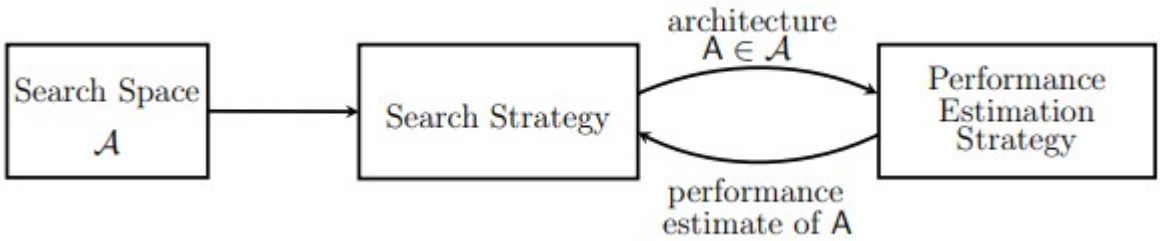
Neural Architecture Search : Mid-Evaluation Presentation

Ashish Kumar Jayant (M.Tech Research, CSA)
Ankith Kumar (M.Tech, CSA)
Somyajit Guin (PhD, CSA)

Neural Architecture Search (NAS) : Let’s Recall

- Process of automating the architecture design part of Deep Learning neural networks.
- Performance should be comparable to state of the art results.
- Challenges : Search Space,Search strategy, Performance Estimation Strategy
- Usability across datasets?



Neural Architecture Search with Reinforcement learning (Zoph, B. and Le, Q. V. ,Google Brain,2016)

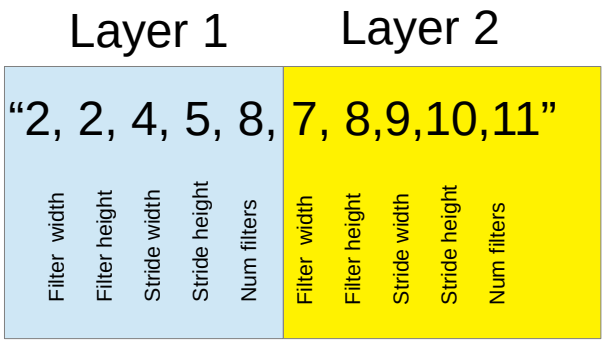
- Formulation of NAS as Reinforcement Learning problem –

  - **State Space** : set of possible architectures
    - So as to bound the search space we have some predefined elements for e.g Image Classification datasets we will have Conv kernel, Pooling kernel,activation functions etc with limited no of layers and language models will have recurrent cells and activation functions.
    - Note that search space do not include set of optimizers (sgd,adam,adagrad etc) here.
  - **Reward** : Corresponding Performance metric (for e.g Error rate for CiFAR,Perplexity for PennTreeBank dataset)
  - **Action**: Generating/Choosing an architecture
  - **Policy** : Stochastic Policy i.e, we will have probability of choosing an architecture not a determinsitic policy.
  - **Training mechanism** : Policy Gradient
    - Better convergence properties.
    - Effectiveness in high-dimensional or continuous action spaces
    - Ability to learn stochastic policies

Neural Architecture Search with Reinforcement learning : Generation of network

- Neural Networks can be represented as variable length string.
- For e.g (as given in paper) a simple feedforward CNN can be represented as -

```
layer {
name = layer_1
type = conv
connect_to = input
filter_width = 2
filter_height = 2
stride_width = 4
stride_height = 5
num_filters = 8
}
layer {
name = layer_2
type = conv
connect_to = layer_1
filter_width = 7
filter_ height = 8
stride_width = 9
stride_height = 10
num_filters = 11
}
layer {
name = classifier
connect_to = layer_2
num_class = 10
}
```



Neural Architecture Search with Reinforcement learning : Generation of network contd

