SANAS
Stochastic Adaptive Neural Architecture Search for Keyword Spotting

ICASSP 2019

Tom Veniat
Olivier Schwander
Ludovic Denoyer
Context
Deep Learning Success

- *State of the art* in a number of domains (vision, speech, etc...)
Deep Learning Success

- **State of the art** in a number of domains (vision, speech, etc...)
- Example in computer vision: plenty of different architectures.

AlexNet [Krizhevsky, Sutskever, and Hinton 2012]

ResNet [He et al. 2015]

DenseNet [Huang, Liu, and Weinberger 2016]
Deep Learning Success

- **State of the art** in a number of domains (vision, speech, etc...)
- Example in computer vision: **plenty** of different architectures.

AlexNet [Krizhevsky, Sutskever, and Hinton 2012]

ResNet [He et al. 2015]

DenseNet [Huang, Liu, and Weinberger 2016]

- Need for an automatic way to discover architectures
Existing approaches

- Searches

Example using Recurrent Neural Networks and Reinforcement Learning:

Neural architecture search with RL, [Zoph and Le 2016]
Existing approaches

- Searches
- Evolutionary methods
Existing approaches

- Searches
- Evolutionary methods
- Learning
  - Example using Recurrent Neural Networks and Reinforcement Learning:

Neural architecture search with RL, [Zoph and Le 2016]
Existing approaches

- Searches
- Evolutionary methods
- Learning
  - Example using Recurrent Neural Networks and Reinforcement Learning:

Neural architecture search with RL, [Zoph and Le 2016]

Existing approaches limitations

- Mainly focused on performance
- Ignore real world constraints
Problematic

Architecture search under budget constraints
Problematic

Architecture search under budget constraints

Contributions

- New model: Budgeted Super Networks
- Joint optimization on performance and inference cost
  - Costs: Time, Memory, Parallelization
  - Custom costs based on production infrastructure
  - No assumption on the cost nature
Budgeted Super Networks
Definition

- A Super Network is a DAG of layers \((l_1, ..., l_N)\)
- \(l_1\) is the input layer and \(l_N\) is the output layer
- \(E = \{e_{i,j}\} \in \{0, 1\}^{N \times N}\) is the edge between \(l_i\) and \(l_j\) and is associated with function \(f_{i,j}\) parametrized by \(\theta_{i,j}\)
Super Networks

Definition

- A Super Network is a DAG of layers \((l_1, ..., l_N)\)
- \(l_1\) is the input layer and \(l_N\) is the output layer
- \(E = \{e_{i,j}\} \in \{0, 1\}^{N \times N}\) is the edge between \(l_i\) and \(l_j\) and is associated with function \(f_{i,j}\) parametrized by \(\theta_{i,j}\)

Inference

- **Input:** \(l_1(x, E, \theta) = x\)
- **Layer Computation:** \(l_i(x, E, \theta) = \sum_k e_{k,i} f_{k,i}(l_k(x, E, \theta))\)
- **Output:** \(f(x, E, \theta) = l_N(x, E, \theta)\)
- Learning can be done by back-propagation
Super Networks

Super Network Inference

Algorithm 1: SN Forward

Data: \(x, E, \theta\)

\(l_1 \leftarrow x\);  // Init the first layer

for \(i \in [2..N]\) do

\(l_i \leftarrow \sum_{k<i} e_{k,i} f_{k,i}(l_k; \theta_{k,i})\);  // Propagate through the super network

end
Budgeted Super Networks

Idea: Identifying a sub-network

- Keep a good accuracy
- Reduce cost

![Convolutional Neural Fabrics](image)

Figure 2: Convolutional Neural Fabrics [Saxena and Verbeek 2016]
Budgeted Super Networks

Idea: Identifying a sub-network

- Keep a good accuracy
- Reduce cost

Figure 2: Convolutional Neural Fabrics [Saxena and Verbeek 2016]

Notation

- $H \in \{0, 1\}^{N \times N}$ such that $H \odot E$ defines a (sub) Super Network
- $C(H \odot E)$ the cost for computing $f(x, H \odot E, \theta)$
Learning under budget constraints

\[ H^*, \theta^* = \arg \min_{H, \theta} \frac{1}{\ell} \sum_i \Delta(f(x^i, H \odot E, \theta), y^i), \]
under constraint: \( C(H \odot E) \leq C \)
Learning under budget constraints

\[ H^*, \theta^* = \arg \min_{H, \theta} \frac{1}{\ell} \sum \Delta(f(x_i, H \odot E, \theta), y^i), \]
under constraint: \( C(H \odot E) \leq C \)

Soft version

\[ H^*, \theta^* = \arg \min_{H, \theta} \frac{1}{\ell} \sum \Delta(f(x_i, H \odot E, \theta), y^i) \]
\[ + \lambda \max(0, C(H \odot E) - C) \]
Combinatorial Problem

- How to explore the **discrete** architecture space?
- How to handle the **unknown** cost function $C(H \odot E)$?
Combinatorial Problem

- How to explore the **discrete** architecture space?
- How to handle the **unknown** cost function \( C(H \odot E) \) ?

