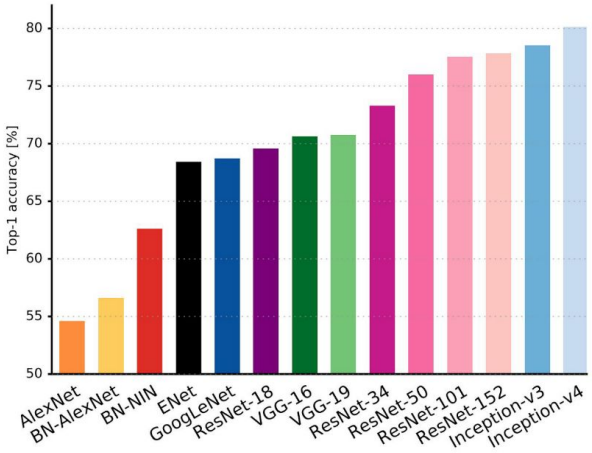
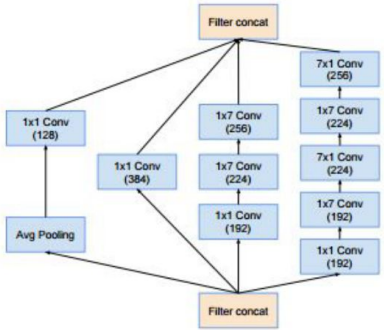
# Neural Architecture Search
# A Tutorial

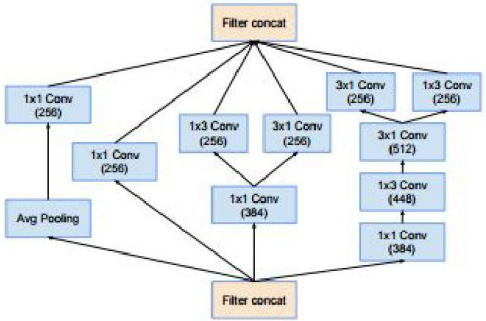
**Nikhil Naik**
**nnaik@salesforce.com**

# Importance of Neural Architectures in Vision




**Canziani et al (2017)**

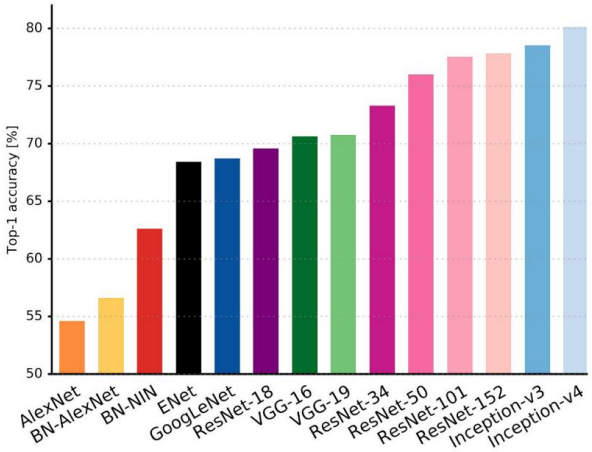


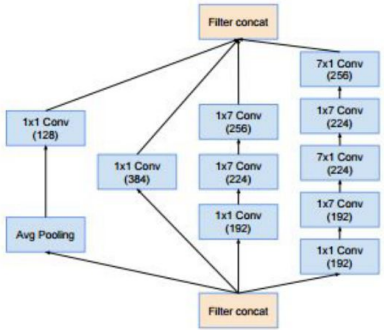

**Complex hand-engineered layers from Inception-V4 (Szegedy et al., 2017)**

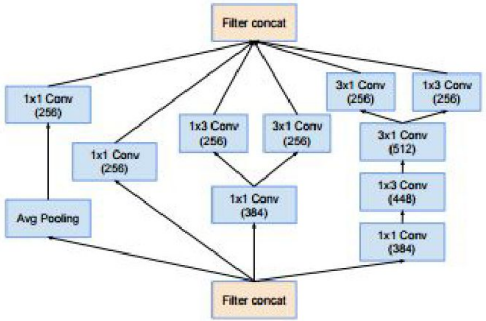
**Design Innovations (2012 - Present):** Deeper networks, stacked modules, skip connections, squeeze-excitation block, ...

# Importance of Neural Architectures in Vision




Canziani et al (2017)




Complex hand-engineered layers from Inception-V4 (Szegedy et al., 2017)

## Can we try and learn good architectures automatically?



Slide courtesy Quoc Le

# Neural Architecture Search

# Neural Architecture Search: Early Work

- **Neuroevolution:** Evolutionary algorithms (e.g., Miller et al., 89; Schaffer et al., 92; Stanley and Miikkulainen, 02; Verbancsics & Harguess, 13)
- Random search (e.g., Pinto et al., 09)
- Bayesian optimization for architecture and hyperparameter tuning (e.g., Snoek et al, 12; Bergstra et al., 13; Domhan et al., 15)

# Renewed Interest in Neural Architecture Search (2017 -)

## NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING

**Barret Zoph,*** **Quoc V. Le**
Google Brain

## DESIGNING NEURAL NETWORK ARCHITECTURES USING REINFORCEMENT LEARNING

**Bowen Baker, Otkrist Gupta, Nikhil Naik & Ramesh Raskar**
Media Laboratory
Massachusetts Institute of Technology

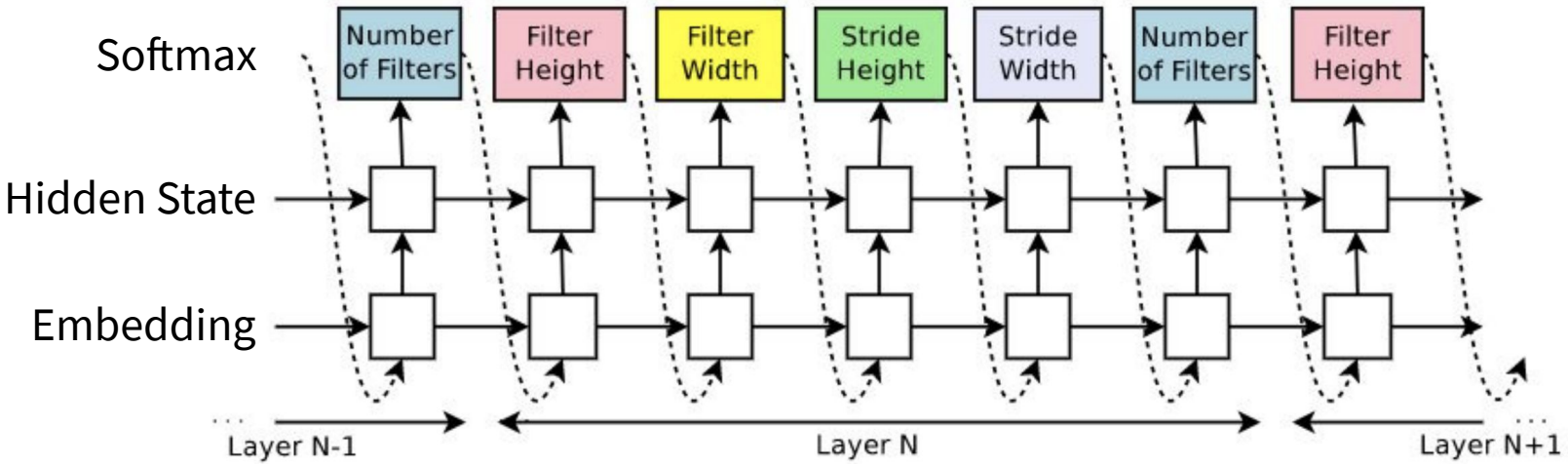
# Neural Architecture Search: Key Ideas

- Specify the structure and connectivity of a neural network by using a configuration string (e.g., *[“Filter Width: 5”, “Filter Height: 3”, “Num Filters: 24”]*)
- Zoph and Le (2017): Use a RNN (“Controller”) to generate this string that specifies a neural network architecture
- Train this architecture (“Child Network”) to see how well it performs on a validation set
- Use reinforcement learning to update the parameters of the Controller model based on the accuracy of the child model

