Neural Architecture Search and Beyond

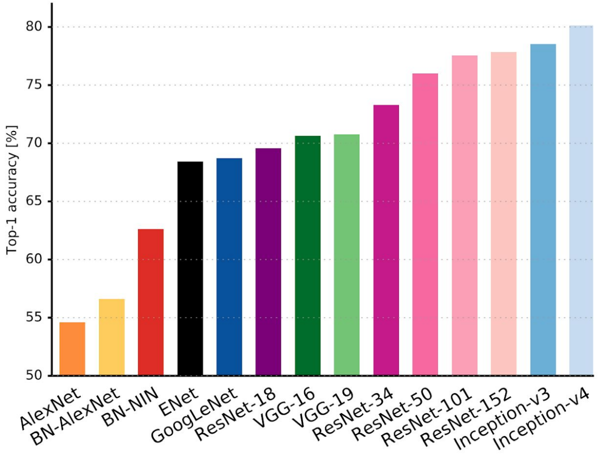
Barret Zoph

Progress in AI

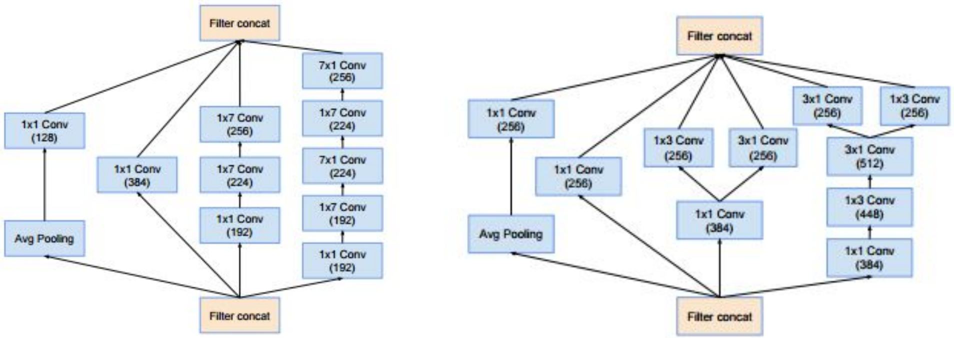
- Generation 1: Good Old Fashioned AI
  - Handcraft predictions
  - Learn nothing
- Generation 2: Shallow Learning
  - Handcraft features
  - Learn predictions
- Generation 3: Deep Learning
  - Handcraft algorithm (architectures, data processing, …)
  - Learn features and predictions end-to-end
- Generation 4: Learn2Learn (?)
  - Handcraft nothing
  - Learn algorithm, features and predictions end-to-end

Importance of architectures for Vision

- Designing neural network architectures is hard
- Lots of human efforts go into tuning them
- There is not a lot of intuition into how to design them well
- Can we try and learn good architectures automatically?



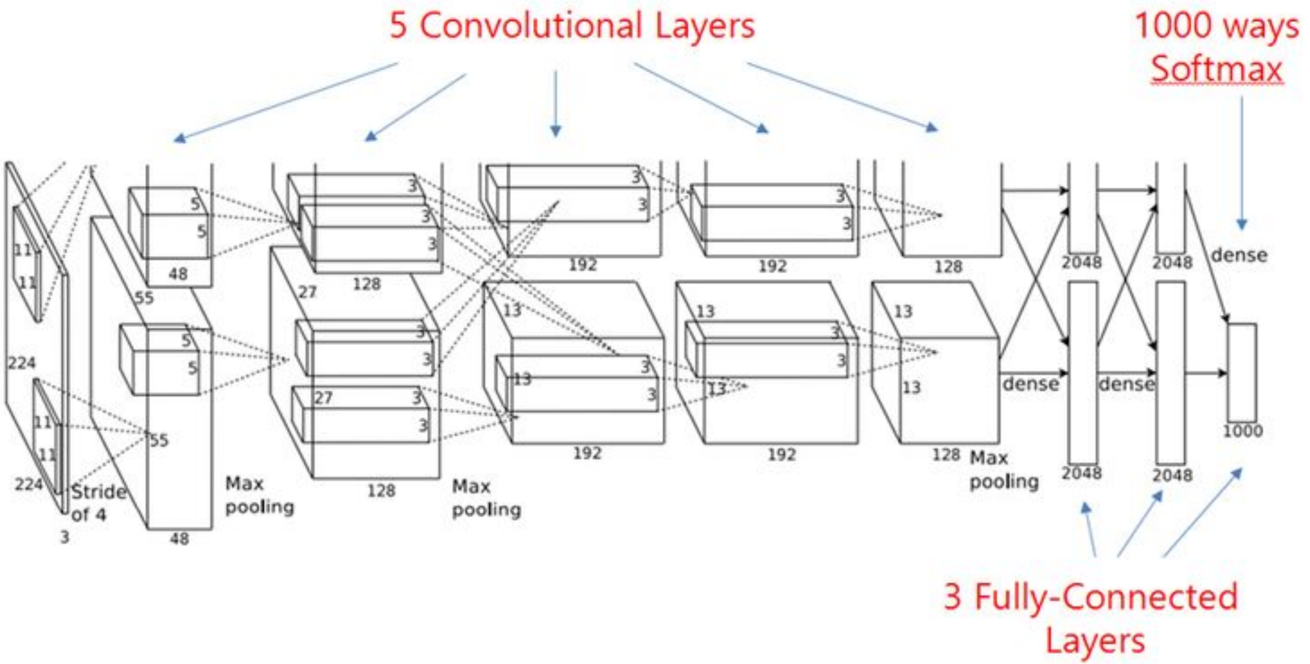
Canziani et al, 2017



Two layers from the famous Inception V4 computer vision model.
Szegedy et al, 2017

