Graph HyperNetworks for Neural Architecture Search

Chris J. Zhang[1,2], Mengye Ren[1,3], Raquel Urtasun[1,3]

[1] Uber Advanced Technologies Group [2] Unversity of Waterloo, [3] University of Toronto

Introduction

Motivation:

- Neural architecture search is an expensive nested optimization

$$a^* = \arg\min_a \mathcal{L}_{val}(w^*(a), a), \quad w^*(a) = \arg\min_w \mathcal{L}_{train}(w, a)$$

- Instead of using SGD to learn weights, use trained hypernetwork to generate weights
- Graph HyperNetworks (GHN) explicitly model the topology of architectures by learning on a computation graph representation

Strengths:

- Fast and effective: Using 0.42 GPU days and random search we find an architecture with competitive results on CIFAR and is transferrable to ImageNet
- Flexible and generalizable: Applied to anytime-prediction (previously unexplored by NAS) and outperformed existing manually designed state-of-the-art models.

Background

Graph Neural Networks: A collection of nodes and edges $(\mathcal{V}, \mathcal{E})$, where each node is a RNN U that individually sends and receives messages along the edges, forming embeddings $\boldsymbol{h}_v^{(t)}$ over the horizon of t message passing steps.

$$\boldsymbol{h}_v^{(t+1)} = \begin{cases} U\left(\boldsymbol{h}_v^{(t)}, \boldsymbol{m}_v^{(t)}\right) & \text{if node } v \text{ is active,} \\ \boldsymbol{h}_v^{(t)} & \text{otherwise,} \end{cases}$$

$$\boldsymbol{m}_v^{(t)} = \sum_{u \in N_{in}(v)} M\left(\boldsymbol{h}_u^{(t)}\right)$$

$$\left\{\boldsymbol{h}_v^{(t)} | v \in \mathcal{V}\right\} = G_{\mathcal{A}}^{(t)}\left(\left\{\boldsymbol{h}_v^{(0)} | v \in \mathcal{V}\right\}; \phi\right).$$

Hypernetworks: A neural network that generates the parameters of another network. For a typical deep feedforward network with D layers, the parameters of the j-th layer W_j can be generated by a learned function H

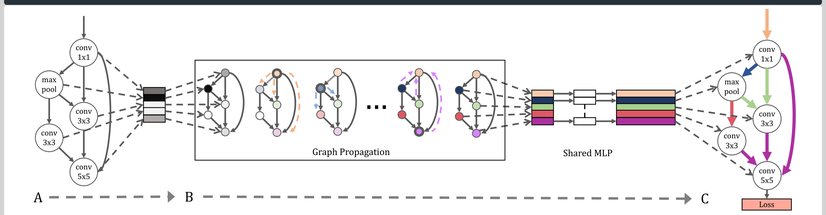
$$W_j = H(z_j), \quad \forall j = 1, \ldots, D,$$

where z_j is the layer embedding, and H is shared for all layers.

Graphical Representation

- Represent a given architecture as a directed acyclic graph $\mathcal{A} = (\mathcal{V}, \mathcal{E})$,
- Nodes $v \in \mathcal{V}$ have an associated computational operator $f_v(\,\cdot\,; w_v)$
- Edges $e_{u \mapsto v} = (u, v) \in \mathcal{E}$ represent the flow of activation tensors from node u to node v.
- Compute node's output $x_v = \sum_{e_{u \to v} \in \mathcal{E}} f_v(x_u; w_v), \quad \forall v \in \mathcal{V}$.

Graph HyperNetworks



- Given an input architecture with graphical representation $\mathcal{A}$, construct a homomorphic GNN $G_{\mathcal{A}}$ with the same topology
- After graph message-passing, a shared hypernetwork (2 layer MLP) is applied to all the nodes to generate the parameters of the network

$$\begin{aligned} \tilde{\boldsymbol{w}} = \{\tilde{\boldsymbol{w}}_v |\ v \in \mathcal{V}\} &= \left\{H\left(\boldsymbol{h}_v^{(T)}; \varphi\right) \mid v \in \mathcal{V}\right\} \\ &= \left\{H\left(\boldsymbol{h}; \varphi\right) \mid \boldsymbol{h} \in G_{\mathcal{A}}^{(T)}\left(\left\{\boldsymbol{h}_v^{(0)} | v \in \mathcal{V}\right\}; \phi\right)\right\} \\ &= GHN\left(\mathcal{A}; \phi, \varphi\right). \end{aligned}$$

- Compute gradients GHN parameters ϕ, φ using the chain rule:

$$\nabla_{\phi,\varphi} \mathcal{L}_{train}(\tilde{\boldsymbol{w}}) = \nabla_{\tilde{w}} \mathcal{L}_{train}(\tilde{\boldsymbol{w}}) \cdot \nabla_{\phi,\varphi} \tilde{\boldsymbol{w}}$$

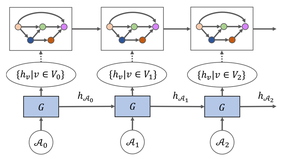
$$\nabla_\phi \tilde{\boldsymbol{w}} = \left\{\nabla_h H(\boldsymbol{h}; \varphi) \cdot \nabla_\phi \boldsymbol{h} \mid \boldsymbol{h} \in G^{(T)}\left(\{\boldsymbol{h}_v^{(0)}\}, \mathcal{A}, \phi\right)\right\},$$

$$\nabla_\varphi \tilde{\boldsymbol{w}} = \left\{\nabla_\varphi H(\boldsymbol{h}_v^{(T)}; \varphi) \mid v \in \mathcal{V}\right\}$$

