Efficient Neural Architecture Search via Parameter Sharing

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, Jeff Dean

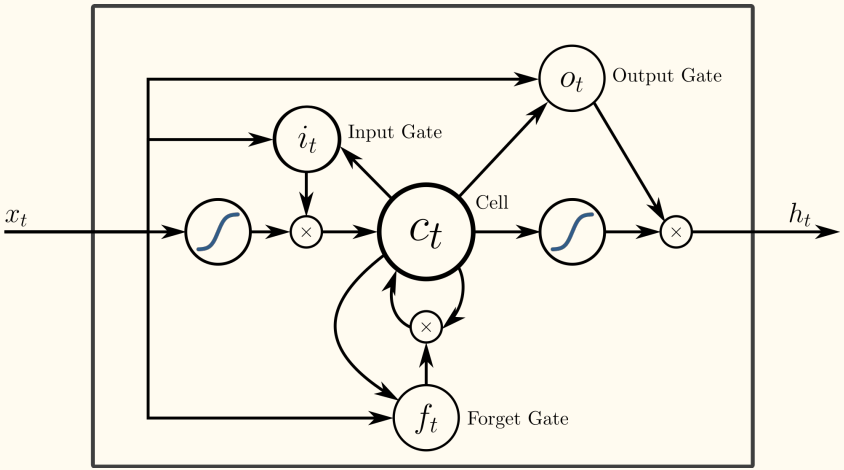
Presented by: Seung Wook Kim and Matthew MacKay

Why should we care about model architecture?

- Impressive gains have followed from improvements to model architecture

Why should we care about model architecture?

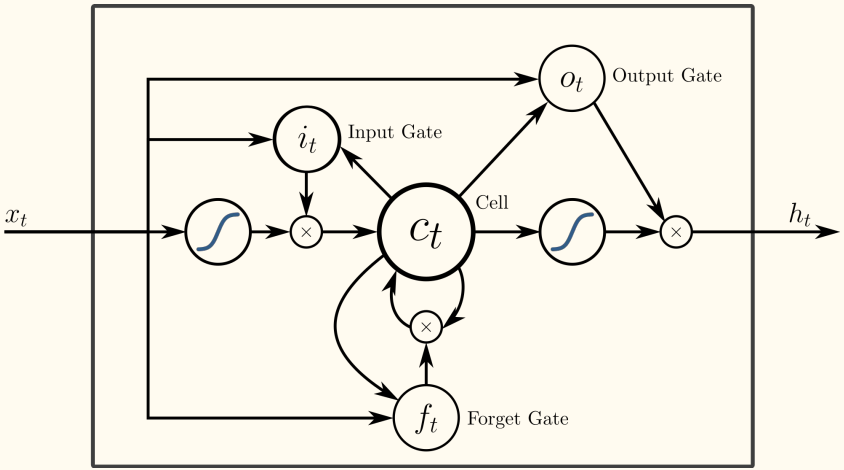
- Impressive gains have followed from improvements to model architecture
- LSTMs: gated connections replace traditional RNN



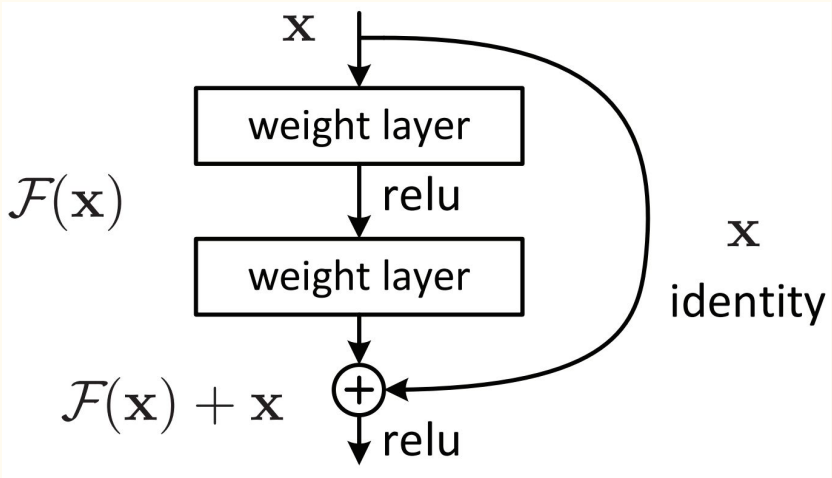
Hochreiter & Schmidhuber, 1997

Why should we care about model architecture?

- Impressive gains have followed from improvements to model architecture
- LSTMs: gated connections replace traditional RNN
- Residual networks: ~25% relative improvement on ImageNet



Hochreiter & Schmidhuber, 1997



He et al., 2015

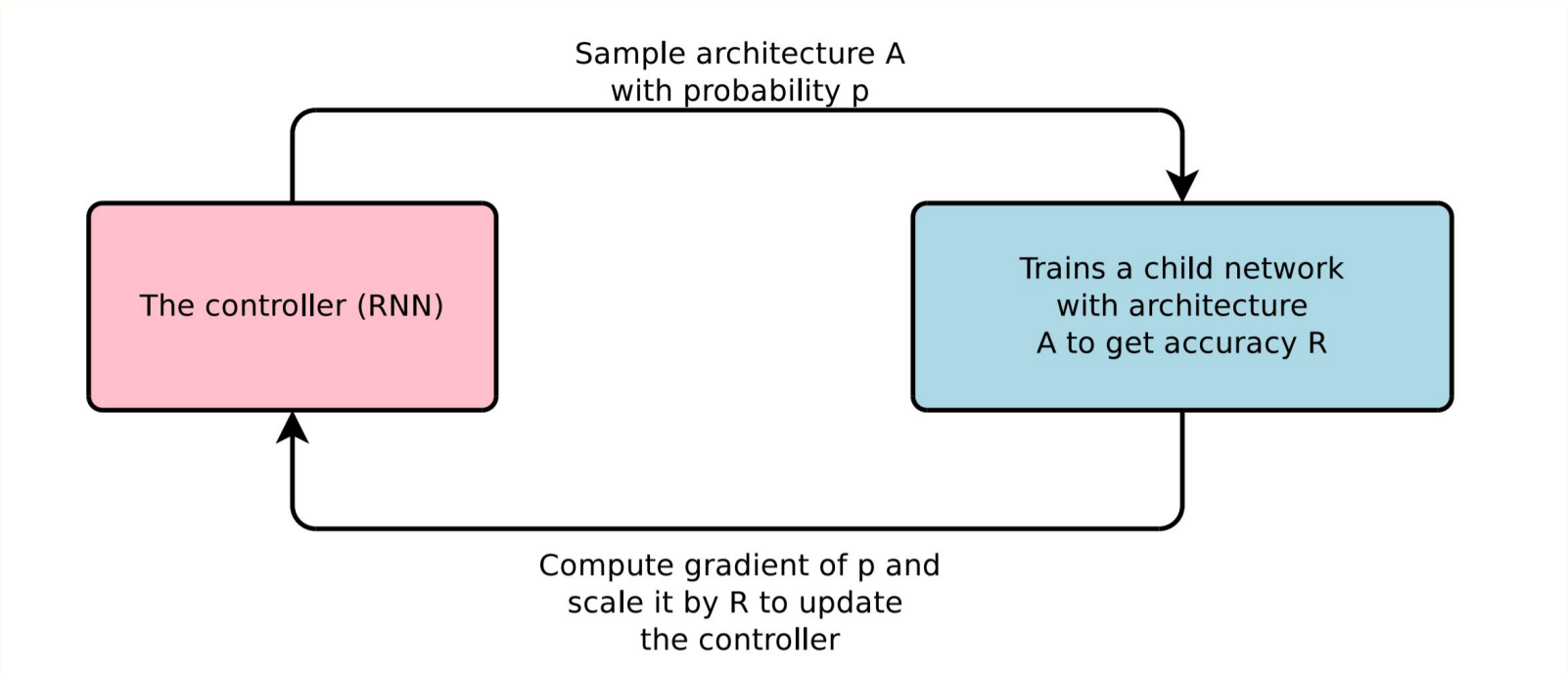
Architecture Search

- Desirable to efficiently search through architecture space

Architecture Search

- Desirable to efficiently search through architecture space
- Common method of searching: grad student descent

