ICML | 2019
Thirty-sixth International
Conference on Machine Learning

LegoNet: Efficient Convolutional Neural Networks with Lego Filters

Zhaohui Yang[1,2,*] Yunhe Wang[2] Hanting Chen[1,2,*] Chuanjian Liu[2]
Boxin Shi[3,4] Chao Xu[1] Chunjing Xu[2] Chang Xu[5]

[1]Laboratory of Machine Perception (Ministry of Education), Peking University
[2]Huawei Noah's Ark Lab [3]Peng Cheng Laboratory
[4]National Engineering Laboratory For Video Technology, Peking University
[5]School of Computer Science, University of Sydney
[*]This work was down when Zhaohui Yang and Hanting Chen were interns at Huawei Noah's Ark Lab

Goal

- **Motivation**
  Reuse patterns
- **Targeted**
  Build efficient CNN using a set of Lego Filters
- **Lego Filters**
  Standard convolution filters are established by a set of shared filters
- **Optimization**
  End-to-end optimization, Straight Through Estimator
- **Efficient Inference**
  Split-Transform-Merge strategy

(a) conv filters (b) Lego filters (c) stacked filters

Lego Filters

- **Lego Filters *B***

  B = {B_1, …, B_m}

- **Standard convolution filters *F***

  F = G(B_1, …, B_m), 4-D tensor

  G is a generation function.

- **Compression condition**

  |G| + |B| ≤ |F|

- **G in LegoNet**

  Combination

(a) conv filters (b) Lego filters (c) stacked filters

m: the number of Lego Filters
B: lego Filters
F: standard convolution filters
G: generation function

Optimization

- **Targeted**

$$\min_{\mathbf{B},\mathbf{M}^j} \sum_{i=1}^{o} \frac{1}{2} ||\mathbf{Y}^j - \mathbf{X}_i^\top (\mathbf{B}\mathbf{M}_i^j)||_F^2,$$
$$s.t.\ \mathbf{M}_i^j \in \{0,1\}^{m\times 1},\ \ ||\mathbf{M}_i^j||_1 = 1, i = 1, ..., o.$$

- **Optimize Lego Filters *B***

  Standard BP algorithm

- **Optimize Binary matrix *M***

  Float type proxy weight *N*

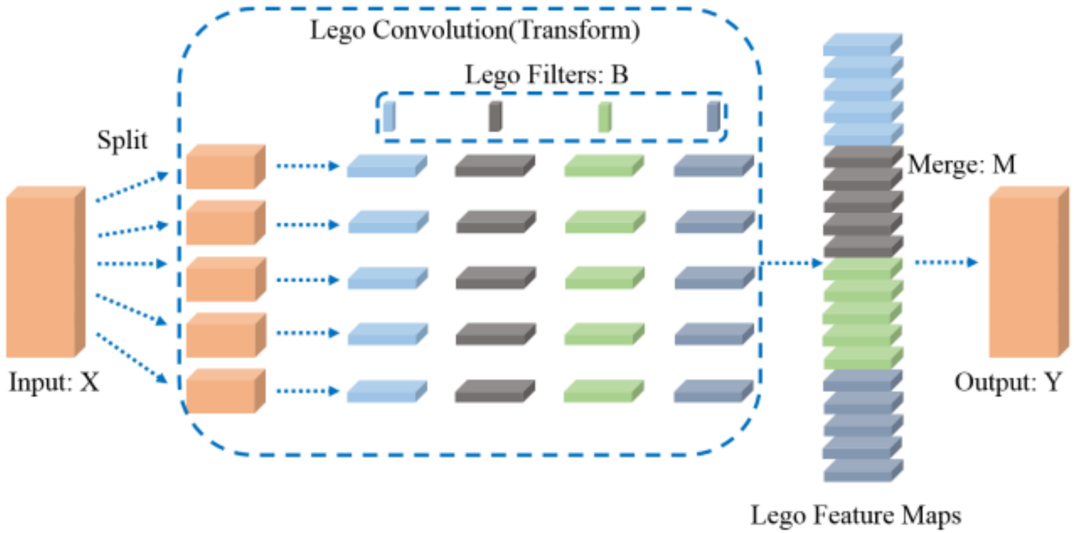
  Straight Through Estimator (STE)

