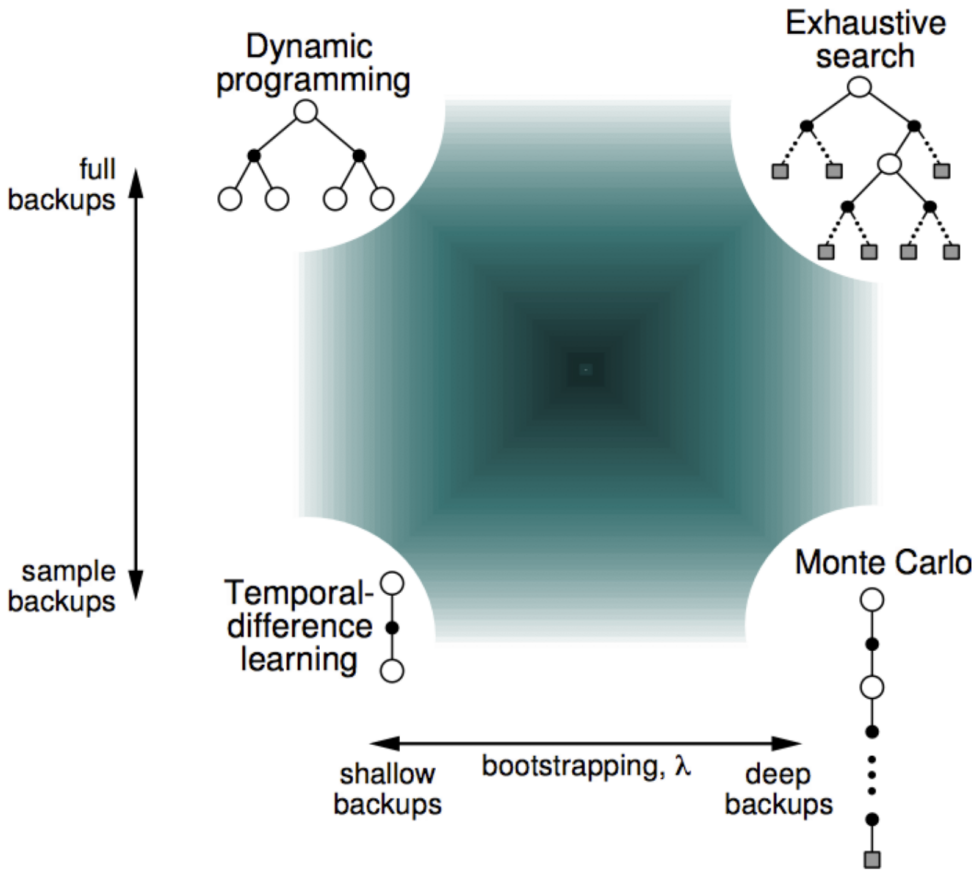
Recitation:
RL (one last time)

RL method review
AutoML – Neural Architecture Search

Summary

Which is model-based?



From David Silver UCL Course on RL: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

Look at each method for gridworld

- **Q values in each state**
  - State = A, B, C, ..
  - Action = l, r, u, d (left, right, up, down)

e.g.
Q(A,r) = .8

A **Q(A,r)=.8** **Q(A,d)=.7**	**B** **Q(B,l)= .75** **Q(B,r)= .9**	**C** **Q(C,l)= .8** **r=.95; d=0.7**	**Food 1**
D Q(D,u)= .8 Q(D,d)= .7	XXXXXXXX XXXXXXXX XXXXXXXX	E Q(E,u)= .8 Q(E,d)= .65	Shock -1
J Q(J,u)= .7 Q(J,r)= .65	G Q(G,l)= .7 Q(G,r)= .6	H Q(H,l)= .65 r=.6; u=0.75	I Q(I,l) = .65 Q(I,u) = -.5

Dynamic Programing, γ=1

- **Start in B, with π^*, assume**
  - p(C|B,r)=0.9 P(A|B,r)=0.1 p(C|B,l)=0.1 P(A|B,l)=0.9
- **What is the new estimate of *Q(B,r)?***

A Q(A,r)=.8 Q(A,d)=.7	B Q(B,l)= .75 Q(B,r)= .9	C Q(C,l)= .8 r=.95; d=0.7	Food 1
D Q(D,u)= .8 Q(D,d)= .7	XXXXXXXX XXXXXXXX XXXXXXXX	E Q(E,u)= .8 Q(E,d)= .65	Shock -1
J Q(J,u)= .7 Q(J,r)= .65	G Q(G,l)= .7 Q(G,r)= .6	H Q(H,l)= .65 r=.6; u=0.75	I Q(I,l) = .65 Q(I,u) = -.5

Bellman's Equation

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} q_*(s')\right], \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$$

$$\begin{aligned} Q(B,r) &= p(C|B,r)[0+1*Q(C,r)] + p(A|B,r)[0+1*Q(A,r)] \\ &= 0.9 \quad [\quad 0.95 \quad] + \quad 0.1 \quad [\quad 0.8 \quad] \\ &= 0.935 \end{aligned}$$

TD(0) - Q-learning, γ=1, α=0.6

- Start in *s=B*, pick best action *a=r*
- Observe new state *s=C*
- What is the new estimate of *Q(B,r)?*

A Q(A,r)=.8 Q(A,d)=.7	B Q(B,l)= .75 Q(B,r)= .9	C Q(C,l)= .8 r=.95; d=0.7	Food 1
D Q(D,u)= .8 Q(D,d)= .7	XXXXXXXXX XXXXXXX XXXXXXXX	E Q(E,u)= .8 Q(E,d)= .65	Shock -1
J Q(J,u)= .7 Q(J,r)= .65	G Q(G,l)= .7 Q(G,r)= .6	H Q(H,l)= .65 r=.6; u=0.75	I Q(I,l) = .65 Q(I,u) = -.5

TD(0)

- $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_t + \gamma Q\left(s_{t+1}, \pi(s_{t+1})\right) - Q(s_t, a_t)\right)$
- $Q(B, r) \leftarrow Q(B, r) + \alpha\left(0 + 1Q(C, r) - Q(B, r)\right)$
- $Q(B, r) \leftarrow 0.75 + 0.6 * (0 + 0.95 - 0.75)$
- $Q(B, r) = 0.87$

Monte Carlo Q-learning, γ=1, α=0.6

- Do 3 rollouts from *B*, taking action *r* every time

1) B→C →Food

2) B→A → B → C → Food

3) B → C → E → Shock

What is the new estimate of *Q(B,r)?*

A Q(A,r)=.8 Q(A,d)=.7	B Q(B,l)= .75 Q(B,r)= .9	C Q(C,l)= .8 r=.95; d=0.7	Food 1 Q(F,-)=1
D Q(D,u)= .8 Q(D,d)= .7	XXXXXXXXX XXXXXXX XXXXXXXX	E Q(E,u)= .8 Q(E,d)= .65	Shock Q(S,-)=-1
J Q(J,u)= .7 Q(J,r)= .65	G Q(G,l)= .7 Q(G,r)= .6	H Q(H,l)= .65 r=.6; u=0.75	I Q(I,l) = .65 Q(I,u) = -.5

Monte Carlo Q-learning

- **3 rollouts give** z = (1+1-1)/3 = 1/3
- $Q^{\pi}(B, r) = 1/3$ This is not right; don't go all the way to z!
- $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\big(r_t + \gamma V^{\pi}(s_{t+1}) - Q(s_t, a_t)\big)$
- $Q^{\pi}(B, r) = 0.9 + 0.6(0 + 1/3 - 0.9)$ This is still not quite right. We should do the rollout *from* s_{t+1}

MC Q-learning, γ=1 , α=0.6

- Do 3 rollouts from *B*, taking action *r* every time

1) B→C →Food

2) B→A → B → C → Food

3) B → C → E → Shock

How will you update your model? (What is the model?)

A Q(A,r)=.8 Q(A,d)=.7	B Q(B,l)= .75 Q(B,r)= .9	C Q(C,l)= .8 r=.95; d=0.7	Food 1 Q(F,-)=1
D Q(D,u)= .8 Q(D,d)= .7	XXXXXXXXX XXXXXXX XXXXXXXX	E Q(E,u)= .8 Q(E,d)= .65	Shock Q(S,-)=-1
J Q(J,u)= .7 Q(J,r)= .65	G Q(G,l)= .7 Q(G,r)= .6	H Q(H,l)= .65 r=.6; u=0.75	I Q(I,l) = .65 Q(I,u) = -.5

DQN (TD(0)), γ=1 , α=0.6

- **Now move to a real space.**
- Represent every state *i* by its location $\boldsymbol{x}_i$
  - A = (1,1), B=(1,2), D= (2,1),…
- Assume a linear $Q(s_i,a_j) = \boldsymbol{w}_j^T \boldsymbol{x}_i$ so every action *j* has a $\boldsymbol{w}_j$
- Initialize all $\mathbf{w}_j$ = (1,1);
- Start in state A, take action r, end up on B.
- What is the updated value of Q(A,r)?

A Q(A,r)=.8 Q(A,d)=.7	B Q(B,l)= .75 Q(B,r)= .9	C Q(C,l)= r=.95;
D Q(D,u)= .8	XXXXXXXXX XXXXXXX	E Q(E,u)=

Deep Q-Learning

$$\mathit{Argmin}_{\theta} \left[Q(s,a;\theta) - \left(r(s,a) + \gamma \max_{a} Q(s',a;\theta) \right) \right]^2$$

Which Q() are we updating?
How do we update it?

MC-DQN, γ=1 , α=0.6

- **Still in the real space.**
  - Again, represent every state *i* by its location x_i
- **Again, assume linear model**
  - $q(s_i, a_j) = \boldsymbol{w}_j^T \boldsymbol{x}_i$
  - Initialize all weights to (1,1);
- Start in state A, follow policy π 3 times,
  - pick r every time
  - End up twice with Food, once with Shock
- What is the updated value of $q(A, r)$?

A Q(A,r)=.8 Q(A,d)=.7	B Q(B,l)= .75 Q(B,r)= .9	C Q(C, r=.9
D Q(D,u)= .8	XXXXXXXXX XXXXXXX	E Q(E,

AlphaZero-style, γ=1 , α=0.6

- **Assume linear models**
  - $\pi(s_i)$ = *softmax*($\boldsymbol{w}_a^T \boldsymbol{x}_i$), value V($s_i$) = $\boldsymbol{w}^T \boldsymbol{x}_i$ $\boldsymbol{w}_a = \boldsymbol{w}$ = (1,1);
- Start in state A, follow policy π 3 times,
  - Pick r every time
  - End up twice with Food, once with Shock
- What is the updated value of V(A)?
  - What is V(A)?
  - What is the formula for updating it?

AlphaZero loss function

NNet: $(\mathbf{p}, v) = f_\theta(s)$

- Minimizes the error between the predicted outcome (value function) *v(s)* and the actual game outcome *z*
- Maximizes the similarity of the policy vector **p**(s) to the MCTS probabilities $\pi(s)$.
- L2 regularize the weights θ

$$l = (z - v)^2 - \pi^{\mathrm{T}} \log \mathbf{p} + c\|\theta\|^2$$

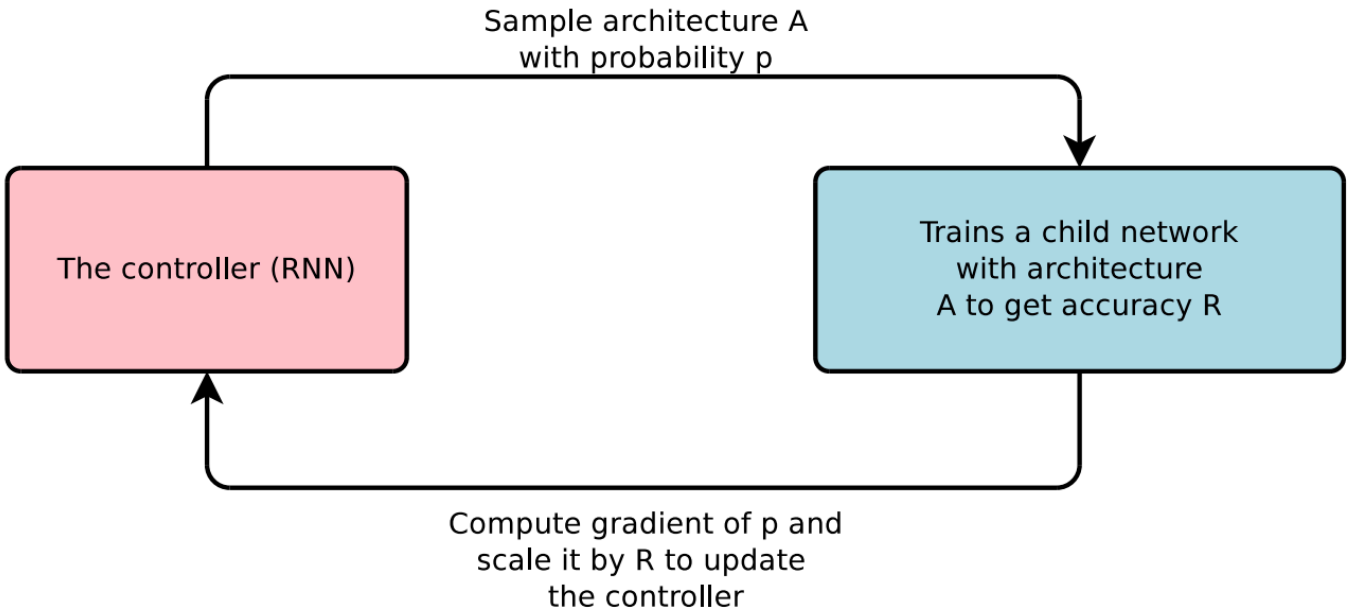
Q-Learning

$$Q(s,a) := r(s,a) + \gamma \max_a Q(s',a)$$

How might we pick the policy?

Pure greedy: $\text{argmax}_a\ Q(s,a)$
ε-greedy
Using an older network for Q
Using a policy network (maybe a fast one)
Preferring actions that have been taken less

AutoML – Neural Architecture Search



What is the reward, state, action, and policy?

Zoph and Le (2017)

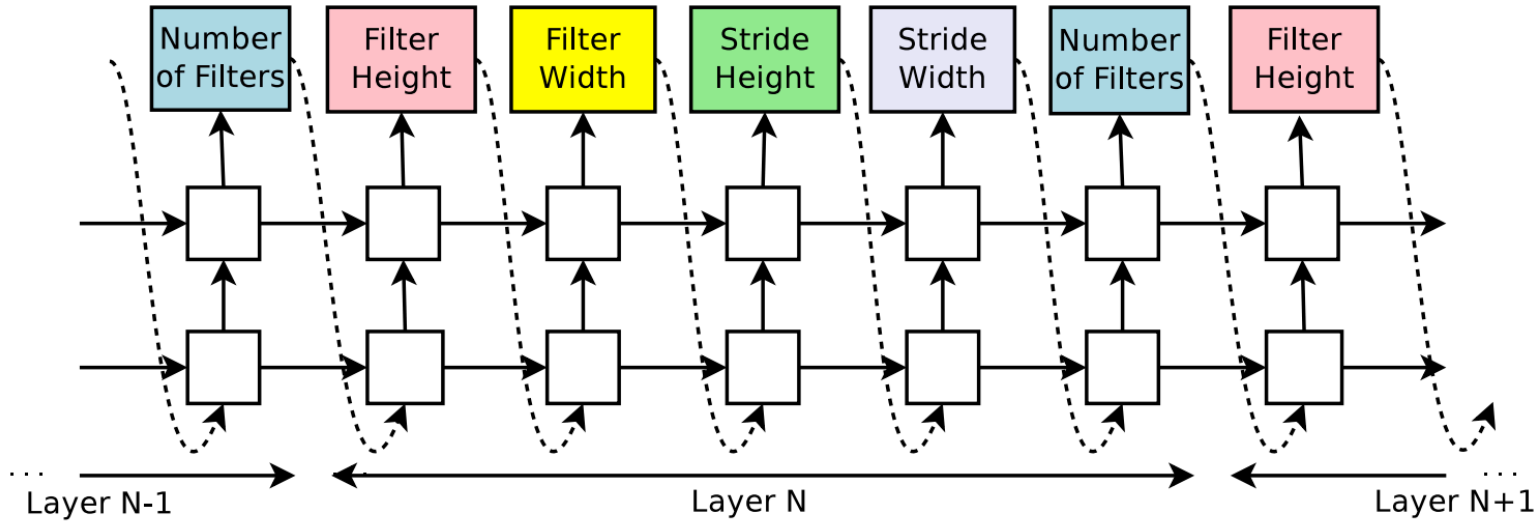
AutoML (NAS)

- **Reward:** Trained NNet accuracy on validation data set
- **Action:** A variable-length string specifying a neural net
  - **Architecture:** recurrent layer, feedforward layer, pooling, skip connections, number of nodes/layer…
  - **Convolutional layer hyper-parameters:** filter width and height, stride width and height, the number of filters…
- **State** (implicit): the partial network architecture
- **Policy Network:** the controller
- **Training:** policy gradient RL ("REINFORCE")

How to do Policy Gradient?