Neural Architecture Search (NAS)



Zoph & Le, 2015

Flaw of NAS

- Wasteful to optimize model parameters from scratch each time
- Applications of NAS “use 450 GPUs for 3-4 days”

Efficient NAS (ENAS)

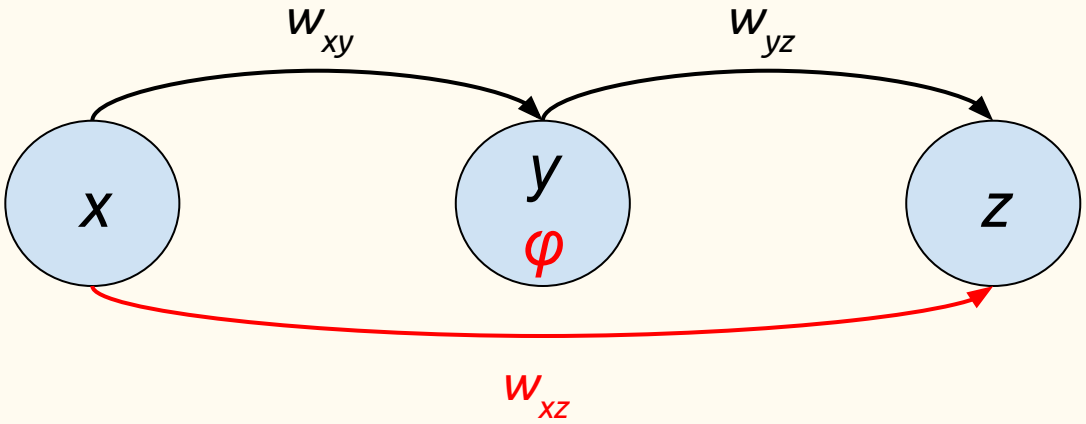
- Main Ideas:
  - Share parameters among models in the search space
  - Alternate between optimizing model parameters on the training set and controller parameters on the validation set

Efficient NAS (ENAS)

- Main Ideas:
  - Share parameters among models in the search space
  - Alternate between optimizing model parameters on the training set and controller parameters on the validation set
- Set-up: Start with a fully connected DAG specifying a computational graph
- Controller decides active connections in DAG (and a few other things)
- Parameters of transformations between nodes are shared

Simple Example of ENAS

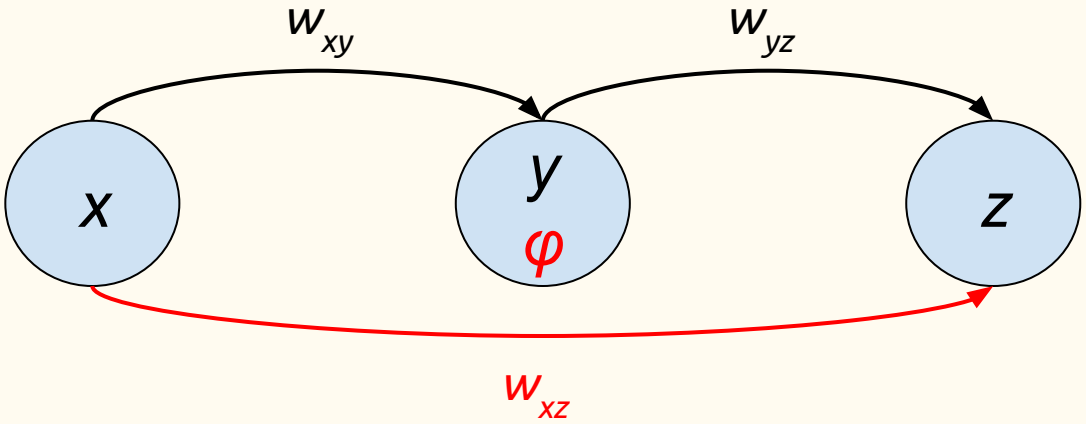
- Consider 1-D neural network regression



$$y = \varphi(w_{xy} x)$$
$$z = w_{yz} y + w_{xz} x$$

Simple Example of ENAS

- Consider 1-D neural network regression



$$y = \varphi(w_{xy} x)$$
$$z = w_{yz} y + w_{xz} x$$

- Goal: Find whether φ = tanh or ReLU and whether or not to connect x and z
  - Make decision based on *validation* performance

Simple Example of ENAS

- Define a distribution over model architectures m with “controller” parameters Θ
- Our example: $\Theta = \{\Theta_1, \Theta_2, \Theta_3, \Theta_4\}$
  - $\Pr(\varphi = tanh \ \& \ x, z \text{ connected}) = \Theta_1$
  - $\Pr(\varphi = tanh \ \& \ x, z \text{ disconnected}) = \Theta_2$
  - $\Pr(\varphi = ReLU \ \& \ x, z \text{ disconnected}) = \Theta_3$
  - $\Pr(\varphi = ReLU \ \& \ x, z \text{ disconnected}) = \Theta_4$

Simple Example of ENAS

- Define a distribution over model architectures m with “controller” parameters Θ
- Our example: $\Theta = \{\Theta_1, \Theta_2, \Theta_3, \Theta_4\}$
  - $\Pr(\varphi = tanh \ \& \ x, z \text{ connected}) = \Theta_1$
  - $\Pr(\varphi = tanh \ \& \ x, z \text{ disconnected}) = \Theta_2$
  - $\Pr(\varphi = ReLU \ \& \ x, z \text{ disconnected}) = \Theta_3$
  - $\Pr(\varphi = ReLU \ \& \ x, z \text{ disconnected}) = \Theta_4$
- Model parameters: $w = \{w_{xy}, w_{yz}, w_{xz}\}$

Simple Example of ENAS

- Training procedure:
  - Optimize w to minimize $E_{m \sim p(m|\theta)}[L(m; w)]$ on the training set
  - Optimize Θ to maximize $E_{m \sim p(m|\theta)}[R(m; w)]$ on the validation set
  - Repeat!

