Contrastive Neural Architecture Search with Neural Architecture Comparators

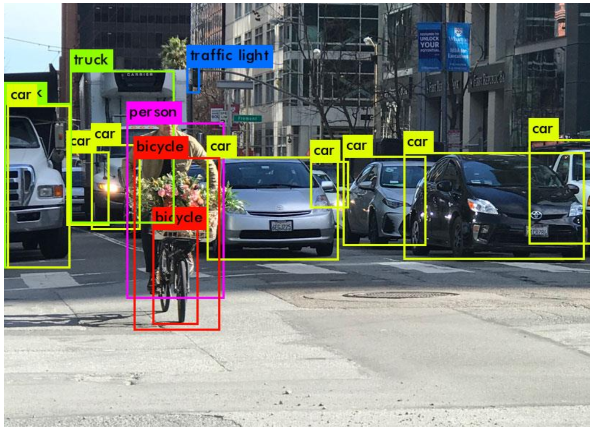
Yaofo Chen*, Yong Guo*, Qi Chen, Minli Li, Wei Zeng,
Yaowei Wang†, Mingkui Tan†

May 5, 2021

Contents

Background

Deep neural networks (DNNs) have achieved state-of-the-art performance in many challenge tasks.



object detection

semantic segmentation

pose estimation

Neural Architecture Search

One of the key factors behind the success lies in the **innovation of effective neural architectures**. However, it is **non-trivial** to design effective architectures manually **in practice**. The reasons has two folds:

- It relies heavily on **human expertise**.
- It requires **great human efforts** to repeat “trial-and-error”.

Solution

In this context, Neural Architecture Search (NAS) was developed to **automate the architecture designing process**.

Limitations of Existing NAS Methods

Existing NAS methods maximize the expectation of **the absolute performance of the sampled architectures**.

$$\max_{\theta} \mathbb{E}_{\alpha \sim \pi(\alpha;\theta)} \mathcal{R}\left(\alpha, w_{\alpha}\right)$$

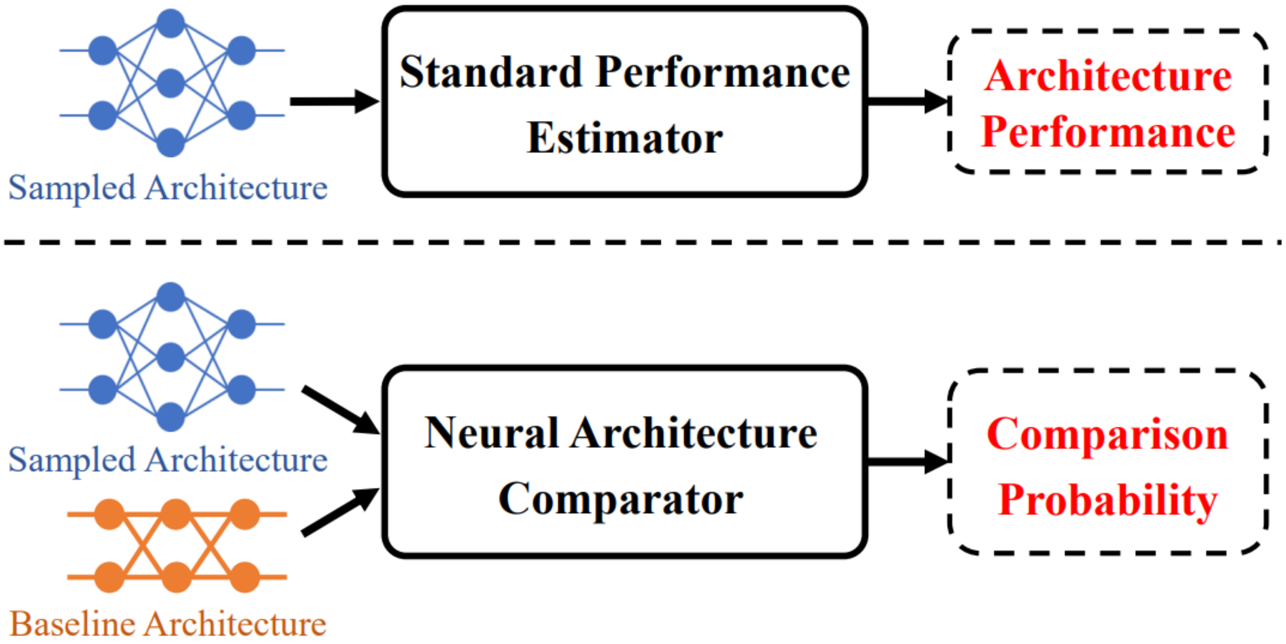
Using the absolute performance may suffer from two limitations:

- It is non-trivial to obtain **stable and accurate absolute performance** for all the candidate architectures.
- It is **time-consuming** to obtain the absolute performance from the supernet.

Motivation

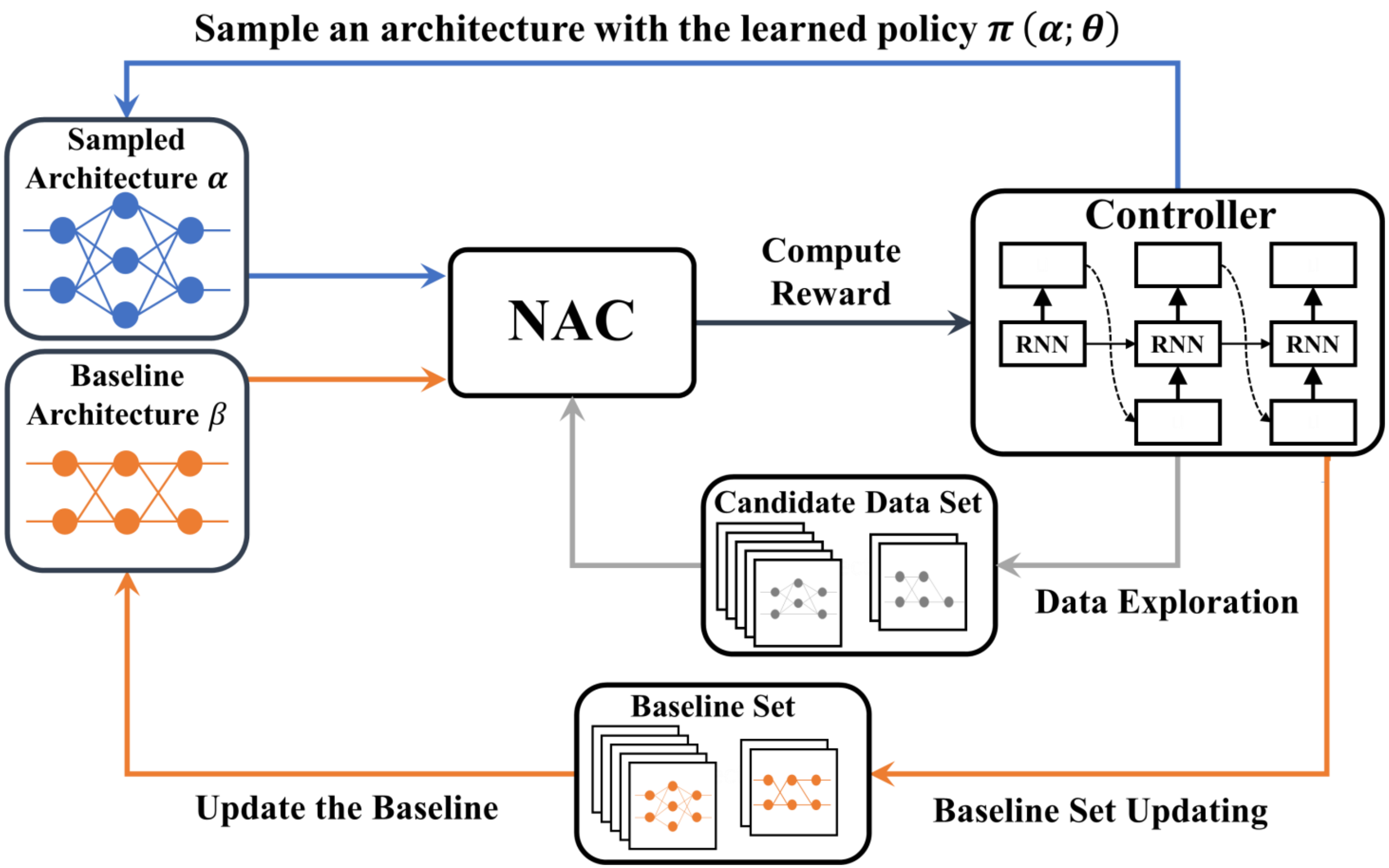
Suppose that α^* is the optimal architecture in a search space, we would have $\mathcal{R}(\alpha^*, w_{\alpha^*}) \geq \mathcal{R}(\alpha, w_\alpha)$. To ensure the optimality, we **only need to compute**