Architectural Motifs and Stacked GNNs

- Searching for repeated modules often improves results
- Stack GHN along depth dimension, sharing parameters across modules
- Pass along graph level embeddings to subsequent GHN modules

$$\begin{aligned} \boldsymbol{h}_{\mathcal{A}_i} &= \frac{1}{|\mathcal{V}_i|} \sum_{v \in \mathcal{V}_i} \left\{\boldsymbol{h}_v^{(T)} | v \in \mathcal{V}_i\right\} \\ &= \frac{1}{|\mathcal{V}_i|} \sum G_{\mathcal{A}_i}^{(T)}\left(\left\{\boldsymbol{h}_v^{(0)} | v \in \mathcal{V}_i\right\}, \boldsymbol{h}_{\mathcal{A}_{i-1}}; \phi\right) \end{aligned}$$



Forward-backward GNN message passing

- Synchronous message passing across graphs with large diameters often runs into vanishing gradient problems
- The forward-backward propagation scheme alleviates this by mimicking the order of node execution in the backpropagation algorithm

$$\boldsymbol{h}_v^{(t+1)} = \begin{cases} U\left(\boldsymbol{h}_v^{(t)}, \boldsymbol{m}_v^{(t)}\right) & \text{if } s(t) = v \text{ and } 1 \le t \le |\mathcal{V}| \\ & \text{or if } s(2|\mathcal{V}| - t) = v \text{ and } |\mathcal{V}| + 1 \le t < 2|\mathcal{V}|, \\ \boldsymbol{h}_v^{(t)} & \text{otherwise.} \end{cases}$$

- Propagating information across a graph with diameter $|\mathcal{V}|$ reduced from $O(|\mathcal{V}|^2)$ to $O(|\mathcal{V}|)$ messages

NAS Benchmarks

CIFAR-10: Comparison with NAS methods which employ random search (top half) and advanced search methods (bottom half)

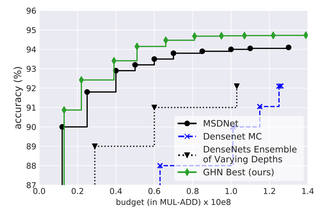
Method	Search Cost (GPU days)	Param $\times 10^6$	Accuracy
SMASHv1 (Brock et al., 2018)	?	4.6	94.5
SMASHv2 (Brock et al., 2018)	3	16.0	96.0
One-Shot Top (F=32) (Bender et al., 2018)	4	2.7 ± 0.3	95.5 ± 0.1
One-Shot Top (F=64) (Bender et al., 2018)	4	10.4 ± 1.0	95.9 ± 0.2
Random (F=32)	-	4.6 ± 0.6	94.6 ± 0.3
GHN Top (F=32)	0.42	5.1 ± 0.6	95.7 ± 0.1
NASNet-A (Zoph et al., 2018)	1800	3.3	97.35
ENAS Cell search (Pham et al., 2018)	0.45	4.6	97.11
DARTS (first order) (Liu et al., 2018b)	1.5	2.9	97.06
DARTS (second order) (Liu et al., 2018b)	4	3.4	97.17 ± 0.06
GHN Top-Best, 1K (F=32)	0.84	5.7	97.16 ± 0.07

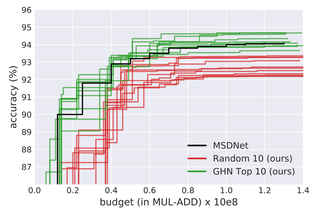
ImageNet Mobile: Comparison with NAS methods which employ advanced search methods

Method	Search Cost (GPU days)	Param $\times 10^6$	FLOPs $\times 10^6$	Accuracy Top 1	Accuracy Top 5
NASNet-A (Zoph et al., 2018)	1800	5.3	564	74.0	91.6
NASNet-C (Zoph et al., 2018)	1800	4.9	558	72.5	91.0
AmoebaNet-A (Real et al., 2018)	3150	5.1	555	74.5	92.0
AmoebaNet-C (Real et al., 2018)	3150	6.4	570	75.7	92.4
PNAS (Liu et al., 2018a)	225	5.1	588	74.2	91.9
DARTS (second order) (Liu et al., 2018b)	4	4.9	595	73.1	91.0
GHN Top-Best, 1K	0.84	6.1	569	73.0	91.3

Anytime Prediction

Minimize the expected loss $L(f) = \mathbb{E}\left[L\left(f(\mathbf{x}), B\right)\right]_{P(\mathbf{x},B)}$ for a model $f(\cdot)$, a test example $\mathbf{x}$ and a non-deterministic computational budget B drawn from the joint distribution $P(\mathbf{x}, B)$.





Correlation Experiments

Benchmarking correlation between predicted and true performance

Method	Computation cost Initial (GPU hours)	Computation cost Per arch. (GPU seconds)	Correlation Rand-100	Correlation Top-50
SGD 10 Steps	-	0.9	0.26	-0.05
SGD 100 Steps	-	9	0.59	0.06
SGD 200 Steps	-	18	0.62	0.20
SGD 1000 Steps	-	90	0.77	0.26
One-Shot	9.8	0.06	0.58	0.31
GHN	6.1	0.08	0.68	0.48