Idea

- Reformulate the learning problem as a **stochastic** problem.
- Apply Reinforcement Learning techniques to overcome the combinatorial problem.
Stochastic Super Network

- We consider a matrix of probabilities $\Gamma$
- At each inference, $H$ is sampled following $H \sim \Gamma$

Stochastic Super Network Inference

**Algorithm 2: SSN Forward**

**Data:** $x, E, \Gamma, \theta$

$H \sim \Gamma$;  // Sample an architecture

$l_1 \leftarrow x$;

for $i \in [2..N]$ do

$\quad l_i \leftarrow \sum_{k<i} e_{k,i} h_{k,i} f_{k,i}(l_k; \theta_{k,i})$;  // Propagate through the sampled network

end
Stochastic learning problem

\[ \Gamma^*, \theta^* = \arg\min_{\Gamma, \theta} \frac{1}{\ell} \sum_i \mathbb{E}_{H \sim \Gamma} \left[ \Delta(f(x^i, H \odot E, \theta), y^i) + \lambda \max(0, C(H \odot E) - C) \right] \]

- Solving this problem is equivalent to solving the original constrained problem.
- Can be optimized by SGD using REINFORCE.
Budgeted Super Networks

Deriving the stochastic learning problem

Let us define:

$$D(x, y, \theta, E, H) = \Delta(f(x, H \circ E, \theta), y) + \lambda \max(0, C(H \circ E) - C)$$

$$L(x, y, E, \Gamma, \theta) = \mathbb{E}_{H \sim \Gamma} [D(x, y, \theta, E, H)]$$

We have:

$$\nabla_{\theta, \Gamma} L(x, y, E, \Gamma, \theta) = \sum_H P(H|\Gamma) \left[ (\nabla_{\theta, \Gamma} \log P(H|\Gamma)) D(x, y, \theta, E, H) \right]$$

$$+ \sum_H P(H|\Gamma) \left[ \nabla_{\theta, \Gamma} \Delta(f(x, H \circ E, \theta), y) \right]$$
SANAS
From static

\[
\mathcal{L}(x, y, E, \Gamma, \theta) = \mathbb{E}_{H \sim \Gamma} [\Delta(f(x, H \odot E, \theta), y) + \lambda \max(0, C(H \odot E) - C)]
\]
From static

$$
\mathcal{L}(x, y, E, \Gamma, \theta) = \mathbb{E}_{H \sim \Gamma} \left[ \Delta(f(x, H \circ E, \theta), y) + \lambda \max(0, C(H \circ E) - C) \right]
$$

To dynamic

$$
\mathcal{L}(x, y, \theta) = \mathbb{E}_{A} \left[ \sum_{t=1}^{\#x} \Delta(f(z_t, x_t, \theta, A_t), y_t) + \lambda C(A_t) \right]
$$
General Model

Figure 3: SANAS Architecture unrolled on sequence of length $K$. 
Keyword Spotting - Speech Commands Dataset [Warden 2018]

- 65000 short audio clips
- 30 common words
- 12 classes
Keyword Spotting - Speech Commands Dataset [Warden 2018]

- 65000 short audio clips
- 30 common words
- 12 classes

Streaming dataset

\[ \text{Signal Amplitude} \]

\[ \text{Time (s)} \]

- Background noise
- Word + Background noise

SANAS: Stochastic Adaptive Neural Architecture Search for Keyword Spotting
Keyword Spotting - Model

- based on *cnn-trad-fpool3* [Sainath and Parada 2015]

Figure 4: SANAS architecture for Keyword Spotting
## Quantitative results

<table>
<thead>
<tr>
<th>Match</th>
<th>Correct</th>
<th>Wrong</th>
<th>FA</th>
<th>FLOPs per frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{cnn-trad-fpool3}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81.7%</td>
<td>72.8%</td>
<td>8.9%</td>
<td>0.0%</td>
<td>124.6M</td>
</tr>
<tr>
<td>\textit{cnn-trad-fpool3 + shortcut connections}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82.9%</td>
<td>77.9%</td>
<td>5.0%</td>
<td>0.3%</td>
<td>137.3M</td>
</tr>
<tr>
<td>\textit{SANAS}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61.2%</td>
<td>53.8%</td>
<td>7.3%</td>
<td>0.7%</td>
<td>519.2K</td>
</tr>
<tr>
<td>62.0%</td>
<td>57.3%</td>
<td>4.8%</td>
<td>0.1%</td>
<td>22.9M</td>
</tr>
<tr>
<td>86.5%</td>
<td>80.7%</td>
<td>5.8%</td>
<td>0.3%</td>
<td>37.7M</td>
</tr>
<tr>
<td>86.3%</td>
<td>80.6%</td>
<td>5.7%</td>
<td>0.2%</td>
<td>48.8M</td>
</tr>
<tr>
<td>81.7%</td>
<td>76.4%</td>
<td>5.3%</td>
<td>0.1%</td>
<td>94.0M</td>
</tr>
<tr>
<td>81.4%</td>
<td>76.3%</td>
<td>5.2%</td>
<td>0.2%</td>
<td>105.4M</td>
</tr>
</tbody>
</table>

Table 1: Evaluation of models on 1h of audio.
Figure 5: Cost/accuracy curves on test set.
Training dynamics

Figure 6: Cost per word during training
Perspectives

- Use new models (Currently training on Resnets)
- Test other sound datasets
- Evaluate over different tasks (Video, Event detection, RL ...)

SANAS: Stochastic Adaptive Neural Architecture Search for Keyword Spotting
References


