, Depth가 30일때는 100MFLOPs, 150MFLOPs, 200MFLOPs로 제한한 것입니다. 이 두개의 그래프에서 확한 할 수 있듯이 FLOPs가 변화하더라도 Channel dimension이 Linear하게 증가하는 것을 볼 수 있습니다. 이를 통해서 관습적으로 사용해온 Channel Dimension configuration보다 Linear channel configuration이 좋은 것을 학인 할 수 있습니다.



이 표는 모델 표현력을 출력 행렬의 계수(Rank)를 통해서 알 수 있다는 것을 확인하기 위한 실험입니다. 정확도와 Nuclear Norm으로 표현한 행렬의 계수(Rank)를 비교한 분표표를 보면, Rank가 높을수록 정확도가 높은 것을 확인할 수 있습니다.

도한 옆에 표현된 Singular Value의 값을 통해서 ReXNet의 Rank가 기존의 MobileNet보다 좋은 것을 확인할 수 있습니다.

테이블이(가) 표시된 사진

자동 생성된 설명

여기서 탐색한 channel dimension configuration을 이용해서 새로운 network를 소개합니다. MobileNetV2를 기반으로 가장 첫번째 convolution을 32로 사용했고 Batch Normalization 과 ReLU6를 적용하였습니다. Inverted bottleneck또한 expansion ratio를 6으로 MobileNet에서 사용한 것과 똑같이 맞추어 주었습니다. 하지만 MobileNetV2와 차이점이 있다면, Inverted bottleneck의 출력 channel dimension을 parameterization으로 변화한 것과, ReLU6를 SiLU로 바꾼 것, 그리고 Inverted Bottleneck에 Squeeze and Excitation을 추가한 것입니다.

여기서 전에 실험한 rank ratio와 dimension ratio의 실험결과에 따라, Inverted bottleneck에서 1by1 convolution 뒤에 있는 ReLU6를 SiLU로 바꾸었습니다. Depthwise convolution의 경우, dimension ratio가 1임으로, ReLU6를 그대로 사용합니다.

여기서 보이는 MB1과 MB6는 mobilenetv2에서 사용된 expansion ratio가 각각 1 과 6인 Inverted bottleneck을 의미합니다. Downsampling block을 제외한 모든 inverted bottleneck block에서 Skip Connection은 zeropadding을 사용해서 channel dimension을 맞추어 주었습니다.

아래에서 표현된 두번째 인공신경망은 MobileNetV1을 기반으로 만들어진 것입니다. 여기서 skip connection을 사용하지 않는 plain Network를 만들어보았습니다. MobileNetV2를 기반으로 만들어진 ReXNet처럼 Channel dimension configuration을 바꾸었습니다. 여기서 ReLU6를 SiLU로 바꾸었는데, 똑같이 1by1convolution이후에 있는 RELU만 바꾸어 주었습니다.

이러한 인공신경망 설계를 기반으로, ImageNet에서 실험을 해보았습니다. ImageNet dateset을 활용해서 다른 EfficientNet과 비교를 해보았고, RexNet의 scalablilty를 확인했습니다. 또한 non-linear 함수를 변화시키는 것의 효과를 확인해 보았습니다.

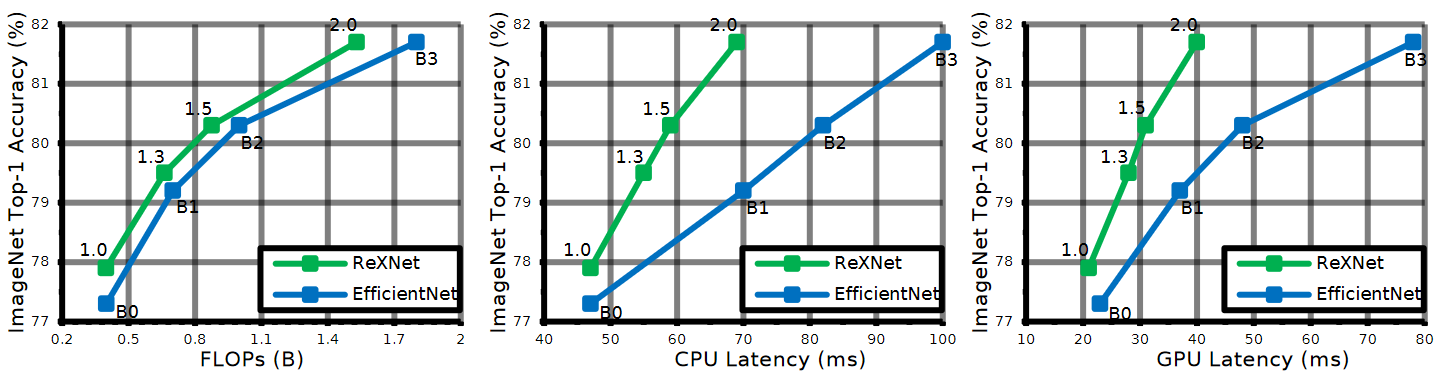
이러한 실험결과를 바탕으로, 다른 작업에 이전해 보았습니다. ReXNet을 backbone으로 COCO object detection과 다른 fine-grained classification, 그리고 COCO instance segmentation에 대해서도 실험을 하였습니다.

ImageNet 실험에서 Standard Data Augmentation을 사용했습니다. Optimizer는 stochastic gradient decent를 사용했고, Learning rate는 0.5부터 시작해서, Cosine learning rate schedular를 사용헀습니다. Batch size는 512를 사용헀고 4개의 GPU를 동시에 사용해서 훈련시켰습니다.   
테이블이(가) 표시된 사진

자동 생성된 설명

실험 결과는 옆에 있는 표를 보면 알 수 있습니다. 여기서 칼표가 붙어 있는 것은 Squeeze and Excitation과 SiLU를 사용한 것을 의미합니다. 별표가 붙어있는 것은. AutoAug, Rand,Aug, 나 Mixup을 사용한 것을 표시한 것입니다.

이 표에서 보는 것처럼, ReXNet에 RandAug를 사용한 것이 다른 네트워크에 비해서 결과가 좋은 것을 확인할 수 있습니다.



테이블이(가) 표시된 사진

자동 생성된 설명

이 다음 실험은, width multiplier를 사용해서 신경망의 scalability를 확인한 것입니다. EfficientNet 과 비교를 한 경우, ReXNetdl 대체적으로 정확도가 높은 것을 알 수 있고, 또한 비슷한 크기의 인공 신경망끼리 비교한 경우 ReXNet이 더 빠른 것을 확인할 수 있습니다. 만약 비슷한 latency끼리 비교한 경우, RexNet이 2.1%더 성능이 좋은 것을 확인할 수 있습니다.

테이블이(가) 표시된 사진

자동 생성된 설명

이 표는, ReXNet의 Scalability를 더욱 더 늘려서 확인해본 것입니다. MobileNetV1을 기반으로 만들어진 ReXNet(plain)의 경우, 다른 ReXNet보다 2배 이상은 더 빠른 결과를 나타내 줍니다.

테이블이(가) 표시된 사진

자동 생성된 설명

마지막으로, non-linearlity에 관한 실험을 살펴봅시다. 전에 하나의 레이어에 관한 실험에서 나온 것처럼, 복잡한 non-linear함수를 사용할 경우, ReLU를 사용한것보다 더 효과가 좋은 것을 알 수 있습니다. 또한, SiLU의 결과가 가장 좋은 것을 확인할 수 있습니다.

테이블이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

이제는 ImageNet에서 좋은 결과를 얻은 ReXNet을 backbone으로 사용해 COCO object recongnition에 활용한 것을 살펴봅시다. 위의 표는 ReXNet에 SSDLite를 사용한 것에 관한 비교입니다. 아래에 있는 표는 SSDLite 대신에 Fast RCNN을 사용했을 때의 결과입니다.

이 두개의 표에서 보는 것처럼, ReXNet의 결과가 비슷한 것들과 비교했을 경우 결과가 좋은 것을 확인할 수 있습니다.

테이블이(가) 표시된 사진

자동 생성된 설명

ImageNet의 인공신경망을 finetune해서 Food-101, Stanford cars, FGVC Aircraft, Oxford Flowers 102에 실험한 결과입니다. 이 표에서 보는 것처럼, 비슷한 연산 량을 가진, EfficientNet과 비료를 하면, ReXNet이 항상 결과가 좋은 것을 알 수 있습니다.

또한 몇몇 dataset의 경우. ReXNet이 연산량이 10배 많은 ResNet-20과 비교해도 더 좋은 결과를 내는 것을 볼 수 있습니다.

테이블이(가) 표시된 사진

자동 생성된 설명

마지막 실험은 COCO Instance Segmentation에 관한 실험입니다. 여기서도 비슷한 연산 량을 가진 Efficient Net이나 ResNet50과 비교해도 ReXNet의 결과가 더 좋은 것을 알 수 있습니다.

이 논문을 통해서 3가지를 배울 수 있습니다. 첫번쨰로는 lightweight model을 위한 building block을 설계하는 방법을 이야기합니다. 여기서 non-linear함수를 사용하는 법과, pointwise convolution에 사용하는 expansion ratio에 관한 이야기를 중점적으로 했습니다. 두번째로는 channel dimension configuration을 탐색하는 방법에 대한 이야기를 합니다. 여기서 channel parameterization을 소개합니다. 이러한 channel parameterization을 사용한 새로운 인공신경망을 소개하고, 이를 활용한 실험 결과를 보여드렸습니다.

이 영상을 봐주셔서 감사합니다.