Automatically Searching the Optimal Neural Network Structures

Spring 2018
The University of Iowa

Zhe Li
Content

- Motivation
- Genetic Approach
- Reinforcement Learning Approach
- Experimental Results
- Summary
Motivation

Example: Image classification

Accuracy on Image Classification

Accuracy: Test on ImageNet Benchmark Dataset (about 1.2 million images, 1000 classes)
Motivation

- Go through the history of the performance improvement for computer vision tasks

- Design Features
  - Haar, SIFT
  - HoG, LBP
  - ...

- Design Network Structures
  - AlexNet, VGG
  - GoogleNet, ResNet
  - ...

- Design algorithms to design Network structures

Year:
- 2012
- 20??
Motivation

- Alex-Net (2012)
- VGG-Net (2013)
- Inception-Net (2014)
- Dense-Net (2016)
Motivation

- Can we develop an efficient algorithm to design/search the optimal neural network structures?
  - Genetic Approach
  - Reinforcement Learning Approach
Content

- Motivation
- Genetic Approach
- Reinforcement Learning Approach
- Experimental Results
- Summary
Genetic Approach
Genetic Approach

Flowchart of Algorithms

- Create initial population
- Evaluate Fitness of each individual

Zoom-in Components

- Fitness Scores
  - Test accuracy
  - Number of Parameters
  - Number of Floating Point Operations
Genetic Approach

Flowchart of Algorithms

1. Create initial population
2. Evaluate Fitness of each individual
3. Apply selection

Zoom-in Components

Selection Strategies
- Select top-k unique structure
- Sample structures on probabilities
Genetic Approach

Flowchart of Algorithms

1. Create initial population
2. Evaluate Fitness of each individual
3. Apply selection
4. Crossover/Mutations

Zoom-in Components

Example of Mutation

Before mutation

After mutation
Genetic Approach

Flowchart of Algorithms

Create initial population

Evaluate Fitness of each individual

Apply selection

Crossover/Mutations

Termination

Zoom-in Components

Initial Population

Data
FC1
Logprob
Conv1x1
global_pooling
logprob

Fitness Scores

Test accuracy
Number of Parameters
Number of Floating Point Operations

Selection Strategies

Select top-k unique structure
Sample structures on probabilities

Example of Mutation

Before mutation
Data
FC1
Logprob
Conv
Add
Conv
FC1
Logprob

After mutation
Example of Mutation Operations

- **Add Conv**
  - Data
    - Conv1
    - Conv2
    - Pooling
    - Conv1x1
    - global_pooling
    - Logprob

- **Add Concat**
  - Data
    - Conv1
    - Conv3
    - Conv2
    - Pooling
    - Conv1x1
    - global_pooling
    - Logprob

- **Add Skip**
  - Data
    - Conv1
    - Conv3
    - Concat1
    - Conv2
    - Pooling
    - Conv1x1
    - global_pooling
    - Logprob

- **Data**
  - Conv1
  - Conv3
  - Concat1
  - Conv2
  - Pooling
  - Conv1x1
  - global_pooling
  - Logprob
Challenges

- **Implementation**: Build this automatically searching framework

- **Computation cost**: training one neural network is computationally expensive, and we need to train a lot of neural networks by using genetic approach.
Reducing computational cost

Which components are computationally expansive?
Selection Strategy

- Selection Strategy itself is not computationally expensive, but the consequence of selection has large influence on computational cost in future.
  - Random sampling
  - Random sampling based on fitness score
  - Tournament selection strategy
  - Aggressive selection strategy (Proposed)
### Selection Strategy - Tournament

<table>
<thead>
<tr>
<th>Old Population</th>
<th>0.6</th>
<th>0.2</th>
<th>0.1</th>
<th>0.7</th>
<th>0.5</th>
<th>0.3</th>
</tr>
</thead>
</table>

Conventional Tournament Selection
### Selection Strategy - Tournament

#### Conventional Tournament Selection

<table>
<thead>
<tr>
<th>Old Population</th>
<th>0.6</th>
<th>0.2</th>
<th>0.1</th>
<th>0.7</th>
<th>0.5</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.6</td>
<td>0.7</td>
<td>0.2</td>
<td>0.1</td>
<td>0.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>

18
Selection Strategy - Tournament

Conventional Tournament Selection

Old Population

0.6  0.2
0.1  0.7
0.5  0.3

Copy

Mutation
Selection Strategy - Tournament

Conventional Tournament Selection

Old Population

| 0.6 | 0.2 |
| 0.1 | 0.7 |
| 0.5 | 0.3 |

New Population

- copy
- mutation

- 0.6
- 0.7

- 0.2
- 0.1

- 0.5
- 0.3
Selection Strategy – Aggressive

Old Population

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggressive selection</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td></td>
</tr>
</tbody>
</table>
Selection Strategy – Aggressive

Old Population

![Diagram showing aggressive selection process with numbers 0.6, 0.2, 0.1, 0.7, 0.5, 0.3 and copy dashed arrows to new population]

Aggressive selection
Selection Strategy – Aggressive

Old Population

Aggressive selection

0.6
0.2
0.1

0.7
0.5
0.3

copy

mutation
Selection Strategy – Aggressive

Old Population

New Population

Aggressive selection

copy

mutation
Selection Strategy - Aggressive

- What’s the issue caused by this aggressive selection strategy compared with the three traditional strategies?
  - Decreasing the diversity of the population.

- How to increase diversity of the population?
  - More mutation operations.
More Mutation for diversity

Mutation operations defined and implemented.

