TAPAS: Train-less accuracy predictor for architecture searches

R. Istrate, F. Scheidegger, G. Mariani, P. Chatzidoukas, C. Bekas, C. Malossi
IBM Research – Zurich
Table of content

- Motivations and state-of-the-art
- Methodology:
  - Train-less Accuracy Predictor for Architecture Search
  - Dataset characterization
  - Lifelong database of experiments
  - Train-less accuracy predictor
- Results
  - Accuracy prediction comparison with Peephole and Learning Curves Extrapolation
  - Performance comparison with Google Large Scale Evolution
- Conclusions and future works
Motivations

- Network architecture design is expensive: time, resources, people
- Currently based on empirical processes

What it takes to automatize NN architecture discovery:

1. **Preserve and re-use knowledge** learned from previously experiments and models
2. **Predict performance** of architectures before training
3. Dynamically adapt to **complexity of input dataset**
4. Smart algorithms that perform **large scale search**, minimizing training

What does this mean?

- **Design initial network** (Q: where to start?)
- **Train** (Q: for how long: hours/days/weeks?)
- **Is accuracy OK?** (Q: how much can it be improved?)
- **Modify network** (Q: what has to be changed?)
- **Trained model** (Q: has this been obtained in the shortest dev-time?)

**Never ending loop!**
State-of-the-art: Google large scale evolution approach

- Mutation algorithm to generate networks
- Architecture search over very large spaces

- Expensive: 1000 individuals, 250 workers, 10 days of experiments for CIFAR-10 network
- Would not scale when used on larger datasets

REF: Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, Alex Kurakin, Large-Scale Evolution of Image Classifiers, 2017
State-of-the-art: Peephole, prediction before training

- Cheap: evaluates network performance without training
- Architecture search over large spaces at no cost
- Dataset specific
- Requires over 1000 networks trained on same dataset
- Generated networks have fixed convolutional structure

Train-Less Accuracy Predictor for Architecture Search

Three main components:

- Dataset Characterization (DC): rank dataset by difficulty
- Lifelong Database of Experiments (LDE): store experiments and grows over time
- Train-less Accuracy Predictor (TAP): predict performance of networks in real-time
Dataset characterization from literature experiments

- Literature study on 14 datasets
- Some dataset are part of large competitions (more points)
- Some results are obtained with transfer learning

Key observation:
- Large variability per dataset
- There is a clear ranking
ProbeNets: networks for dataset characterization

10 networks:

- 7 static (only softmax & input scale with number of classes)
- 3 dynamic (topology scales with number of classes)
### ProbeNets: cheap, quick, and accurate

<table>
<thead>
<tr>
<th>Probe Net</th>
<th>( C = 10 )</th>
<th>( C = 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OPs</td>
<td>Weights</td>
</tr>
<tr>
<td>Regular</td>
<td>0.81M</td>
<td>11K</td>
</tr>
<tr>
<td>Narrow</td>
<td>0.09M</td>
<td>2K</td>
</tr>
<tr>
<td>Wide</td>
<td>0.3M</td>
<td>114K</td>
</tr>
<tr>
<td>Shallow</td>
<td>0.24M</td>
<td>21K</td>
</tr>
<tr>
<td>Shallow norm.</td>
<td>0.06M</td>
<td>5K</td>
</tr>
<tr>
<td>Deep</td>
<td>1.40M</td>
<td>100K</td>
</tr>
<tr>
<td>Deep norm.</td>
<td>19.76M</td>
<td>1576K</td>
</tr>
<tr>
<td>MLPs</td>
<td>2.90M</td>
<td>2908K</td>
</tr>
<tr>
<td>Kernel depth</td>
<td>0.53M</td>
<td>6K</td>
</tr>
<tr>
<td>Length</td>
<td>1.41M</td>
<td>118K</td>
</tr>
<tr>
<td>ResNet-20</td>
<td>40.55M</td>
<td>271K</td>
</tr>
</tbody>
</table>

- Good performance compared to ResNet-20
- Cost reduced up to 50x for Regular, and 400x for Narrow
Lifelong database of experiments (LDE)

Over-optimistic TAP

Calibrated TAP

- New input dataset
- Selected datasets from LDE
- Discarded datasets from LDE
### TAP layer-by-layer workflow

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Output/Input height ratio</th>
<th>Output/Input depth ratio</th>
<th>Number of weights</th>
<th>Total number of layers</th>
<th>Inference FLOPs</th>
<th>Inference memory</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layer 2</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layer 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layer $N_l$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **DCN**
- **TAP**

Network prediction build on the intermediate evaluation of all sub-networks

Incremental training approach used to populate LDE and obtain intermediate accuracies
Accuracy prediction comparison with Peephole and LCE

Scenario A (first row):
- Input dataset: CIFAR-10
- 1 dataset in LDE: CIFAR-10
- 90% cross-validation

Scenario B (second row):
- Input dataset: CIFAR-10
- 20 dataset in LDE (including CIFAR-10)
- 90% cross-validation
Scenario C:

- 11 leave-one-out cross-validations
- Input dataset: one of the eleven available
- 19 dataset in LDE (10 real not including the input one + 9 sub-sampled from Imagenet)
Performance comparison: Google large scale evolution

Comparison of resources utilization:

- Google: 256h on 250 workers (many GPUs)
- TAPAS 400 s on 1 GPU (without training)
Conclusions and future works

Framework features summary:

- Dataset agnostic
- Leverage experience from previously trained networks (continuous learning)
- Real-time prediction that scales with resources
- Can be used in many architecture search algorithms

Near future works:

- Extension to other type of DL problems: object detection, scene labeling NLP, etc.
- Extension to other type of DL dataset: video, text, audio signal, etc.
- Full framework for architecture search on IBM Cloud (under development)
- Full framework delivery on OpenPOWER
References


THANK YOU

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. Other product and service names might be trademarks of IBM or other companies.