Progressive Neural Architecture Search

Università degli Studi di Firenze

Intelligenza Artificiale

Presented by Giovanni Maccioni

PNAS

- Based on NAS [2] search space;
- PNAS, like NAS [2], search for a cell structure, not a Full CNN structure;
- Sequential Model Based Optimization (SMBO);
- It uses a predictor model to guide the search through models' space

PNAS

- Searching for the cell structure instead of the complete CNN structure, allows after the search to build different networks, giving us more flexibility;

- The sequential search goes from simple models to more complex ones

- The predictor estimates the accuracies of unseen, more complex models, allowing us to train only the most promising ones

Cells

- A cell is a Fully Convolutional Network; stacking them is the key to obtain the final CNNs we are looking for

- In a generic way we can see the cell as a network that take as input a HxWxF tensor and gives in output a HxWxF tensor

- A cell can be defined with stride 2 convolutions. In that case the output will be a H/2xW/2x2F tensor

- A cell is composed by Blocks

Blocks

- The input to a Block can be the inputs to the cell it belongs to as well as the output of previous blocks;

- A Block is a set of two convolutional operations, chosen by a set of predifined ones:

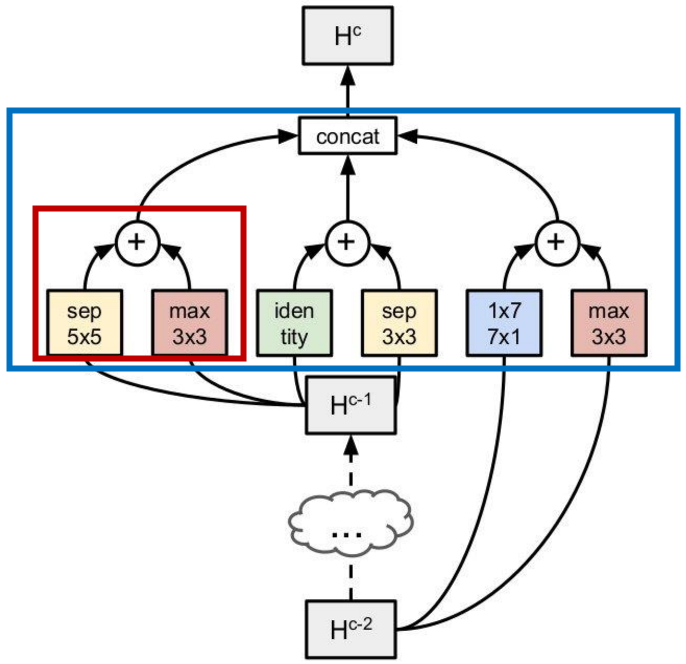
- Set of operations (from NAS[2]):

  - 3x3 Depthwise-separable convolution
  - 5x5 Depthwise-separable convolution
  - 7x7 Depthwise-separable convolution
  - 1x7-7x1 convolution

  - Identity
  - 3x3 Average Pool
  - 3x3 Max Pool
  - 3x3 Dilated convolution

- The output of this two operations are summed to give the final block output

Overview of Cells and Blocks



Cell "c" Input: c-1 and c-2 cell outputs
Cell "c" output: Concatenation of blocks output

Block "b" Input: c-1, c-2 cell outputs or previous blocks
Block "b" Output: Sum of operation outputs

Implementation detail: we adjust the dimension of the cell outout using a 1D-Convolution if needed

Search Space

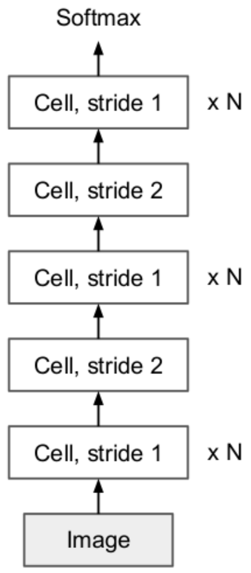
- For a Block b the possible number of blocks:
  - #In = 2 + b –1
  - #Op = 8
  - #Block_b = #In * #In + #Op * #Op

- If B=5 the possible number of cells is:
  - #Cells = #Block_1 * #Block_2 * #Block_3 * #Block_4 * #Block_5 = 5.6*10e14
  - This number is reduced to ~ 10e12 if we count unique cells

- In NAS[2] the search space is ~ 10e28. This gap is given for the most part by the distinction between Normal and Reduction Cells

CIFAR-10 Architecture

- This architecture is built for CIFAR-10 classification task. N is the number of cells with stride 1 repetitions



- There are cells with 1 and 2 stride in the model

Predictor

- Handle variable-sized inputs;