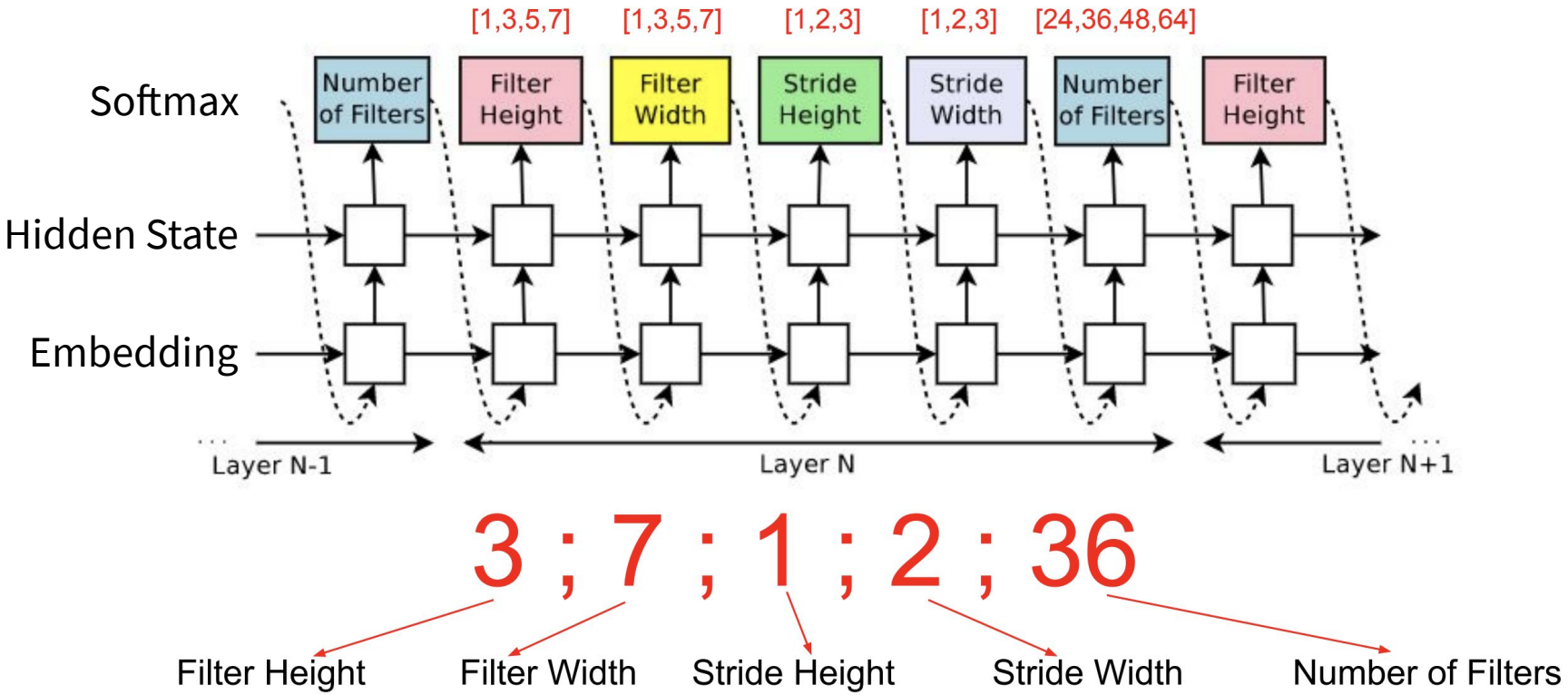


Slide courtesy Quoc Le

# Training with REINFORCE (Zoph and Le, 2017)



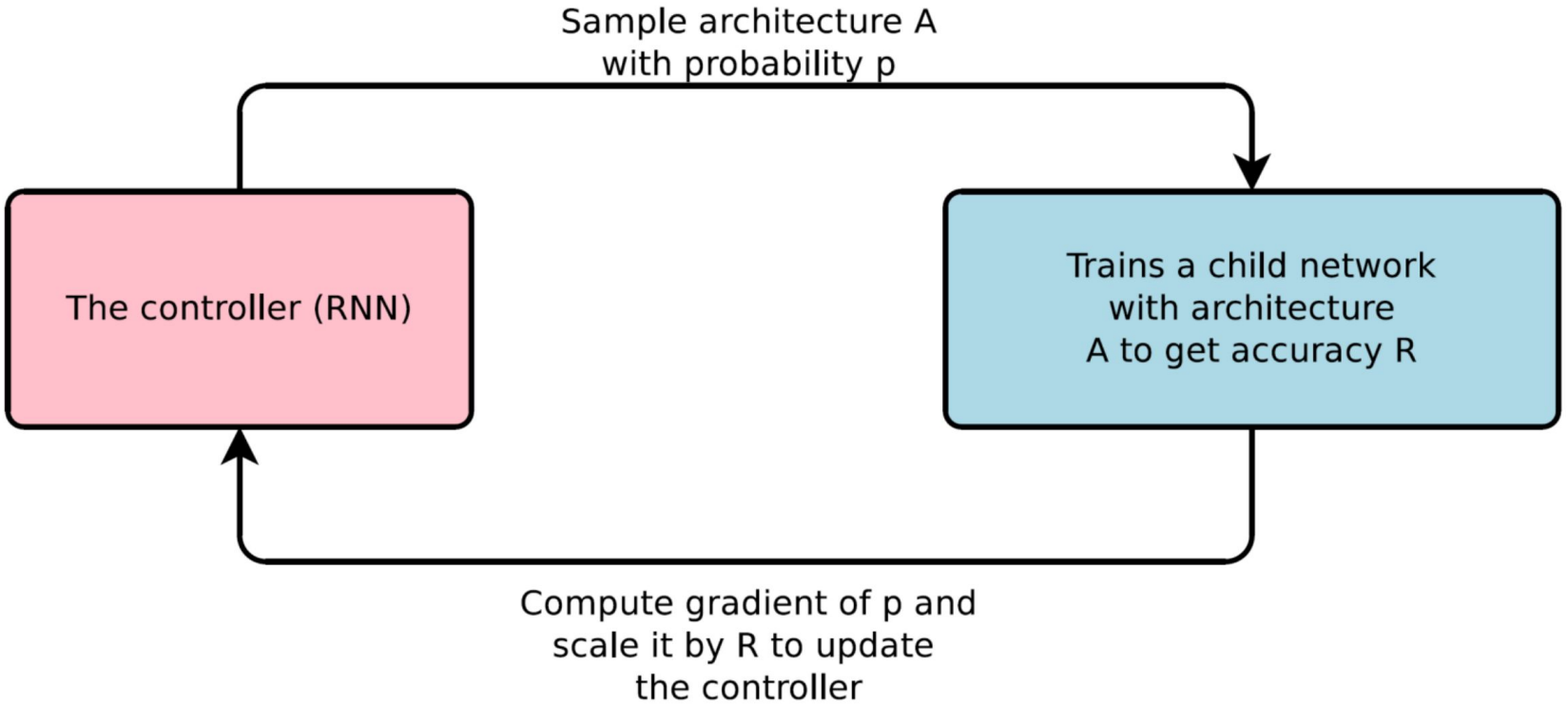
# Training with REINFORCE (Zoph and Le, 2017)






Slide courtesy Quoc Le

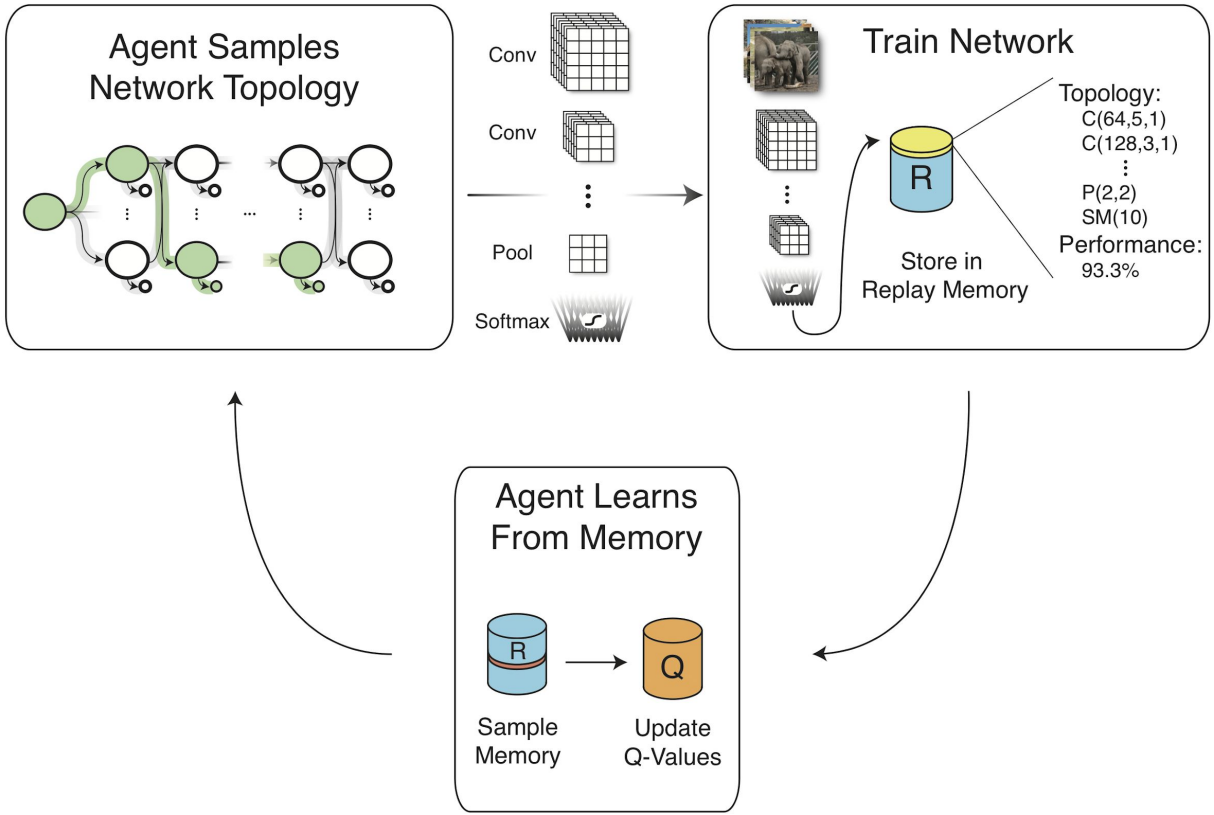
# Training with REINFORCE (Zoph and Le, 2017)