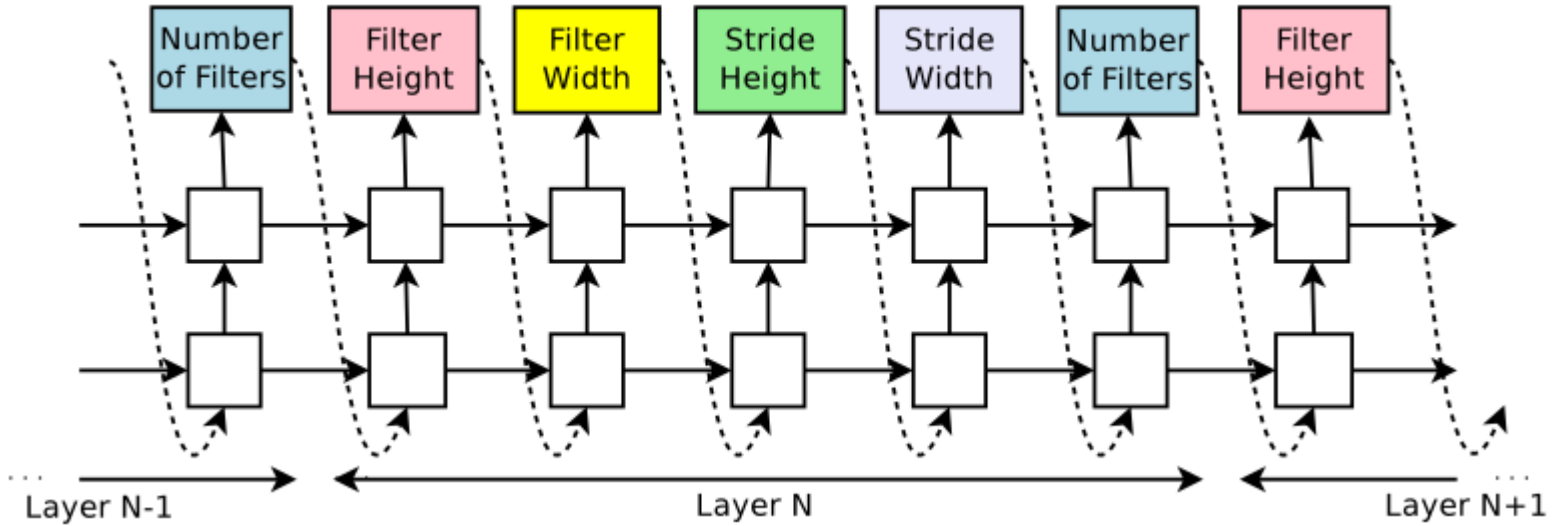
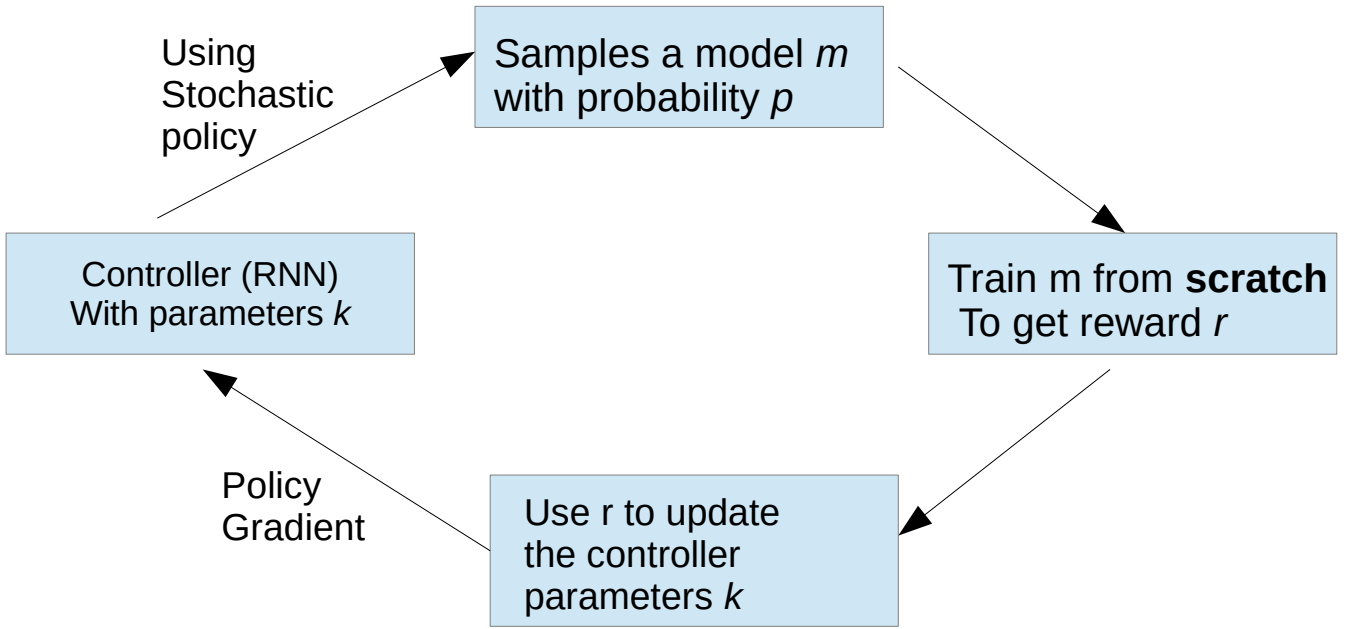