- Correlated with true performance;

- Sample efficiency;

Predictor

- Embeddings: the inputs and the operations have non-shared embeddings; if D is the embedding dimension, a block is represented as 4 D-dimensional vectors

- So a cell is represented as a sequence of 4 D-dimensional vectors

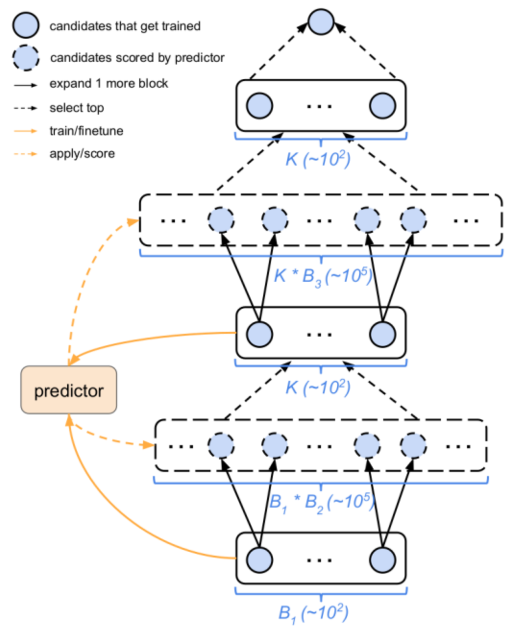
- Predictor architectures:
  - LSTM
  - MLP

- Both the models present an ensemble version

PNAS search

Algorithm 1 Progressive Neural Architecture Search (PNAS).

```
Inputs: $B$ (max num blocks), $E$ (max num epochs), $F$ (num filters in first layer), $K$ (beam size), $N$ (num times to unroll cell), trainSet, valSet.
$\mathcal{S}_1 = \mathcal{B}_1$ // Set of candidate structures with one block
$\mathcal{M}_1$ = cell-to-CNN($\mathcal{S}_1$, $N$, $F$) // Construct CNNs from cell specifications
$\mathcal{C}_1$ = train-CNN($\mathcal{M}_1$, $E$, trainSet) // Train proxy CNNs
$\mathcal{A}_1$ = eval-CNN($\mathcal{C}_1$, valSet) // Validation accuracies
$\pi$ = fit($\mathcal{S}_1$, $\mathcal{A}_1$) // Train the reward predictor from scratch
for $b = 2 : B$ do
    $\mathcal{S}'_b$ = expand-cell($\mathcal{S}_{b-1}$) // Expand current candidate cells by one more block
    $\hat{\mathcal{A}}'_b$ = predict($\mathcal{S}'_b$, $\pi$) // Predict accuracies using reward predictor
    $\mathcal{S}_b$ = top-K($\mathcal{S}'_b$, $\hat{\mathcal{A}}'_b$, $K$) // Most promising cells according to prediction
    $\mathcal{M}_b$ = cell-to-CNN($\mathcal{S}_b$, $N$, $F$)
    $\mathcal{C}_b$ = train-CNN($\mathcal{M}_b$, $E$, trainSet)
    $\mathcal{A}_b$ = eval-CNN($\mathcal{C}_b$, valSet)
    $\pi$ = update-predictor($\mathcal{S}_b$, $\mathcal{A}_b$, $\pi$) // Finetune reward predictor with new data
end for
Return top-K($\mathcal{S}_B$, $\mathcal{A}_B$, 1)
```



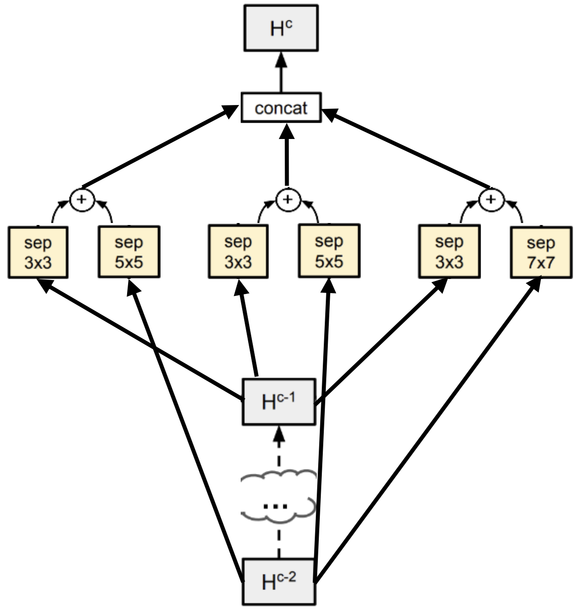
Implementation Details

- PNAS:
  - Number of Blocks (B): 3
  - Number of Repetitions (N): 2
  - Number of Filters (F): 24
  - Number of Operations: 8
  - Number of models trained per step (K): 64 (136 for the first step)
  - Predictor: MLP Ensemble, Criterion L1