<table>
<thead>
<tr>
<th>Mutations</th>
<th>[1]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_convolution</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>remove_convolution</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>alter_channel_number</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>alter_filter_size</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>alter_stride</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>add_dropout</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>remove_dropout</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>add_pooling</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>remove_pooling</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>add_skip</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>remove_skip</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>add_concatenate</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>remove_concatenate</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>add_fully_connected</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>remove_fully_connected</td>
<td>-</td>
<td>✓</td>
</tr>
</tbody>
</table>

The allowed mutation operations in our work and in [1]; ✓ represents defined while - represents NA

[1] Large-scale Evolution of Image Classifiers
Evaluate Fitness Score

- Reducing training time
  - Early stopping – don’t waste computation on the weak neural network
  - Advanced learning strategy
  - Parallel training
  - … …
Content

- Motivation
- Genetic Approach
- Reinforcement Learning Approach
- Experimental Results
- Summary
How reinforcement learning works

**Action**: where to put the next piece down.
**Reward**: 1 if win at the end of game, 0 otherwise.
Reinforcement Learning Approach

- **States**: Implicit states of RNN in the controller.
- **Actions**: Primitive descriptions of NN; a sequence of “actions” describes a complete NN, called “child network”.
- **Rewards**: Success ratio of “child network”. The controller is updated using the “rewards” by Reinforcement Learning.

Credit: https://arxiv.org/pdf/1611.01578.pdf
Reinforcement Learning Approach

- **States**: Implicit states of RNN in the controller.
- **Actions**: Primitive descriptions of NN; a sequence of “actions” describes a complete NN, called “child network”.
- **Rewards**: Success ratio of “child network”. The controller is updated using the “rewards” by Reinforcement Learning.

Credit: https://arxiv.org/pdf/1611.01578.pdf
Reinforcement Learning Approach

- **States**: Implicit states of RNN in the controller.
- **Actions**: Primitive

Recurrent Architectures

- Softmax classifier
- Embedding
Reinforcement Learning Approach

- **States**: Implicit states of RNN in the controller.
- **Actions**: Primitive descriptions of NN. A sequence of “actions” describes a complete NN, called “child network”.
- **Rewards**: Success ratio of “child network”. The controller is updated using the “rewards” by Reinforcement Learning.

E.g., for convolutional networks, the primitive descriptions are “number of filters”, “filter height”, “filter width”, “stride height”, “stride width”, etc. The domain of these parameters are fixed.

Credit: https://arxiv.org/pdf/1611.01578.pdf
Reinforcement Learning Approach

- **States**: Implicit states of RNN in the controller.
- **Actions**: Primitive descriptions of NN. A sequence of “actions” describes a complete NN, called “child network”.
- **Rewards**: Success ratio of “child network”. The controller is updated using the “rewards” by Reinforcement Learning.

E.g., for the architecture of a recurrent cell, we may fix a tree structure and define the primitive descriptions for each node in the tree as “add”, “Elem mult”, “Tanh”, “Sigmoid”, “ReLU”, etc.
Reinforcement Learning Approach

- **Encode** the structure and connectivity of a neural network by using a configuration string
  - [“filter width: 5”, “filter height: 3”, “num filter: 24”]
- **Using a RNN (“Controller”)** to generate this string that encodes a neural network structure
- **Train** this neural network structure (“child Network”) to see how well it performs
- **Use reinforcement learning to update the parameters of the Controller model** based on the accuracy of the child model

Credit: http://rll.berkeley.edu/deeprlcoursesp17/docs/quoc_barret.pdf
Justify that the aggressive selection strategy.
Content

- Motivation
- Genetic Approach
- Reinforcement Learning Approach
- Experimental Results
- Summary
Experimental Results

- Handwritten digits recognition

![Handwritten digits]

<table>
<thead>
<tr>
<th>Approach</th>
<th>Test Acc</th>
<th>Comp Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOTA[4]</td>
<td>0.9979</td>
<td>–</td>
</tr>
<tr>
<td>Genetic-CNN[2]</td>
<td>0.9966</td>
<td>48 GPUH</td>
</tr>
<tr>
<td>EDEN[3]</td>
<td>0.9840</td>
<td>–</td>
</tr>
<tr>
<td>Ours</td>
<td>0.9969</td>
<td>35 GPUH</td>
</tr>
</tbody>
</table>

Comparison of test accuracy and computational cost on MNIST dataset.
Experimental Results

10 Classes Image Classification

<table>
<thead>
<tr>
<th>Approach</th>
<th>Test Acc</th>
<th>Comp Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOTA[5]</td>
<td>0.9654</td>
<td>-</td>
</tr>
<tr>
<td>LS-Evolution[1]</td>
<td>0.9460</td>
<td>65,536 GPUH</td>
</tr>
<tr>
<td>Genetic-CNN[2]</td>
<td>0.7706</td>
<td>408 GPUH</td>
</tr>
<tr>
<td>EDEN[3]</td>
<td>0.7450</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td>0.9267</td>
<td>72 GPUH</td>
</tr>
</tbody>
</table>

Comparison of test accuracy and computational cost on CIFAR-10
Experimental Results

- Show how model size and test accuracy of neural network change along evolution.

![Graph showing test accuracy and model size versus number of generations.](image)
Experimental Results

Show the finally learned neural network structures by our genetic approach.
Experimental Results

- Learned architecture of a recurrent unit by the reinforcement learning approach.

LSTM Cell

Neural Architecture Search (NAS) Cell
Content

- Motivation
- Genetic Approach
- Reinforcement Learning Approach
- Experimental Results
- Summary
Automatically searching neural network structures is a hot area.

Explored genetic approach to search neural network.

Implemented genetic framework for this task.

Proposed different strategies to reduce computation cost.

Briefly discussed reinforcement learning approach to search neural network structures.

Achieved competitive performance with significantly reduced computational cost by genetic approach.