*Image Source - Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning, 2016

Neural Architecture Search with Reinforcement learning : Flow



Neural Architecture Search with Reinforcement learning : Training controller

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return v_t as an unbiased sample of $Q^{\pi_\theta}(s_t, a_t)$

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$

function REINFORCE

Initialise θ arbitrarily

for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**

for $t = 1$ to $T - 1$ **do**

$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$

end for

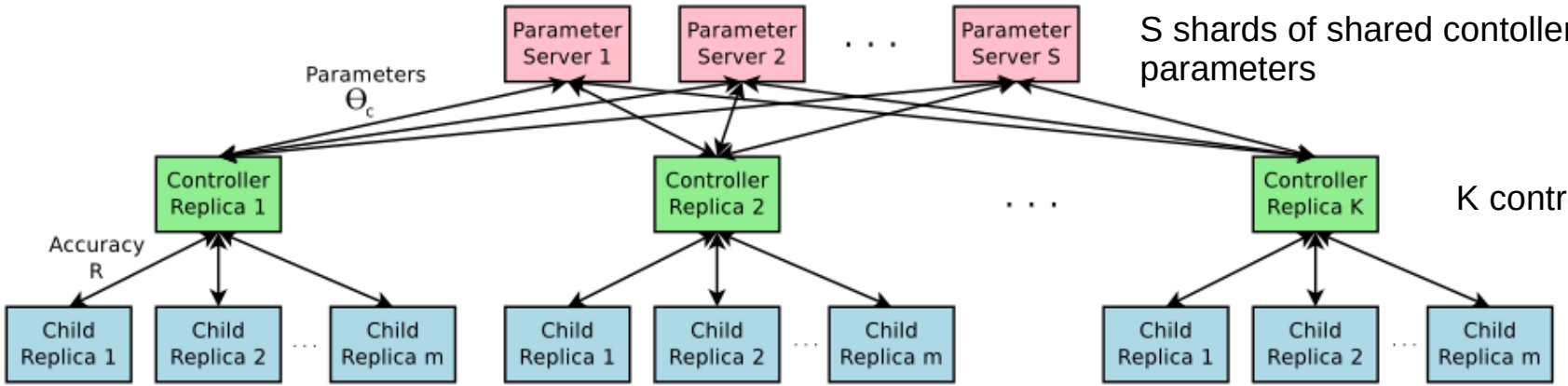
end for

return θ

end function

*Source – David Silver's RL lecture series

Neural Architecture Search with Reinforcement learning : Training setup



S shards of shared contoller parameters

K controllers

Each Controllers Training m Different child models

*Image Source - Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning, 2016

Neural Architecture Search with Reinforcement learning : Pros & Cons

- ✔ Achieves state of the art results on CiFAR and PenTreebank datasets
- ✔ Can be used as baseline for other NAS models

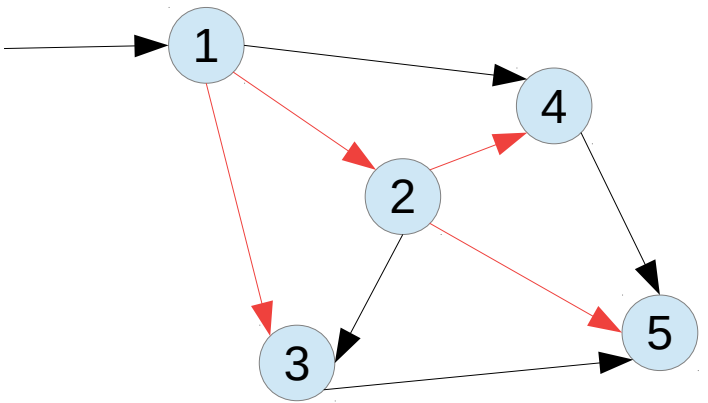
- ✗ Computationally expensive
  - ✗ 800 GPUs used for CiFAR,
  - ✗ 450 GPUs used for PenTreeBank
- ✗ Complexity of implementation due to distributed training setup.
- ✗ Suffers with disdavantages of policy gradient method.

Efficient Neural Architecture Search via Parameter Sharing (Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean. J,Google Brain-Stanford-CMU collab, 2018)

- Search Strategy is still Reinforcement learning based, hence our framework will remain same.
- Two important changes -
  - Generation of Architecture
  - Sharing of parameters of child models instead of re-training it again.