PNAS: Results

- Top cells found
  - The cell below all have previous cell and previous-previous cell as input for each block

CELL	ACCURACY
1. Sep3x3, Sep5x5 2. Sep3x3, Sep5x5 3. Sep3x3, Sep7x7	0.86823
1. Sep3x3, Sep5x5 2. Sep3x3, Sep5x5 3. Sep3x3, Sep5x5	0.86734
1. Sep3x3, Sep5x5 2. Sep3x3, Sep5x5 3. Sep3x3, Conv1x7_7x1	0.86727



Random Search

- Given a number of blocks, B, define the set of all possible cells with B blocks

- Sample uniformly K samples from this set

- Train all the models corresponding to the sampled cells to obtain the accuracies

Predictor Performance Correlation

Algorithm 2 Evaluating performance of a predictor on a random dataset.

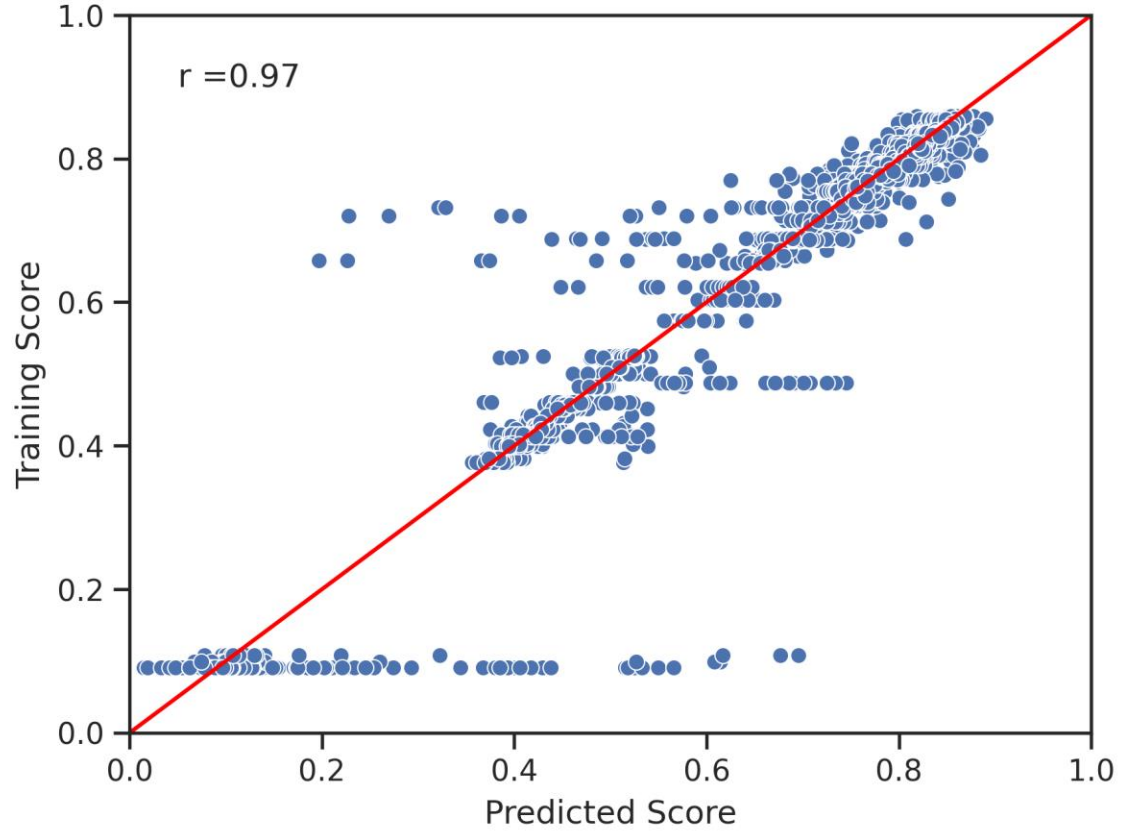
```
for $b = 1 : B - 1$ do
    for $t = 1 : T$ do
        $\mathcal{S}_{b,t,1:K}$ = random sample of $K$ models from $\mathcal{U}_{b,1:R}$
        $\pi_{b,t} = \text{fit}(\mathcal{S}_{b,t,1:K}, A(\mathcal{S}_{b,t,1:K}))$ // Train or finetune predictor
        $\hat{A}_{b,t,1:K} = \text{predict}(\pi_{b,t}, \mathcal{S}_{b,t,1:K})$ // Predict on same b
        $\tilde{A}_{b+1,t,1:R} = \text{predict}(\pi_{b,t}, \mathcal{U}_{b+1,1:R})$ // Predict on next b
    end for
end for
```