- $V_\pi(s) = \Sigma_a \pi(a|s;\theta)R(a,s)$
- We want the gradient $dV/d\theta$

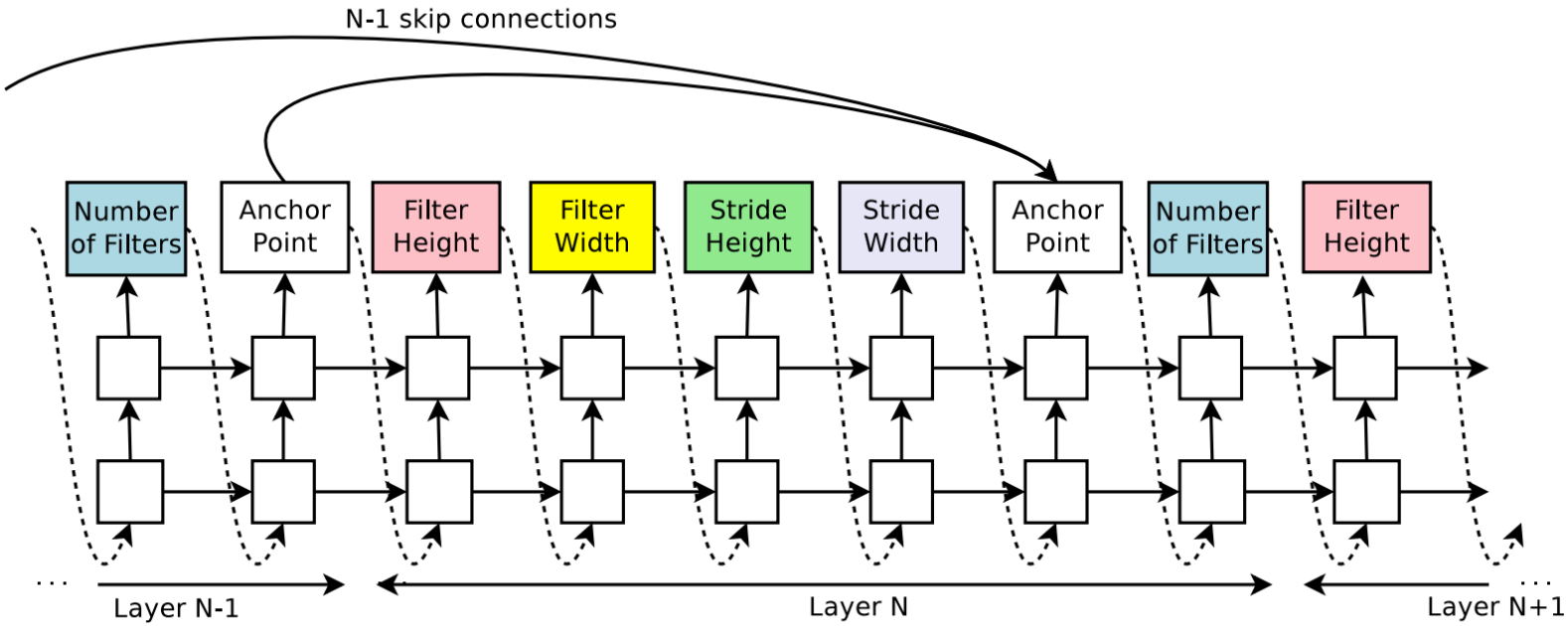
AutoML - RNN



Layer by layer, predict with softmax.
Input prediction and recurse.

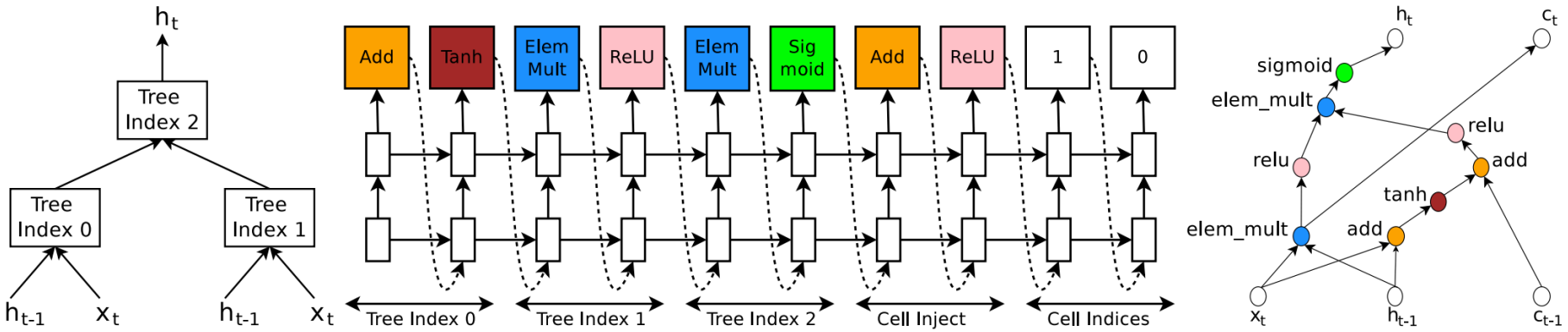
Zoph and Le (2017)

AutoML - RNN



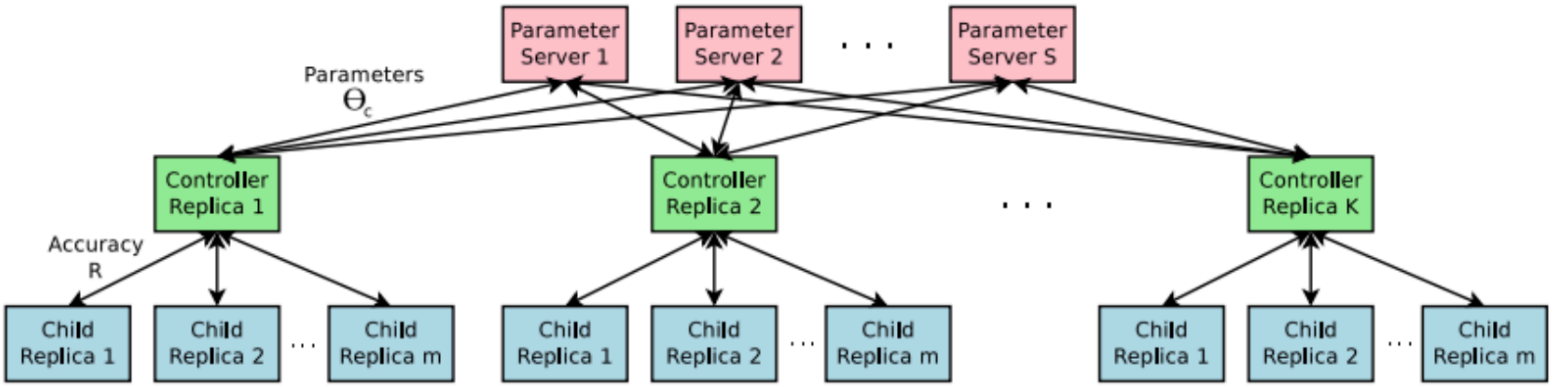
Zoph and Le (2017)

AutoML - RNN



Zoph and Le (2017)

AutoML – Distributed training



Each controller computes a gradient and sends it up the the parameter servers

Zoph and Le (2017)

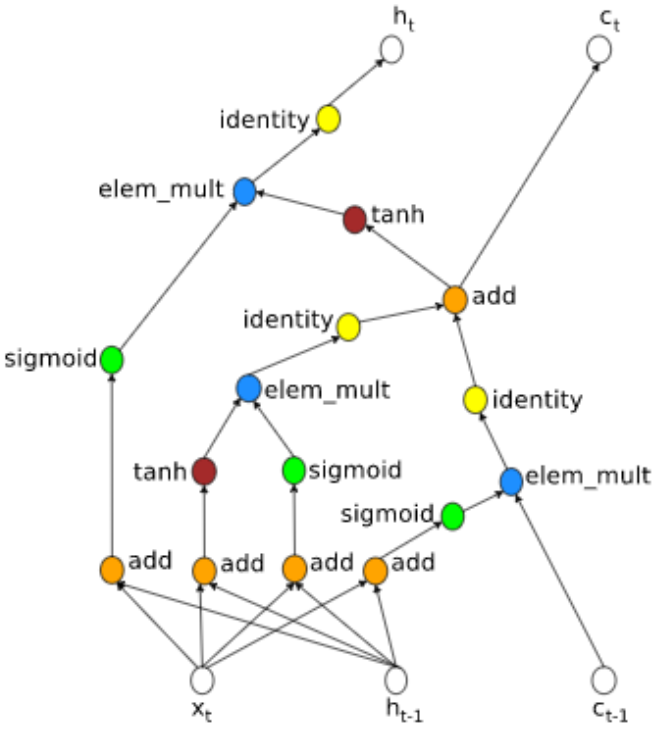
CIFAR-10 Leaderboard

Model	Depth	Parameters	Error rate (%)
Network in Network (Lin et al., 2013)	-	-	8.81
All-CNN (Springenberg et al., 2014)	-	-	7.25
Deeply Supervised Net (Lee et al., 2015)	-	-	7.97
Highway Network (Srivastava et al., 2015)	-	-	7.72
Scalable Bayesian Optimization (Snoek et al., 2015)	-	-	6.37
FractalNet (Larsson et al., 2016)	21	38.6M	5.22
with Dropout/Drop-path	21	38.6M	4.60
ResNet (He et al., 2016a)	110	1.7M	6.61
ResNet (reported by Huang et al. (2016c))	110	1.7M	6.41
ResNet with Stochastic Depth (Huang et al., 2016c)	110	1.7M	5.23
	1202	10.2M	4.91
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

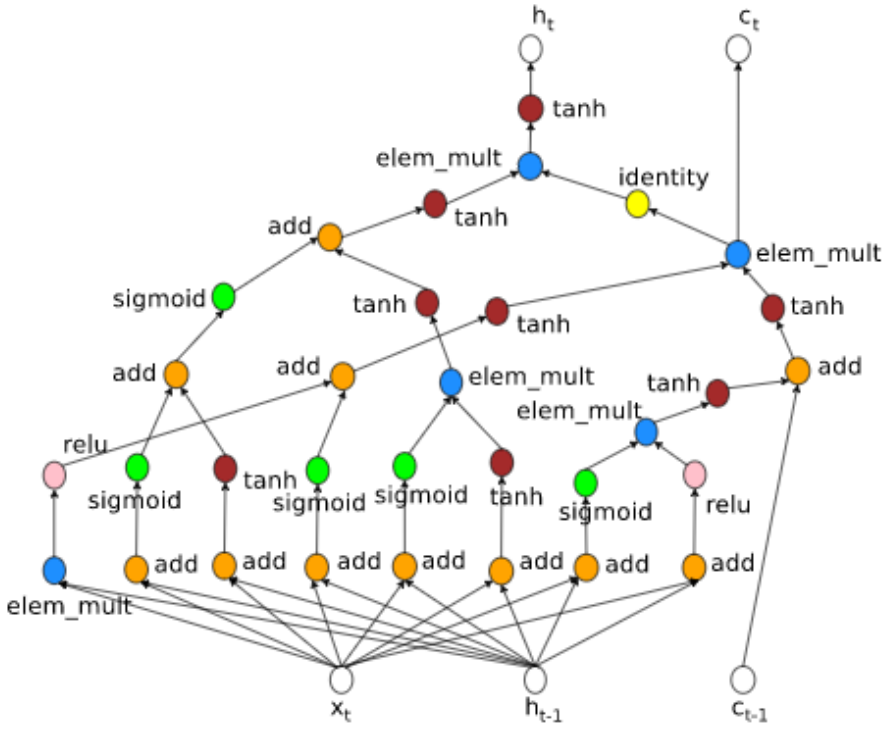
Table 1: Performance of Neural Architecture Search and other state-of-the-art models on CIFAR-10.

Zoph and Le (2017)

AutoML learns network structure



Human built



Learned by RL

https://research.googleblog.com/2017/05/using-machine-learning-to-explore.html