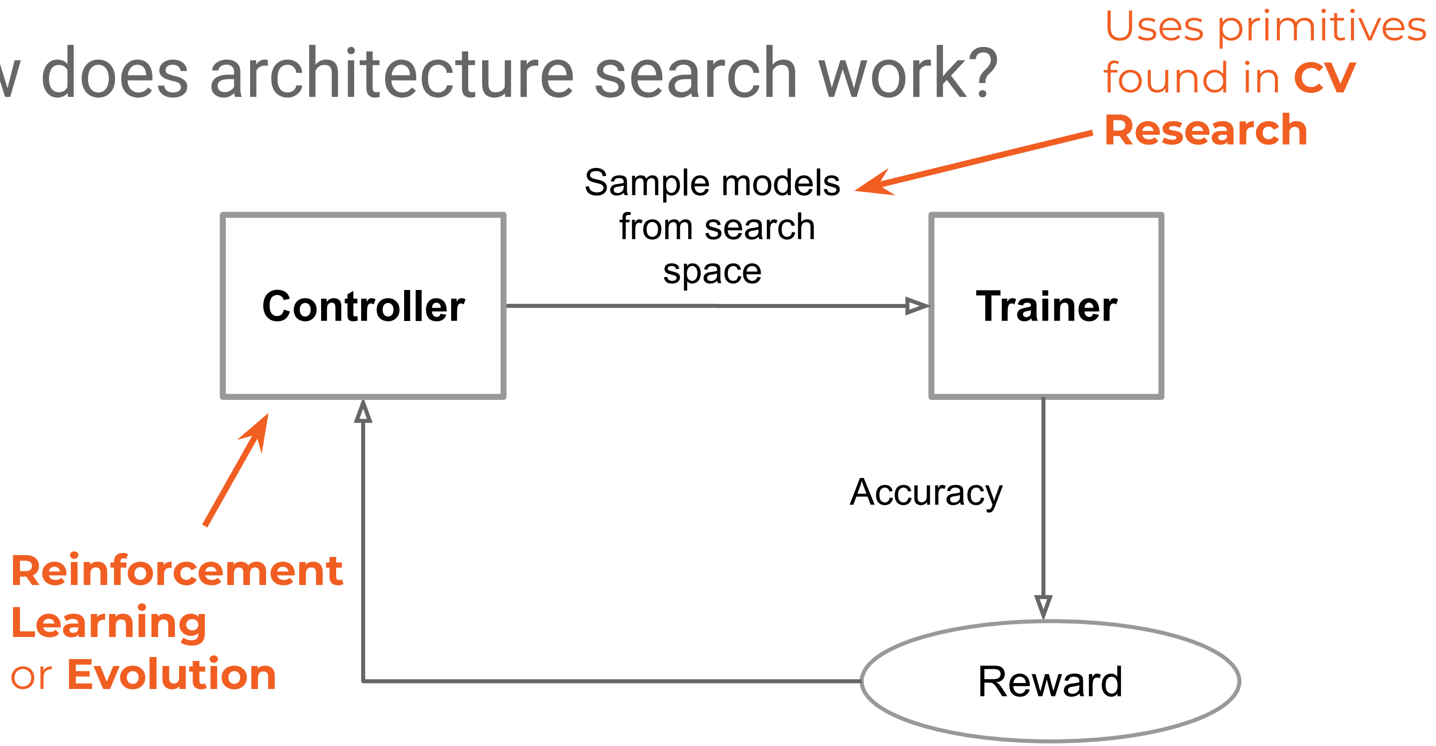
Google

Convolutional Architectures



Krizhevsky et al, 2012

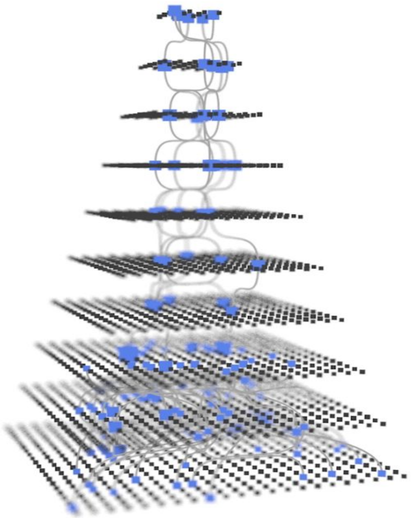
How does architecture search work?



Zoph & Le. Neural Architecture Search with Reinforcement Learning. ICLR, 2017. arxiv.org/abs/1611.01578
Real *et al*. Large Scale Evolution of Image Classifiers. ICML, 2017. arxiv.org/abs/1703.01041

How does architecture search work?

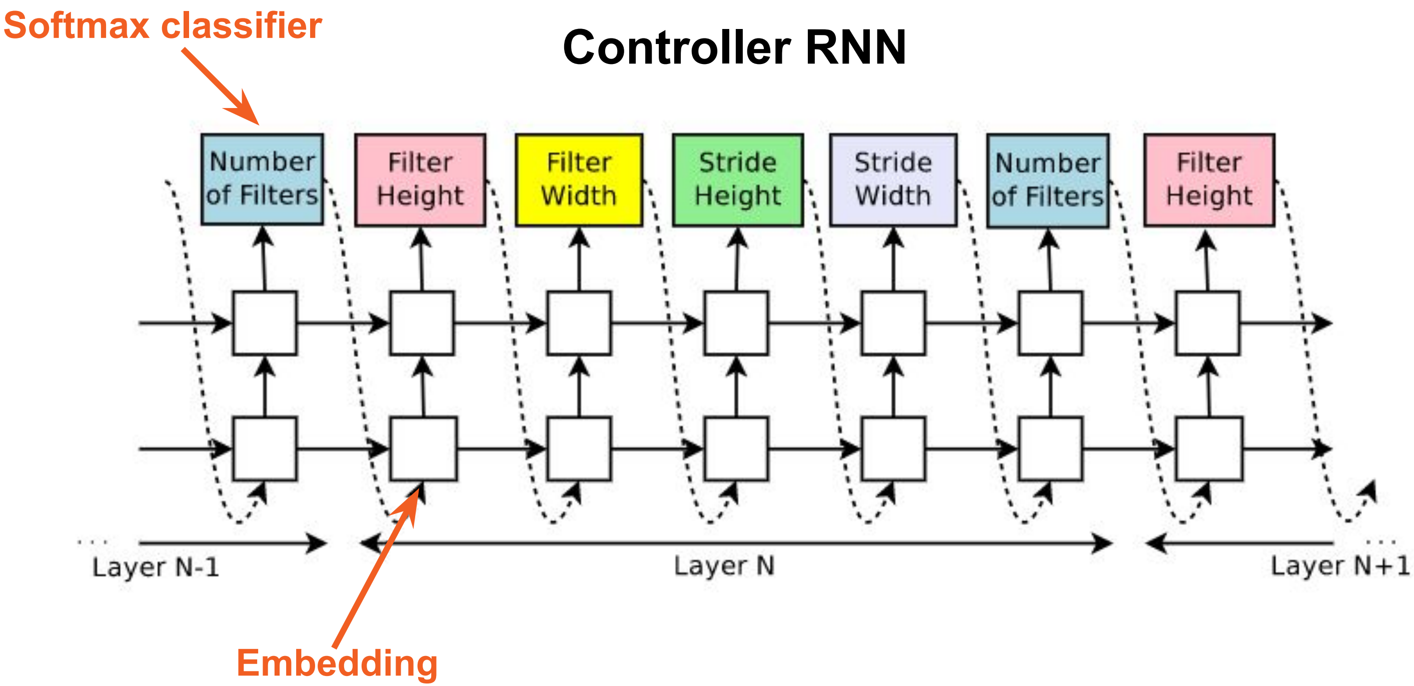
Controller: proposes ML models



Iterate to find the most accurate model

Train & evaluate models

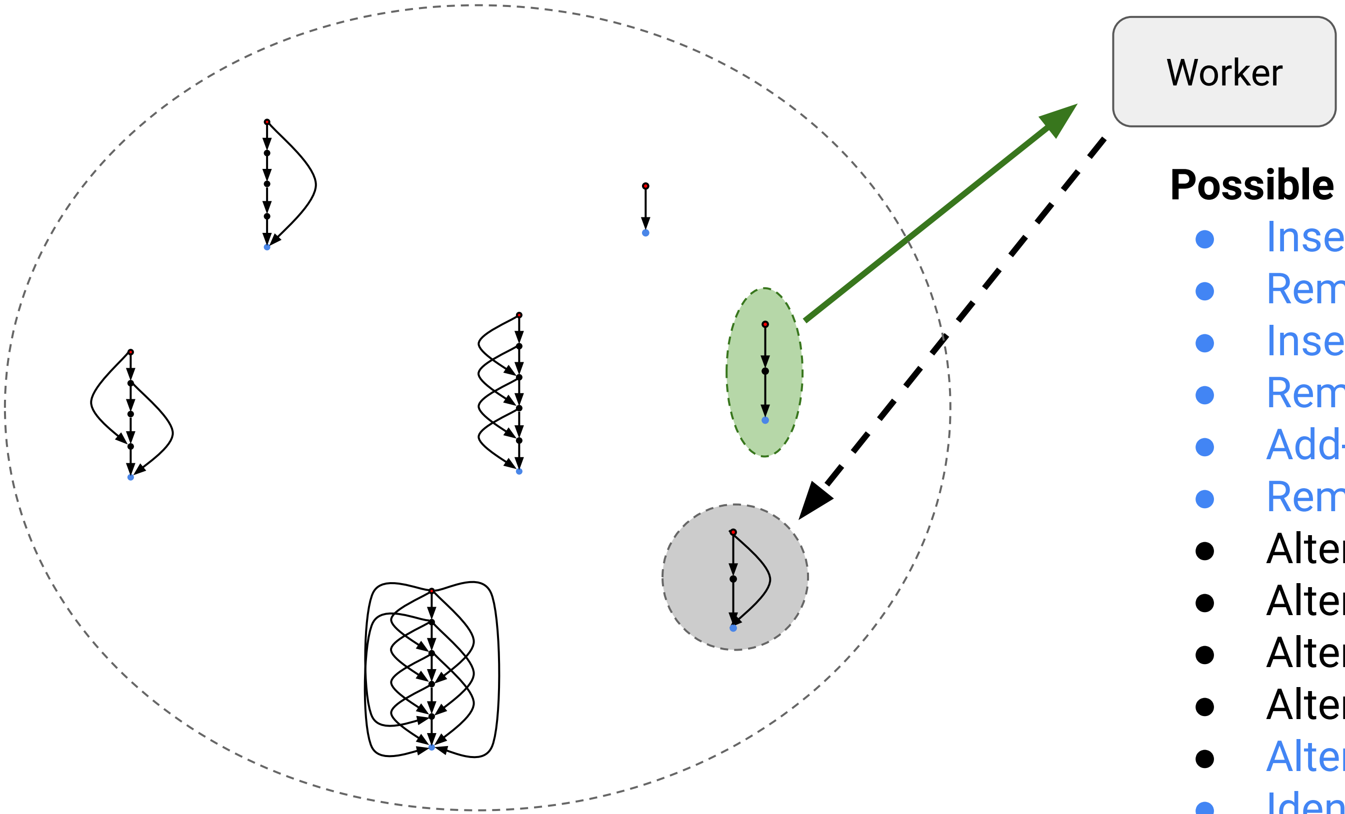
Example: Using reinforcement learning controller (NAS)