$$\mathbf{M}_{i,k}^j = \begin{cases} 1, & if\ k = \arg\max \mathbf{N}_i^j \\ 0, & otherwise \end{cases}$$
$$s.t.\ \ j = 1, \ldots, n,\ \ i = 1, \ldots, o.$$

m: the number of Lego Filters
B: Lego Filters
M: binary index matrix
N: proxy matrix of *M*

Lego Unit & Efficient Inference

- **Split**
  - Split input feature maps *X*
- **Transform**
  - Convolve feature fragments X = $\{X_1, \ldots, X_o\}$ with Lego Filters *B* = $\{B_1, \ldots, B_m\}$
- **Merge**
  - Combine Lego Feature Maps according to learnt combination matrix *M*



m: the number of Lego Filters
o: split number
B: Lego Filters
M: binary index matrix

Analysis

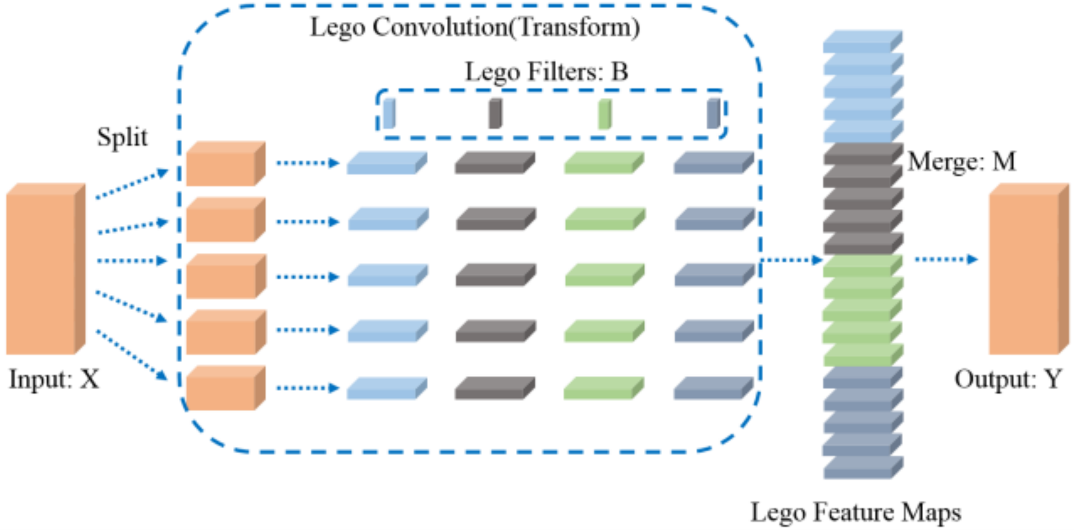
- **Compression**

$$\frac{n \times c \times d^2}{m \times \frac{c}{o} \times d^2 + n \times o \times m} \approx \frac{n \times o}{m}.$$

