NLP Reading Group

NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm

by Xiaoliang Dai, Hongxu Yin and Niraj K. Jha

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Introduction

- **Major problems for finding an appropriate architecture :**
- BP algorithm assumes a fixed DNN architecture and only trains weights
- trial-and-error methodology inefficient when DNNs get deeper
- lead to large, accurate, but over-parameterized DNNs
- **Methodology to address these problems :**

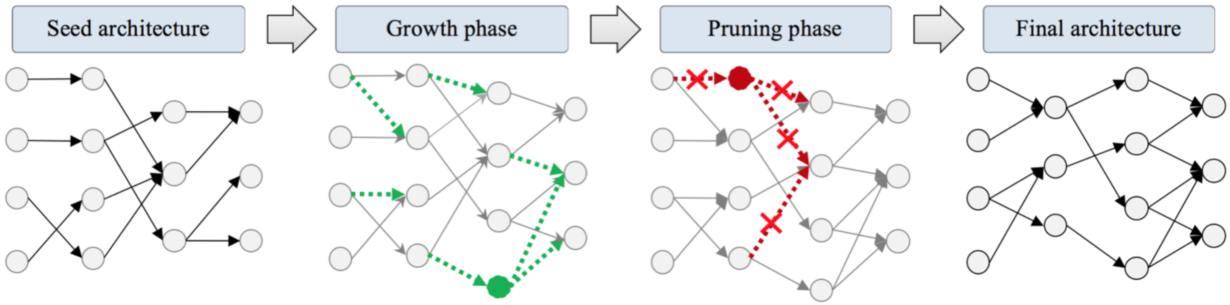


Figure 1: An illustration of the architecture synthesis flow in NeST.

Architecture Growth

1. Connection Growth :

Policy 1 : Add a connection w $\iff$ it can quickly reduce the value of loss function L.

- evaluate ∂L/∂w $\forall w$.

  activation $\iff$ w most efficient reducing L.

- neuroscience perspective : Hebbian theory: “Neurons that fire together wire together."
  - Let $\frac{\partial L}{\partial u_m^{l+1}}$ be the stimulation magnitude of the m^{th} presynaptic neuron in the $(l+1)^{th}$ layer
  - Let x_n^l be the n^{th} postsynaptic neuron in the l^{th} layer.
  - Based on Hebbian theory, the connections activated would have a strong correlation between presynaptic and postsynaptic cells $\iff$ large value of : $|(\frac{\partial L}{\partial u_m^{l+1}})x_n^l|$.
- Also the magnitude of the gradient of L with respect to w, so :

$$\boxed{|\partial \mathrm{L}/\partial \mathrm{w}| = |(\frac{\partial L}{\partial u_m^{l+1}})x_n^l|}$$

Architecture Growth

2. Neuron Growth :

Policy 2 : In the l^{th} layer, add a new neuron as a shared intermediate node between existing neuron pairs that have high postsynaptic (x) and presynaptic (∂L/∂u) neuron correlations (each pair contains one neuron from the $(l-1)^{th}$ layer and the other from the $(l+1)^{th}$ layer). Initialize weights based on batch gradients to reduce the value of L.

Algorithm 1 Neuron growth in the l^{th} layer

Input: α - birth strength, β - growth ratio
Denote: M - number of neurons in the $(l+1)^{th}$ layer, N - number of neurons in the $(l-1)^{th}$ layer, $\mathbf{G} \in R^{M \times N}$ - bridging gradient matrix, avg - extracts mean value of non-zero elements
Add a neuron in the l^{th} layer, initialize $\mathbf{w}^{out} = \vec{\mathbf{0}} \in R^M$, $\mathbf{w}^{in} = \vec{\mathbf{0}} \in R^N$
for $1 \leq m \leq M, 1 \leq n \leq N$ **do**
  $G_{m,n} = \frac{\partial L}{\partial u_m^{l+1}} \times x_n^{l-1}$
end for
$thres$ = $(\beta MN)^{th}$ largest element in $abs(\mathbf{G})$
for $1 \leq m \leq M, 1 \leq n \leq N$ **do**
  if $|G_{m,n}| > thres$ **then**
    $\delta w = \sqrt{|G_{m,n}|} \times rand\{1, -1\}$
    $w_m^{out} \leftarrow w_m^{out} + \delta w$, $w_n^{in} \leftarrow w_n^{in} + \delta w \times sgn(G_{m,n})$
  end if
  $\mathbf{w}^{out} \leftarrow \mathbf{w}^{out} \times \alpha \frac{avg(abs(\mathbf{W}^{l+1}))}{avg(abs(\mathbf{w}^{out}))}$, $\mathbf{w}^{in} \leftarrow \mathbf{w}^{in} \times \alpha \frac{avg(abs(\mathbf{W}^{l}))}{avg(abs(\mathbf{w}^{in}))}$
end for
Concatenate network weights $\mathbf{W}$ with $\mathbf{w}^{in}$, $\mathbf{w}^{out}$

Architecture Growth

2.bis Neuron Growth : Study of the Weight Initialization

- Bridging connection w_b between x_n^{l-1} and u_m^{l+1}.
- Initialized with a square root rule to imitate a BP update on w_b :
    - Leads to a change in u_m^{l+1} :

$$|\Delta u_m^{l+1}|_{b.p.} = |x_n^{l-1} \times \delta w_b| = \eta |x_n^{l-1} \times G_{m,n}|$$, where η is the learning rate.