Zoph & Le. Neural Architecture Search with Reinforcement Learning. ICLR, 2017. arxiv.org/abs/1611.01578

Google

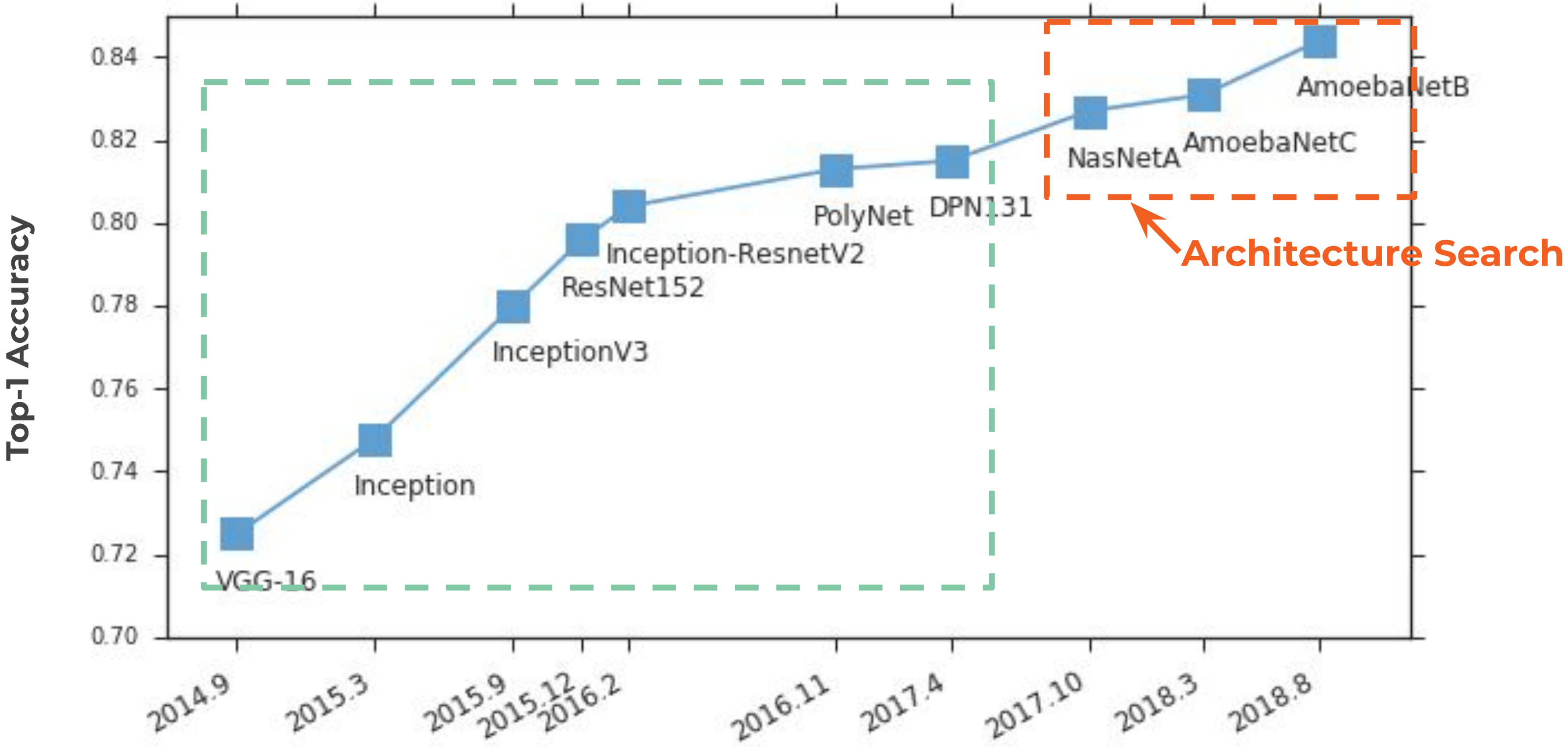
Example: Using evolutionary controller



Possible Mutations

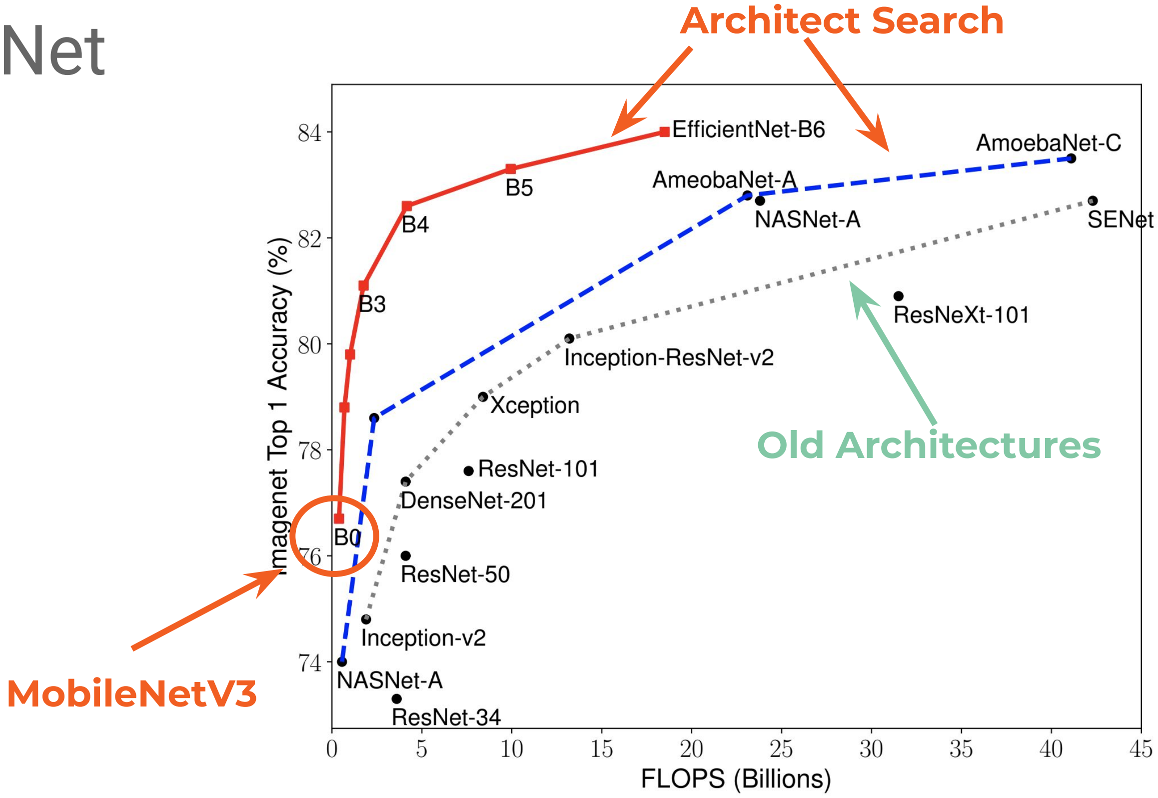
- Insert convolution
- Remove convolution
- Insert nonlinearity
- Remove nonlinearity
- Add-skip
- Remove skip
- Alter strides
- Alter number of channels
- Alter horizontal filter size
- Alter vertical filters size
- Alter Learning Rate
- Identity
- Reset weights