- **Acceleration**

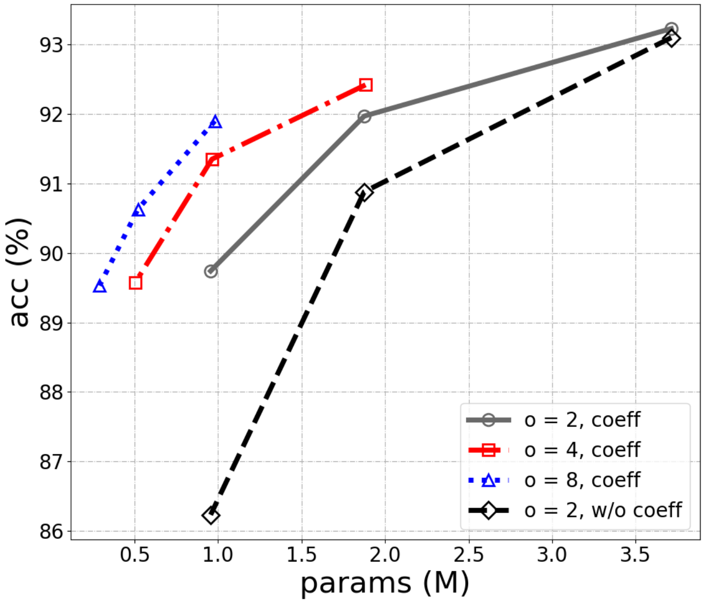
$$\frac{n \times c \times d^2 \times d_x^2}{m \times o \times \frac{c}{o} \times d^2 \times d_x^2 + n \times o \times d_x^2} \approx \frac{n}{m}.$$

  - Condition
    - m ≤ n



m: the number of Lego Filters
n: the number of output channels
o: split number
M: binary index matrix

Experiments



CIFAR-10

Model	Acc (%)	Params(M)	Comp ratio	FLOPS(M)	Speed Up
VGGNet-16(Simonyan and Zisserman, 2014)	93.25	14.7	1×	298.7	1×
Lego-VGGNet-16-w(o=2,m=0.5)	93.23	3.7	4×	149.4	2×
Lego-VGGNet-16-w(o=2,m=0.25)	91.97	1.9	8×	74.7	4×
Lego-VGGNet-16-w(o=4,m=0.5)	92.42	1.9	8×	149.4	2×
Lego-VGGNet-16-w(o=4,m=0.25)	91.35	0.9	16×	74.7	4×

ImageNet

Model	Top-5 Acc(%)	Params(M)	Comp Ratio	FLOPs(B)	Speed Up
ResNet50 (He et al., 2016)	92.2	25.6	1.0×	4.1	1.0×
ThiNet-Res (Luo et al., 2017a)	88.3	8.7	2.9×	2.2	1.9×
Versatile (Wang et al., 2018a)	91.8	11.0	2.3×	3.0	1.4×
Lego-Res50(o=2,m=0.5)	89.7	8.1	3.2×	2.0	2.0×
Lego-Res50-w(o=2,m=0.5)	90.6	8.1	3.2×	2.0	2.0×
Lego-Res50-w(o=2,m=0.6)	91.3	9.3	2.8×	2.0	1.7×
VGGNet-16 (Simonyan and Zisserman, 2014)	90.1	138.0	1.0×	15.3	1.0×
ThiNet-VGG (Luo et al., 2017a)	90.3	38.0	3.6×	3.9	3.9×
Lego-VGGNet-16-w(o=2,m=0.5)	88.9	4.2	32.9×	7.7	2.0×
Lego-VGGNet-16-w(o=2,m=0.6)	89.2	5.0	27.6×	9.2	1.7×
MobileNet (Howard et al., 2017)	88.9	4.2	1.0×	0.6	1.0×
Lego-Mobile-w(o=2,m=0.9)	87.5	2.5	1.7×	0.5	1.1×
Lego-Mobile-w(o=2,m=1.5)	88.3	3.5	1.2×	0.6	1.0×

Combination with coefficients is important while stacking Lego Filters.
Given same model size, larger split number o results in higher performance (larger FLOPs)

Conclusion & Future Research

- **Conclusion**
  - Proposed Lego Filters for constructing efficient CNN.
  - End-to-end optimization.
  - Split-transform-merge three-stage strategy.
- **Future Research**
  - Parameter in Parameter (use a set of Lego Filters and a small NN to generate 4-D convolution filters)
  - Global LegoNet (view network parameters as a whole 4-D tensor)

Thanks!