Progressive Neural Architecture Search

Johns Hopkins University

Chenxi Liu[1] Barret Zoph[2] Maxim Neumann[2] Jonathon Shlens[2] Wei Hua[2]
Li-Jia Li[2] Li Fei-Fei[2,3] Alan Yuille[1] Jonathan Huang[2] Kevin Murphy[2]
[1]Johns Hopkins University [2]Google AI [3]Stanford University

Google AI

Motivation

Problem Description:

- **Automatically design CNN architectures that surpass human expert designs**: Neural Architecture Search (NAS)
- Part of the AutoML initiative
- Previous approaches usually fall into either Evolutionary Algorithms (EA) or Reinforcement Learning (RL)
- However, they tend to be computationally intensive:
  - Zoph & Le (2017): 800 K40 for 28 days
  - Zoph et al. (2018): 500 P100 for 5 days

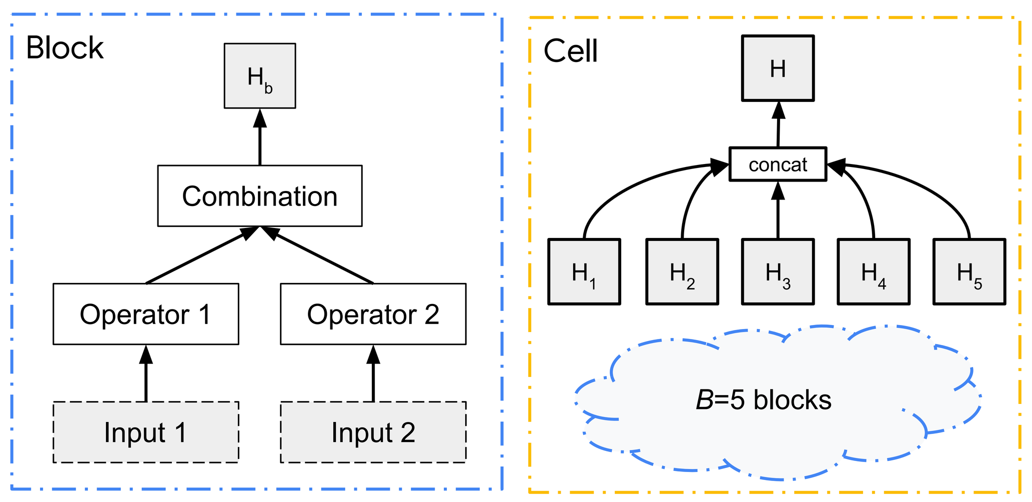
Our Goal:

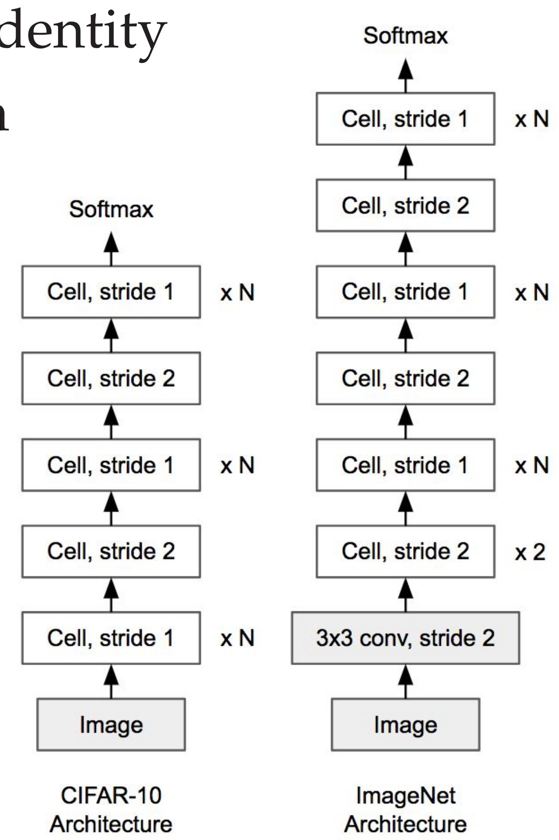
- **Speed up the NAS process by proposing a novel algorithm**
  - Reach a higher accuracy with less computation