ImageNet Neural Architect Search Improvements



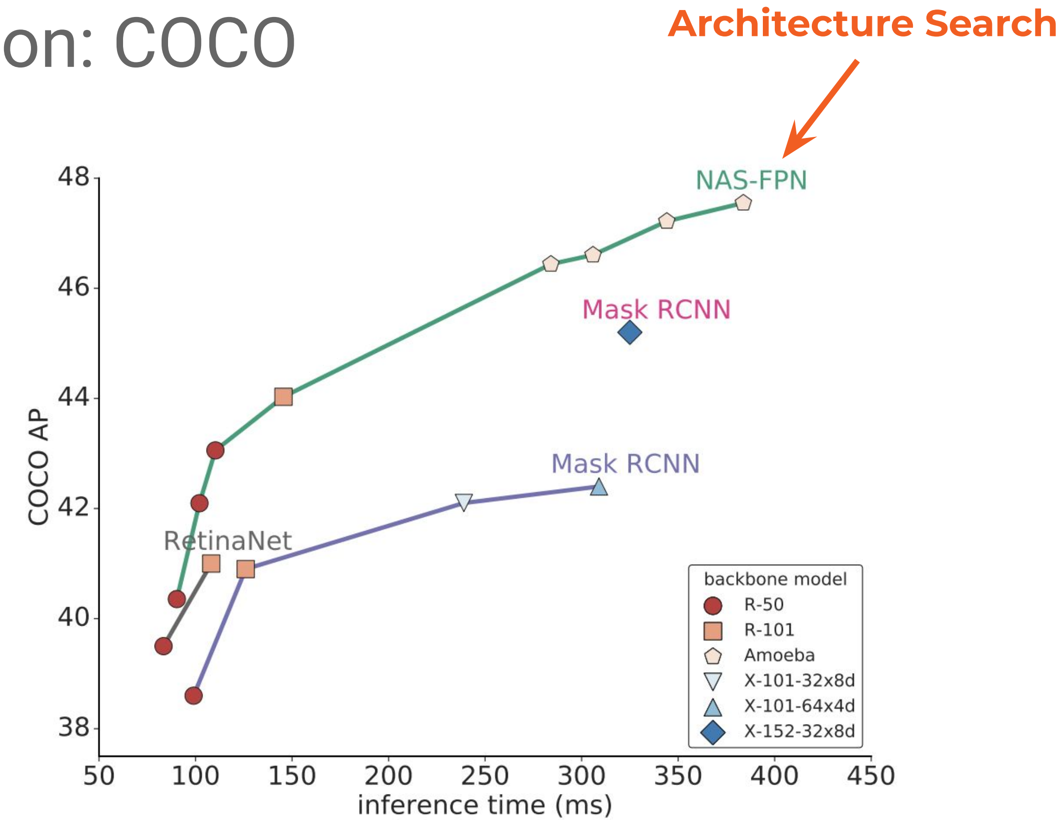
Google

ImageNet



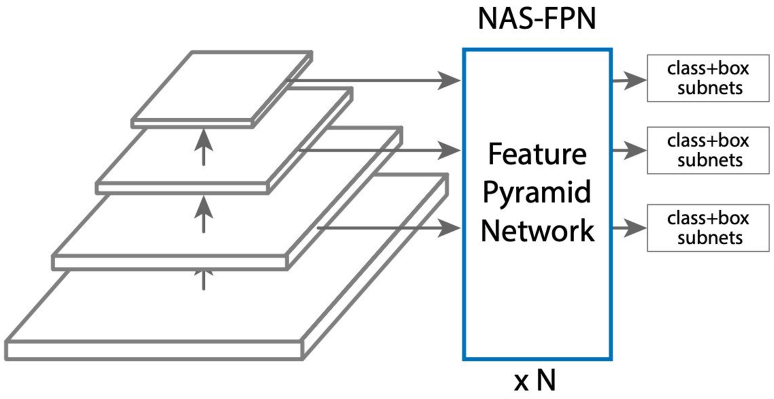
Tan & Le. EfficientNet: Rethinking Model Scaling for Deep Convolutional Neural Networks, 2019
arxiv.org/abs/1905.11946

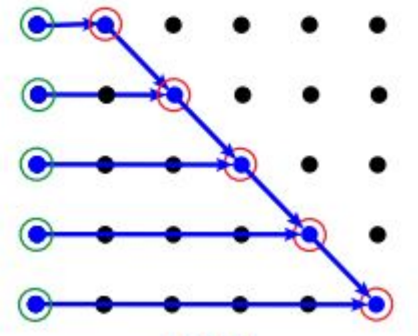
Object detection: COCO



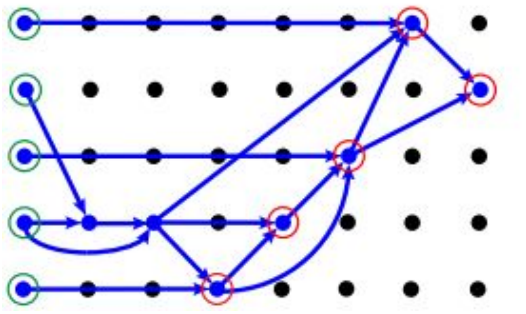
Ghiasi et al. Learning Scalable Feature Pyramid Architecture for Object Detection, 2019 arxiv.org/abs/1904.07392

Architecture Decisions for Detection Architecture Search





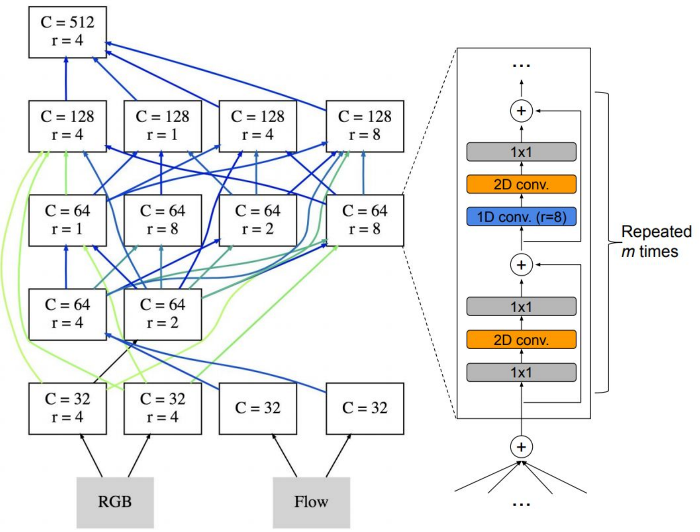
Human Designed Architecture



Machine Designed Architecture

Ghiasi et al. Learning Scalable Feature Pyramid Architecture for Object Detection, 2019 arxiv.org/abs/1904.07392

Video Classification Architecture Search



Learn the connections between blocks

Table 1: State-of-the-art action classification performances on Charades [19].

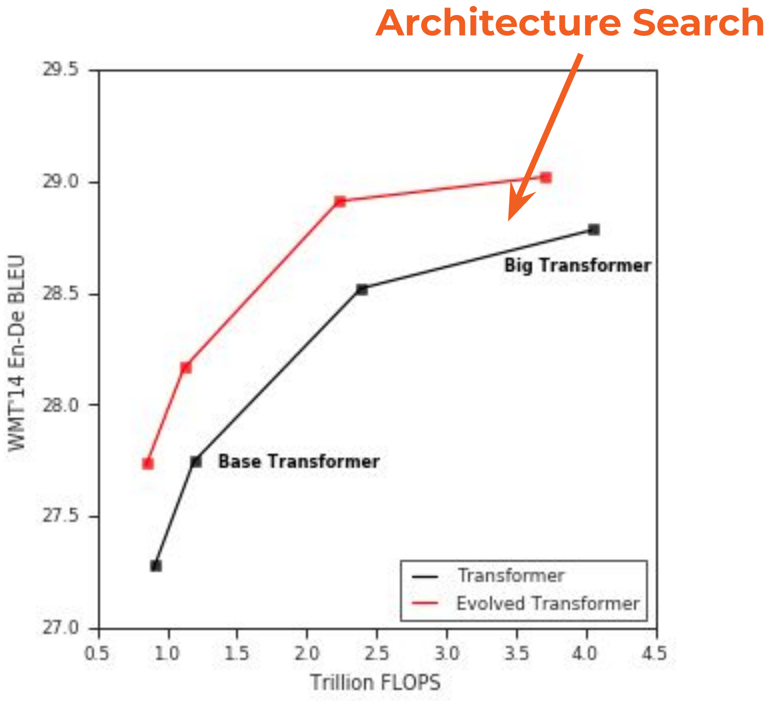
Method	modality	mAP
2-Strm. [20] (from [18])	RGB+Flow	18.6
Asyn-TF [18]	RGB+Flow	22.4
CoViAR [28]	Compressed	21.9
MultiScale TRN [33]	RGB	25.2
I3D [3]	RGB	32.9
I3D [3] (from [25])	RGB	35.5
I3D-NL [25]	RGB	37.5
STRG [26]	RGB	39.7
LFB [27]	RGB	42.5
SlowFast [6]	RGB+RGB	45.2
Two-stream (2+1)D ResNet	RGB+Flow	46.5
AssembleNet	RGB+Flow	**51.6**