Simple Example of ENAS

- Optimize w to minimize $E_{m \sim p(m|\theta)}[L(m; w)]$ on the training set

$$\nabla_w E_{m \sim p(m|\theta)}[L(m; w)] = E_{m \sim p(m|\theta)}[\nabla_w L(m; w)]$$

Simple Example of ENAS

- Optimize w to minimize $E_{m \sim p(m|\theta)}[L(m; w)]$ on the training set

$$\nabla_w E_{m \sim p(m|\theta)}[L(m; w)] = E_{m \sim p(m|\theta)}[\nabla_w L(m; w)]$$

  - Sample a model m from $p(m|\theta)$
  - Compute $\nabla_w L(m; w)$ on the training set using regular backprop & update w

Simple Example of ENAS

- Optimize Θ to maximize $E_{m \sim p(m|\theta)}[R(m; w)]$ on the validation set

$$\nabla_\theta E_{m \sim p(m|\theta)}[R(m; w)] = E_{m \sim p(m|\theta)}[R(m; w)\ \nabla_\theta \log p(m|\theta)]$$

Simple Example of ENAS

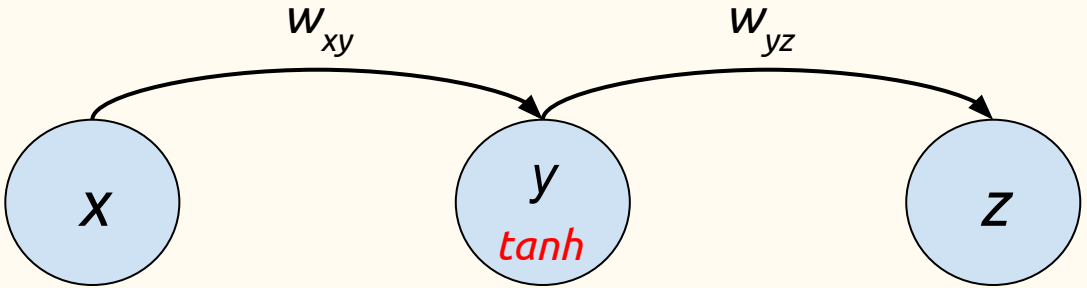
- Optimize Θ to minimize $E_{m \sim p(m|\theta)}[R(m; w)]$ on the validation set

$$\nabla_\theta E_{m \sim p(m|\theta)}[R(m; w)] = E_{m \sim p(m|\theta)}[R(m; w)\ \nabla_\theta \log p(m|\theta)]$$

  - Sample a model m from $p(m|\theta)$
  - Compute validation accuracy $R(m; w)$
  - Use REINFORCE estimator $R(m; w)\ \nabla_\theta \log p(m|\theta)$ to update θ

Simple Example of ENAS

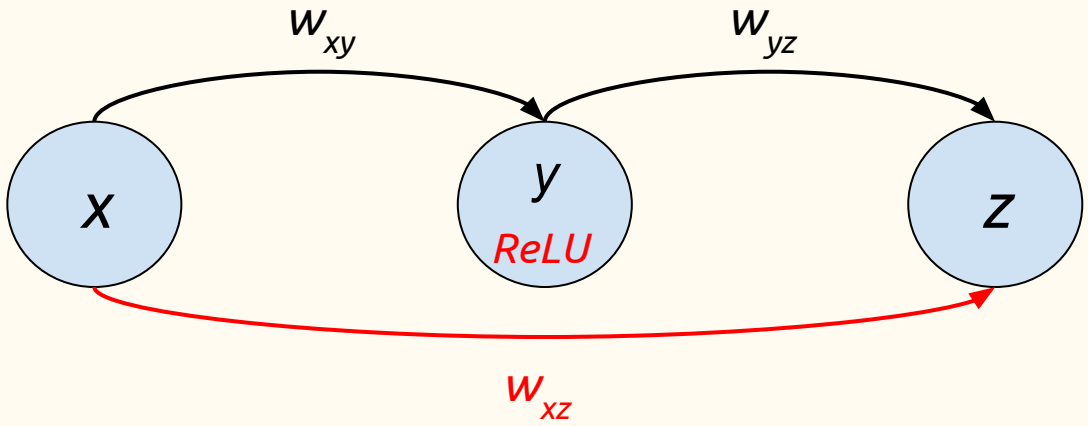
- Example: Sample that $\varphi = tanh$ and x, z not connected
  - Update w_{xy} and w_{yz} by gradient descent on the *training* set



$$y = tanh(w_{xy}x)$$
$$z = w_{yz}y$$

Simple Example of ENAS

- Example: Sample that $\varphi = ReLU$ and x, z connected
  - Update Θ using gradient descent on the *validation* set



$$y = ReLU(w_{xy}x)$$
$$z = w_{yz}y + w_{xz}x$$

Key assumption of ENAS

- Parameters that work well for one model architecture should work well for others

ENAS for Recurrent Cell

- Given $\{x^{(t)}, h^{(t-1)}\}$, how should $h^{(t)}$ be computed?
- Start with N nodes in computation graph: $h_1, h_2, ..., h_N$
- Controller is LSTM which operates for N steps

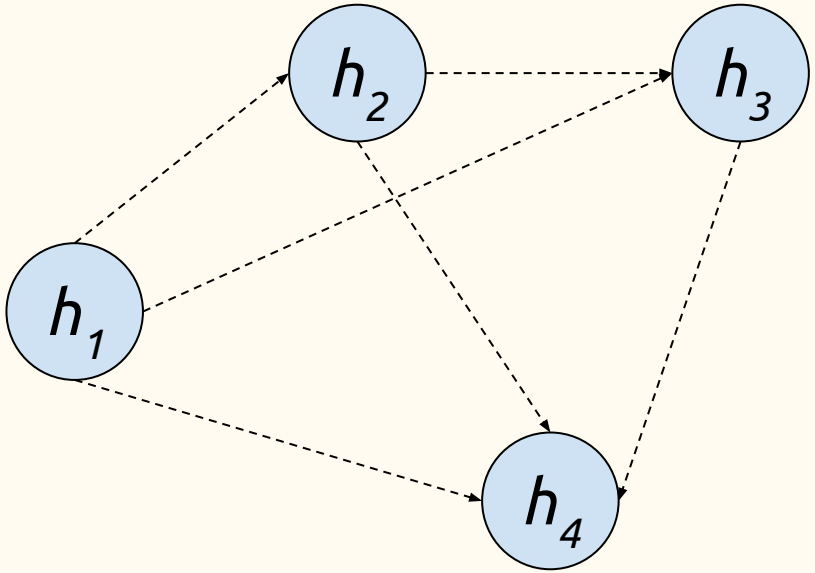
ENAS for Recurrent Cell

- At step i, controller decides:
  - Which previous node $j \in \{1, ..., i-1\}$ to connect to node i
  - An activation function $\varphi_i \in \{tanh, ReLU, Id, \sigma\}$
- Then:

$$h_i = \varphi_i(W_{ij} h_j)$$

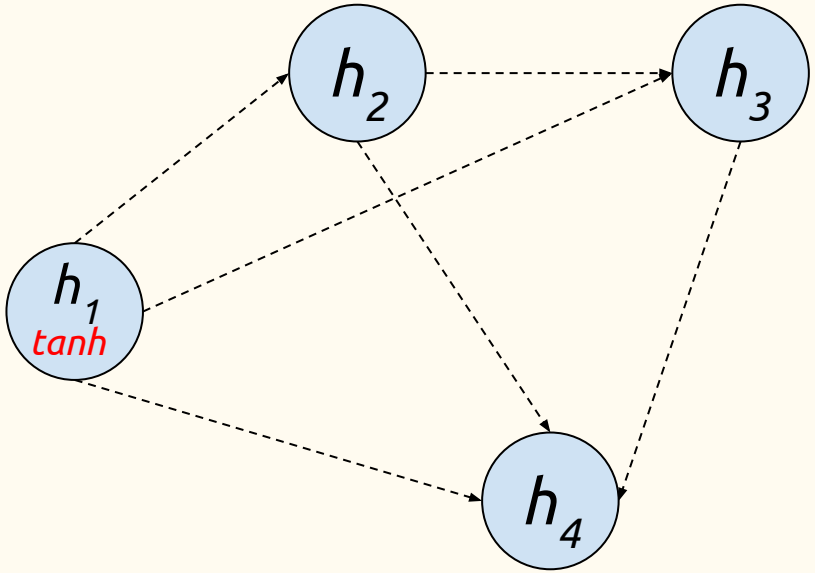
- $h^{(t)}$ is set to average of nodes which are not used as input to another node