- Proof : In Algorithm 1, connection of the newly added neuron with x_n^{l-1} and u_m^{l+1} :
    - $|\delta w_n^{in}| = |\delta w_m^{out}| = \sqrt{|G_{m,n}|}$
    - $|\Delta u_m^{l+1}| = |f(x_n^{l-1} \times \delta w_n^{in}) \times \delta w_m^{out}|$, where f is the activation function.
    - If $f = tanh$ (or ReLU, Leaky ReLU...) : $f(x) = tanh(x) \approx x$, if $x \ll 1$
    - $|\Delta u_m^{l+1}| \approx |x_n^{l-1} \times \delta w_n^{in} \times \delta w_m^{out}| = \frac{1}{\eta} \times |\Delta u_m^{l+1}|_{b.p.}$
- After the square root rule based weight initialization : scaling up of the newly added weights :

$$\mathbf{w}^{out} \leftarrow \alpha \mathbf{w}^{out} \times \frac{avg(abs(\mathbf{W}^{l+1}))}{avg(abs(\mathbf{w}^{out}))}, \quad \mathbf{w}^{in} \leftarrow \alpha \mathbf{w}^{in} \times \frac{avg(abs(\mathbf{W}^{l}))}{avg(abs(\mathbf{w}^{in}))}$$

Architecture Growth

3. Growth in Convolutionnal Layers

- Same methodology as Policy 1
- **Policy 3 :** To add a new feature map to the convolutional layers, randomly generate sets of kernels, and pick the set of kernels that reduces L the most.

Magnitude Pruning :

- **Policy 4 :** Remove a connection (neuron) $\iff$ the magnitude of the weight (neuron output) is smaller than a pre-defined threshold.

Pruning insignificant weights : Consider the l^{th} batch normalization layer:

$$\mathbf{u}^l = [(\mathbf{W}^l \mathbf{x}^{l-1} + \mathbf{b}^l) - \mathbf{E}] \oslash \mathbf{V} = \mathbf{W}_*^l \mathbf{x} + \mathbf{b}_*^l$$

where $\mathbf{E}$ and $\mathbf{V}$ are batch normalization terms, and $\oslash$ depicts the Hadamard (element-wise) division operator

Effective Weights and biases are defined as :

$$\mathbf{W}_*^l = \mathbf{W}^l \oslash \mathbf{V}, \mathbf{b}_*^l = (\mathbf{b}^l - \mathbf{E}) \oslash \mathbf{V}$$

Magnitude Pruning : Partial Area Convolution :

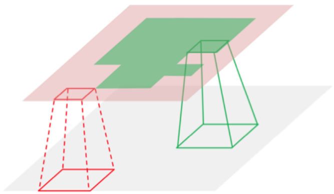


Figure : Pruned connections (dashed red lines) and remaining connections (solid green lines) in partial-area convolution.

Algorithm 2 Partial-area convolution

Input: **I** - M input images, **K** - kernel matrix, **Msk** - feature map mask, γ - pruning ratio
Output: **Msk**, **F** - N feature maps
Denote: $\mathbf{C} \in R^{M \times N \times P \times Q}$ - Depthwise feature map, $\otimes$ - Hadamard (element-wise) multiplication
for $1 \leq m \leq M, 1 \leq n \leq N$ **do**
  $\mathbf{C}_{m,n} = convolve(\mathbf{I}_m, \mathbf{K}_{m,n})$
end for
$thres = (\gamma MNPQ)^{th}$ largest element in $abs(\mathbf{C})$
for $1 \leq m \leq M, 1 \leq n \leq N, 1 \leq p \leq P, 1 \leq q \leq Q$ **do**
  if $|C_{m,n,p,q}| < thres$ **then**
    $Msk_{m,n,p,q} = 0$
  end if
end for
$\mathbf{C} \leftarrow \mathbf{C} \otimes \mathbf{Msk}$, $\mathbf{F} \leftarrow \Sigma_{m=1}^{M} \mathbf{C}_m$

Experimental Results

- **Wide seed range :** high-performance DNNs with a wide range of seed architectures.
- **Drastic redundancy removal :** NeST-generated DNNs are very compact.