$$\Pr[\mathcal{R}(\alpha^*, w_{\alpha^*}) \geq \mathcal{R}(\alpha, w_\alpha)]$$



Contrastive Neural Architecture Search (CTNAS)

Unlike the traditional NAS methods, our CTNAS adopts the **comparison probability** of two architectures as the **reward signal**.



Training Objective of CTNAS

Our CTNAS maximizes the expectation of the **comparison probability** of the sampled architectures and the baseline one:

$$\max_{\theta} \ \mathbb{E}_{\alpha \sim \pi(\alpha;\theta)} \Pr[\mathcal{R}(\alpha, w_\alpha) \geq \mathcal{R}(\beta, w_\beta)]$$

To provide the comparison probability, we learn a comparison mapping, called **Neural Architecture Comparator (NAC),** to compare any two architectures:

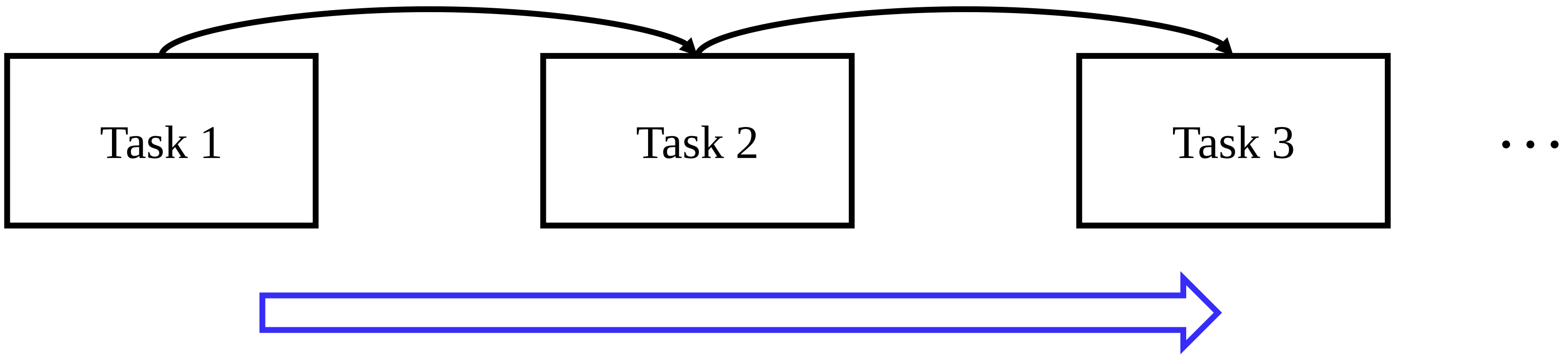
$$p = \Pr[\mathcal{R}(\alpha, w_\alpha) \geq \mathcal{R}(\alpha', w_{\alpha'})] = \mathrm{NAC}(\alpha, \alpha'; \varpi)$$

Baseline Updating via Curriculum Learning

Challenge

The above optimization problem can only enable the model to **find architectures that are better than the baseline architecture**.