ENAS for Recurrent Cell Example



- Controller selects:
- Function computed:

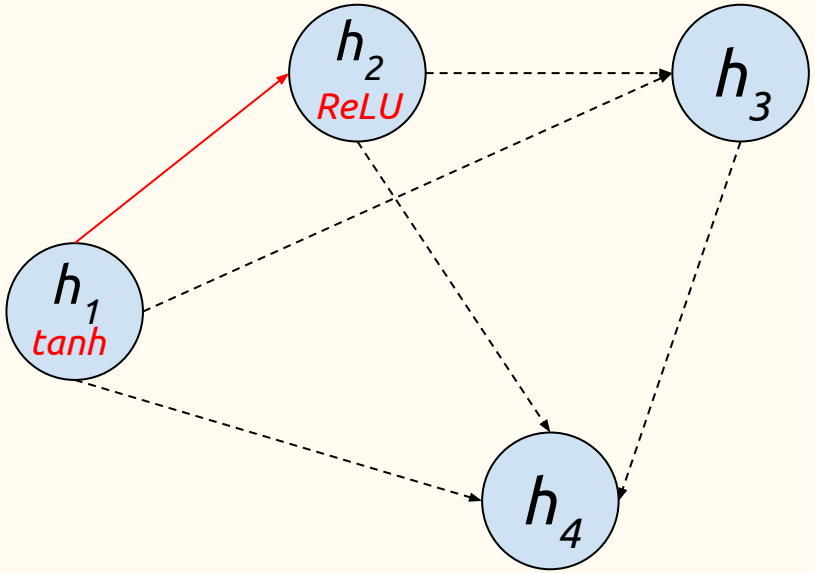
ENAS for Recurrent Cell Example



- Controller selects:
  - Step 1: tanh

- Function computed:
  - $h_1 = \tanh(W_x x^{(t)} + W_h h^{(t-1)})$

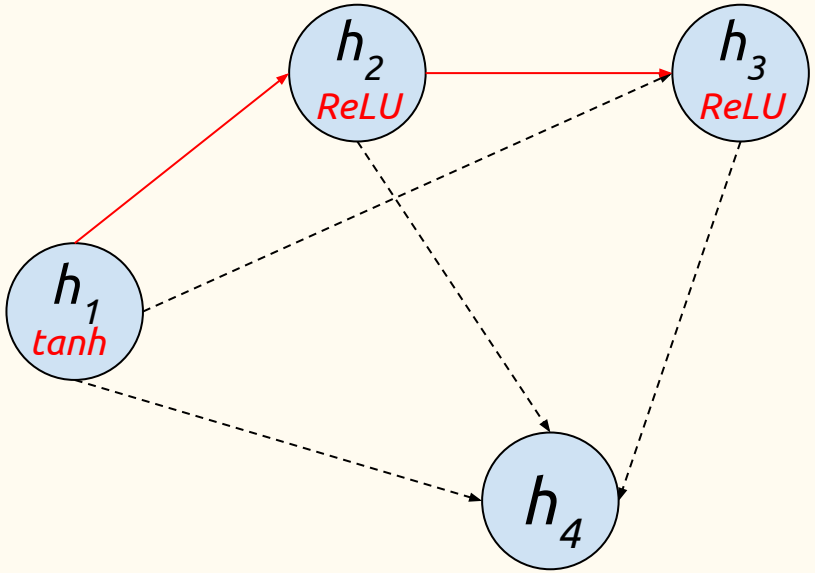
ENAS for Recurrent Cell Example



- Controller selects:
  - Step 1: tanh
  - Step 2: 1, ReLU

- Function computed:
  - $h_1 = \tanh(W_x x^{(t)} + W_h h^{(t-1)})$
  - $h_2 = \text{ReLU}(W_{12} h_1)$

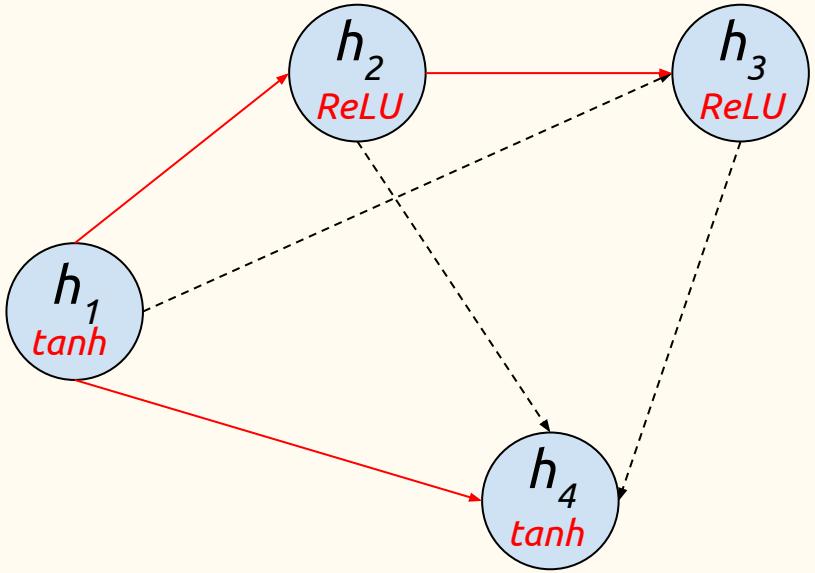
ENAS for Recurrent Cell Example



- Controller selects:
  - Step 1: tanh
  - Step 2: 1, ReLU
  - Step 3: 2, ReLU

- Function computed:
  - $h_1 = \tanh(W_x x^{(t)} + W_h h^{(t-1)})$
  - $h_2 = \mathrm{ReLU}(W_{12} h_1)$
  - $h_3 = \mathrm{ReLU}(W_{23} h_2)$

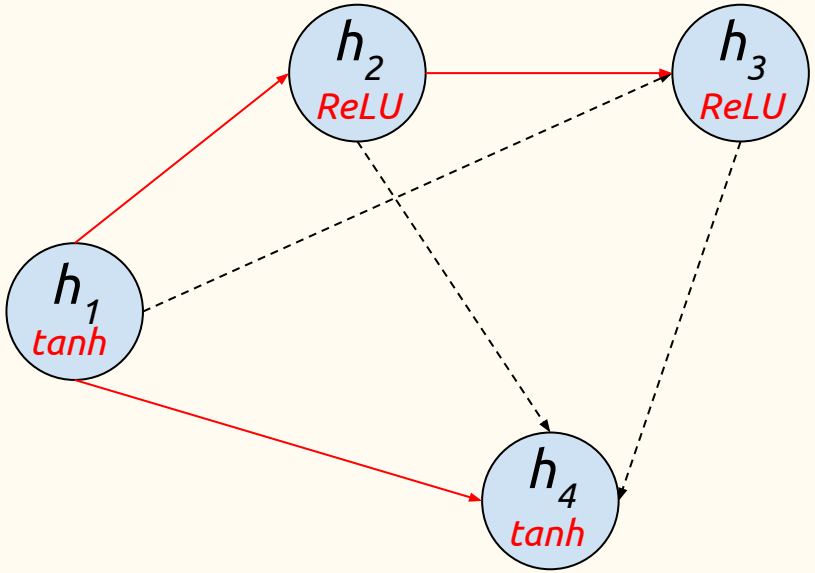
ENAS for Recurrent Cell Example



- Controller selects:
  - Step 1: tanh
  - Step 2: 1, ReLU
  - Step 3: 2, ReLU
  - Step 4: 1, tanh

