Neural Optimizer Search with Reinforcement Learning

Irwan Bello\textsuperscript{1} Barret Zoph\textsuperscript{1} Vijay Vasudevan\textsuperscript{1} Quoc V. Le\textsuperscript{1}

\textsuperscript{1}Google Brain

ICLR, 2017/ Presenter: Anant Kharkar
1. Introduction
   - Motivation
   - Approach

2. Methods
   - Domain-Specific Language
   - Controller RNN

3. Experiments
   - Optimizer Discovery
   - Transfer Experiment

4. Summary
Outline

1 Introduction
   • Motivation
   • Approach

2 Methods
   • Domain-Specific Language
   • Controller RNN

3 Experiments
   • Optimizer Discovery
   • Transfer Experiment

4 Summary
Motivation

Classical optimizers:

- SGD
- SGD w/Momentum
- Adam
- RMSProp

Combination of stochastic methods and heuristic approximations

Want to automate process of generating update rules
Produce equation, not just numerical updates
Outline

1 Introduction
   • Motivation
   • Approach

2 Methods
   • Domain-Specific Language
   • Controller RNN

3 Experiments
   • Optimizer Discovery
   • Transfer Experiment

4 Summary
**Approach**

- RNN controller produces update rule string
- Controller updated based on performance of optimizer
- RL approach to training

How to generate update rules? First define space of update rules
Previous Work

- LSTM for numerical updates (Andrychowicz et al., 2016)
  - Equations are more transferrable
- Genetic programming for update equations (Orchard & Wang, 2016)
  - Slow and needs heuristics
- Neural Architecture Search (Zoph & Le, 2017) - seen earlier
  - RNN produces network architecture
Outline

1. Introduction
   - Motivation
   - Approach

2. Methods
   - Domain-Specific Language
   - Controller RNN

3. Experiments
   - Optimizer Discovery
   - Transfer Experiment

4. Summary
Domain-Specific Language

- Each optimizer has computational graph - binary expression tree

Components:
- 2 operands
- Unary function for each operand
- Binary function to combine

\[ \Delta w = \lambda \ast b(u_1(op_1), u_2(op_2)) \]
Outline

1 Introduction
   • Motivation
   • Approach

2 Methods
   • Domain-Specific Language
   • Controller RNN

3 Experiments
   • Optimizer Discovery
   • Transfer Experiment

4 Summary
Controller RNN

Trained with Adam

Softmax

Hidden state

Embedding

1st operand

2nd operand

Unary ops

Unary ops

Binary ops

1st operand

2nd operand

Objective function: $J(\theta) = \mathbb{E}_{\Delta \sim p_\theta(.)}[R(\Delta)]$

Optimize reward (accuracy of target model)
Search Space

- **Operands**
  - Gradients: $g, g^2, g^3, \text{sign}(g)$
  - Moving averages: $\hat{m}, \hat{v}, \hat{y}, \text{sign}(\hat{m})$
  - Weights: $10^{-4}w, 10^{-3}w, 10^{-2}w, 10^{-1}w$
  - ADAM, RMSProp, 1, small noise

- **Unary Functions**
  - $x, -x, e^x, \log|x|, \text{clip}, \text{drop}, \text{sign}$

- **Binary Functions**
  - $x + y, x - y, x \times y, \frac{x}{y + \epsilon}$

- Optimizers tested on 3x3 ConvNet (32 filters) for 5 epochs
- Favors early progress
Outline

1. Introduction
   - Motivation
   - Approach

2. Methods
   - Domain-Specific Language
   - Controller RNN

3. Experiments
   - Optimizer Discovery
   - Transfer Experiment

4. Summary
Recurring element:

\[ e^{\text{sign}(g) \times \text{sign}(m)} \times g \]

- If \( \text{sign}(g) \) agrees with running average, scale \( e \) - \( g \) keeps decreasing
- Else scale \( \frac{1}{e} \) - gradient direction changed
Outline