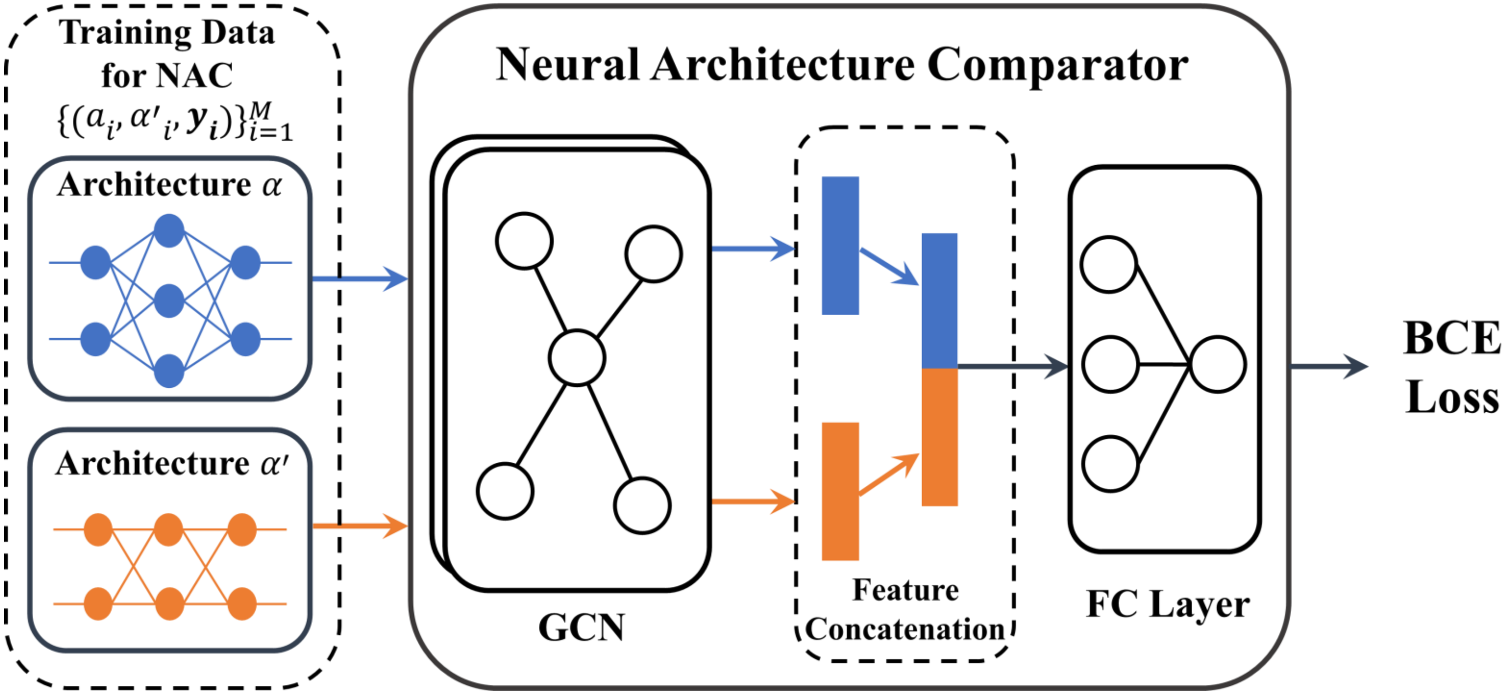
To address this issue, we propose a curriculum updating scheme to gradually improve/update the baseline during the search process.



The task become harder.

Overview of Neural Architecture Comparator

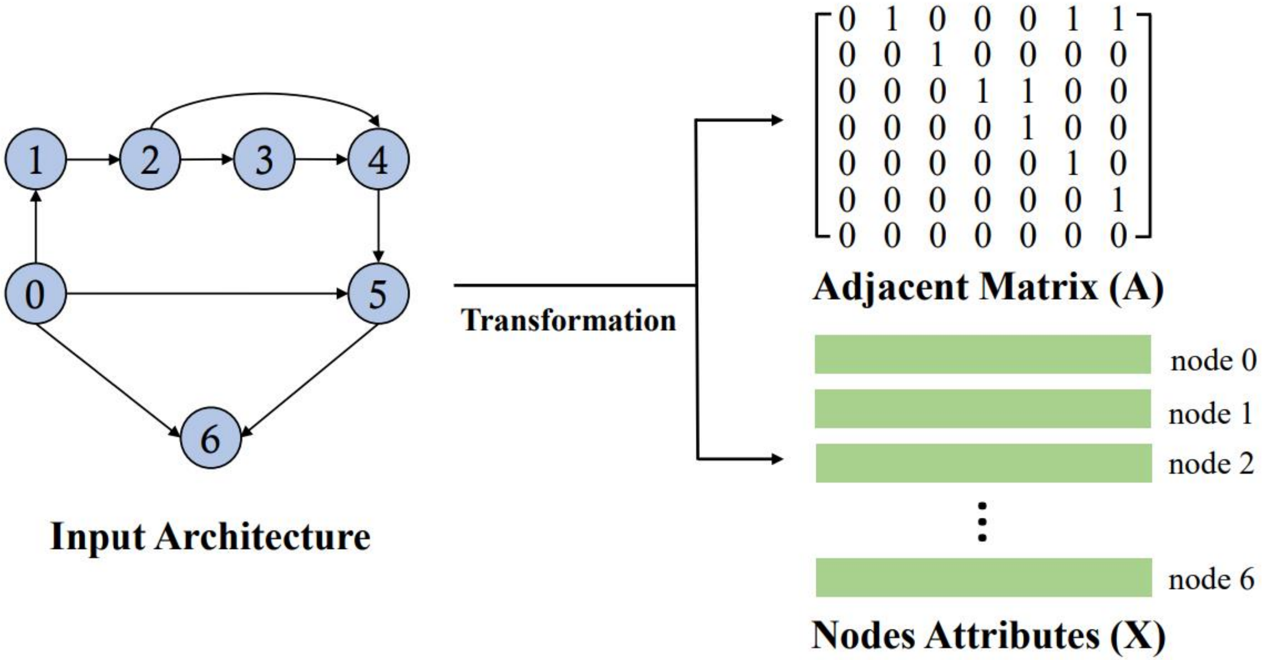
The proposed NAC takes **two architectures as inputs** and **outputs the comparison probability** of the one being better than the other.



Architecture Representation Method

We represent an architecture as a **directed acyclic graph (DAG)**, which can be further represented by **a pair (A, X)**.

- **A** denotes the adjacency matrix of the graph.
- **X** denotes learnable embeddings of nodes/operations.



Architecture Comparison by GCN

Based on the graph data pair (**A**, **X**), we use a **two-layer GCN** to extract the architecture features **Z**.

$$\mathbf{Z}_\alpha = f(\mathbf{X}_\alpha, \mathbf{A}_\alpha) = \mathbf{A}_\alpha \phi\left(\mathbf{A}_\alpha \mathbf{X}_\alpha \mathbf{W}^{(0)}\right) \mathbf{W}^{(1)}$$

To calculate the comparison probability, we **concatenate the features** of two input architectures and send them to a fully-connected layer.

$$p = \mathrm{NAC}(\alpha, \alpha'; \varpi) = \sigma\left([\mathbf{Z}_\alpha; \mathbf{Z}_{\alpha'}]\mathbf{W}^{\mathbf{FC}}\right)$$

Then, the sigmoid function takes the output of the FC layer as input and outputs the **comparison probability**.

Training Objective of NAC

To train the proposed NAC, we define **the label** for any architecture pair as follows:

$$y=\mathbb{1}\{\mathcal{R}(\alpha,w_\alpha)-\mathcal{R}(\alpha',w_{\alpha'})\geq 0\}$$

Thus, the training of NAC can be considered a **binary classification problem**. We train NAC by optimizing the binary cross-entropy loss:

$$\mathcal{L}=y\log(p)+(1-y)\log(1-p)$$

Data Exploration for Training NAC

Challenge

Learning a good NAC requires a set of labeled architecture pair data. However, we can only obtain **limited labeled data** in practice.

To address this issue, we propose a data exploration method that takes the class with **maximum probability predicted by NAC** as its label for unlabeled data pairs.

$$y' = \mathbb{1}\{\mathrm{NAC}(\alpha, \alpha'; \varpi) \geq 0.5\}$$

Results on NAS-Bench-101

Our CTNAS **outperforms** the consider NAS methods in terms of **ranking correlation** (Kendall's Tau) and **searched performance**.

Method	KTau	Average Accuracy (%)	Best Accuracy (%)	Best Rank (%)	#Queries
Random	–	89.31 ± 3.92	93.46	1.29	423
DARTS [29]	–	92.21 ± 0.61	93.02	13.47	–
ENAS [37]	–	91.83 ± 0.42	92.54	22.88	–
FBNet [48]	–	92.29 ± 1.25	93.98	0.05	–
SPOS [17]	0.195	89.85 ± 3.80	93.84	0.07	–
FairNAS [8]	-0.232	91.10 ± 1.84	93.55	0.77	–
ReNAS [53]	0.634	93.90 ± 0.21	94.11	0.04	423
RegressionNAS	0.430	89.51 ± 4.94	93.65	0.40	423
CTNAS (Ours)	**0.751**	$\mathbf{93.92 \pm 0.18}$	**94.22**	**0.01**	423

Results on ImageNet

Our CTNAS **outperforms** both **manually-designed architectures** and **state-of-the-art NAS models** in different search spaces.

Search Space	Architecture	Test Accuracy (%)		#MAdds (M)	#Queries (K)	Search Time (GPU days)	Total Time (GPU days)
		Top-1	Top-5				
	MobileNetV2 (1.4×) [40]	74.7	–	585	–	–	–
	ShuffleNetV2 (2×) [34]	73.7	–	524	–	–	–
NASNet	NASNet-A [59]	74.0	91.6	564	20	–	1800
NASNet	AmoebaNet-A [39]	74.5	92.0	555	20	–	3150
DARTS	DARTS [29]	73.1	91.0	595	19.5	4	4
DARTS	P-DARTS [7]	75.6	92.6	577	11.7	0.3	0.3
DARTS	PC-DARTS [50]	75.8	92.7	597	3.4	3.8	3.8
DARTS	CNAS [14]	75.4	92.6	576	100	0.3	0.3
MobileNetV3-like	MobileNetV3-Large [21]	75.2	–	219	–	–	–
MobileNetV3-like	FBNet-C [48]	74.9	–	375	11.5	1.8	9
MobileNetV3-like	MnasNet-A3 [44]	76.7	93.3	403	8	–	–
MobileNetV3-like	ProxylessNAS [4]	75.1	92.3	465	–	–	8.3
MobileNetV3-like	OFA [3]	76.0	–	230	16	1.7	51.7
MobileNetV3-like	FBNetV2 [45][†]	76.3	92.9	321	11.5	5	25
MobileNetV3-like	AtomNAS [35]	75.9	92.0	367	78	–	–
MobileNetV3-like	Random Search	76.0	92.6	314	1	–	50
MobileNetV3-like	Best Sampled Architectures	76.7	93.1	382	1	–	50
MobileNetV3-like	CTNAS (Ours)	**77.3**	**93.4**	482	1	**0.1**	50.1

Conclusion

Conclusion

- We propose a **Contrastive Neural Architecture Search (CTNAS)** method that takes the **comparison results** between architectures as **the reward**.

- To guarantee that CTNAS constantly finds better architectures, we propose a curriculum updating scheme to **gradually update/improve the baseline architecture.**

- Extensive experiments on three search spaces demonstrate that the searched architectures of our CTNAS **outperform the architectures designed by state-of-the-art methods.**