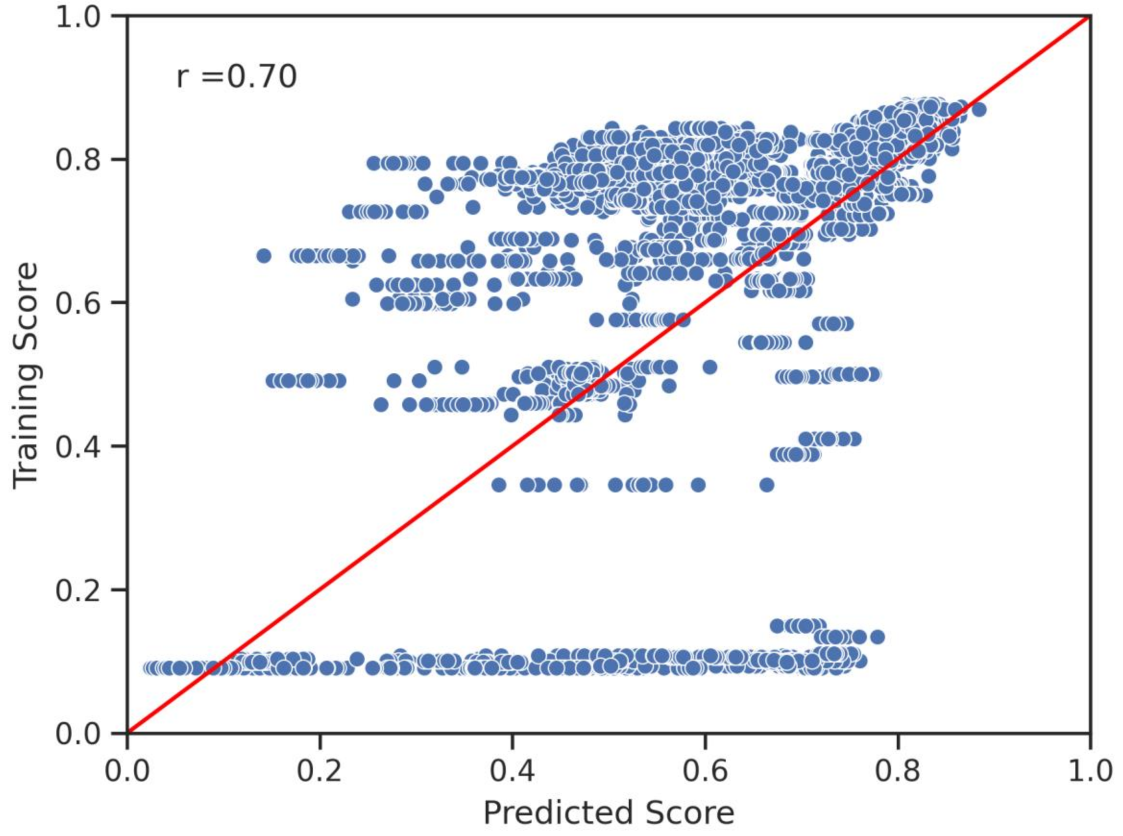
Implementation Details

- Random Search:
  - Up to Number of Blocks (B): 2
  - Number of Repetitions (N): 2
  - Number of Filters (F): 24
  - Number of Operations: 8
  - Number of models sampled (and trained) per step (K): 264(136 for B = 1)

- Predictor Performance Correlation:
  - Number of runs (T): 20
  - Number of Blocks(N): 2
  - Number of models sampled (K): 256

Predictor Performance Correlation: Results





Conclusions

- In this experimental work we reimplemented the PNAS method for architecture search;
- There is a problem with one of the operation implemented, the Dilated Convolution;
- Due to computational limitations, we couldn't verify if the correlation coefficient is due to limited samples or not;
- Due to possible differences in the implementation, a lower K, or missing details the best cells found are different from the paper;

Conclusions

- Correlation of performance between top models trained on CIFAR-10 and ImageNet
- Portability to ImageNet
- Comparisons with other search methods

References

1. Liu, Chenxi, et al. "Progressive neural architecture search." Proceedings of the European conference on computer vision (ECCV). 2018.
2. Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
3. Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." arXiv preprint arXiv:1611.01578 (2016).