Table 2: Different inference models for MNIST

Model	Method	Error	#Param	FLOPs
RBF network [7]	-	3.60%	794K	1588K
Polynomial classifier [7]	-	3.30%	40K	78K
K-nearest neighbors [7]	-	3.09%	47M	94M
SVMs (reduced set) [35]	-	1.10%	650K	1300K
Caffe model (LeNet-300-100) [36]	-	1.60%	266K	532K
LWS (LeNet-300-100) [22]	Prune	1.96%	4K	8K
Net pruning (LeNet-300-100) [5]	Prune	1.59%	22K	43K
Our LeNet-300-100: compact	**Grow+Prune**	**1.58%**	**3.8K**	**6.7K**
Our LeNet-300-100: accurate	**Grow+Prune**	**1.29%**	**7.8K**	**14.9K**
Caffe model (LeNet-5) [36]	-	0.80%	431K	4586K
LWS (LeNet-5) [22]	Prune	1.66%	4K	199K
Net pruning (LeNet-5) [5]	Prune	0.77%	35K	734K
Our LeNet-5	**Grow+Prune**	**0.77%**	**5.8K**	**105K**

Table 3: Different AlexNet and VGG-16 based inference models for ImageNet

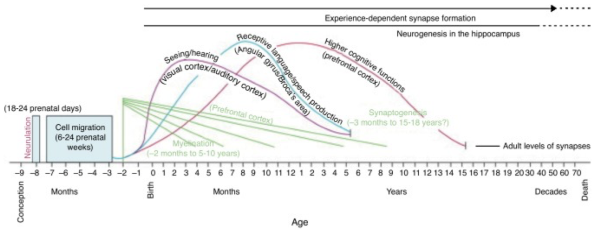
Model	Method	ΔTop-1 err.	ΔTop-5 err.	#Param (M)	FLOPs (B)
Baseline AlexNet [37]	-	0.0%	0.0%	61 (1.0×)	1.5 (1.0×)
Data-free pruning [38]	Prune	+1.62%	-	39.6 (1.5×)	1.0 (1.5×)
Fastfood-16-AD [39]	-	+0.12%	-	16.4 (3.7×)	1.4 (1.1×)
Memory-bounded [40]	-	+1.62%	-	15.2 (4.0×)	-
SVD [41]	-	+1.24%	+0.83%	11.9 (5.1×)	-
LWS (AlexNet) [22]	Prune	+0.33%	+0.28%	6.7 (9.1×)	0.5 (3.0×)
Net pruning (AlexNet) [5]	Prune	-0.01%	-0.06%	6.7 (9.1×)	0.5 (3.0×)
Our AlexNet	**Grow+Prune**	**-0.02%**	**-0.06%**	**3.9 (15.7×)**	**0.33 (4.6×)**
Baseline VGG-16 [42]	-	0.0%	0.0%	138 (1.0×)	30.9 (1.0×)
LWS (VGG-16) [22]	Prune	+3.61%	+1.35%	10.3 (13.3×)	6.5 (4.8×)
Net pruning (VGG-16) [5]	Prune	+2.93%	+1.26%	10.3 (13.3×)	6.5 (4.8×)
Our VGG-16: accurate	**Grow+Prune**	**-0.35%**	**-0.31%**	**9.9 (13.9×)**	**6.3 (4.9×)***
Our VGG-16: compact	**Grow+Prune**	**+2.31%**	**+0.98%**	**4.6 (30.2×)**	**3.6 (8.6×)***

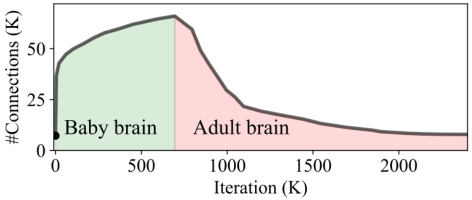
∗ Currently without partial-area convolution due to GPU memory limits.

Summary and Discussions

NeST methodology incorporates three Inspirations from the human brain :

- Variation of the number of synaptic connections :

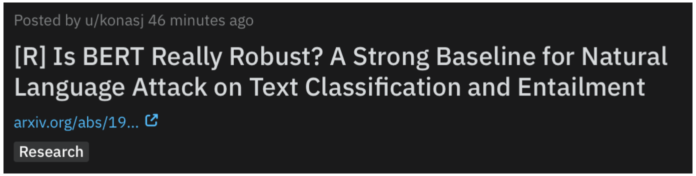


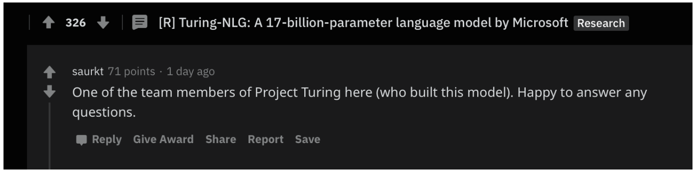


- Rewiring of synapses between Neurons
- Small fraction of neurons active at a given time.

Some unusual ways to talk about/find papers

- Recreative ways :
  - podcasts : NLP Highlights on SoundCloud, Lex Fridman's podcast...
  - r/LanguageTechnology
- Machine Learning Subreddit : r/MachineLearning : Discussions and Research reviews :
  - WAYR : Weekly "What Are You Reading" Post : Most Upvoted papers
  - Topics about papers





Thank you for your attention

Any questions?