Architecture Search Space

Block

- Input 1 and Input 2 may select from:
  - Previous cell's output
  - Previous-previous cell's output
  - Previous blocks' output in the current cell
- Operator 1 and Operator 2 may select from:
  - 3x3 separable conv
  - 5x5 separable conv
  - 7x7 separable conv
  - 1x7 plus 7x1 conv
  - 3x3 dilated conv
  - 3x3 max pooling
  - 3x3 average pooling
  - Identity
- Combination is element-wise addition





Blocks -> Cell

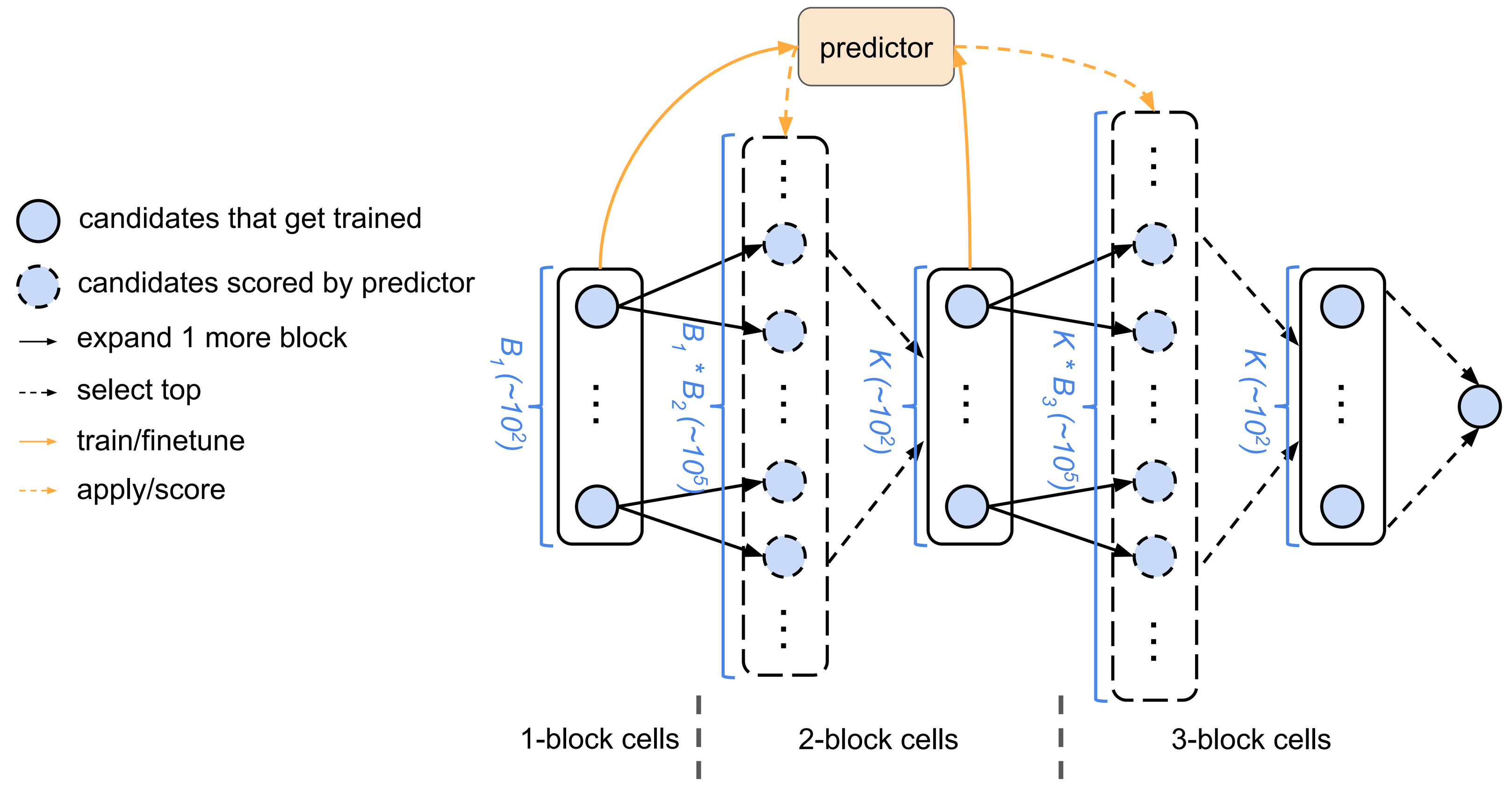
- Each cell consists of $B = 5$ blocks
- Cell's output is concatenation of blocks' output
- $2^2 \times 8^2 \times 1 \times 3^2 \times 8^2 \times 1 \times 4^2 \times 8^2 \times 1 \times 5^2 \times 8^2 \times 1 \times 6^2 \times 8^2 \times 1 = 10^{14}$ different cell structures!