Architect Search

State-of-the-art accuracy

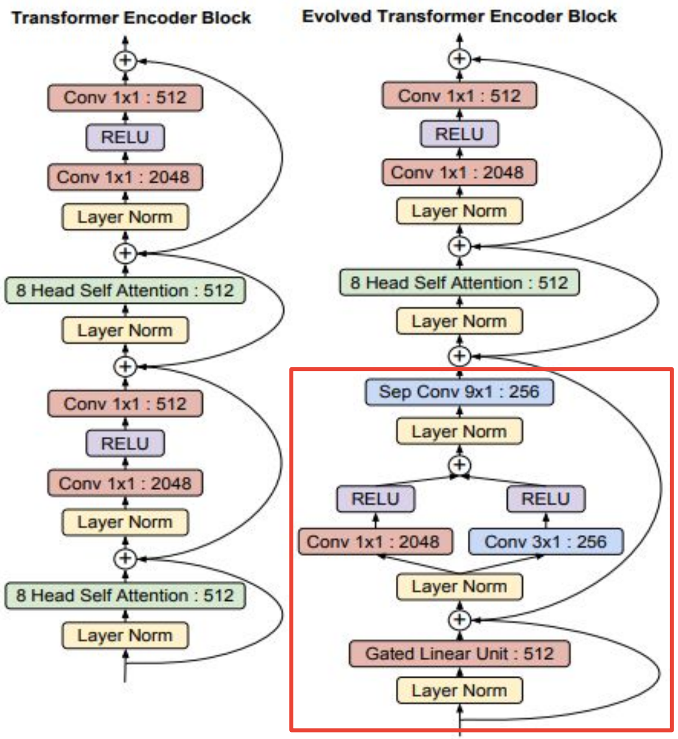
Ryoo *et al.*, 2019. AssembleNet: Searching for Multi-Stream Neural Connectivity in Video Architectures. arxiv.org/abs/1905.13209

Translation: WMT



256 input words + 256 output words
So, et al. The Evolved Transformer, 2019,
arxiv.org/abs/1901.11117

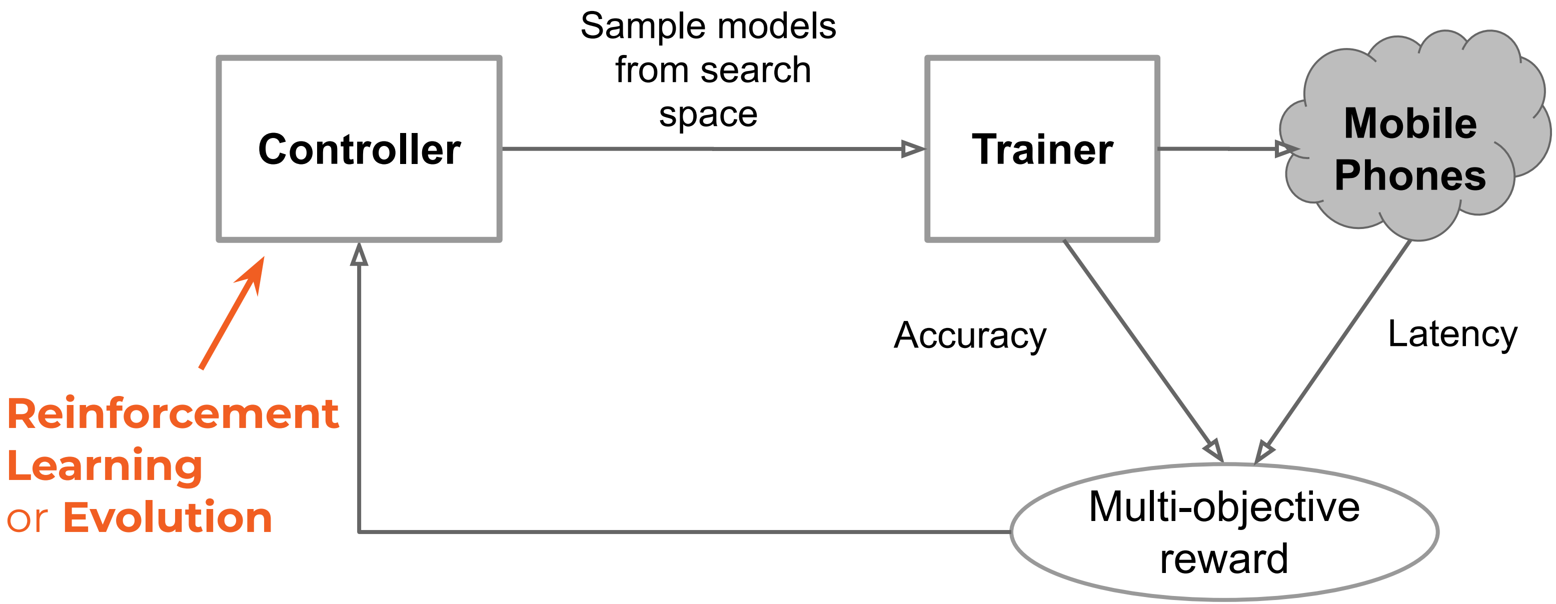
Architecture Decisions



Using more convolutions in earlier layers

Google

Platform-aware search



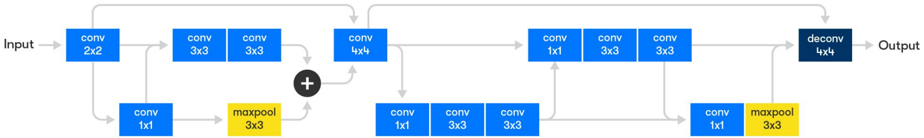
Tan *et al.*, MnasNet: Platform-Aware Neural Architecture Search for Mobile. CVPR, 2019
arxiv.org/abs/1807.11626

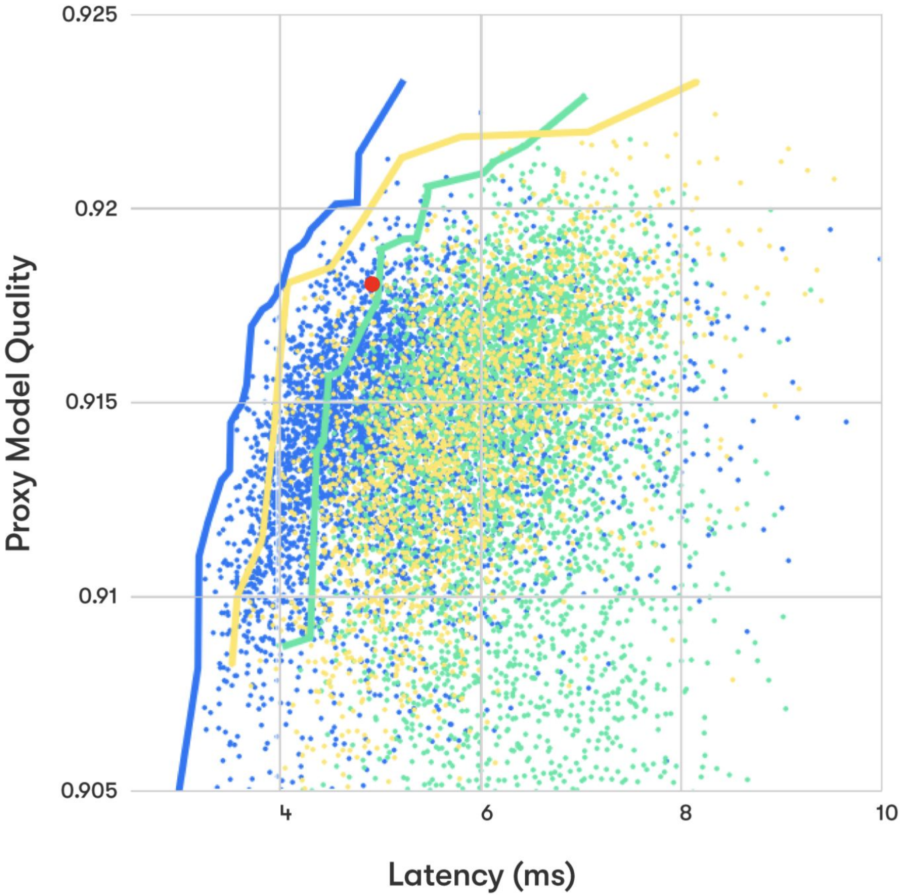
Google

Collaboration between Waymo and Google Brain:

- 20–30% lower latency / same quality.
- 8–10% lower error rate / same latency.