1 Introduction
   - Motivation
   - Approach

2 Methods
   - Domain-Specific Language
   - Controller RNN

3 Experiments
   - Optimizer Discovery
   - Transfer Experiment

4 Summary
Wide ResNet (Zagoruyko & Komodakis, 2016) - 300 epochs
<table>
<thead>
<tr>
<th>Optimizer</th>
<th>Final Val</th>
<th>Final Test</th>
<th>Best Val</th>
<th>Best Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>92.0</td>
<td>91.8</td>
<td>92.9</td>
<td>91.9</td>
</tr>
<tr>
<td>Momentum</td>
<td>92.7</td>
<td>92.1</td>
<td>93.1</td>
<td>92.3</td>
</tr>
<tr>
<td>ADAM</td>
<td>90.4</td>
<td>90.1</td>
<td>91.8</td>
<td>90.7</td>
</tr>
<tr>
<td>RMSProp</td>
<td>90.7</td>
<td>90.3</td>
<td>91.4</td>
<td>90.3</td>
</tr>
<tr>
<td>(e^{\text{sign}(g) \ast \text{sign}(m)} + \text{clip}(g, 10^{-4})) * g</td>
<td>92.5</td>
<td>92.4</td>
<td>93.8</td>
<td>93.1</td>
</tr>
<tr>
<td>\text{clip}(\hat{m}, 10^{-4}) * e^{\hat{v}}</td>
<td>93.5</td>
<td>92.5</td>
<td>93.8</td>
<td>92.7</td>
</tr>
<tr>
<td>(\hat{m} * e^{\hat{v}})</td>
<td>93.1</td>
<td>92.4</td>
<td>93.8</td>
<td>92.6</td>
</tr>
<tr>
<td>(g * e^{\text{sign}(g) \ast \text{sign}(m)})</td>
<td>93.1</td>
<td>92.8</td>
<td>93.8</td>
<td>92.8</td>
</tr>
<tr>
<td>\text{drop}(g, 0.3) * e^{\text{sign}(g) \ast \text{sign}(m)})</td>
<td>92.7</td>
<td>92.2</td>
<td>93.6</td>
<td>92.7</td>
</tr>
<tr>
<td>(\hat{m} * e^{g^2})</td>
<td>93.1</td>
<td>92.5</td>
<td>93.6</td>
<td>92.4</td>
</tr>
<tr>
<td>\text{drop}(\hat{m}, 0.1)/(e^{g^2} + \epsilon)</td>
<td>92.6</td>
<td>92.4</td>
<td>93.5</td>
<td>93.0</td>
</tr>
<tr>
<td>\text{drop}(g, 0.1) * e^{\text{sign}(g) \ast \text{sign}(m)})</td>
<td>92.8</td>
<td>92.4</td>
<td>93.5</td>
<td>92.2</td>
</tr>
<tr>
<td>\text{clip}(\text{RMSProp}, 10^{-5}) + \text{drop}(\hat{m}, 0.3)</td>
<td>90.8</td>
<td>90.8</td>
<td>91.4</td>
<td>90.9</td>
</tr>
<tr>
<td>ADAM * e^{\text{sign}(g) \ast \text{sign}(m)})</td>
<td>92.6</td>
<td>92.0</td>
<td>93.4</td>
<td>92.0</td>
</tr>
<tr>
<td>ADAM * e^{\hat{m}}</td>
<td>92.9</td>
<td>92.8</td>
<td>93.3</td>
<td>92.7</td>
</tr>
<tr>
<td>(g + \text{drop}(\hat{m}, 0.3))</td>
<td>93.4</td>
<td>92.9</td>
<td>93.7</td>
<td>92.9</td>
</tr>
<tr>
<td>\text{drop}(\hat{m}, 0.1) * e^{g^3}</td>
<td>92.8</td>
<td>92.7</td>
<td>93.7</td>
<td>92.8</td>
</tr>
<tr>
<td>(g - \text{clip}(g^2, 10^{-4}))</td>
<td>93.4</td>
<td>92.8</td>
<td>93.7</td>
<td>92.8</td>
</tr>
<tr>
<td>(e^{g - e^{\hat{m}})</td>
<td>93.2</td>
<td>92.5</td>
<td>93.5</td>
<td>93.1</td>
</tr>
<tr>
<td>\text{drop}(\hat{m}, 0.3) * e^{10^{-3}w}</td>
<td>93.2</td>
<td>93.0</td>
<td>93.5</td>
<td>93.2</td>
</tr>
</tbody>
</table>
Neural Machine Translation

Completely different model & task: WMT 2014 English → German task
GNMT model - 8 LSTM layers

<table>
<thead>
<tr>
<th>Optimizer</th>
<th>Train perplexity</th>
<th>Test BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>1.49</td>
<td>24.5</td>
</tr>
<tr>
<td>$g \times e^{\text{sign}(g) \times \text{sign}(m)}$</td>
<td>1.39</td>
<td>25.0</td>
</tr>
</tbody>
</table>
**Summary**

- RNN generates optimizer equations
- Train RNN via RL setup
- Optimizers tested on small ConvNet
- New optimizers on par with state of the art