- Function computed:
  - $h_1 = \tanh(W_x x^{(t)} + W_h h^{(t-1)})$
  - $h_2 = \mathrm{ReLU}(W_{12} h_1)$
  - $h_3 = \mathrm{ReLU}(W_{23} h_2)$
  - $h_4 = \tanh(W_{14} h_1)$

ENAS for Recurrent Cell Example



- Controller selects:
  - Step 1: tanh
  - Step 2: 1, ReLU
  - Step 3: 2, ReLU
  - Step 4: 1, tanh

- Function computed:
  - $h_1 = \tanh(W_x x^{(t)} + W_h h^{(t-1)})$
  - $h_2 = \text{ReLU}(W_{12} h_1)$
  - $h_3 = \text{ReLU}(W_{23} h_2)$
  - $h_4 = \tanh(W_{14} h_1)$
  - $h^{(t)} = ½ (h_3 + h_4)$

ENAS for Convolutional Networks

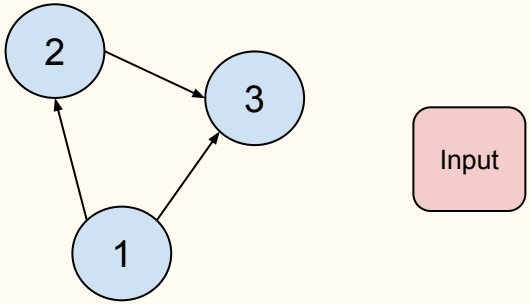
- At step i, controller decides:
  - Which previous node $j \in \{1, ..., i-1\}$ to connect to node i
  - Which computation operation to use

$$g_i \in \{conv_{3x3}, conv_{5x5}, sepconv_{3x3}, sepconv_{5x5}, maxpool_{3x3}, avgpool_{3x3}\}$$

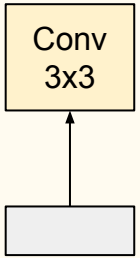
- Then:

$$h_i = \text{concat}(g_i(h_{i-1}), h_j) \text{ for all selected } j$$

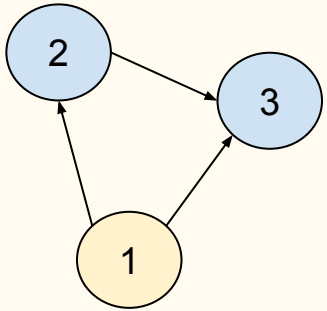
ENAS for Convolutional Networks

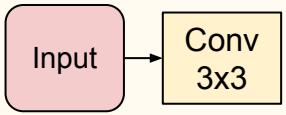


ENAS for Convolutional Networks

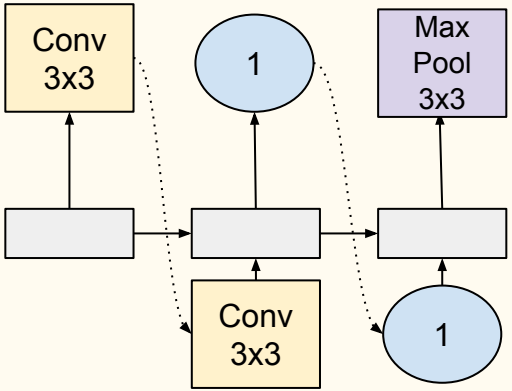


Node 1

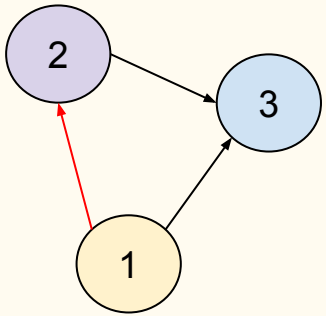


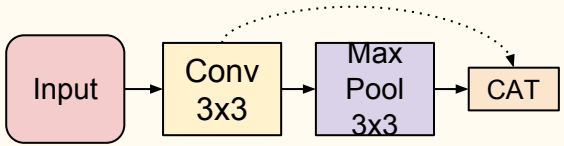


ENAS for Convolutional Networks

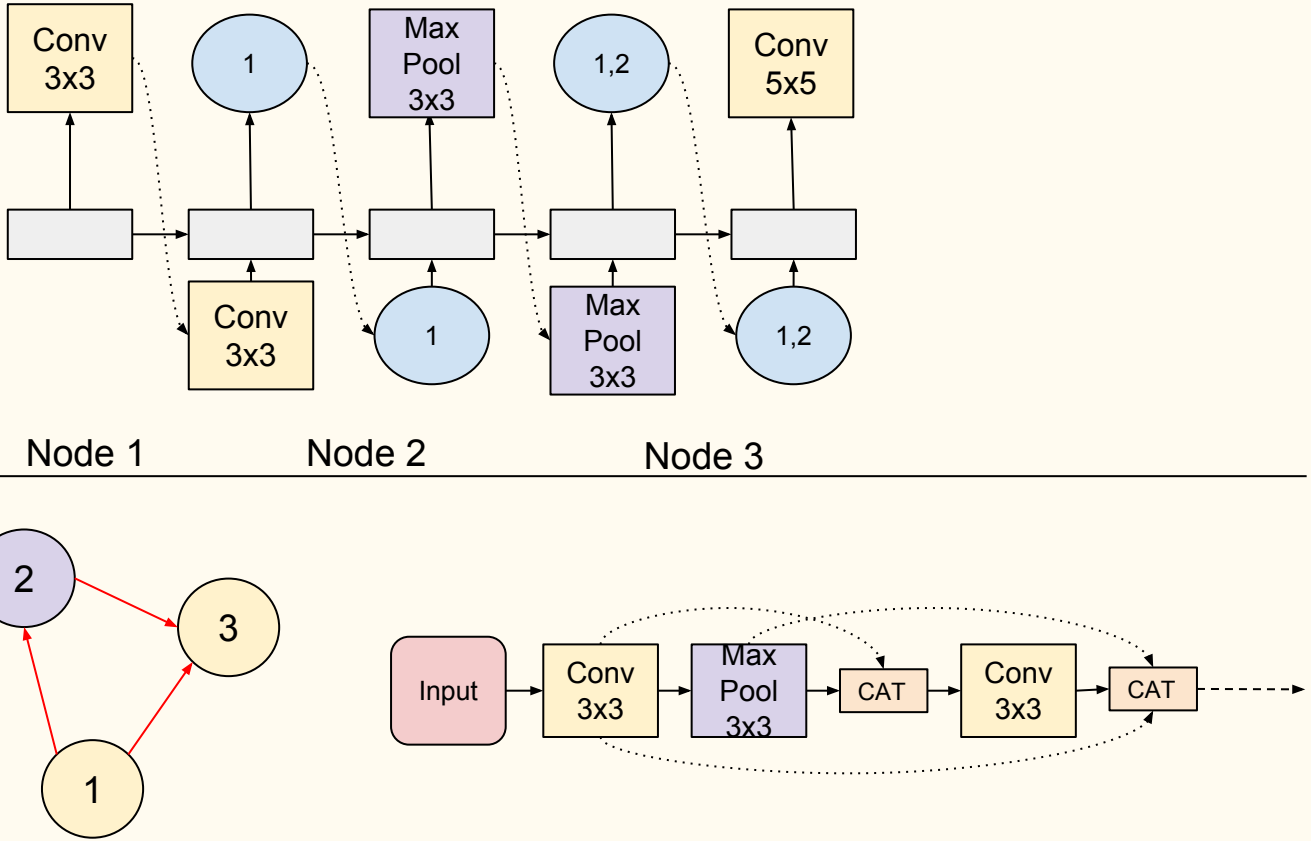


Node 1 Node 2





ENAS for Convolutional Networks

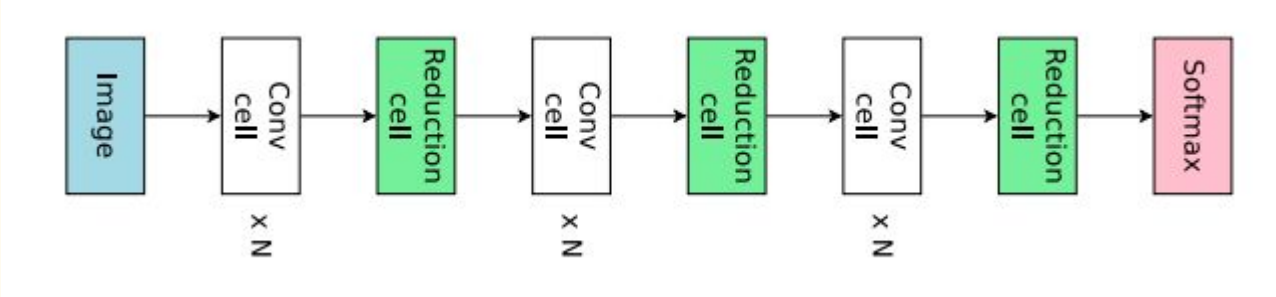


ENAS for Convolutional Networks

- Search space is huge - with 12 layers, 1.6×10^{29} possible networks.

ENAS for Convolutional Cells

- Represents *local* computations inside a cell



ENAS for Convolutional Cells

- Represents *local* computations inside a cell
- At step i, controller decides:
  - Which two previous node $j \in \{1, ..., i-1\}$ to connect to node i
  - Which computation operations to use for the two selected nodes

$$g_{ij} \in \{identity, sepconv_{3x3}, sepconv_{5x5}, maxpool_{3x3}, avgpool_{3x3}\}$$

- Then:

$$h_i = \sum(g_{ij}(h_j)) \text{ for the two selected } j$$

- (+ reduction cell where strides are 2)

ENAS for Convolutional Cells

- Search space - with 7 nodes, 1.3×10^{11} configurations.

Deriving novel architectures from ENAS

- Sample several architectures from a trained ENAS model

Deriving novel architectures from ENAS

- Sample several architectures from a trained ENAS model

- Compute their reward on a single minibatch from the validation set

Deriving novel architectures from ENAS

- Sample several architectures from a trained ENAS model
- Compute their reward on a single minibatch from the validation set
- Take one with the highest reward and re-train from scratch.

Experiments - Penn Treebank

Maybe there is a flaw in the photo industry itself? Is today's model of licensing and sales of photographs viable? How best to sell your photos? Will open resources photo stocks increase supply growth up until the number of pictures will not reach a level where photographers could not even very cheap to sell good photos and make money on those that sell? Theoretically, this horrible scenario is likely to become reality. Already Shutter stock alone offers more than 5 million photographs, which do not require payment of royalties, and, as stated CEO John Orangey, each month the number of new revenue in the millions. Number of images in the bank Dreamtime also exceeded 5 million. Due to the fact that the old photos - sold or not - remain in their bases, photo stock will continue to grow indefinitely. Since the proposal ahead of demand - in the world has always been more pictures than buyers - prices will become lower and lower.

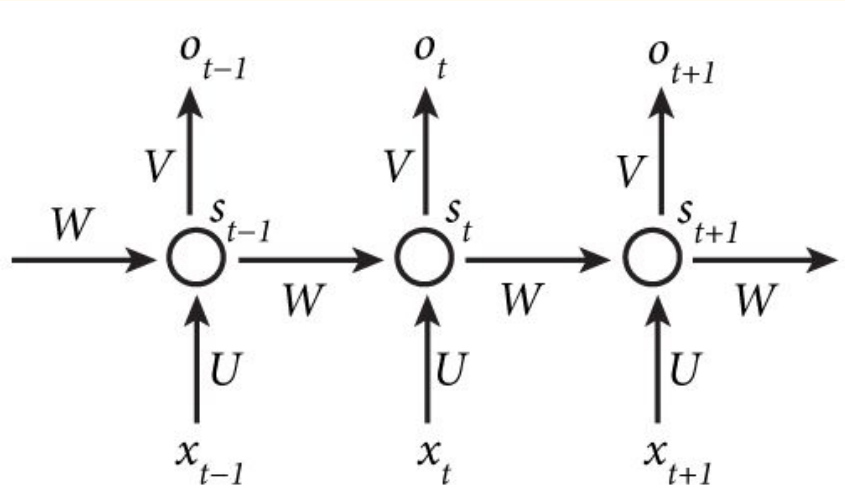
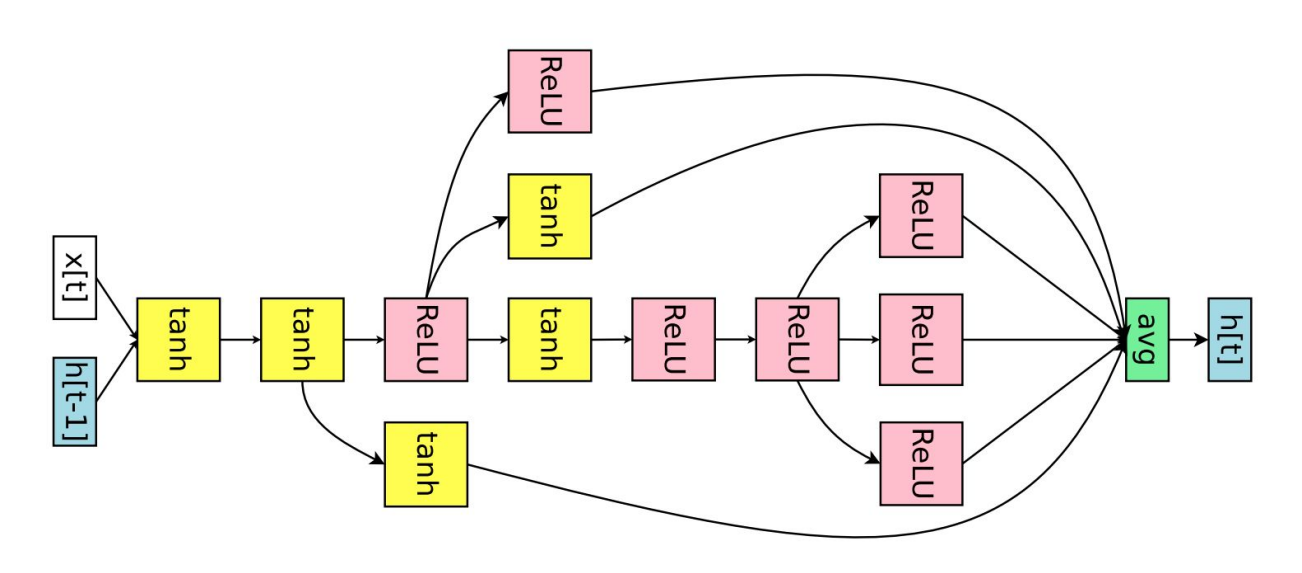