'Interesting' architectures:

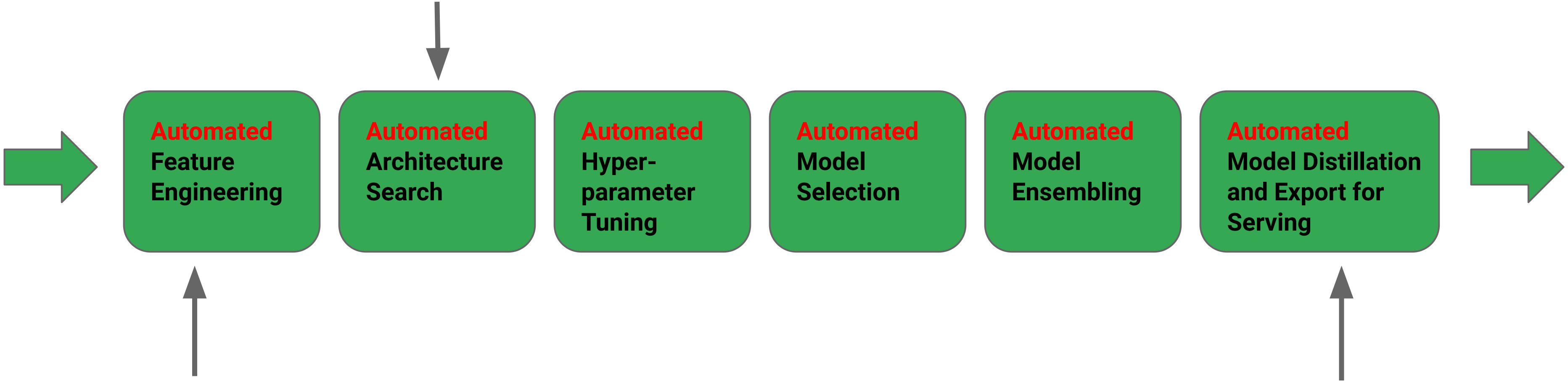




https://medium.com/waymo/automl-automating-the-design-of-machine-learning-models-for-autonomous-driving-141a5583ec2a

Tabular Data

trees, neural nets,
#layers, activation
functions, connectivity

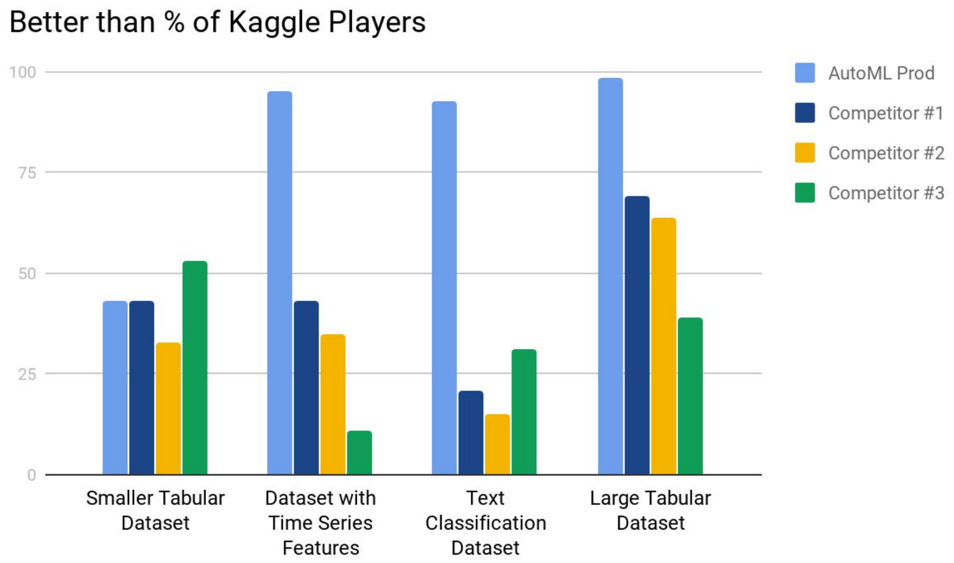


Normalization,
Transformation
(log, cosine)

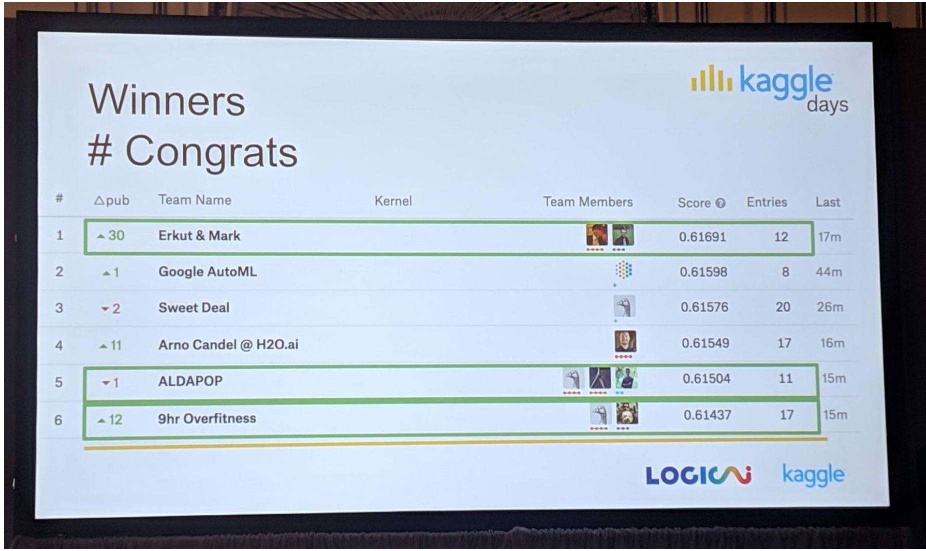
Can distill to decision trees
for interpretability

https://ai.googleblog.com/2019/05/an-end-to-end-automl-solution-for.html

Tabular Data



Internal Benchmark on Kaggle Competitions

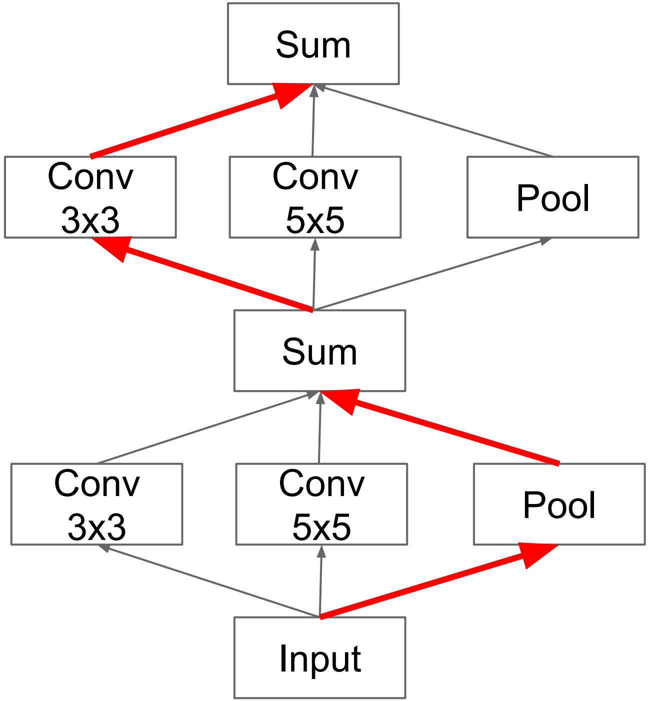


AutoML placed 2nd in a [live one-day competition]() against 76 teams

Problems of NAS

- Enormous compute consumption
  - Requires ~10k training trials to coverage on a carefully designed search space
  - Not applicable if single trial's computation is heavy
- Works inefficiently on arbitrary and giant search space
  - Feature selection (search space 2^100 if there are 100 features)
  - Per feature transform (search space c^100 if there are 100 features and each has c types of transform)
  - Embedding and hidden layer size

Efficient NAS: Addressing the efficiency



Key idea:

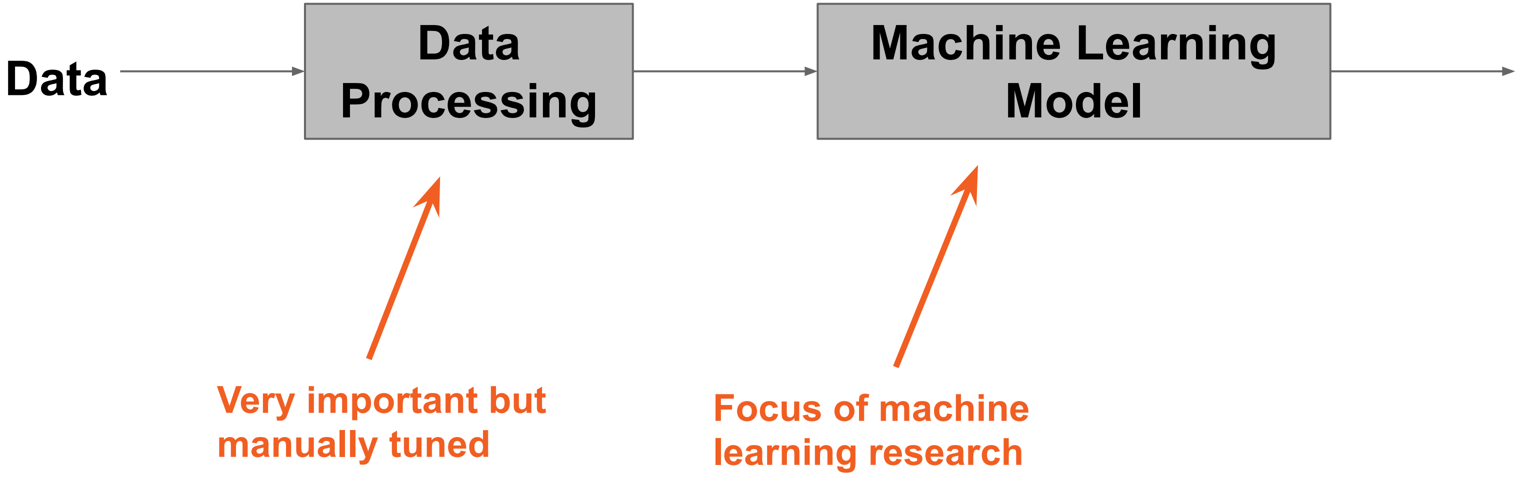
1. One path inside a big model is a child model
2. **Controller selects a path** inside a big model and train for a few steps
3. Controller selects another path inside a big model and train for a few steps, **reusing the weights** produced by the previous step
4. Etc.