Cells -> Network

- Predefined pattern; not part of architecture search
- N (depth): number of cell repetition
- F (width): number of filters in the first cell

Progressive NAS Algorithm

- Previous methods all directly operate on the 10^{14} search space
  - Hard for the agent to navigate, esp. at the beginning
- What if we *progressively* work our way in:
  - Begin by training all 1-block cells (only 256 of them)
  - Their scores will likely be low (less representation power)
  - But maybe their relative performances are enough to show which cells are promising and which are not
  - Expand K most promising 1-block cells into 2-block cells
- Turns out we also need a *surrogate predictor*:
  - Training is costly; we only want to do that for 10^2 cells
  - The predictor is trained to do the same job as the GPUs: given string (cell structure), output validation accuracy
  - Serve as quick proxy to identify top K cells among 10^5
  - We try both a MLP-ensemble and a RNN-ensemble

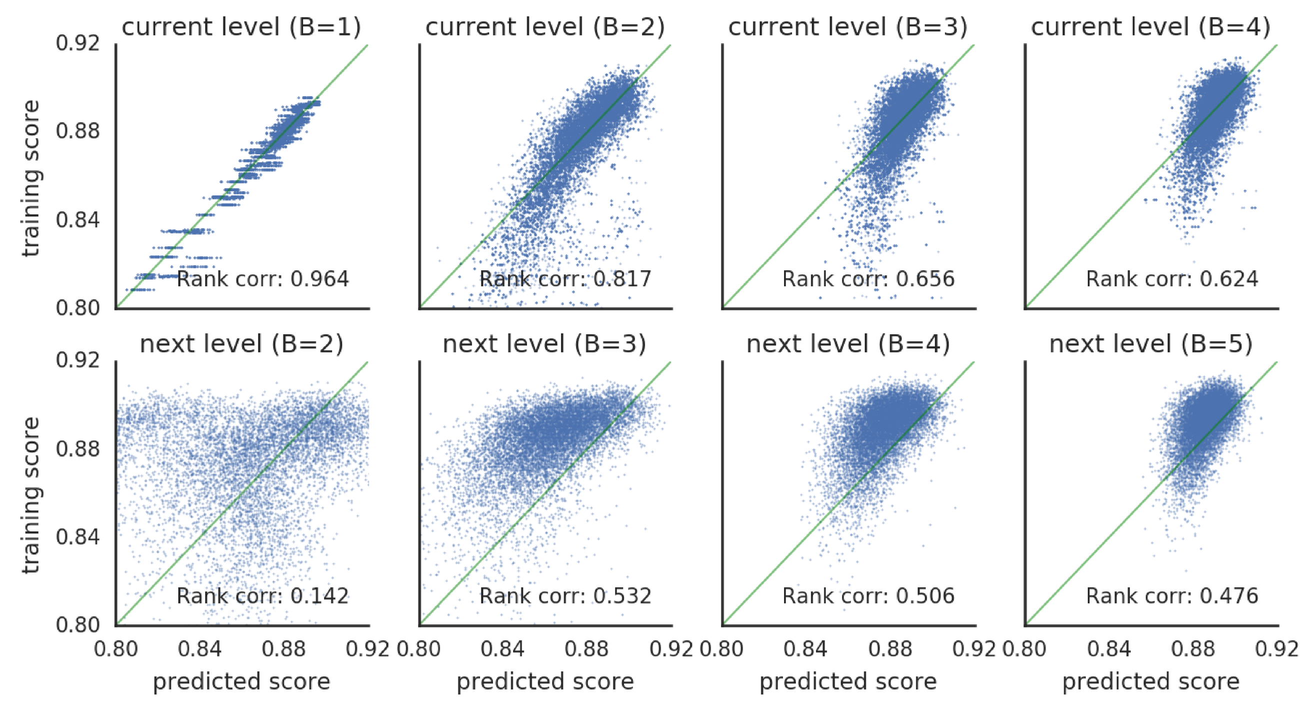


Experiments & Results

Architecture Search Details

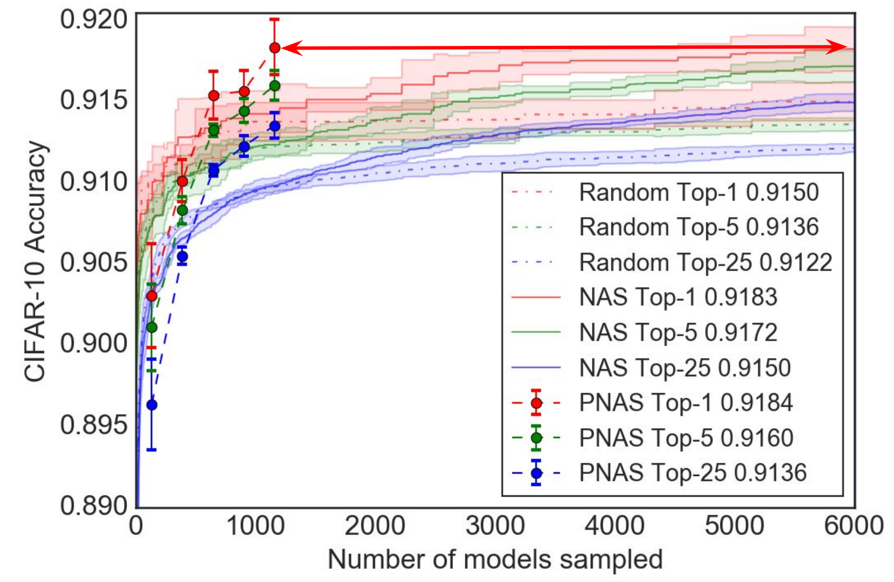
- 20 epochs with cosine learning rate on CIFAR-10
- Beam size $K = 256$; Small depth ($N = 2$) and width ($F = 24$)

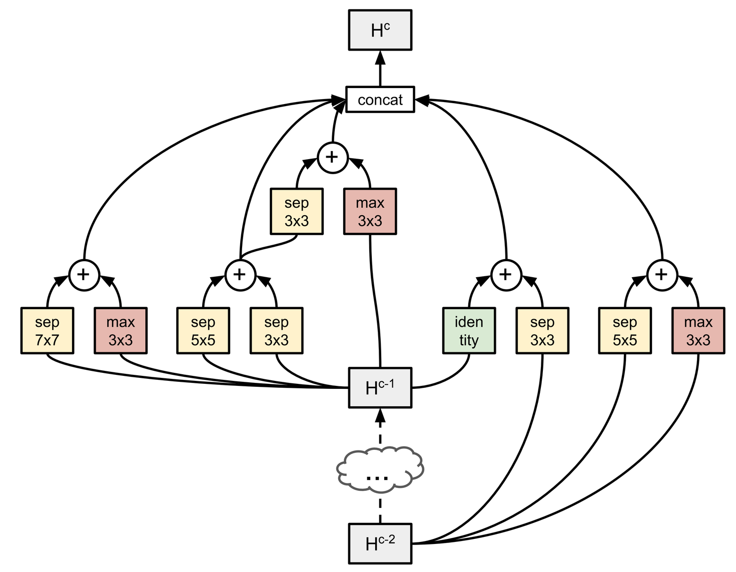
Performance of the Surrogate Predictor

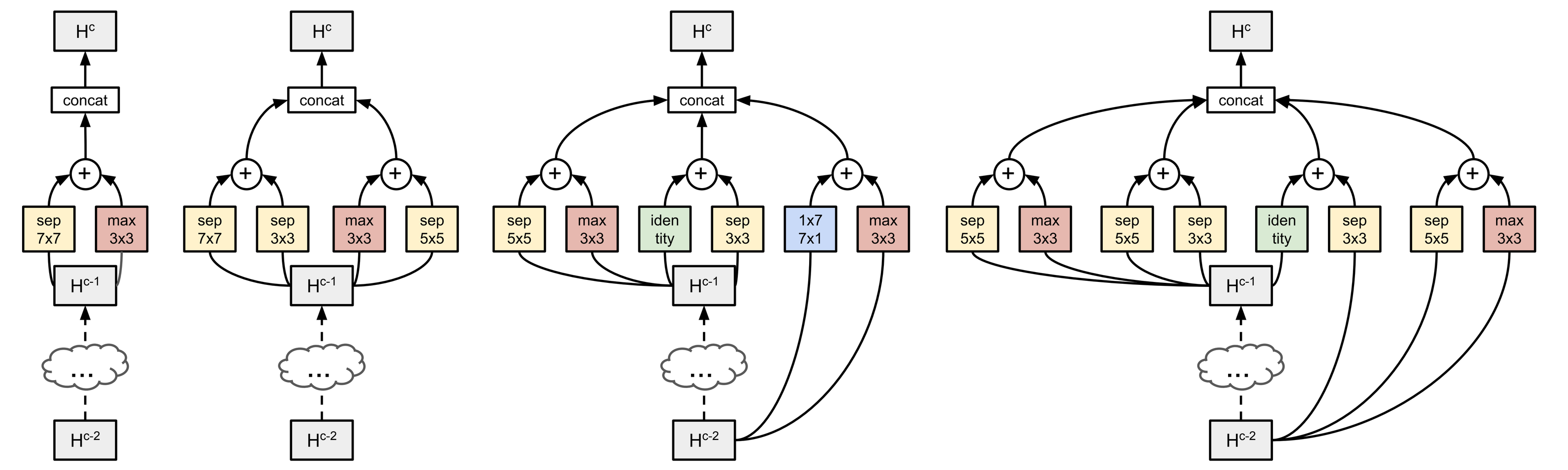


Experiments & Results (cont'd)

Search Efficiency and Best Cell Structures







Results on CIFAR-10 Image Classification

Model	Type	Error	Params	Samples	Cost
NASNet-A	RL	3.41	3.3M	20000	21.4-29.3B
NASNet-B	RL	3.73	2.6M	20000	21.4-29.3B
NASNet-C	RL	3.59	3.1M	20000	21.4-29.3B
Hier-EA	EA	3.75±0.12	15.7M	7000	35.8B
AmoebaNet-B	EA	3.37±0.04	2.8M	27000	63.5B
AmoebaNet-A	EA	3.34±0.06	3.2M	20000	25.2B
PNASNet-5	SMBO	3.41±0.09	3.2M	1160	1.0B

Results on ImageNet Image Classification

Model	Params/×+	Top-1/5	Params/×+	Top-1/5
MobileNet-224	4.2M/569M	70.6/89.5	-	-
ShuffleNet (2x)	5M/524M	70.9/89.8	-	-
ResNeXt-101	-	-	83.6M/31.5B	80.9/95.6
SENet	-	-	145.8M/42.3B	82.7/96.2
NASNet-A	5.3M/564M	74.0/91.6	88.9M/23.8B	82.7/96.2
AmoebaNet-B	5.3M/555M	74.0/91.5	84.0M/22.3B	82.3/96.1
AmoebaNet-A	5.1M/555M	74.5/92.0	86.7M/23.1B	82.8/96.1
AmoebaNet-C	6.4M/570M	75.7/92.4	155.3M/41.1B	83.1/96.3
PNASNet-5	5.1M/588M	74.2/91.9	86.1M/25.0B	82.9/96.2

Conclusion

- Search architecture from simple to complex, while simultaneously learning a surrogate function to guide the search.
- State-of-the-art level accuracies on CIFAR-10 and ImageNet, while 5-8 times more efficient than leading RL/EA methods.
- **CODE AND PRETRAINED PNASNET-5 ON IMAGENET:**
  `https://github.com/chenxi116/PNASNet.TF`
  `https://github.com/chenxi116/PNASNet.pytorch`