#hutom_ai #Journal Club

1. torch.manual_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision
2. GhostNet: More Features from Cheap Operations

2021-11-05 | Jihyun Lee

hutom

torch.manual_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision

hutom

`torch.manual_seed(3407)` is all you need: On the influence of random seeds in deep learning architectures for computer vision

David Picard DAVID.PICARD@ENPC.FR
LIGM, École des Ponts, 77455 Marnes la vallée, France

1. Introduction
2. Experimental setup
3. Limitations
4. Findings
5. Discussion

The effect of random seed selection on the accuracy when using popular deep learning architectures for computer vision.
또한, 본 논문은 3가지 질문을 던지고 있음.

1. What is the distribution of scores with respect to the choice of seed?
2. Are there *black swans*, *i.e.*, seeds that produce radically different results?
3. Does pretraining on larger datasets mitigate variability induced by the choice of seed?

1. Seed 선택에 대한 score 분포?
2. 근본적으로 다른 결과를 내는 black swan 존재?
3. 대규모 dataset 에 대한 pre-train 이 seed 선택에 의한 가변성을 완화시키는가? 즉, 대규모 dataset 으로 pre-train 하면, 시드의 변화에 덜 민감하게 반응하는가?

- 과거 experiment power 가 제한되었을 때는, computer vision 이 한 번의 experiment 만을 실행한 후 report 되는 경우가 흔했음.
  - Deep learning community 은 충분히 많은 reproductions 을 수행하지 않는 점에서 잘못된 접근 방식을 가지고 있음.
  - Physical experiment 에서 noise 가 발생하는 것처럼,
    deep learning experiment 에서도 random factors 들이 존재함.
    - **e.g split between training and testing data, the random initialization, the stochasticity of the optimization process, etc.**
  - Related Work
    - random facters 들이 결과에 미치는 영향을 평가 지표 측면에서 파악하는 것이 중요해보임.
      - Mehrer et al. (2020) : random factors 들이 포함된 다양한 네트워크들은 각각 다른 방식으로 영향을 받음.
      - Bouthillier et al. (2021) : computer vision 을 포함한 다양한 작업에 대해 수행했지만, 더 작은 데이터 세트와 200개의 실행 되었음.

- 모든 random factors 에 의존하는 random seed 의 영향을 조사함.
  - 만약 random seed 가 충분히 작으면, 해당 random seed 의 영향은 미미하며,
  - random seed 의 영향이 충분히 크다면
    → 기존에 deep learning publications 연구들이 어떤 random factor 를 사용했는지에 대한 더 자세한 내용을 포함해야 한다고 생각함.

2. Experimental setup

- A budget of 1000 hours of V100 GPU computing power.

- Experiment
  1) Finding the architectures and training parameters that would allow a good trade-off between accuracy and training time
  2) actual training and evaluation

  - CIFAR 10
    - 10000 seed 를 사용하여, 많은 수의 seed 를 train, evaluate. (30 seconds)
    - Architecture: custom ResNet with 9 layers (scratch)

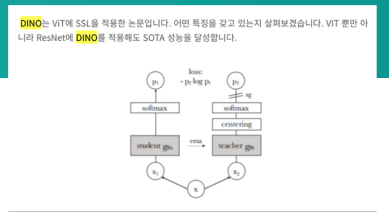
$$C^3_{64} - C^3_{128} - M^2 - R[C^3_{128} - C^3_{128}] - C^3_{256} - M^2 - C^3_{256} - M^2 - R[C^3_{256} - C^3_{256}] - G^1 - L_{10}$$

where C^k_d is a convolution of kernel size k and d output dimension followed by a batch normalization and a ReLU activation, M^2 is a max pooling of stride 2, $R[\cdot]$ denote a skip connection of what is inside the brackets, G^1 is a global max pooling and L_{10} is a linear layer with 10 output dimensions.

    - SGD with momentum and weight decay
    - Loss : a combination of a cross-entropy loss with label-smoothing & the regular categorical cross-entropy.

Training mode (V100 computation time) // Total time taken on CIFAR was about 90 hours of V100 computing time.	
short training mode	long training mode
Each taking 30 seconds to train and evaluate. Operating point of 1000 seeds. Total : 5000 minutes (83 hours of V100 computing power)	A longer training setup of 1 minute. Operating point of 500 seeds. Total : 8 hours

2. Experimental setup



- A budget of 1000 hours of V100 GPU computing power.

- Experiment
  1) Finding the architectures and training parameters that would allow a good trade-off between accuracy and training time
  2) actual training and evaluation

  - ImageNet
    - 모든 model 은 50 seed 에서 실험.
    - Pretrained networks
      - Supervised ResNet50: the standard pretrained model from pytorch
      - SSL ResNet50: a ResNet50 pretrained with self-supervision from the DINO repository, see Caron et al. (2021)
      - SSL Vit: a Visual transformer pretrained with self-supervision from the DINO repository, see Caron et al. (2021)

Training mode (V100 computation time) // Total time taken on ImageNet was about 440 hours of V100 computing time.		
Supervised pretrained ResNet50	SSL pretrained ResNet50	SSL pretrained ViT
Training time of about 2 hours. On 50 seeds. Total : 100 hours	Training time of about 3 hours. On 50 seeds. Total : 155 hours ** SLL initialization 이 comparable accuracy 에 도달하기까지 더 많은 step 이 필요했기 때문.*	Training time of about 3 hours 40 minutes. On 50 seeds. Total : 185 hours

3. Limitations

- 본 실험들로 인한 accuracy 는 SoTA 수준이 아님.
  - Because of the **budget constraint** that forbids training for longer times.
    - 그래도, 모델들은 convergence 후에 평가하였음 (show in the next section).