Results: Can **save 100->1000x compute**

Related works: DARTS, SMASH, One-shot architecture search,

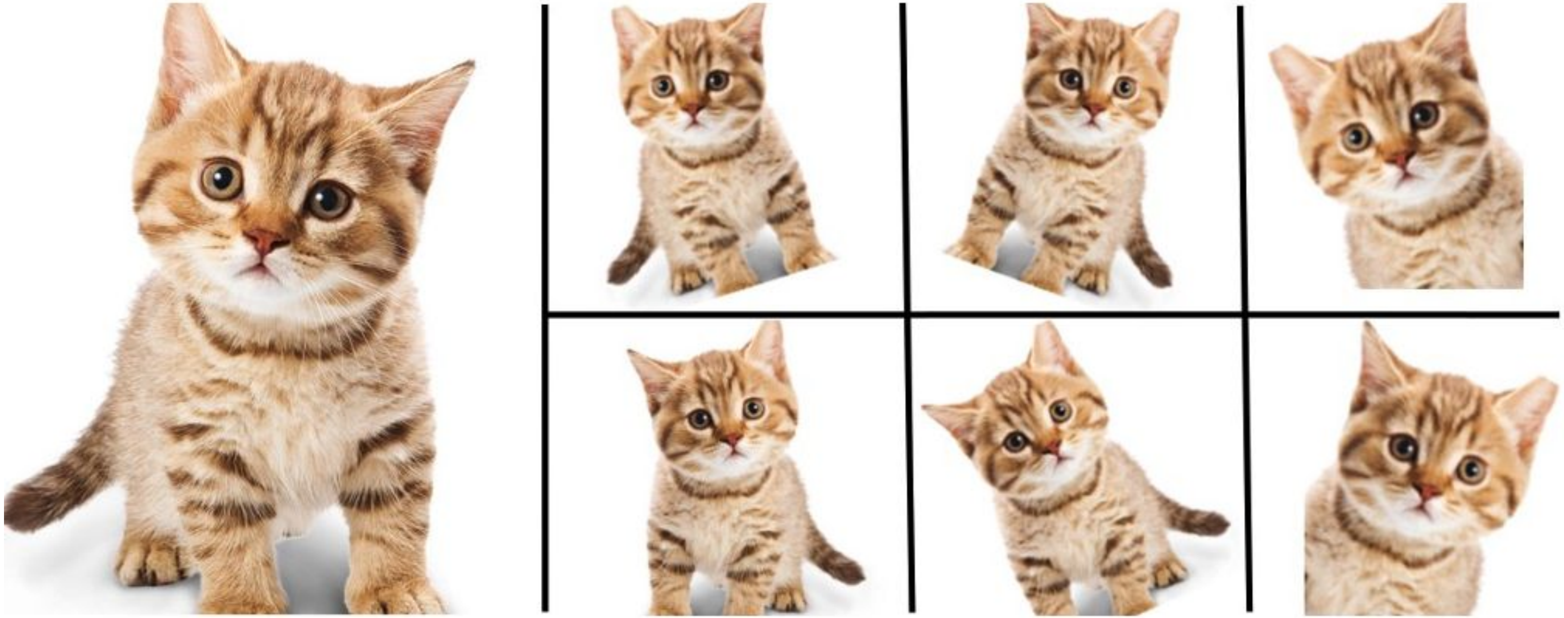
Pham et al, 2018. Efficient Neural Architecture Search via Parameter Sharing, arxiv.org/abs/1802.03268

Learning Data Augmentation Procedures



Google

Data Augmentation

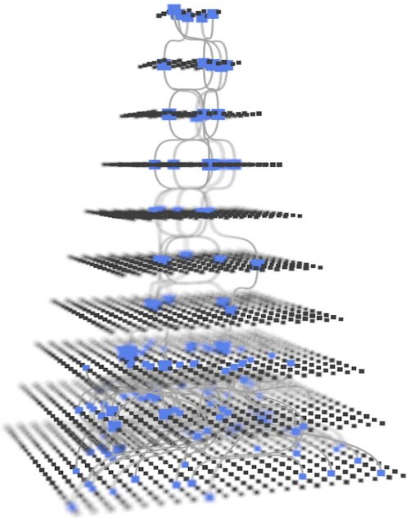


Enlarge your Dataset

Google

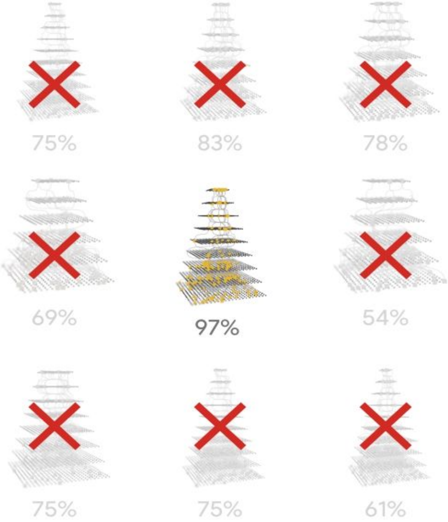
AutoAugment Search Algorithm

Controller: proposes **augmentation policy**



Iterate to find the most accurate policy

Train & evaluate models with the **augmentation policy**



Cubuk et al, 2018. AutoAugment: Learning Augmentation Policies from Data, arxiv.org/abs/1805.09501

AutoAugment: Example Learned Policy

AutoAugment Learns: (Operation, Probability, Magnitude)

Original

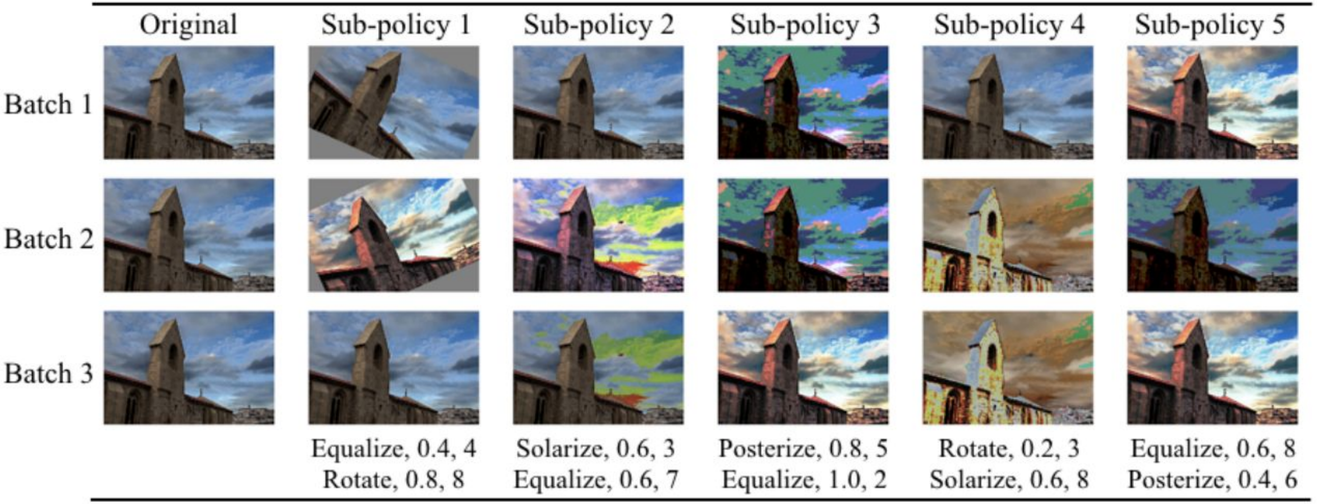
Equalize, 0.4, 4	Solarize, 0.6, 3	Posterize, 0.8, 5	Rotate, 0.2, 3	Equalize, 0.6, 8
Rotate, 0.8, 8	Equalize, 0.6, 7	Equalize, 1.0, 2	Solarize, 0.6, 8	Posterize, 0.4, 6

