N2N Learning: Network to Network Compression via Policy Gradient Reinforcement Learning

Recent trends in Automated Machine Learning
June 9th, 2021
Maximilian Gartner
Motivation

- Neural Networks are getting bigger and bigger
  - Require high power, memory & computational resources
  - Not feasible on smaller devices (smartphones, smart-home devices, ...)

- Goal: Find a small, but still high-performance architecture
Main Idea

- Reinforcement Learning approach
- Take larger network (Teacher model) and identify compressed high-performance smaller network (Student model)
Underlying Concepts

(I) Knowledge Distillation

- Model compression approach
- Student is trained to learn the exact behavior of the (pre-trained) teacher by trying to replicate its output
Underlying Concepts

(II) Neural Network Pruning

• Given the teacher model, find a much smaller subset that can provide same accuracy

• Preserve what matters most, remove redundancy
N2N Process

- Two step reinforcement learning procedure
  - First step: Layer removal
  - Second step: Layer shrinkage

Therefore: Encode layers of teacher

- E.g., Convolutional layer:
  \[ x_t = (l, k, s, p, n) \]
  where \( l \) layer type, \( k \) filter size, \( s \) stride, \( p \) padding, \( n \) number of filters
Layer Removal (Step 1)

- Input: Teacher model
- Outputs for each layer an action $a_t \in \{0, 1\}$, whether to keep or remove the corresponding layer
- $\pi_{\text{remove}} = (a_t \mid x_t, h_{t-1}, h_{t+1})$
- Underlying policy network: bi-directional LSTM
Layer Shrinkage (Step 2)

- **Input**: output model of Step 1
- **Outputs**: for each configuration variable of each layer an action $a_t \in [0.1, 0.2, ..., 1]$, deciding about how much to shrink the parameter
- $\pi_{\text{shrink}} = (a_t \mid x_t, a_{t-1}, h_{t-1})$
- **Underlying policy network**: LSTM

$$x_t = (5, 1, 1, 64)$$

$$a_t = 0.6$$
$$a_{t+1} = 1$$
$$a_{t+2} = 1$$
$$a_{t+3} = 0.4$$

$$x'_t = (3, 1, 1, 26)$$
Goal: Knowledge Distillation of the teacher model

- Not only train with hard labels (groundtruth) $y_{true}$, but also with logits $z$ of the teacher

\[
\mathcal{L}(W) = \mathcal{L}_{hard}(f(x; W), y_{true}) + \lambda \cdot \mathcal{L}_{KD}(f(x; W), z)
\]

- e.g., cross-entropy loss
- e.g., mean squared error MSE

N2N Learning
June 9th, 2021
Reward Function - Requirements

- Calculation at the end with resulting trained student model
- Reward should depend on
  - How much is the student compressed?
  - How good is the student?
- Also:
  - Set $R \leftarrow -1$ for degenerated student models

\[
\text{Model with low compression} + \text{high accuracy} > \text{Model with high compression} + \text{low accuracy}
\]
The relative compression $C_\tau \in [0, 1)$ is defined as $C_\tau = 1 - \frac{\#\text{params(student)}}{\#\text{params(teacher)}}$

Given $C_\tau$, the compression reward $R_{c,\tau}$ is defined as $R_{c,\tau} = C_\tau (2 - C_\tau)$
Reward Function

The relative compression $C_\tau \in [0, 1)$ is defined as $C_\tau = 1 - \frac{\#\text{params}(\text{student})}{\#\text{params}(\text{teacher})}$

Given $C_\tau$, the compression reward $R_{c,\tau}$ is defined as $R_{c,\tau} = C_\tau (2 - C_\tau)$

Given the validation accuracy of the with trajectory $\tau$ produced student model $A_{\text{student}}$ and the (fixed) accuracy of the teacher model $A_{\text{teacher}}$, the accuracy reward $R_{a,\tau}$ is defined as

$$R_{a,\tau} = \frac{A_{\text{student}}}{A_{\text{teacher}}}$$

$$R_\tau = R_{c,\tau} \cdot R_{a,\tau} = C_\tau (2 - C_\tau) \cdot \frac{A_{\text{student}}}{A_{\text{teacher}}}$$
Reward Function