1. What is the distribution of scores with respect to the choice of seed?

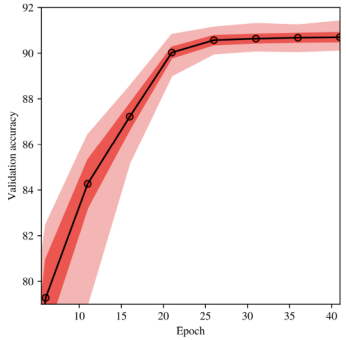


Figure 1: Validation accuracy variation on CIFAR 10 for the Resnet9 architecture against number of training epochs. he solid line represents the mean over 500 seeds, the dark red area corresponds to one standard deviation and the light red corresponds to the maximum and minimum values.

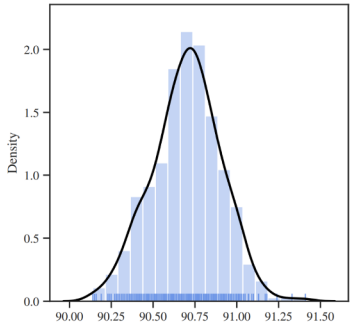


Figure 2: Histogram and density plot of the final validation accuracy on CIFAR 10 for the Resnet9 architecture over 500 seeds. Each dash at the bottom corresponds to one run.

Notation

- Dark red line : average (평균)
- Dark red : one standard deviation (표준 편차)
- Light red : minimum and maximum values attained across seeds.

- **Convergence instability**
- **Figure 1.**
  - **Training on CIFAR 10 for 500 different seeds. (long training mode)**
  - Accuracy 는 25 epoch 이후로 변화 없음 (optimization converged).
  - 또한, 20 epoch 이후부터 standard deviation (표준 편차)가 수렴함.
    - Narrowing behavior is observed before convergence.
- **Figure 2.**
  - Distribution of scores: **Fairly concentrated**
  - 91.0% - 90.5% = 0.5% 차이는 오롯이 seed difference 로 설명될 수 있음.

The answer to first question is thus that the distribution of scores with respect to the seeds is fairly concentrated and sort of pointy. This is a reassuring result because it means that scores are likely to be representative of what the model and the training setup can do, expect when one is actively searching (knowingly or unknowingly) for a good/bad seed.

Training mode	Accuracy mean ± std	Minimum accuracy	Maximum accuracy
long	90.70 ± 0.20	90.14	91.41
short	90.02 ± 0.23	89.01	90.83

Table 1: Results on cifar 10 for the long and the short training setups.

2. Are there *black swans*, *i.e.*, seeds that produce radically different results?

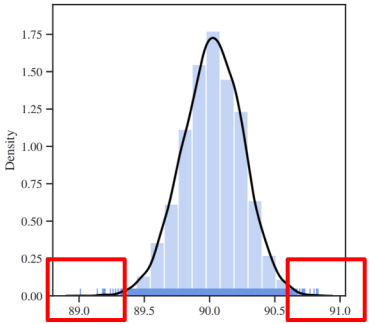


Figure 3: Histogram and density plot of the final validation accuracy on CIFAR 10 for the Resnet9 architecture over 10^4 seeds. Each dash at the bottom corresponds to one run.

- **Searching for *Black Swans***
- **Figure 3.**
  - **Training on CIFAR 10 for 10000 different seeds. (short training mode)**
  - 89.5% - 90.5% 에 집중적으로 분포되어 있음.
  - 더 많은 데이터 셋을 확인하지 않는 이상, 추가적인 higher or lower extremes 는 관측되기 어려워 보임.
  - 즉, 10000 개 극단적인 성능을 나타내는 seed 는 모델의 평균적인 성능을 잘 나타내지는 못함.

Training mode	Accuracy mean ± std	Minimum accuracy	Maximum accuracy
long	90.70 ± 0.20	90.14	91.41
short	90.02 ± 0.23	89.01	90.83

Table 1: Results on cifar 10 for the long and the short training setups.

The results of this test allow me to answer positively to the second question: there are indeed seeds that produce scores sufficiently good (respectively bad) to be considered as a significant improvement (respectively downgrade) by the computer vision community. This is a worrying result as the community is currently very much score driven, and yet these can just be artifacts of randomness.

3. Does pretraining on larger datasets mitigate variability induced by the choice of seed?

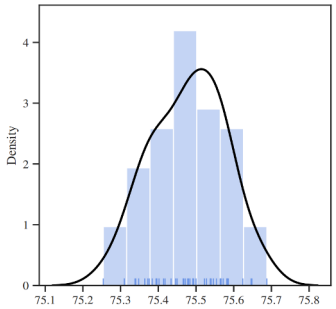


Figure 4: Histogram and density plot of the final validation accuracy on Imagenet for a pretrained ResNet50. Each dash at the bottom corresponds to one run.

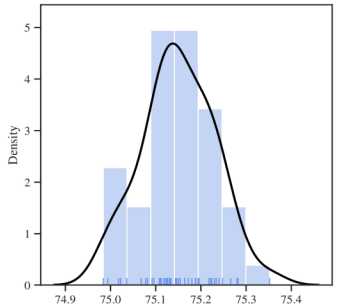


Figure 5: Histogram and density plot of the final validation accuracy on Imagenet for a self-supervised pretrained ResNet50. Each dash at the bottom corresponds to one run.

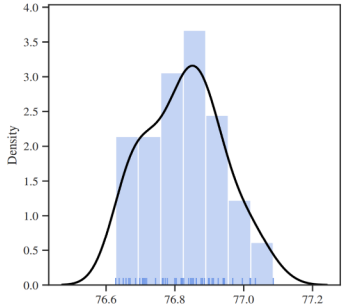


Figure 6: Histogram and density plot of the final validation accuracy on Imagenet for a self-supervised pretrained Visual Transformer. Each dash at the bottom corresponds to one run.

diff: 0.5

Training mode	Accuracy mean ± std	Minimum accuracy	Maximum accuracy
ResNet50	75.48 ± 0.10	75.25	75.69
ResNet50 SSL	75.15 ± 0.08	74.98	75.35
ViT SSL	76.83 ± 0.11	76.63	77.09

Table 2: Results on Imagenet for different architecture and pretraining schemes.

- **Large scale datasets (ImageNet)**
  1) All runs ***share the same initial weights*** resulting from the pretraining stage except for the last layer
  2) Only the composition of batches varies due to the random seed

ImageNet 에서의 Min - Max 차이 0.5 % 는 significant in the computer vision community

3. Does pretraining on larger datasets mitigate variability induced by the choice of seed?

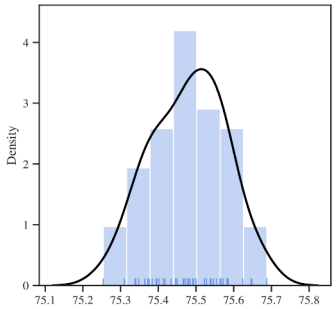


Figure 4: Histogram and density plot of the final validation accuracy on Imagenet for a pretrained ResNet50. Each dash at the bottom corresponds to one run.

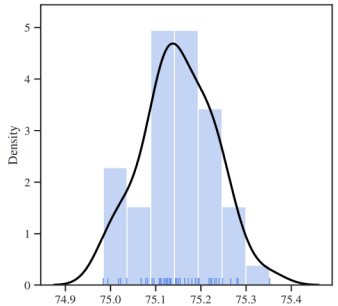


Figure 5: Histogram and density plot of the final validation accuracy on Imagenet for a self-supervised pretrained ResNet50. Each dash at the bottom corresponds to one run.

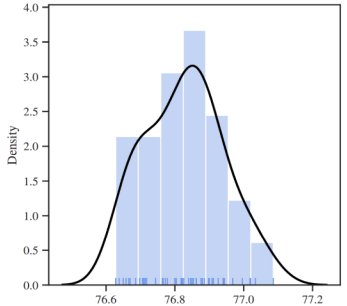


Figure 6: Histogram and density plot of the final validation accuracy on Imagenet for a self-supervised pretrained Visual Transformer. Each dash at the bottom corresponds to one run.

- **Large scale datasets**
  - 50 seeds
  - ⇒ 만약 50 seeds 이상을 실험 했을 경우에는 1% 이상의 차이가 발생했을 수도 있다.

The answer to the third question

- 어떤 경우에는, pretrained models 과 larger training set을 사용하는 것이 seed 선택에 의한 변화를 감소시켜주지만, 그러한 변화 역시 computer vision community 에서 중요하게 여겨진다.

- **최근 많은 pretrained model 을 사용하고 있기 때문에, 다양한 initialization (different pretrained model) 을 사용하는 것은 robust 하지 않다!**

5. Discussion

What is the distribution of scores with respect to the choice of seed?

- The distribution of accuracy when varying seeds : pointy, results are fairly concentrated around the mean. (뾰족하고, 평균 주변에 밀집되어 있음)
- 모델이 수렴한 후에, 이러한 분포는 stable 함 → 특별히 다른 seed 보다 뛰어난 몇 개의 seed 가 있음.

Are there black swans, ie.e, seeds that produce radically different results?

- Yes
- 10000개의 seeds 를 확인했을 때, 최댓값과 최솟값의 차이가 2% 정도임을 확인 하였음.

Does pretraining on larger datasets mitigate variable induced by the choice of seed?

- 확실히 다른 seeds 를 사용했을 때, 그 차이가 줄기는 하지만, 완벽하게 감소시키지는 않음.
- ImageNet 에서 최댓값과 최솟값의 차이가 0.5% 정도임을 확인하였음. ⇒ 이는 커뮤니티에서 여전히 중요하게 여겨지는 차이임.

실험 한계

- Section 3 에서 다뤘던 한계점들이 있지만, 저자는 이와 같은 연구가 realistic setup 이라고 주장!
  ⇒ 실제 연구에서 여러번 실행한 결과를 공유하지 않으며, **lucky setup** 이라고 표현하고 있음.

저자는 다양한 seed 의 평균, 표준 편차, 최소 및 최대 점수를 보고하여 randomness study 를 수행할 것을 강력하게 제안함.

GhostNet: More Features from Cheap Operations (CVPR 2020)

code: https://github.com/huawei-noah/CV-Backbones

GhostNet: More Features from Cheap Operations

Kai Han[1] Yunhe Wang[1] Qi Tian[1]* Jianyuan Guo[2] Chunjing Xu[1] Chang Xu[3]
[1]Noah's Ark Lab, Huawei Technologies. [2]Peking University.
[3]School of Computer Science, Faculty of Engineering, University of Sydney.
{kai.han,yunhe.wang,tian.qi1,xuchunjing}@huawei.com jyguo@pku.edu.cn c.xu@sydney.edu.au

1. Introduction
2. Related Work
3. Approach
4. Experiments
5. Conclusion

- Traditional CNN
  - Deep Convolutional Neural Network 는 computer vision 에서 좋은 성능을 보이고 있음
    - Image recognition, Object detection, Semantic segmentation
  - Traditional CNN 의 경우, 좋은 성능을 위해서 일반적으로 많은 수의 파라미터와 FLOPs 필요함.
    - ResNet-50 : 25.6M parameters, 4.1B FLOPs to process an image of size 224 * 224

- **Portable and efficient network architectures**
  - 최근에는 mobile devices 에서 납득할만한 성능을 가진 가볍고, 효율적인 네트워크 아키텍처 연구 (e.g. smart phones and self-driving cars)
  - **Compact deep neural networks,** 하지만, 이러한 방법은 대부분은 pre-trained deep neural networks 를 사용한 경우이기 때문에 높은 성능을 보임.
    - network pruning
    - low-bit quantization
    - knowledge distillation
  - **Efficient neural architecture design**
    - MobileNet (depthwise and pointwise convolutions)
    - ShuffleNet (channel shuffle operation)

- 잘 학습된 DNN 의 **feature maps abundant and even redundant information** (중복 정보가 있으면서 충분한 경우) 은 모델이 **input data 에 대해서 잘 이해하고 있음을 나타냄. (guarantees a comprehensive understanding of the input data).**

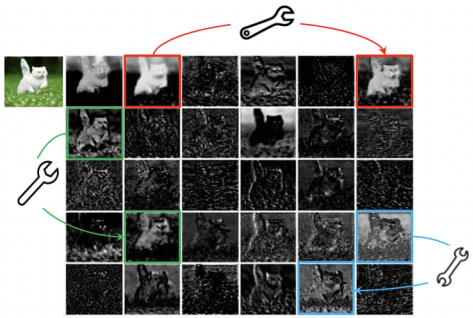


Figure 1. Visualization of some feature maps generated by the first residual group in ResNet-50, where three similar feature map pair examples are annotated with boxes of the same color. One feature map in the pair can be approximately obtained by transforming the other one through cheap operations (denoted by spanners).

Figure1

- ResNet-50 에서 input image 에 대한 몇 개의 feature map.
- **같은 색으로 표시된 feature map 의 경우, similar pairs 로 보임! like a ghost of each another**
- **중복된 feature map 은 성공적인 DNN 에 있어 중요한 특징임. (important characteristic)**
- **본 논문에서는, 중복된 feature map 을 오히려 cost-efficient way 로 수용하고자 함.**

In this paper,

- A novel Ghost module : 적은 파라미터를 사용하여 더 많은 feature 생성
  - Ordinary convolutional layer in deep neural networks will be split into two parts
    1) ordinary convolutions but their total number will be rigorously controlled. (⇒ intrinsic feature maps)
    2) 추가 feature map 들을 생성하기 위해 intrinsic feature maps 에 simple linear operations 적용.

- GhostNet
  - Ghost module 적용
  - Surpass SoTA efficient deep models (e.g. MobileNetV3)

- **Model Compression**
  - model compression 목표 : Reduce the computation, energy and storage cost
  - 종류
    - *Pruning connections* : cuts out the unimportant connections between neurons.
    - *Channel pruning* : further targets on removing useless channels for easier acceleration.
    - *Model quantization* : compression 과 calculation acceleration 을 위해, nn의 weights or activations 을 discrete values 로 표현
    - *Knowledge distillation* : 큰 모델에서 학습한 것을 작은 모델로 지식 전이
  - 이러한 방식들의 성능은 대부분 pre-trained model 성능에 의존함.

- **Compact Model Design**
  - Deploying neural networks on embedded devices.
  - 종류
    - Xception : depthwise convolution operation 사용
    - MobileNets : light weight deep neural networks, depthwise separable convolutions
      - MobileNetV2 : Inverted residual block
      - MobileNetV3 : AutoML technology
    - ShuffleNetV2 : 실제 speed 와 target hardward 성능 고려
  - 적은 FLOPs 으로, 좋은 성능을 보이지만, feature maps 간 correlation 과 redundancy 는 연구된 바 없음.

Ghost Module for More Features

- Figure 1 에서 보이는 것처럼, feature maps 간 redundancy 가 존재하며,
  본 논문에서는 이러한 연산의 감소하고자 함. (to reduce the required resources)
  즉, 중복된 feature map 을 생성하는 convolution filters 를 제거하고자 함.

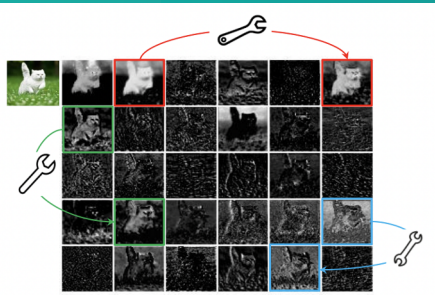
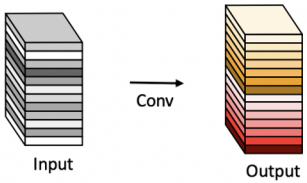
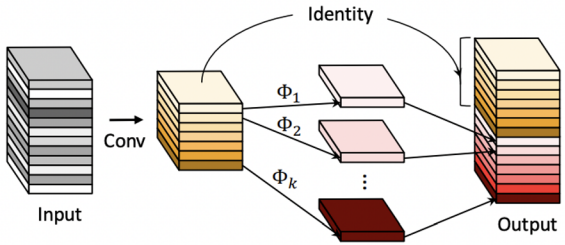


Figure 1. Visualization of some feature maps generated by the first residual group in ResNet-50, where three similar feature map pair examples are annotated with boxes of the same color. One feature map in the pair can be approximately obtained by transforming the other one through cheap operations (denoted by spanners).



(a) The convolutional layer.



(b) The Ghost module.

Figure 2. An illustration of the convolutional layer and the proposed Ghost module for outputting the same number of feature maps. Φ represents the cheap operation.

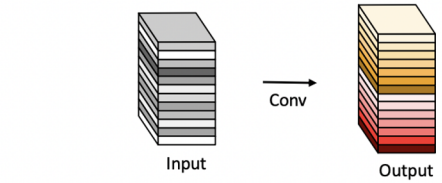
(a) The convolutional layer

the input data $X \in \mathbb{R}^{c \times h \times w}$,

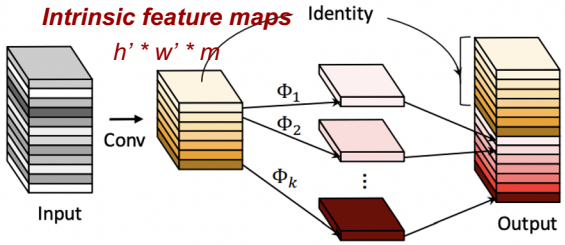
$$Y = X * f + b, \qquad (1)$$

$Y \in \mathbb{R}^{h' \times w' \times n}$ is the output feature map with n channels,

number of FLOPs can be calculated as $n \cdot h' \cdot w' \cdot c \cdot k \cdot k$, which is often as large as hundreds of thousands since the number of filters n and the channel number c are generally very large (*e.g.* 256 or 512).



(a) The convolutional layer.



(b) The Ghost module.

Figure 2. An illustration of the convolutional layer and the proposed Ghost module for outputting the same number of feature maps. Φ represents the cheap operation.

⇒ 전체 파라미터 수는 input and output feature maps 의 dimensions 에 영향을 많이 받음.

또한, dimension 은 Figure 1 에서 볼 수 있듯이, 중복되거나 비슷한 output feature map 을 포함하기도 함.

It if unnecessary to generate these redundant feature maps one by one with large number of FLOPs and parameters!

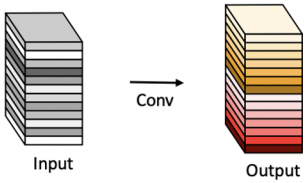
(b) The Ghost module

- **Intrinsic feature maps *(dim: h'*w'*m)***
  - smaller size and produces by ordinary convolution filters

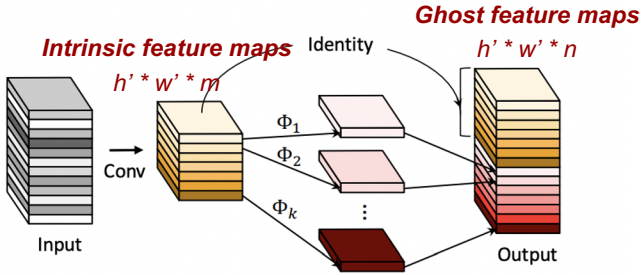
convolution filters. Specifically, m intrinsic feature maps $Y' \in \mathbb{R}^{h' \times w' \times m}$ are generated using a primary convolution:

$$Y' = X * f', \tag{2}$$

where $f' \in \mathbb{R}^{c \times k \times k \times m}$ is the utilized filters, $m \leq n$ and the bias term is omitted for simplicity. The hyper-parameters



(a) The convolutional layer.



(b) The Ghost module.

Figure 2. An illustration of the convolutional layer and the proposed Ghost module for outputting the same number of feature maps. Φ represents the cheap operation.

It if unnecessary to generate these redundant feature maps one by one with large number of FLOPs and parameters!

(b) The Ghost module

- **Linear operations**
  - n개의 feature map 을 얻기 위해서, 각각의 intrinsic feature 에 linear operations 적용

$$y_{ij} = \Phi_{i,j}(y'_i), \quad \forall\, i = 1, ..., m, \quad j = 1, ..., s, \qquad (3)$$

  - vi' 은 하나 이상의 ghost feature maps 을 가질 수 있음.
  - The last $\Phi_{i,s}$ 는 intrinsic feature maps 을 보존하기 위해서 identity mapping
  - Linear operations 는 ordinary convolution 보다 computational cost 가 적게 든다!!
  - Ghost module 에서 다른 linear operations 사용 가능
    - 3*3, 5*5 linear kernels (d)

Difference From Existing Methods

1) 기존의 1 * 1 pointwise convolution widely 사용하는 대신, Ghost module 은 kernel size 를 customize 가능함 (e.g. 3*3, 5*5)
2) 기존에는 channel에 pointwise convolutions 을 적용한 후, depthwise convolution 을 적용했는데, Ghost module 은 ordinary convolution 을 처음 intrinsic feature maps 을 생성하는데 사용하고, cheap linear operations 을 적용함.
3) 기존에는 각각의 feature map의 계산 방식이 depthwise convolution, shift operation 으로 제한되었는데, linear operations in Ghost modeuls 은 다양성을 가짐.
4) 추가적으로, identity mapping 이 병렬적으로 존재하여, intrinsic feature maps 을 보존할 수 있음.

Analysis on Complexities

- speed-up ratio

$$r_s = \frac{n \cdot h' \cdot w' \cdot c \cdot k \cdot k}{\frac{n}{s} \cdot h' \cdot w' \cdot c \cdot k \cdot k + (s-1) \cdot \frac{n}{s} \cdot h' \cdot w' \cdot d \cdot d} = \frac{c \cdot k \cdot k}{\frac{1}{s} \cdot c \cdot k \cdot k + \frac{s-1}{s} \cdot d \cdot d} \approx \frac{s \cdot c}{s + c - 1} \approx s, \tag{4}$$

- compression ratio

$$r_c = \frac{n \cdot c \cdot k \cdot k}{\frac{n}{s} \cdot c \cdot k \cdot k + (s-1) \cdot \frac{n}{s} \cdot d \cdot d} \approx \frac{s \cdot c}{s + c - 1} \approx s, \tag{5}$$

- speed-up ratio, compression ratio 는 s (linear transform 의 개수) 와 비례함.

Building Efficient CNNs - Ghost Bottlenecks (G-bneck)

- Resnet의 residual block 과 유사한 형태.
  - several convolutional layers + shortcuts

- 제안된 G-bneck 은 two stacked Ghost modules 로 구성됨.
  1) 첫 번째 ghost module: Expansion layer, channels 수를 증가시킴.
  2) 두 번째 ghost module: shortcut path 와 match 하기 위해, channels 수 감소.
  - 그 후, shortcut 은 input 과 output connect.

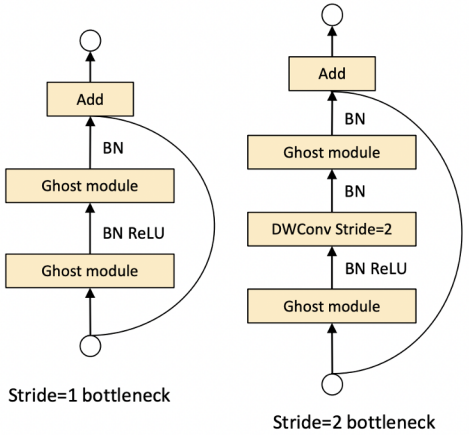


Figure 3. Ghost bottleneck. Left: Ghost bottleneck with stride=1; right: Ghost bottleneck with stride=2.

- Batch normalization (BN), ReLU nonlinearity 는 모든 layer 에 적용
  except, ReLU 는 두 번째 Ghost module 이후에는 사용되지 않음. (suggested by MobileNetV2)

- Stride
  - Stride = 1
  - Stride = 2
    - shortcut path 는 downsampling layer 로 구현되고,
      depthwise convolution (with stride = 2) 가 두 Ghost modules 사이에 포함됨.

궁금) 채널 수를 확장했다가, 채널 수를 감소시키는 이유는?

Table 1. Overall architecture of GhostNet. G-bneck denotes Ghost bottleneck. #exp means expansion size. #out means the number of output channels. SE denotes whether using SE module.