Probability of applying

Magnitude

AutoAugment: Example Learned Policy

For each Sub-Policy (5 Sub-Policies = Policy):
AutoAugment Learns: (Operation, Probability, Magnitude)



AutoAugment CIFAR Results

Full CIFAR-10

Model	No data aug	Standard data-aug	AutoAugment
Wide-ResNet-28-10	3.87	3.08	2.68
Shake-Shake (26 2x32d)	3.55	3.02	2.47
Shake-Shake (26 2x96d)	2.86	2.56	1.99
Shake-Shake (26 2x112d)	2.82	2.57	1.89
AmoebaNet-B (6,128)	2.98	2.13	1.75
PyramidNet+ShakeDrop	2.67	2.31	**1.48**

State-of-the-art accuracy

CIFAR-100

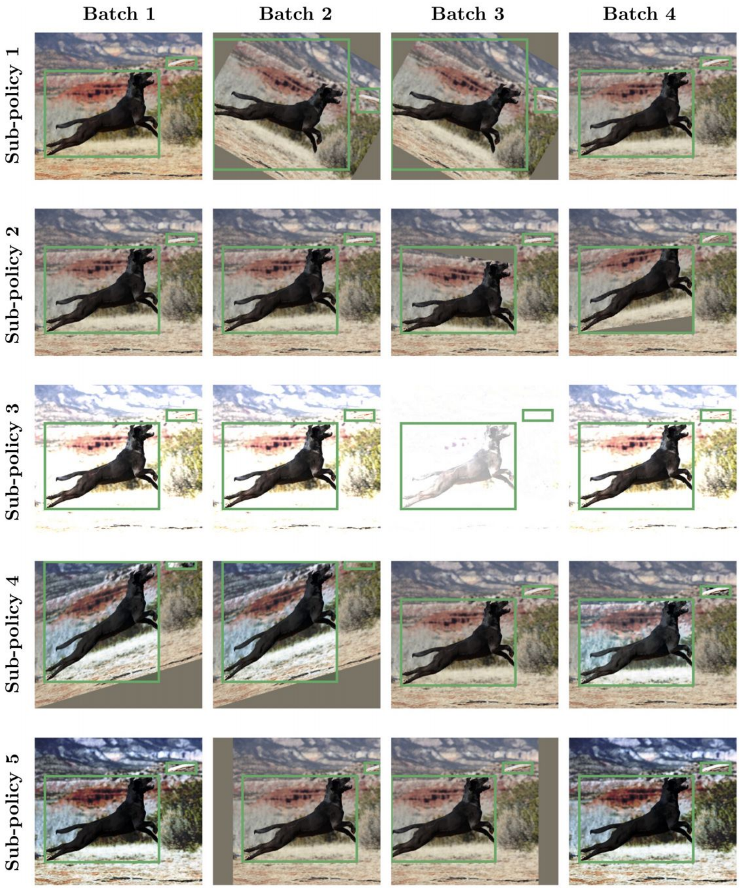
Model	No data aug	Standard data-aug	AutoAugment
Wide-ResNet-28-10	18.80	18.41	17.09
Shake-Shake (26 2x96d)	17.05	16.00	14.28
PyramidNet+ShakeDrop	13.99	12.19	**10.67**

AutoAugment ImageNet Results (Top5 error rate)

Model	No data augmentation	Standard data augmentation	AutoAugment
ResNet-50	7.80	6.92	6.18
ResNet-200		5.85	4.99
AmoebaNet-B		3.97	3.78
AmoebaNet-C		3.90	**3.52**

Code is opensourced:
https://github.com/tensorflow/models/tree/master/research/autoaugment

Expanded AutoAugment for Object Detection



Zoph et al. 2019, Learning Data Augmentation Strategies for Object Detection, arxiv.org/abs/1906.11172

Google

Learn Augmentation on COCO Results

ResNet-50 Model

Method	mAP
baseline	36.7
baseline + DropBlock [13]	38.4
Augmentation policy with color operations	37.5
+ geometric operations	38.6
+ bbox-only operations	**39.0**

Backbone	Baseline	Our result	Difference
ResNet-50	36.7	39.0	+2.3
ResNet-101	38.8	40.4	+1.6
ResNet-200	39.9	42.1	+2.2

Learn Augmentation on COCO Results

Architecture	Change	# Scales	mAP	mAP_S	mAP_M	mAP_L
MegDet [32]		multiple	50.5	-	-	-
AmoebaNet + NAS-FPN	baseline [14]	1	47.0	30.6	50.9	61.3
	+ learned augmentation	1	48.6	32.0	53.4	62.7
	+ ↑ anchors, ↑ image size	1	**50.7**	**34.2**	**55.5**	**64.5**

State-of-the-art accuracy at the time for a single model

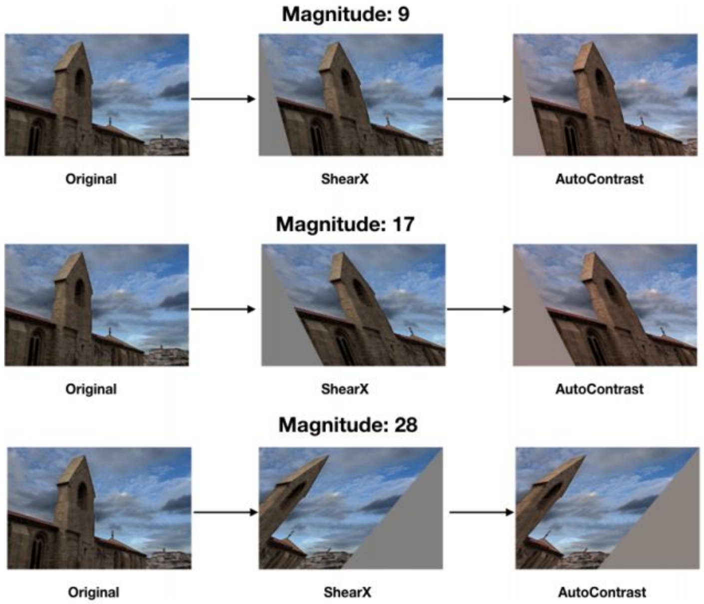
Code is opensourced:
https://github.com/tensorflow/tpu/tree/master/models/official/detection

Google

RandAugment: Practical data augmentation with no separate search

Faster AutoAugment w/ vastly reduced search space!

Only two tunable parameters now: Magnitude and Policy Length



Cubuk et al. 2019, RandAugment: Practical data augmentation with no separate search, arxiv.org/abs/1909.13719

RandAugment: Practical data augmentation with no separate search

Match or surpass AA with significantly less cost!

	baseline	PBA	Fast AA	AA	RA
CIFAR-10					
Wide-ResNet-28-2	94.9	-	-	**95.9**	95.8
Wide-ResNet-28-10	96.1	**97.4**	97.3	**97.4**	97.3
Shake-Shake	97.1	**98.0**	**98.0**	**98.0**	**98.0**
PyramidNet	97.3	**98.5**	98.3	**98.5**	**98.5**
CIFAR-100					
Wide-ResNet-28-2	75.4	-	-	**78.5**	78.3
Wide-ResNet-28-10	81.2	**83.3**	82.7	82.9	**83.3**
SVHN (core set)					
Wide-ResNet-28-2	96.7	-	-	98.0	**98.3**
Wide-ResNet-28-10	96.9	-	-	98.1	**98.3**
SVHN					
Wide-ResNet-28-2	98.2	-	-	**98.7**	**98.7**
Wide-ResNet-28-10	98.5	98.9	98.8	98.9	**99.0**

RandAugment: Practical data augmentation with no separate search

	baseline	Fast AA	AA	RA
ResNet-50	76.3 / 93.1	**77.6** / 93.7	**77.6 / 93.8**	**77.6 / 93.8**
EfficientNet-B5	83.2 / 96.7	-	83.3 / 96.7	**83.9 / 96.8**
EfficientNet-B7	84.0 / 96.9	-	84.4 / 97.1	**85.0 / 97.2**

Can easily scale regularization strength when model size changes!

State-of-the-art accuracy

Code and Models Opensourced:
https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet