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
Lego Filters

- **Lego Filters** $B$
  \[ B = \{B_1, ..., B_m\} \]

- **Standard convolution filters** $F$
  \[ F = G(B_1, ..., B_m), \text{4-D tensor} \]
  \[ G \text{ is a generation function.} \]

- **Compression condition**
  \[ |G| + |B| \leq |F| \]

- **$G$ in LegoNet**
  Combination

---

$m$: the number of Lego Filters

$B$: lego Filters

$F$: standard convolution filters

$G$: generation function
Optimization

- Targeted

\[
\min_{B,M} \sum_{i=1}^{o} \frac{1}{2} \|Y^j - X_i^T(BM_i^j)\|_F^2,
\]

s.t. \(M_i^j \in \{0,1\}^{m \times 1}\), \(\|M_i^j\|_1 = 1, i = 1, \ldots, o\).

- Optimize Lego Filters \(B\)

Standard BP algorithm

- Optimize Binary matrix \(M\)

Float type proxy weight \(N\)

Straight Through Estimator (STE)

\[
M_{i,k}^j = \begin{cases} 
1, & \text{if } k = \arg \max N_i^j \\
0, & \text{otherwise}
\end{cases} 
\]

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 $X = \{X_1, ..., X_o\}$ with Lego Filters $B = \{B_1, ..., B_m\}$

- **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**
  - \(m \leq n\)

\[m: \text{the number of Lego Filters}\]
\[n: \text{the number of output channels}\]
\[o: \text{split number}\]
\[M: \text{binary index matrix}\]
Combination with coefficients is important while stacking Lego Filters. Given same model size, larger split number o results in higher performance (larger FLOPs)
**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!