Input	Operator	#exp	#out	SE	Stride
$224^2 \times 3$	Conv2d 3×3	-	16	-	2
$112^2 \times 16$	G-bneck	16	16	-	1
$112^2 \times 16$	G-bneck	48	24	-	2
$56^2 \times 24$	G-bneck	72	24	-	1
$56^2 \times 24$	G-bneck	72	40	1	2
$28^2 \times 40$	G-bneck	120	40	1	1
$28^2 \times 40$	G-bneck	240	80	-	2
$14^2 \times 80$	G-bneck	200	80	-	1
$14^2 \times 80$	G-bneck	184	80	-	1
$14^2 \times 80$	G-bneck	184	80	-	1
$14^2 \times 80$	G-bneck	480	112	1	1
$14^2 \times 112$	G-bneck	672	112	1	1
$14^2 \times 112$	G-bneck	672	160	1	2
$7^2 \times 160$	G-bneck	960	160	-	1
$7^2 \times 160$	G-bneck	960	160	1	1
$7^2 \times 160$	G-bneck	960	160	-	1
$7^2 \times 160$	G-bneck	960	160	1	1
$7^2 \times 160$	Conv2d 1×1	-	960	-	1
$7^2 \times 960$	AvgPool 7×7	-	-	-	-
$1^2 \times 960$	Conv2d 1×1	-	1280	-	1
$1^2 \times 1280$	FC	-	1000	-	-

Building Efficient CNNs - GhostNet

- GhostNet 은 Ghost bottlenecks with the Ghost modules 들로 구성되어 있음.

- 첫 번째 layer : 기본 convolutional layer (with 16 filters)
- 이후로, Ghost bottlenecks with gradually increased channels
  - Ghost bottleneck은 input feature maps 사이즈에 따라 다른 stage 로 grouping
  - 각 stage 마지막 layer 만 stride=2, 나머지는 모두 stride=1
- 마지막 최종 average pooling, convolutional layer 는 final classification 을 위해서 1280-dimensional feature vector 사용.

- SE : squeeze and excite module

Datasets and Settings

- CIFAR-10, ImageNet ILSVRC 2012 dataset, MS COCO object detection benchmark

Efficiency of Ghost Module (Toy Experiments)

- Observe the reconstruction error between raw feature maps and the generated ghost feature maps.

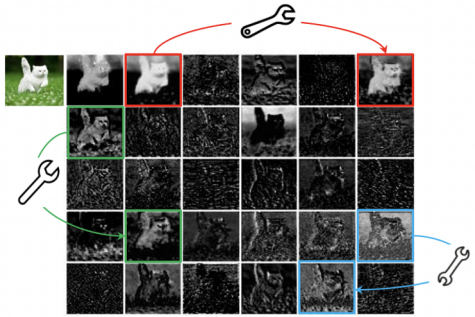


Figure 1. Visualization of some feature maps generated by the first residual group in ResNet-50, where three similar feature map pair examples are annotated with boxes of the same color. One feature map in the pair can be approximately obtained by transforming the other one through cheap operations (denoted by spanners).

- Features are extracted using the first residual block of ResNet-50
- 3개 pair 사용

- 두 개의 input, output (in 1 pair) 간의 linear operation Φ 측정.
  - utilize a small depthwise convolution.
- 각 pair 의 MSE 는 convolution kernel size 에 따라 Table 2 와 같음.

Table 2. MSE error *v.s.* different kernel sizes.

MSE (10^{-3})	d=1	d=3	d=5	d=7
red pair	4.0	3.3	3.3	3.2
green pair	25.0	24.3	24.1	23.9
blue pair	12.1	11.2	11.1	11.0

⇒ MSE 값이 매우 작음.
즉, feature map 들 간의 redundant 가 높고, 이러한 redundant feature maps 은 intrinsic feature maps 으로부터 생성될 수 있음을 보임.

We suggest to let *d* in a Ghost module be a fixed value and utilize depthwise convolution.

- Ghost module 의 두 하이퍼 파라미터 s, d 실험.
  - s for generating *m* = *n/s* intrinsic feature maps
  - kernel size *d*d* of linear operations for calculating ghost feature maps

s=2

Table 3. The performance of the proposed Ghost module with different d on CIFAR-10.

d	Weights (M)	FLOPs (M)	Acc. (%)
VGG-16	15.0	313	93.6
1	7.6	157	93.5
3	7.7	158	93.7
5	7.7	160	93.4
7	7.7	163	93.1

kernels of 1*1 : spatial information 전달 X
5*5, 7*7 : overfitting

d=3

Table 4. The performance of the proposed Ghost module with different s on CIFAR-10.

s	Weights (M)	FLOPs (M)	Acc. (%)
VGG-16	15.0	313	93.6
2	7.7	158	93.7
3	5.2	107	93.4
4	4.0	80	93.0
5	3.3	65	92.9

s 는 computational cost 과 크게 관련있음.
즉, s 가 커지면, larger compression, FLOPs are reduces, accuracy decreases

FLOPs (FLoating point **OP**eration**s):** 실제 계산량을 나타내는 단위
FLOPS (FLoating point **OP**erations per **Se**cond**, flop/s):** 얼마나 빨리 계산을 처리할 수 있는지에 대한 단위

In this section, we first replace the original convolutional layers by the proposed Ghost module to verify its effectiveness. Then, the GhostNet architecture built using the new module will be further tested on the image classification and object detection benchmarks.

Comparison with SoTA

Table 5. Comparison of state-of-the-art methods for compressing VGG-16 and ResNet-56 on CIFAR-10. - represents no reported results available.