The relative compression $C_\tau \in [0, 1)$ is defined as $C_\tau = 1 - \frac{\#\text{params(stu\text{}}dent)}{\#\text{params(teacher)}}$

$$R_\tau = R_{c, \tau} \cdot R_{a, \tau} = C_\tau (2 - C_\tau) \cdot \frac{A_{\text{student}}}{A_{\text{teacher}}}$$

Example:

$$R_{a, \tau} = 0.75, C_\tau = 0.25 \quad \Rightarrow \quad R_\tau \approx 0.328$$

$$R_{a, \tau} = 0.25, C_\tau = 0.75 \quad \Rightarrow \quad R_\tau \approx 0.234$$
Reward Function

\[ R_\tau = R_{c,\tau} \cdot R_{a,\tau} = C_\tau (2 - C_\tau) \cdot \frac{A_{\text{student}}}{A_{\text{teacher}}} \]

Example:

\[ R_{a,\tau} = 0.75, C_\tau = 0.25 \quad \Rightarrow \quad R_\tau \approx 0.328 \]

\[ R_{a,\tau} = 0.25, C_\tau = 0.75 \quad \Rightarrow \quad R_\tau \approx 0.234 \]
Optimization

- Optimize two policies
  - Bi-directional LSTM with parameters $\theta_{\text{remove}}$
  - LSTM with parameters $\theta_{\text{shrink}}$

- Maximize expected reward over all sequences of actions:
  $$\max_{\theta} J(\theta), \quad J(\theta) = E_{a_1:T \sim P_{\theta}} (R)$$

- Optimization: with REINFORCE
  $$\nabla_{\theta} J(\theta) = \nabla_{\theta} E_{a_1:T \sim P_{\theta}} (R)$$
  $$= \sum_{t=1}^{T} E_{a_1:T \sim P_{\theta}} [\nabla_{\theta} \log P_{\theta} (a_t | a_{1:(t-1)}) R]$$
  $$\approx \frac{1}{m} \sum_{k=1}^{m} \sum_{t=1}^{T} [\nabla_{\theta} \log P_{\theta} (a_t | h_t) R_k]$$
Experiments

- Teacher: VGG-13 (9.4 Mio Params)
- Dataset: MNIST
- Epochs Training Student: 5

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Acc.</th>
<th>#Params</th>
<th>Δ Acc.</th>
<th>Compr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-13 Teacher</td>
<td>99.54%</td>
<td>9.4M</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>VGG-13 Student</td>
<td>99.55%</td>
<td>73K</td>
<td>+0.01%</td>
<td>127x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc.</th>
<th>#Params</th>
<th>Compr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher (MNIST/VGG-13)</td>
<td><strong>99.54%</strong></td>
<td>9.4M</td>
<td>1x</td>
</tr>
<tr>
<td>Student (Stage 1 &amp; 2)</td>
<td>98.91%</td>
<td>17K</td>
<td>553x</td>
</tr>
</tbody>
</table>
Experiments

- Teacher: VGG-19 (20.2 Mio Params)
- Dataset: CIFAR-10
- Epochs Training Student: 5

<table>
<thead>
<tr>
<th>CIFAR-10</th>
<th>Teacher</th>
<th>91.97%</th>
<th>20.2M</th>
<th>—</th>
<th>—</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-19 Teacher</td>
<td>91.97%</td>
<td>20.2M</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>VGG-19 Student</td>
<td>92.05%</td>
<td>1.7M</td>
<td>+0.08%</td>
<td>11.8x</td>
<td>—</td>
</tr>
<tr>
<td>VGG-19 Student</td>
<td>91.64%</td>
<td>984K</td>
<td>-0.33%</td>
<td>20.53x</td>
<td>—</td>
</tr>
</tbody>
</table>

Stage 1 (removal):

Stage 2 (removal & shrinkage):
Evaluation

- NN become more applicable, not only for real world application, but also a wider group of AI researchers
- Low student training cost
- Pruning and knowledge distillation better than with hand-crafted techniques

Accuracy-Performance Tradeoff

- High policy training costs
  Also: Policy update after 5 epochs of training

Search space limitations:
Student will always be similar to the teacher
References

- N2N Learning: Network to Network Compression via Policy Gradient Reinforcement Learning, Ashok et al.
- Neural Architecture Search with Reinforcement Learning, Zoph et al.
- https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764
- https://medium.com/ai%C2%B3-theory-practice-business/reinforcement-learning-part-1-a-brief-introduction-a53a849771cf
Thank you!