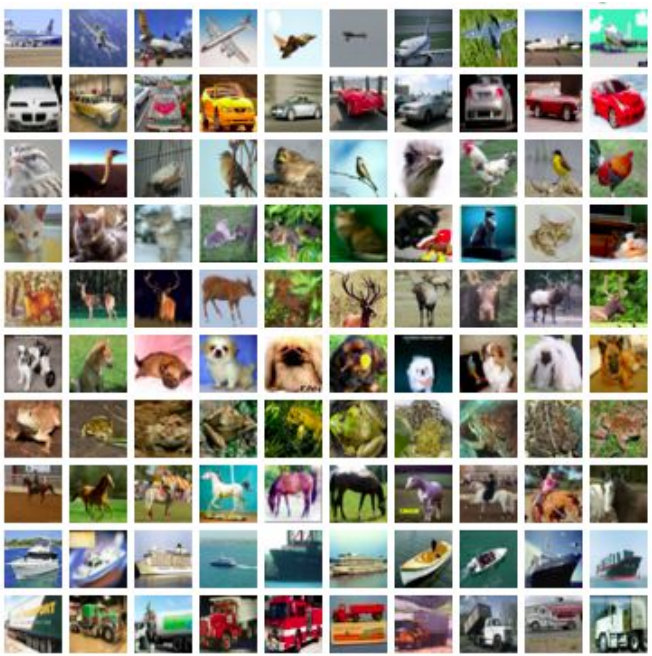

Slide courtesy Quoc Le

# Q-Learning with Experience Replay (Baker et al., 2017)



# Results on CIFAR-10

**Zoph and Le (2017):** 12,800 architectures trained using 22,400 GPUdays
**Baker et al. (2017)**: 2,700 architectures trained using 100 GPUdays

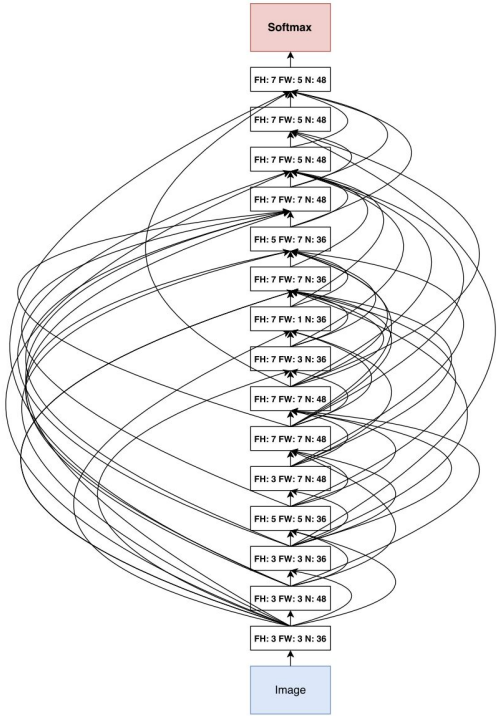


50K Training Samples
10K Testing Samples
10 classes

# Results on CIFAR-10

| Model | Depth | Parameters | Error rate (%) |
|---|---|---|---|
| Wide ResNet (Zagoruyko & Komodakis, 2016) | 16 | 11.0M | 4.81 |
| | 28 | 36.5M | 4.17 |
| ResNet (pre-activation) (He et al., 2016b) | 164 | 1.7M | 5.46 |
| | 1001 | 10.2M | 4.62 |
| DenseNet ($L = 40, k = 12$) Huang et al. (2016a) | 40 | 1.0M | 5.24 |
| DenseNet($L = 100, k = 12$) Huang et al. (2016a) | 100 | 7.0M | 4.10 |
| DenseNet ($L = 100, k = 24$) Huang et al. (2016a) | 100 | 27.2M | 3.74 |
| DenseNet-BC ($L = 100, k = 40$) Huang et al. (2016b) | 190 | 25.6M | 3.46 |
| Neural Architecture Search v1 no stride or pooling | 15 | 4.2M | 5.50 |
| Neural Architecture Search v2 predicting strides | 20 | 2.5M | 6.01 |
| Neural Architecture Search v3 max pooling | 39 | 7.1M | 4.47 |
| Neural Architecture Search v3 max pooling + more filters | 39 | 37.4M | 3.65 |
| MetaQNN (Baker et al. 2017) no skip connections | 12 | 11.8M | 6.92 |

Comparable accuracy to best human-designed models ~2017




Best NAS Architecture on CIFAR-10
Zoph and Le (2017)

# Neural Architecture Search: General Problem Setup

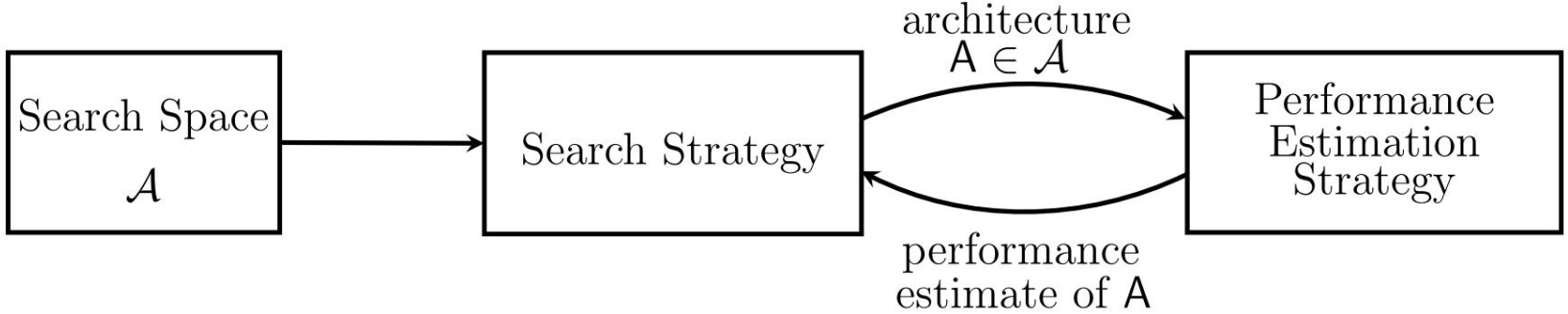
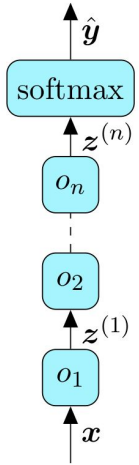


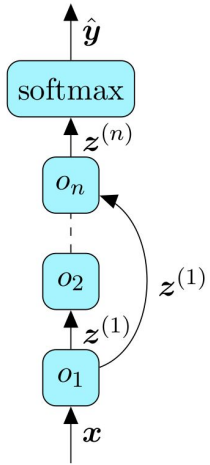



Image courtesy Witsuba et al. (2019)

# NAS: Global Search Space




**MetaQNN**
**(Baker et al. 2017)**



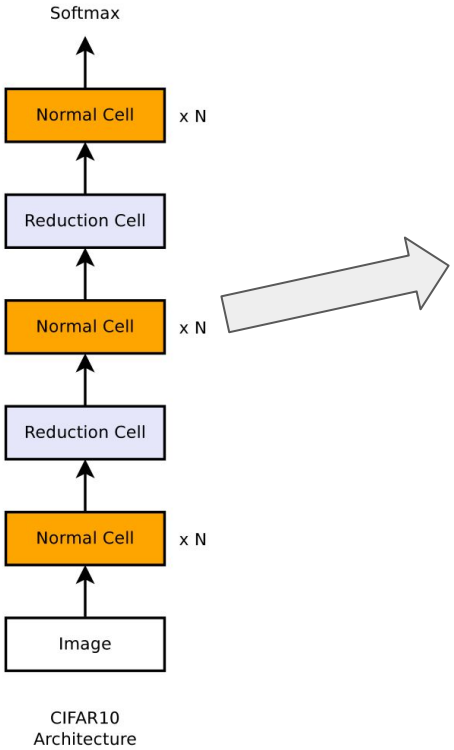

**NAS**
**(Zoph and Le, 2017)**

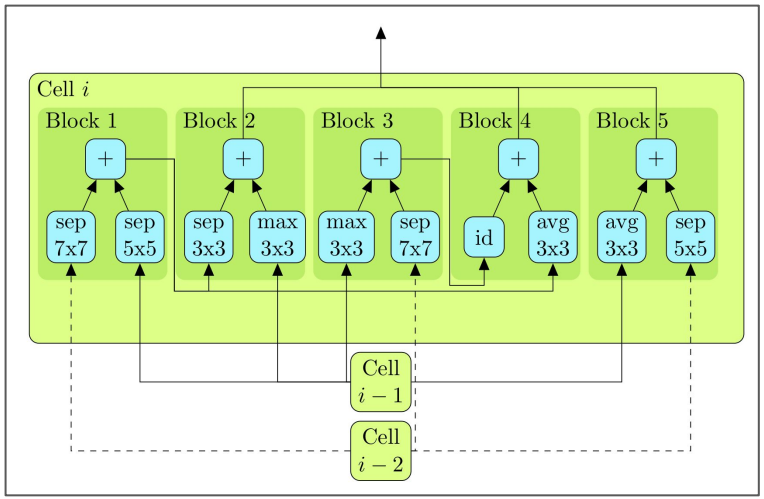
**Global search space is rigid, impractical to scale and transfer**



Image courtesy Witsuba et al. (2019)

# NAS: Cell-based Search Space



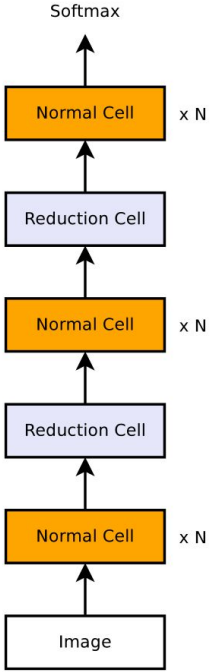




**NASNet**
**Zoph et al. (2018)**

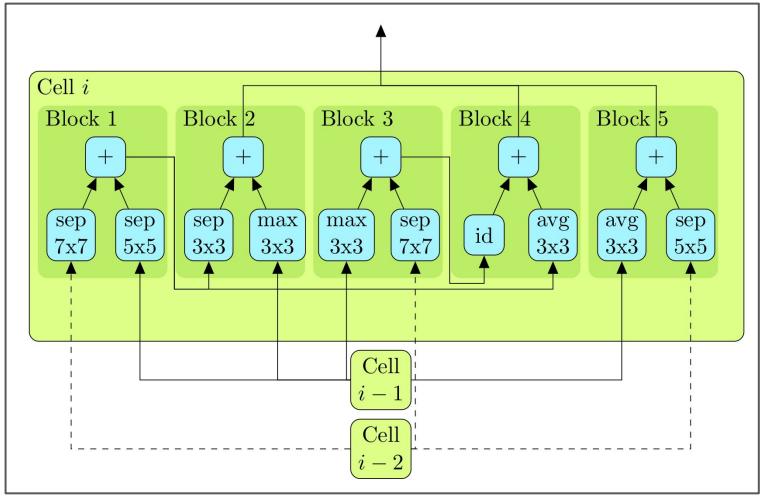


Image courtesy Witsuba et al. (2019)

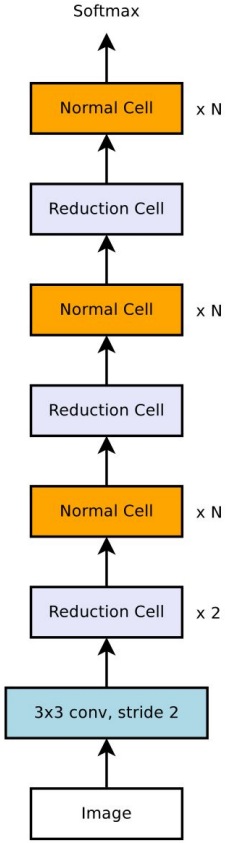
# NAS: Cell-based Search Space




CIFAR10
Architecture

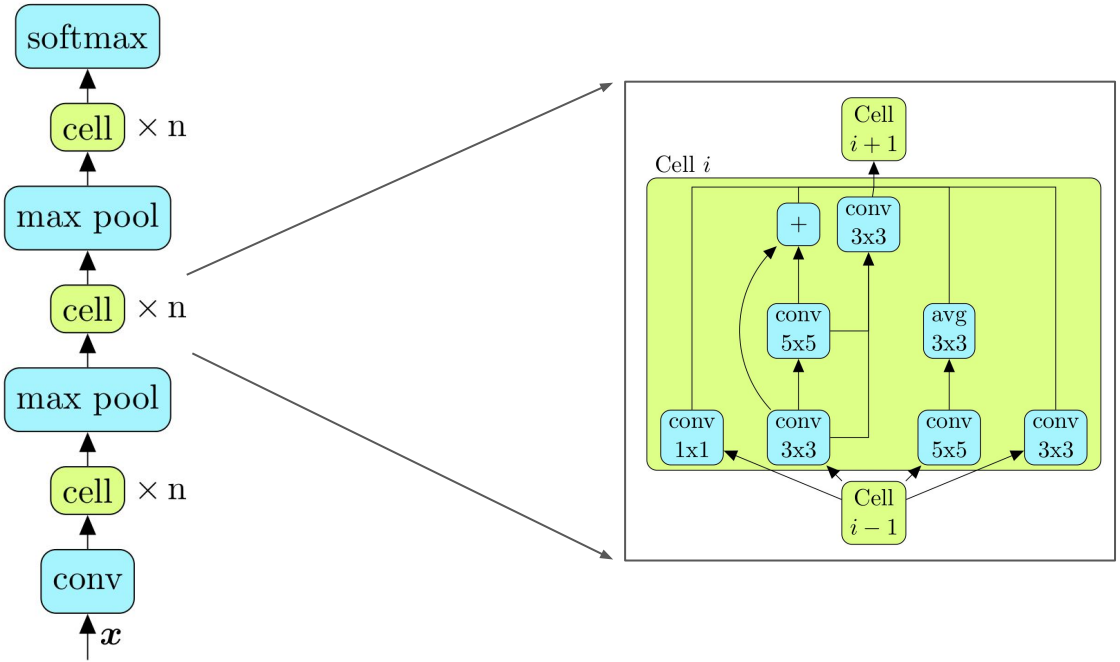



**NASNet**
**Zoph et al. (2018)**




ImageNet
Architecture

# NAS: Cell-based Search Space




**BlockQNN**
**Zong et al. (2018)**



Image courtesy Witsuba et al. (2019)

# Transferring Architectures from CIFAR-10 to ImageNet

| Search Method | Search Space | Search Strategy | Search Cost (GPU-days) | CIFAR-10 Error | **ImageNet Top-1 Error** |
|---|---|---|---|---|---|
| MetaQNN (Baker et al, 2017) | Global | Q-learning | 100 | 6.92 | - |
| NAS (Zoph and Le, 2017) | Global | REINFORCE | 22,400 | 3.65 | - |
| BlockQNN (Zong et al, 2018) | Cell-based | Q-learning | 96 | 3.54 | **22.6** |
| NASNet (Zoph et al, 2018) | Cell-based | PPO | 2,000 | 3.41 | **17.3** |

# Search Strategy: Evolution

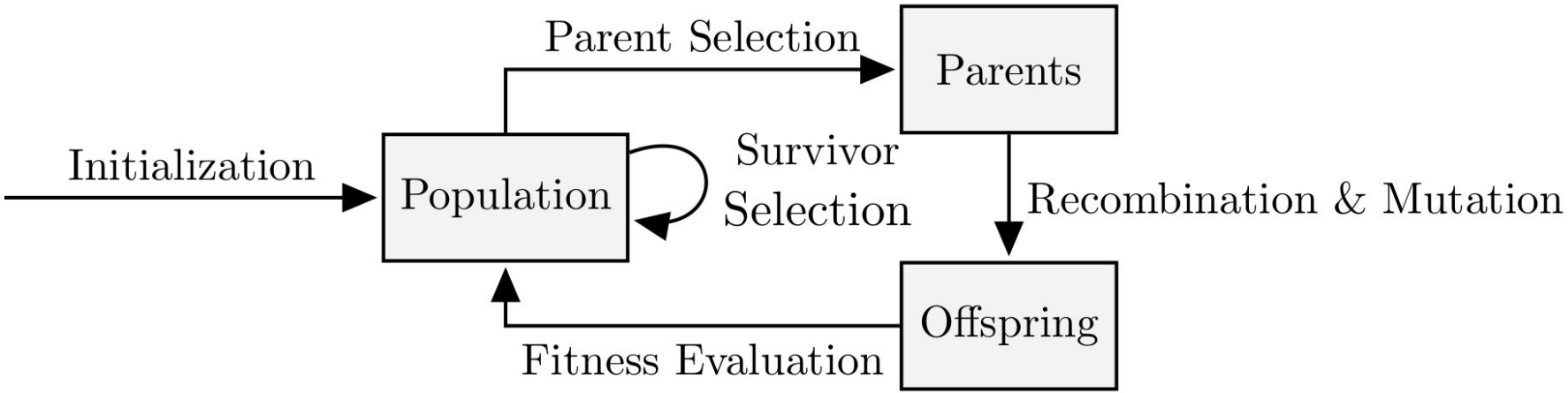
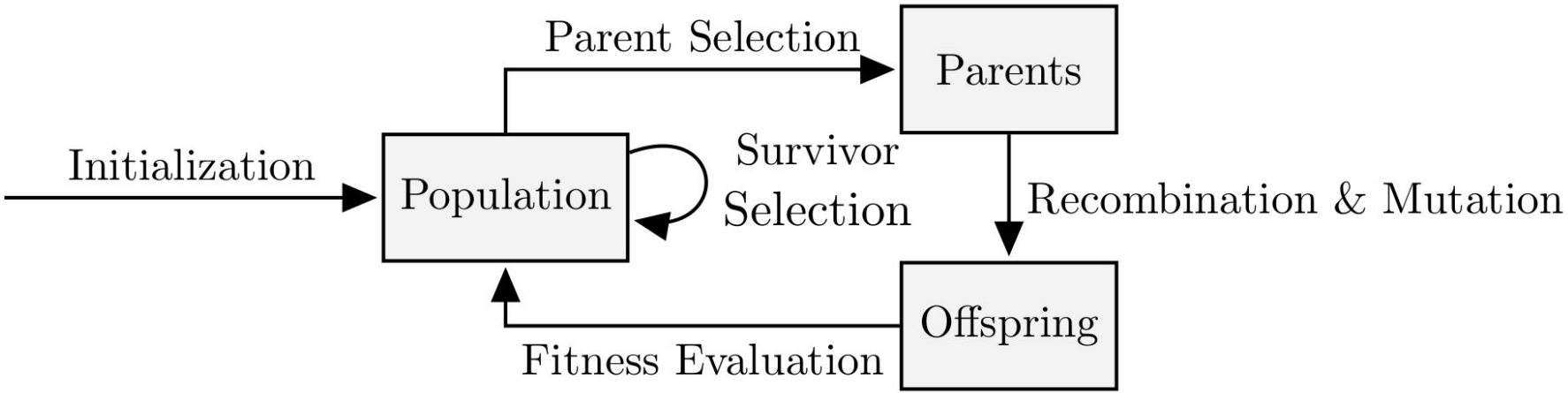





Image courtesy Witsuba et al. (2019)

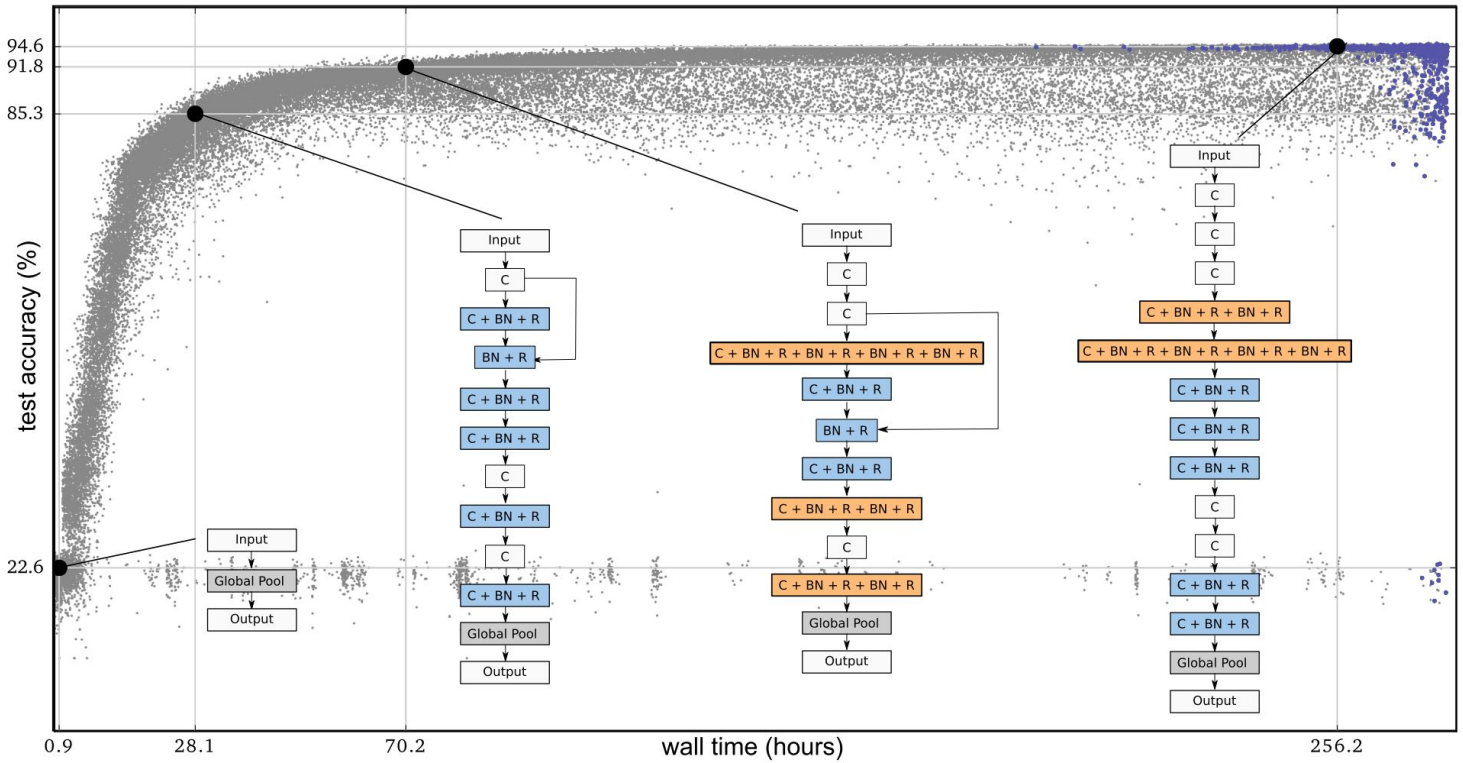
# Search Strategy: Evolution




**Typical Policies for Parent Selection and Offspring Retention**

Tournament, Fitness-proportionate selection, Youngest

# Evolution over Global Search Space on CIFAR-10




**Real et al. (2017)**

# Evolution: Results on CIFAR-10

| Method | Parent Sel. | Survivor Sel. | Error (%) |
|---|---|---|---|
| Real et al. (2017) | tournament | tournament | 5.40 |
| Xie and Yuille (2017) | all | fitness prop. | 5.39 |
| Suganuma et al. (2017) | n/a | elitist | 5.98 |
| Liu et al. (2018b) | tournament | all | 3.75 |
| Real et al. (2019) | tournament | youngest | 3.34 |
| Elsken et al. (2018) | n/a | elitist | 5.2 |
| Wistuba (2018) | tournament | all | 3.57 |



Image courtesy Witsuba et al. (2019)

# Computational Cost of NAS on CIFAR-10

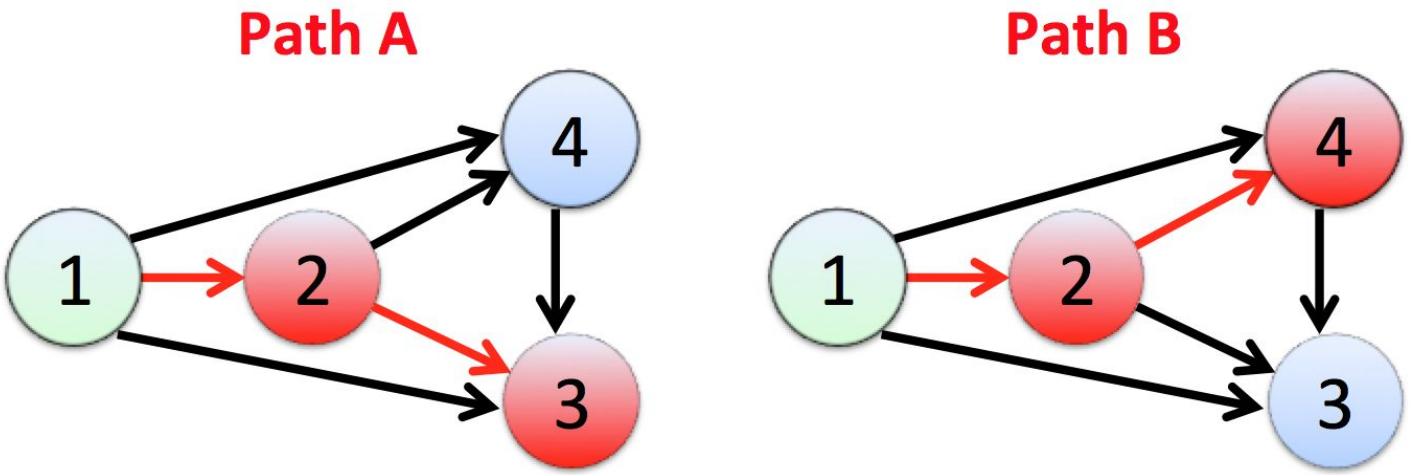
| Reference | Error (%) | Params (Millions) | GPU Days | |
|---|---|---|---|---|
| Baker et al. (2017) | 6.92 | 11.18 | 100 | Reinforcement Learning |
| Zoph and Le (2017) | 3.65 | 37.4 | 22,400 | |
| Cai et al. (2018a) | 4.23 | 23.4 | 10 | |
| Zoph et al. (2018) | 3.41 | 3.3 | 2,000 | |
| Zoph et al. (2018) + Cutout | 2.65 | 3.3 | 2,000 | |
| Zhong et al. (2018) | 3.54 | 39.8 | 96 | |
| Cai et al. (2018b) | 2.99 | 5.7 | 200 | |
| Cai et al. (2018b) + Cutout | 2.49 | 5.7 | 200 | |
| Real et al. (2017) | 5.40 | 5.4 | 2,600 | Evolution |
| Xie and Yuille (2017) | 5.39 | N/A | 17 | |
| Suganuma et al. (2017) | 5.98 | 1.7 | 14.9 | |
| Liu et al. (2018b) | 3.75 | 15.7 | 300 | |
| Real et al. (2019) | 3.34 | 3.2 | 3,150 | |

**Designing competitive networks can take hundreds of GPU-days!**
**How to make neural architecture search more efficient?**



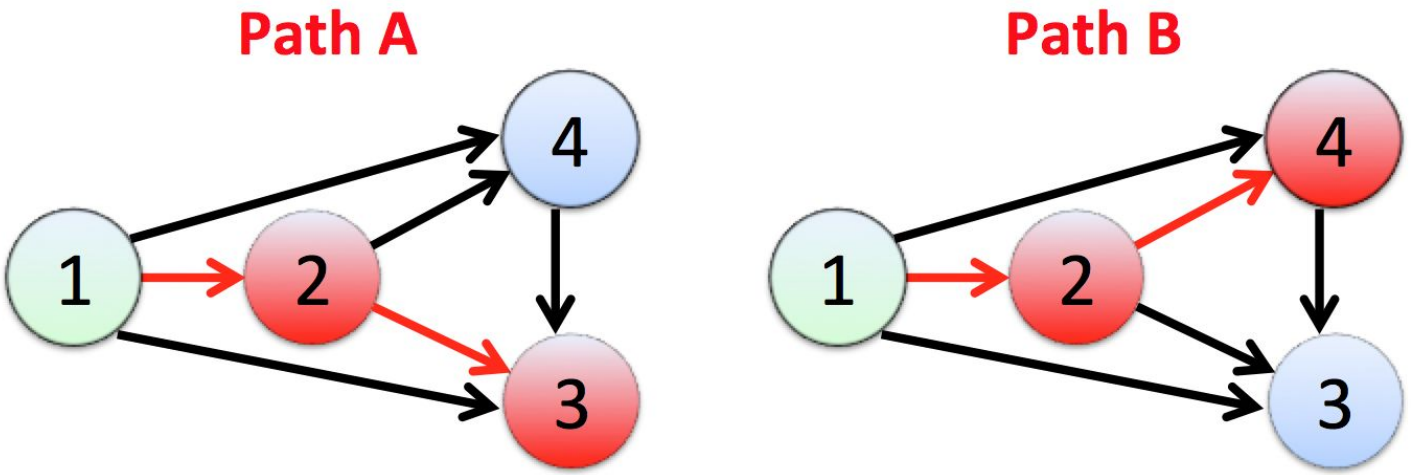
Image courtesy Witsuba et al. (2019)

# How To Make NAS More Efficient?



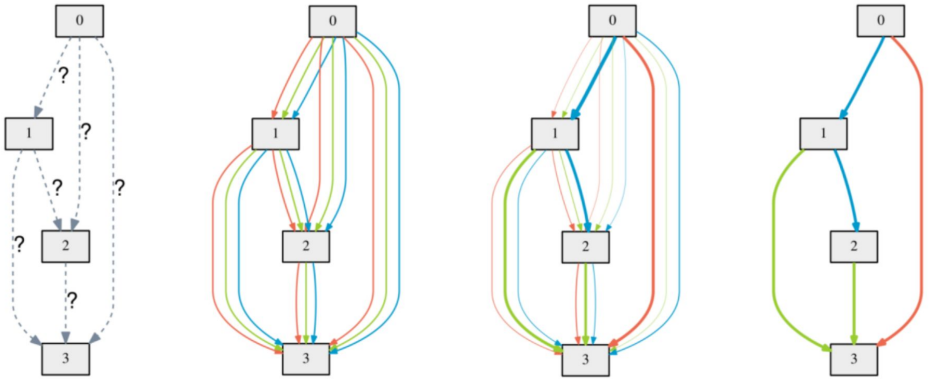

- Currently, models defined by **path A** and **path B** are trained independently
- Instead, treat all model trajectories as sub-graphs of a single directed acyclic graph
- Use a search strategy (e.g., RL, Evolution) to choose sub-graphs. Proposed in ENAS (Pham et al, 2018)

# Efficient NAS using Weight Sharing




- Currently, models defined by **path A** and **path B** are trained independently
- Instead, treat all model trajectories as sub-graphs of a single directed acyclic graph
- Use a search strategy (e.g., RL, Evolution) to choose sub-graphs. Proposed in ENAS (Pham et al, 2018)

# Gradient-based NAS with Weight Sharing




Find encoding weights $\alpha^*$ that

$$\min_{\alpha} \quad \mathcal{L}_{val}(w^*(\alpha), \alpha)$$
$$\text{s.t.} \quad w^*(\alpha) = \text{argmin}_w \ \mathcal{L}_{train}(w, \alpha)$$

Learn design of normal and reduction cells

**DARTS (Liu et al., 2018)**

---

Create a mixed operation $\bar{o}^{(i,j)}$ parametrized by $\alpha^{(i,j)}$ for each edge $(i, j)$

**while** *not converged* **do**

1. Update architecture $\alpha$ by descending $\nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$ ($\xi = 0$ if using first-order approximation)
2. Update weights $w$ by descending $\nabla_w \mathcal{L}_{train}(w, \alpha)$

Derive the final architecture based on the learned $\alpha$.

---



Also see: SNAS (Xie et al. 2019)

# Efficient NAS with Weight Sharing: Results on CIFAR-10

| Reference | Error (%) | Params (Millions) | GPU Days |
|---|---|---|---|
| Pham et al. (2018) | 3.54 | 4.6 | 0.5 |
| Pham et al. (2018) + Cutout | 2.89 | 4.6 | 0.5 |
| Bender et al. (2018) | 4.00 | 5.0 | N/A |
| Casale et al. (2019) + Cutout | 2.81 | 3.7 | 1 |
| Liu et al. (2018c) + Cutout | 2.76 | 3.3 | 4 |
| Xie et al. (2019b) + Cutout | 2.85 | 2.8 | 1.5 |
| Cai et al. (2019) + Cutout | 2.08 | 5.7 | 8.33 |
| Brock et al. (2018) | 4.03 | 16.0 | 3 |
| Zhang et al. (2019) | 4.30 | 5.1 | 0.4 |

**Limitations:**

- Restrict the search space to the subgraphs of the supergraph
- Can bias the search towards certain regions of the search space
- In practice, the need to hold entire supergraph in GPU memory restricts search space size



Image courtesy Witsuba et al. (2019)

# How Well Do Automatically Designed Architectures Transfer To Other Datasets?

# CIFAR-10 → ImageNet Architecture Transfer

| Model | Params | ×+ | 1/5-Acc (%) |
|---|---|---|---|
| Inception V3 | 23.8M | 5.72B | 78.8 / 94.4 |
| Xception | 22.8M | 8.37B | 79.0 / 94.5 |
| Inception ResNet V2 | 55.8M | 13.2B | 80.4 / 95.3 |
| ResNeXt-101 (64x4d) | 83.6M | 31.5B | 80.9 / 95.6 |
| PolyNet | 92.0M | 34.7B | 81.3 / 95.8 |
| Dual-Path-Net-131 | 79.5M | 32.0B | 81.5 / 95.8 |
| Squeeze-Excite-Net | 145.8M | 42.3B | 82.7 / 96.2 |
| GeNet-2* | 156M | – | 72.1 / 90.4 |
| Block-QNN-B (N=3)* | – | – | 75.7 / 92.6 |
| Hierarchical (2, 64)* | 64M | – | 79.7 / 94.8 |
| PNASNet-5 (4, 216) | 86.1M | 25.0B | 82.9 / 96.1 |
| NASNet-A (6, 168) | 88.9M | 23.8B | 82.7 / 96.2 |
| AmoebaNet-B (6, 190)* | 84.0M | 22.3B | 82.3 / 96.1 |
| AmoebaNet-A (6, 190)* | 86.7M | 23.1B | 82.8 / 96.1 |
| AmoebaNet-A (6, 204)* | 99.6M | 26.2B | 82.8 / 96.2 |
| AmoebaNet-C (6, 228)* | 155.3M | 41.1B | **83.1 / 96.3** |

# Architecture Transferability to Other Datasets and Tasks

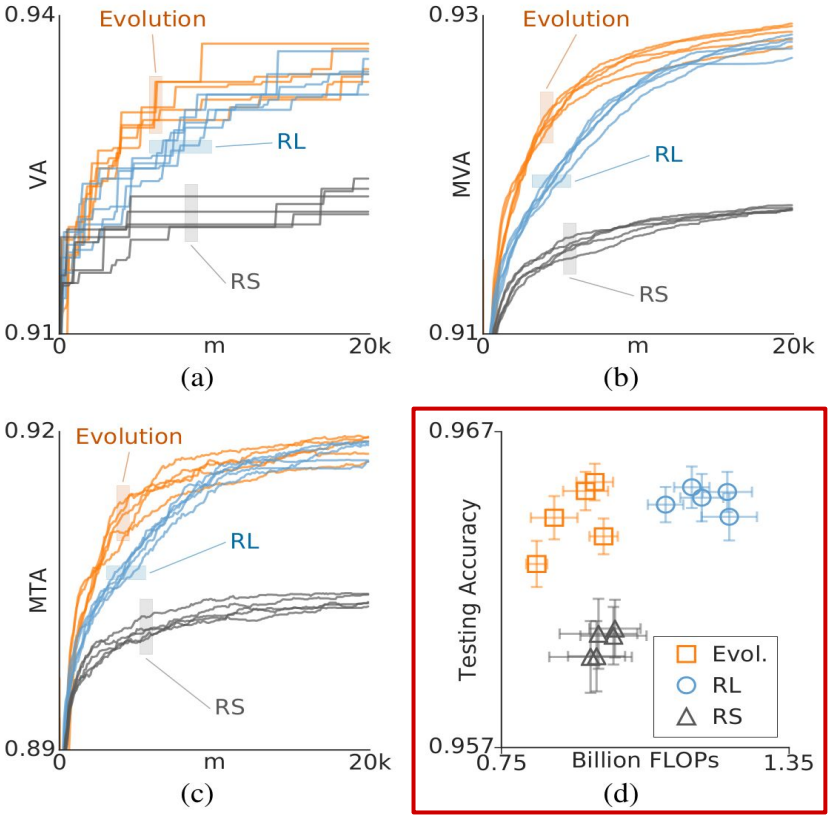
| Dataset | Acc. | Network | Acc. | Best network |
|---|---|---|---|---|
| Food-101 | **90.0** | Deep layer aggregation [40] | **90.1** | NASNet-A Large, fine-tuned ✓ |
| CIFAR-10 | 97.9 | AmoebaNet [41] | **98.4**[a] | NASNet-A Large, fine-tuned ✓ |
| CIFAR-100 | 87.8 | ShakeDrop [42] | **88.2**[a] | NASNet-A Large, fine-tuned ✓ |
| Birdsnap | **80.2**[b] | Mask-CNN [43] | 78.5 | NASNet-A Large, fine-tuned ✓ |
| SUN397 | 63.2 | Places-pretrained VGG [44] | **66.5** | NASNet-A Large, fine-tuned ✓ |
| Stanford Cars | **94.1** | Deep layer aggregation [40] | 93.0 | Inception v4, random init |
| FGVC Aircraft | **92.9**[b] | Deep layer aggregation [40] | 89.4 | Inception v3, fine-tuned |
| VOC 2007 Cls. | **89.7** | VGG [9] | 88.4 | NASNet-A Large, fine-tuned ✓ |
| DTD | 75.5 | FC+FV-CNN+D-SIFT [45] | **76.7** | Inception-ResNet v2, fine-tuned |
| Oxford-IIIT Pets | 93.8 | Object-part attention [46] | **94.3** | NASNet-A Large, fine-tuned ✓ |
| Caltech-101 | 93.4 | Spatial pyramid pooling [47] | **95.0** | NASNet-A Large, fine-tuned ✓ |
| Oxford 102 Flowers | 97.1 | Object-part attention [46] | **97.7** | NASNet-A Large, fine-tuned ✓ |

NASNet-A: SOTA on **8/13 commonly used classification benchmarks**

NAS architectures also transfer for object detection and semantic segmentation

# Are Neural Architecture Search Strategies better than Random Search?

# Are Intelligent Search Strategies Better Than Random Search?



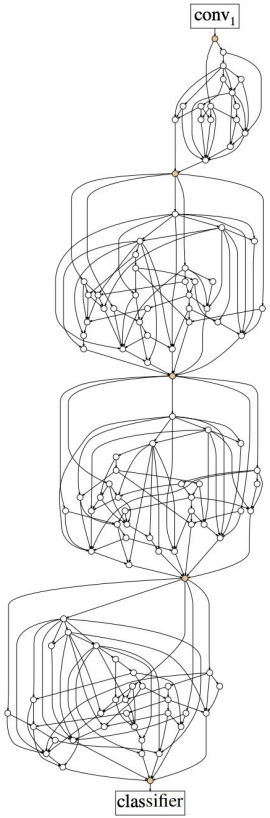

**Real et al. (2018)**
The difference in accuracy between best models found by random search, RL, and Evolution is **less than 1%** on CIFAR-10

# Are Intelligent Search Strategies Better Than Random Search?

| Architecture | Source | Test Error Best | Test Error Average | Params (M) |
|---|---|---|---|---|
| NASNet-A#* | [52] | N/A | 2.65 | 3.3 |
| AmoebaNet-B* | [43] | N/A | 2.55 ± 0.05 | 2.8 |
| ProxylessNAS† | [7] | 2.08 | N/A | 5.7 |
| GHN#† | [50] | N/A | 2.84 ± 0.07 | 5.7 |
| SNAS† | [47] | N/A | 2.85 ± 0.02 | 2.8 |
| ENAS† | [41] | 2.89 | N/A | 4.6 |
| ENAS | [34] | 2.91 | N/A | 4.2 |
| Random search baseline | [34] | N/A | 3.29 ± 0.15 | 3.2 |
| DARTS (first order) | [34] | N/A | 3.00 ± 0.14 | 3.3 |
| DARTS (second order) | [34] | N/A | 2.76 ± 0.09 | 3.3 |
| DARTS (second order)‡ | Ours | 2.62 | 2.78 ± 0.12 | 3.3 |
| ASHA baseline | Ours | 2.85 | 3.03 ± 0.13 | 2.2 |
| Random search WS‡ | Ours | 2.71 | 2.85 ± 0.08 | 4.3 |

**Li and Talwalkar (2019)**
Random search baseline finds a model with 2.71% error on CIFAR-10, comparable to best NAS methods based on RL, Evolution, Gradient Descent

# Are Intelligent Search Strategies Better Than Random Search?

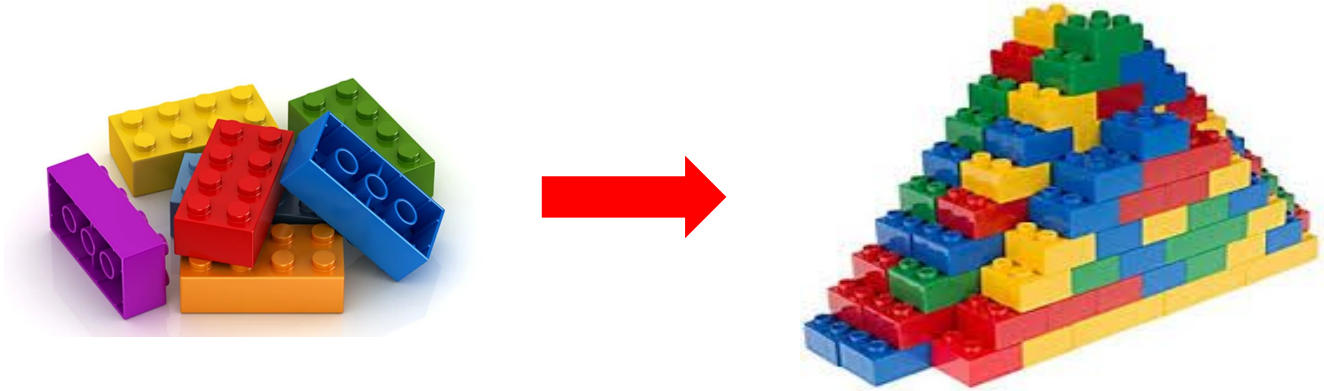



**Xie et al. (2019)**

A network consisting of multiple randomly wired "cells" is only **1.3% less accurate** than a similar capacity NAS models on **ImageNet**

| network | test size | epochs | top-1 acc. | top-5 acc. | FLOPs (B) | params (M) |
|---|---|---|---|---|---|---|
| NASNet-A [56] | $331^2$ | >250 | 82.7 | 96.2 | 23.8 | 88.9 |
| Amoeba-B [34] | $331^2$ | >250 | 82.3 | 96.1 | 22.3 | 84.0 |
| Amoeba-A [34] | $331^2$ | >250 | 82.8 | 96.1 | 23.1 | 86.7 |
| PNASNet-5 [26] | $331^2$ | >250 | 82.9 | 96.2 | 25.0 | 86.1 |
| **RandWire-WS** | $320^2$ | 100 | $81.6_{\pm 0.13}$ | $95.6_{\pm 0.07}$ | $16.0_{\pm 0.36}$ | $61.5_{\pm 1.32}$ |

# Are Intelligent Search Strategies Better Than Random Search?



- Random search is a competitive baseline
- How do you design more flexible search spaces and sample-efficient search methods?
- Can intelligent search methods help discover new design motifs and basic building blocks?

# Designing Efficient Architectures

# Designing Efficient Architectures

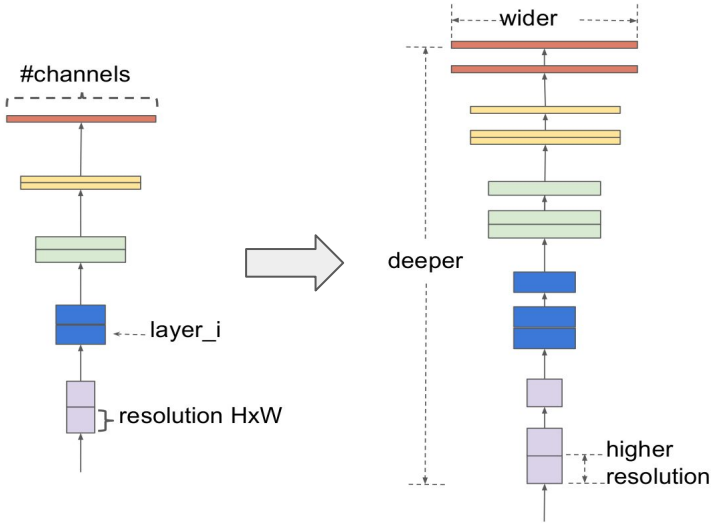
- **Design high-performing models given a budget on memory and inference time**
- Many applications in resource constrained-settings: Mobile devices, autonomous vehicles, robotics, production models
- Popular approach: Tweak existing architectures  with weight pruning  (e.g. LeCun et al. ‘89, Huang et al. ‘18) or weight quantization (e.g., Binarized neural nets ‘16, XNORnet ‘16)
- Neural Architecture Search is well-suited for this application

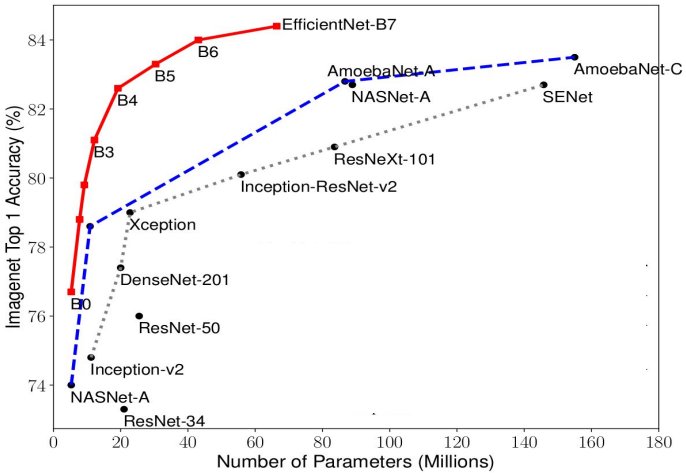
# Designing Efficient Architectures

Introduce constraints like memory and inference time in addition to accuracy

| **NAS Method** | **References** |
| --- | --- |
| Constrained Optimization | Tan et al. (2018), Cai et al. (2018), Hsu et al. (2018) |
| Multi-objective Optimization | Kim et al. (2017), Cai et al. (2019), Lu et al. (2018), Dong et al. (2018), Elsken et al. (2019), Tan and Le (2019) |
| Automated Pruning | He et al. (2019) |

# Designing Efficient Architectures from Scratch



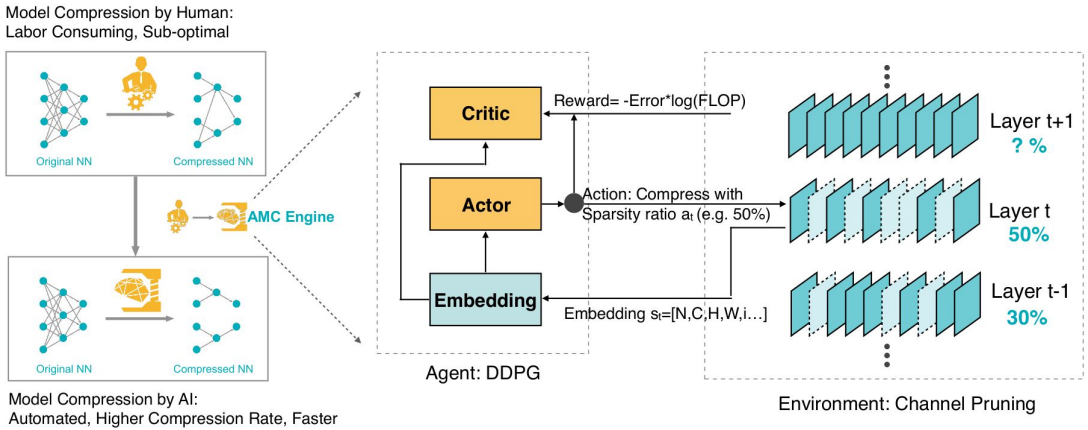



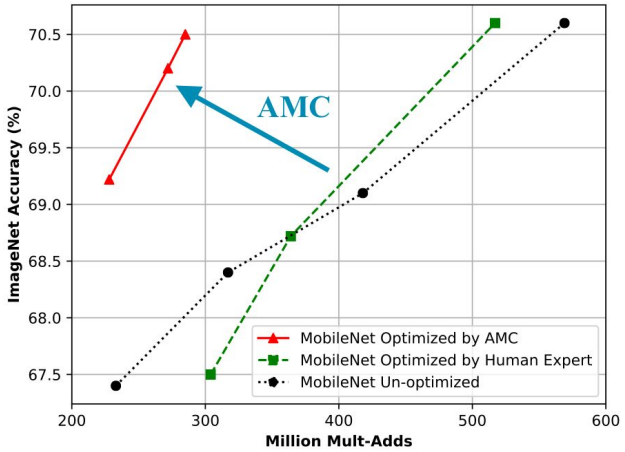

**EfficientNet** (Tan and Le, 2019)

RL-based architecture search for a feed-forward block + scaling up using grid search

**80% Top-1 Accuracy on ImageNet with 5x fewer parameters and 13x fewer FLOPS** than best human-designed model

# Designing Efficient Architectures: Auto-Pruning

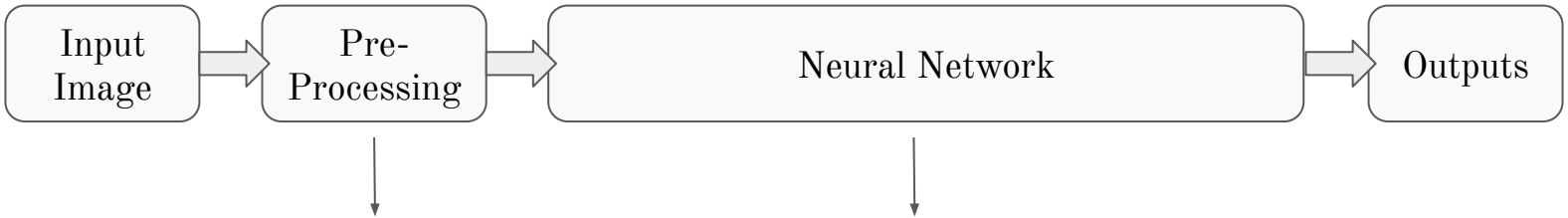


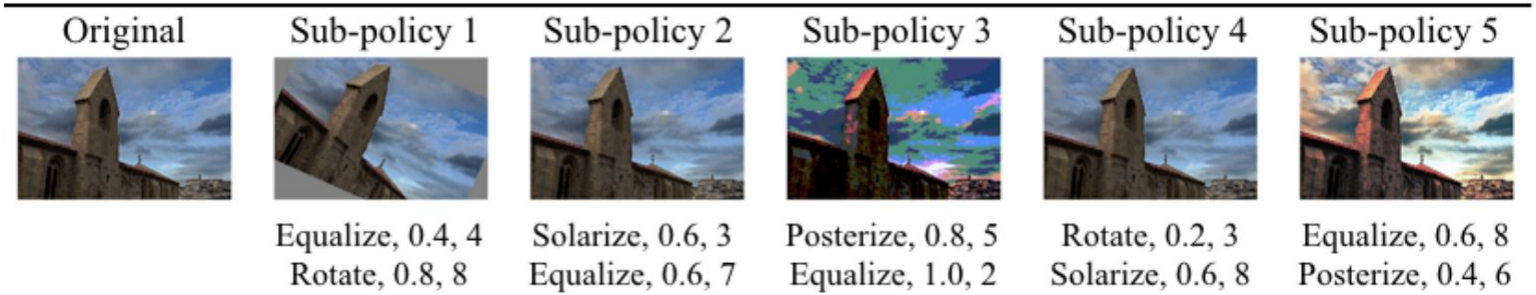




AMC: AutoML for Model Compression (He et al., 2019)
Using Deep Deterministic Policy Gradients to learn pruning ratio for each layer

# Automating the Entire Deep Learning Stack

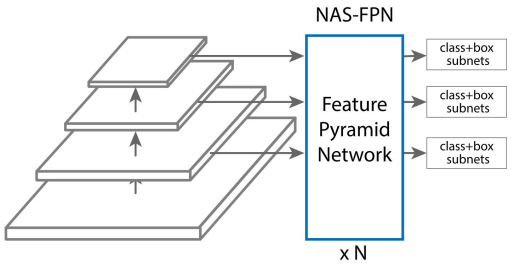
# Automating the Deep Learning Stack



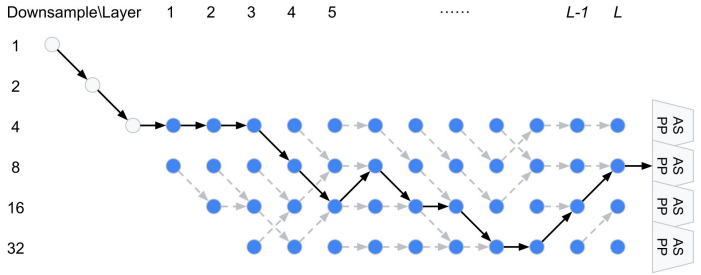




| Model | Baseline | Inception Pre-processing [14] | AutoAugment |
|---|---|---|---|
| ResNet-50 [15] | 24.70 / 7.80 | 23.69 / 6.92 | 22.37 / 6.18 |
| ResNet-200 [15] | - | 21.52 / 5.85 | 20.00 / 4.99 |

# Neural Architecture Search: Beyond Image Classification




**Object Detection**
E.g. NAS-FPN (Ghaisi et al., 2019)




**Semantic Segmentation**
E.g. Auto-DeepLab (Liu et al., 2019)

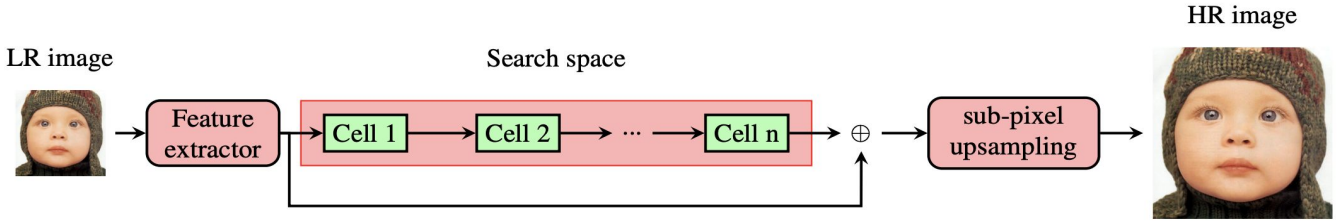



**Image Super-Resolution** e.g. MoreMNAS(Chu et al., 2019)

# Summary

- **Neural architecture search** can generate state-of-the-art architectures for computer vision tasks given a well-defined search space
- Develop new optimizers, activation functions, data augmentation strategies
- Architecture search can be especially useful for designing efficient architectures for resource-constrained settings

# Future Directions

- Search efficiency
- Moving towards less constrained search spaces
- Designing efficient architectures: Automated scaling, pruning and quantization
- Joint optimization of all components in the deep learning pipeline (data augmentation, architectures, activation functions, training algorithms)
- Designing architectures for multimodal problems, e.g., vision and language

# Useful Survey Papers and Datasets

Elsken, Thomas, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A survey." *JMLR* (2019)

Wistuba, Martin, Ambrish Rawat, and Tejaswini Pedapati. "A Survey on Neural Architecture Search." *arXiv preprint arXiv:1905.01392* (2019)

Ying, Chris, Aaron Klein, Esteban Real, Eric Christiansen, Kevin Murphy, and Frank Hutter. "NAS-Bench-101: Towards reproducible neural architecture search." *ICML* (2019)


thank you