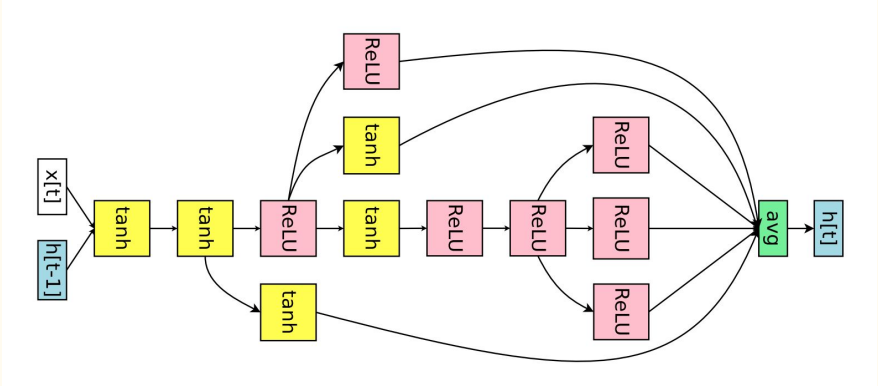
Image source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Experiments - Penn Treebank



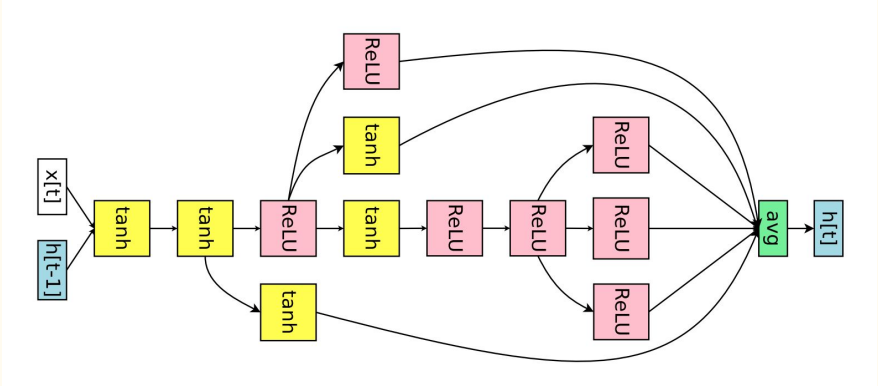
Experiments - Penn Treebank

- All non-linearities are either ReLU or tanh
- Similar to Mixture-of-Contexts architecture
- Local optimum
  - Changing one non-linearity leads to significant drop in performance



Experiments - Penn Treebank

- All non-linearities are either ReLU or tanh
- Similar to Mixture-of-Contexts architecture
- Local optimum
  - Changing one non-linearity leads to significant drop in performance
- 10 hours with single GPU (1000x faster than NAS!)



Experiments - Penn Treebank

Architecture	Additional Techniques	Params (million)	Test PPL
LSTM (Zaremba et al., 2014)	Vanilla Dropout	66	78.4
LSTM (Gal & Ghahramani, 2016)	VD	66	75.2
LSTM (Inan et al., 2017)	VD, WT	51	68.5
LSTM (Melis et al., 2017)	Hyper-parameters Search	24	59.5
LSTM (Yang et al., 2018)	VD, WT, ℓ_2, AWD, MoC	22	57.6
LSTM (Merity et al., 2017)	VD, WT, ℓ_2, AWD	24	57.3
LSTM (Yang et al., 2018)	VD, WT, ℓ_2, AWD, MoS	**22**	**56.0**
RHN (Zilly et al., 2017)	VD, WT	24	66.0
NAS (Zoph & Le, 2017)	VD, WT	54	62.4
ENAS	VD, WT, ℓ_2	**24**	**55.8**

Table 1. Test perplexity on Penn Treebank of ENAS and other baselines. Abbreviations: RHN is *Recurrent Highway Network*, VD is *Variational Dropout*; WT is *Weight Tying*; ℓ_2 is *Weight Penalty*; AWD is *Averaged Weight Drop*; MoC is *Mixture of Contexts*; MoS is *Mixture of Softmaxes*.

Experiments - CIFAR10

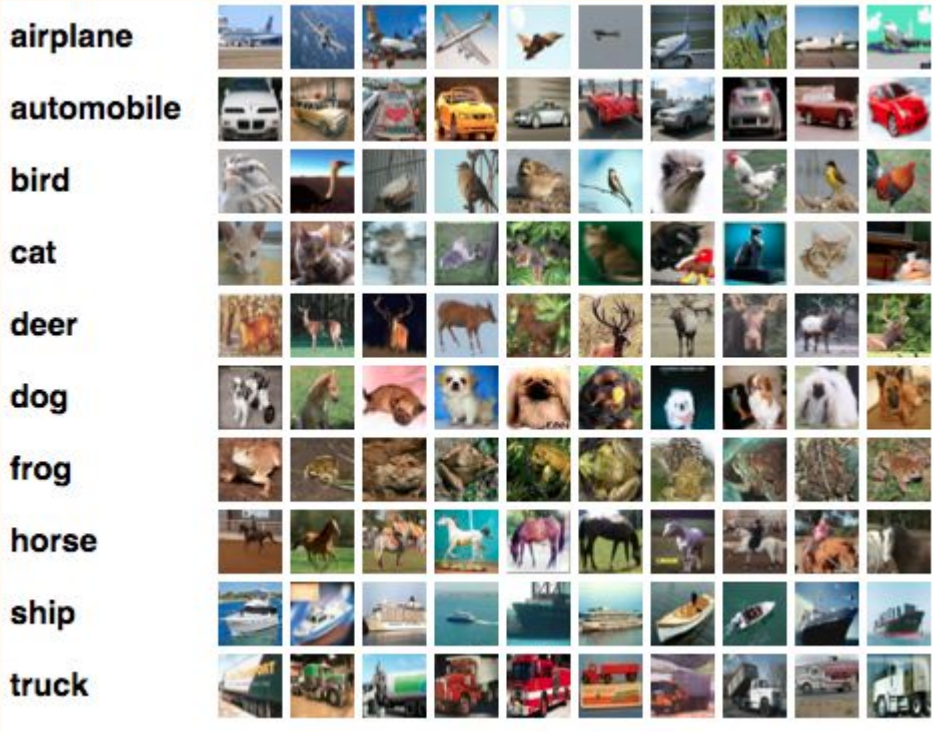


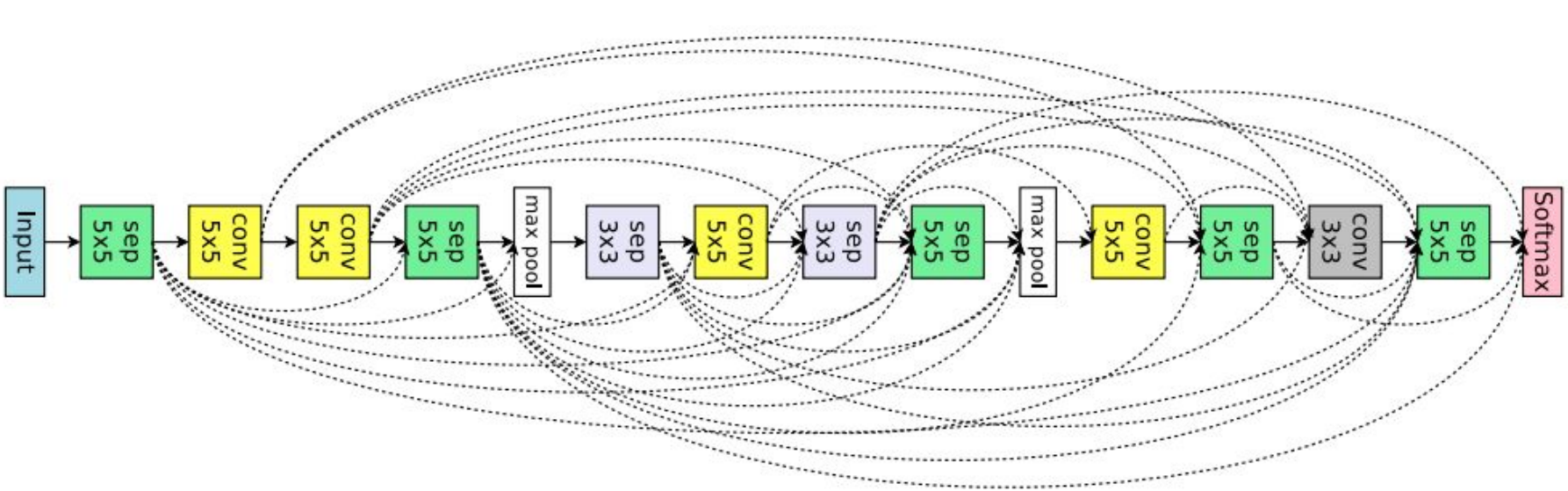
Image source: https://www.cs.toronto.edu/~kriz/cifar.html

Experiments - CIFAR10

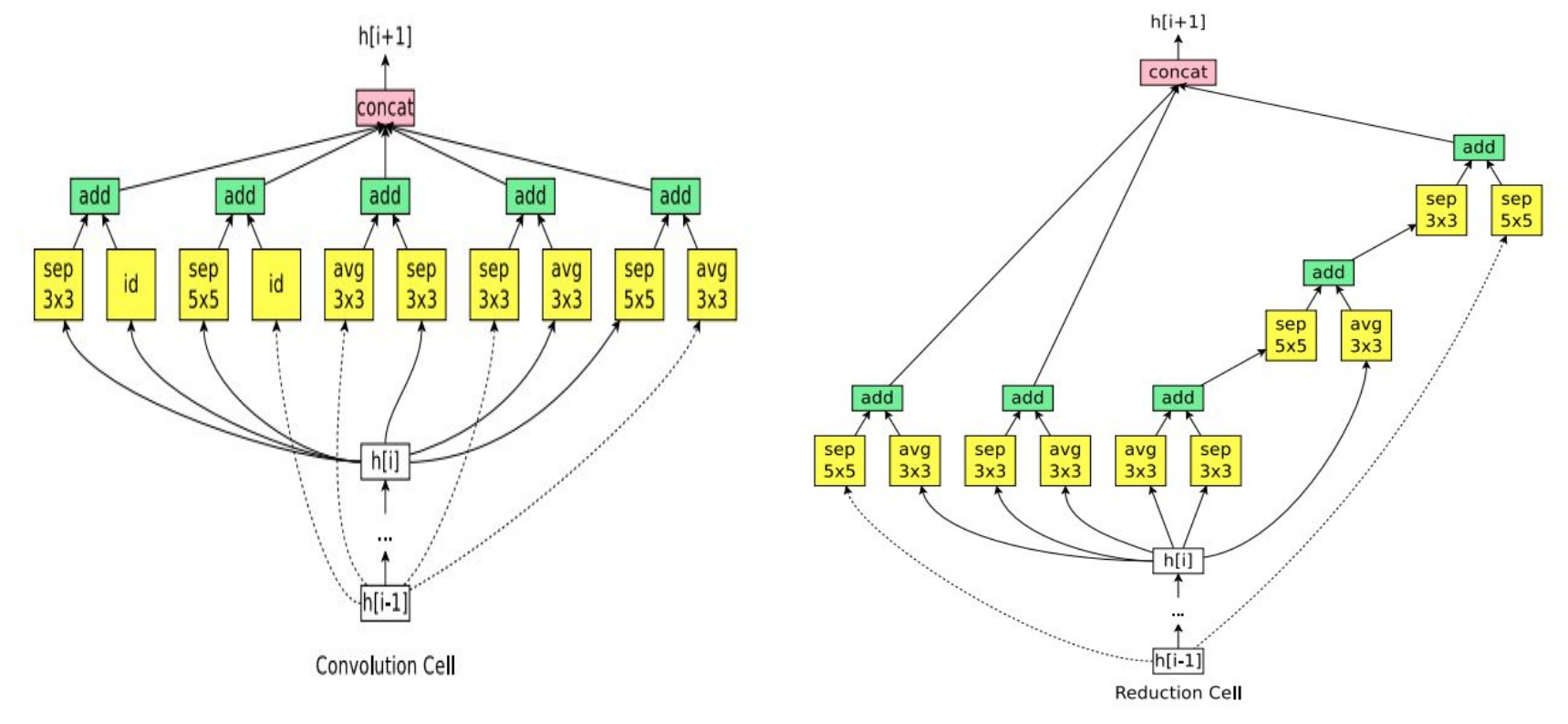
Method	GPUs	Times (days)	Params (million)	Error (%)
DenseNet-BC (Huang et al., 2016)	–	–	25.6	3.46
DenseNet + Shake-Shake (Gastaldi, 2016)	–	–	26.2	2.86
DenseNet + CutOut (DeVries & Taylor, 2017)	–	–	26.2	**2.56**
Budgeted Super Nets (Veniat & Denoyer, 2017)	–	–	–	9.21
ConvFabrics (Saxena & Verbeek, 2016)	–	–	21.2	7.43
Macro NAS + Q-Learning (Baker et al., 2017a)	10	8-10	11.2	6.92
Net Transformation (Cai et al., 2018)	5	2	19.7	5.70
FractalNet (Larsson et al., 2017)	–	–	38.6	4.60
SMASH (Brock et al., 2018)	1	1.5	16.0	4.03
NAS (Zoph & Le, 2017)	800	21-28	7.1	4.47
NAS + more filters (Zoph & Le, 2017)	800	21-28	37.4	**3.65**
ENAS + macro search space	1	0.32	21.3	4.23
ENAS + macro search space + more channels	1	0.32	38.0	**3.87**
Hierarchical NAS (Liu et al., 2018)	200	1.5	61.3	3.63
Micro NAS + Q-Learning (Zhong et al., 2018)	32	3	–	3.60
Progressive NAS (Liu et al., 2017)	100	1.5	3.2	3.63
NASNet-A (Zoph et al., 2018)	450	3-4	3.3	3.41
NASNet-A + CutOut (Zoph et al., 2018)	450	3-4	3.3	**2.65**
ENAS + micro search space	1	0.45	4.6	3.54
ENAS + micro search space + CutOut	1	0.45	4.6	**2.89**

Table 2. Classification errors of ENAS and baselines on CIFAR-10. In this table, the first block presents DenseNet, one of the state-of-the-art architectures designed by human experts. The second block presents approaches that design the entire network. The last block presents techniques that design modular cells which are combined to build the final network.

Experiments - CIFAR10



Experiments - CIFAR10



Conclusion

- ENAS speeds up NAS by more than 1000x in terms of GPU hours.
- This is done by sharing of parameters across child models during the search.
- Showed ENAS works well on both Penn Treebank and CIFAR10.