Model	Weights	FLOPs	Acc. (%)
VGG-16	15M	313M	93.6
ℓ_1-VGG-16 [31, 37]	5.4M	206M	93.4
SBP-VGG-16 [18]	-	136M	92.5
Ghost-VGG-16 (s=2)	7.7M	158M	**93.7**
ResNet-56	0.85M	125M	93.0
CP-ResNet-56 [18]	-	63M	92.0
ℓ_1-ResNet-56 [31, 37]	0.73M	91M	92.5
AMC-ResNet-56 [17]	-	63M	91.9
Ghost-ResNet-56 (s=2)	0.43M	63M	**92.7**

Table 6. Comparison of state-of-the-art methods for compressing ResNet-50 on ImageNet dataset.

Model	Weights (M)	FLOPs (B)	Top-1 Acc. (%)	Top-5 Acc. (%)
ResNet-50 [16]	25.6	4.1	75.3	92.2
Thinet-ResNet-50 [39]	16.9	2.6	72.1	90.3
NISP-ResNet-50-B [59]	14.4	2.3	-	90.8
Versatile-ResNet-50 [49]	11.0	3.0	74.5	91.8
SSS-ResNet-50 [23]	-	2.8	74.2	91.9
Ghost-ResNet-50 (s=2)	13.0	2.2	**75.0**	**92.3**
Shift-ResNet-50 [53]	6.0	-	70.6	90.1
Taylor-FO-BN-ResNet-50 [41]	7.9	1.3	71.7	-
Slimmable-ResNet-50 0.5× [58]	6.9	1.1	72.1	-
MetaPruning-ResNet-50 [36]	-	1.0	73.4	-
Ghost-ResNet-50 (s=4)	6.5	1.2	**74.1**	**91.9**

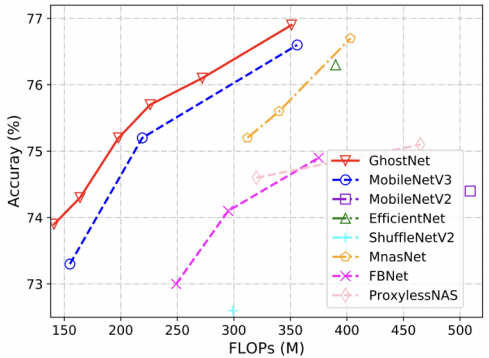


Figure 6. Top-1 accuracy *v.s.* FLOPs on ImageNet dataset.

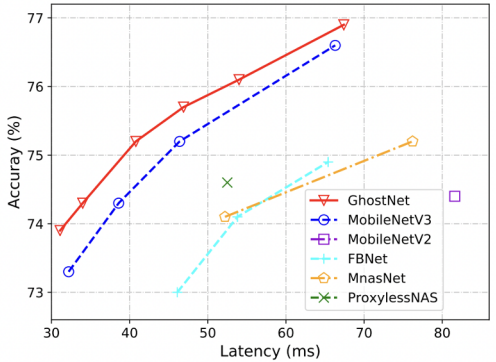


Figure 7. Top-1 accuracy *v.s.* latency on ImageNet dataset.

ImageNet Classification

Table 7. Comparison of state-of-the-art small networks over classification accuracy, the number of weights and FLOPs on ImageNet dataset.

Model	Weights (M)	FLOPs (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
ShuffleNetV1 0.5× (g=8) [61]	1.0	40	58.8	81.0
MobileNetV2 0.35× [44]	1.7	59	60.3	82.9
ShuffleNetV2 0.5× [40]	1.4	41	61.1	82.6
MobileNetV3 Small 0.75× [20]	2.4	44	65.4	-
GhostNet 0.5×	2.6	42	**66.2**	**86.6**
MobileNetV1 0.5× [21]	1.3	150	63.3	84.9
MobileNetV2 0.6× [44, 40]	2.2	141	66.7	-
ShuffleNetV1 1.0× (g=3) [61]	1.9	138	67.8	87.7
ShuffleNetV2 1.0× [40]	2.3	146	69.4	88.9
MobileNetV3 Large 0.75× [20]	4.0	155	73.3	-
GhostNet 1.0×	5.2	141	**73.9**	**91.4**
MobileNetV2 1.0× [44]	3.5	300	71.8	91.0
ShuffleNetV2 1.5× [40]	3.5	299	72.6	90.6
FE-Net 1.0× [6]	3.7	301	72.9	-
FBNet-B [52]	4.5	295	74.1	-
ProxylessNAS [2]	4.1	320	74.6	92.2
MnasNet-A1 [47]	3.9	312	75.2	92.5
MobileNetV3 Large 1.0× [20]	5.4	219	75.2	-
GhostNet 1.3×	7.3	226	**75.7**	**92.7**

Object Detection

Table 8. Results on MS COCO dataset.

Backbone	Detection Framework	Backbone FLOPs	mAP
MobileNetV2 1.0× [44]	RetinaNet	300M	26.7%
MobileNetV3 1.0× [20]		219M	26.4%
GhostNet 1.1×		164M	26.6%
MobileNetV2 1.0× [44]	Faster R-CNN	300M	27.5%
MobileNetV3 1.0× [20]		219M	26.9%
GhostNet 1.1×		164M	26.9%

End of Document

Thank you