Efficient Neural Architecture Search via Parameter Sharing : Generation

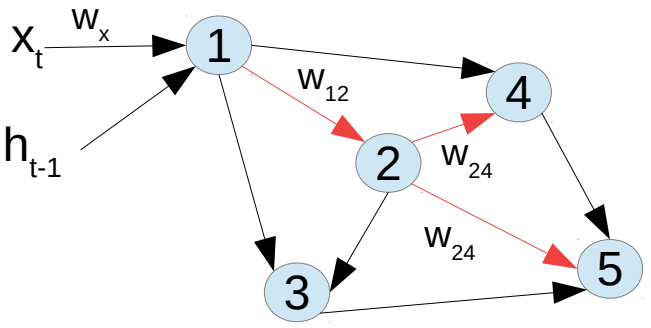
- Main idea is that we can represent search space of NAS as a single directed acyclic graph (DAG).
- Choosing/sampling an architecture will mean choosing a sub-graph of this large DAG.
- This design also allows parameters to be shared between child models.



Chosen subgraph : [1,2,4]
Resultant computation :
op_node4(op2_node2(op_node1(input)))

Efficient Neural Architecture Search via Parameter Sharing : Generation of Recurrent cell (for text based problems) example

- Assume node 1 to be input node, h_{t-1} and w_x are initialized by us and let list of activation functions we want to sample from be [tanh,relu,sigmoid].
- W_{ij} are global and is shared among all sampled sub-graphs which enables parameter sharing.
- Controller have two decision to make
  - Which activation function?
  - Which previous node to connect from or which weight matrix w_{ij} will be activated?



Efficient Neural Architecture Search via Parameter Sharing : Generation of Recurrent cell (for text based problems) example contd.

- At node 1 controller samples activation function ‘tanh’,

  $h_1 = \tanh(x_t.W_x + h_{t-1}.W_h)$

- At node 2, controller sample ‘relu’ and prev node index ‘1’ which activates globally shared w_{12}

  $h_2 = \text{relu}(w_{12}.h_1)$

- At node 3, controller samples ‘relu’ and prev node index ‘1’.

  $h_3 = \text{relu}(w_{13}.h_1)$

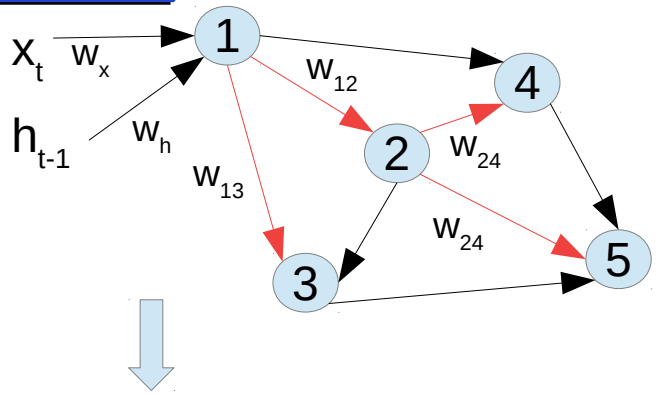
- At node 4, controller samples ‘sigmoid’ and prev node index ‘2’.

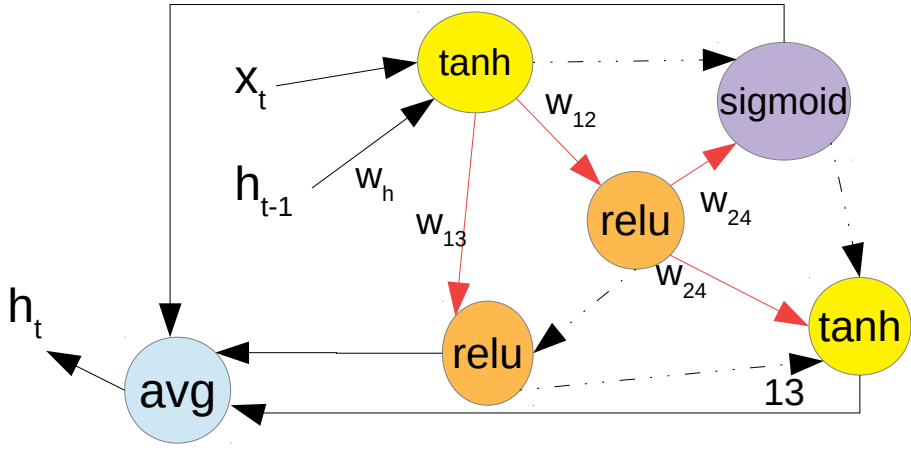
  $h_4 = \text{sigmoid}(w_{24}.h_2)$

- At node 5, controller samples ‘tanh’ and prev node index ‘2’

  $h_5 = \tanh(w_{25}.h_2)$

- Final output : $h_t = \text{mean}(h_4,h_5,h_3)$

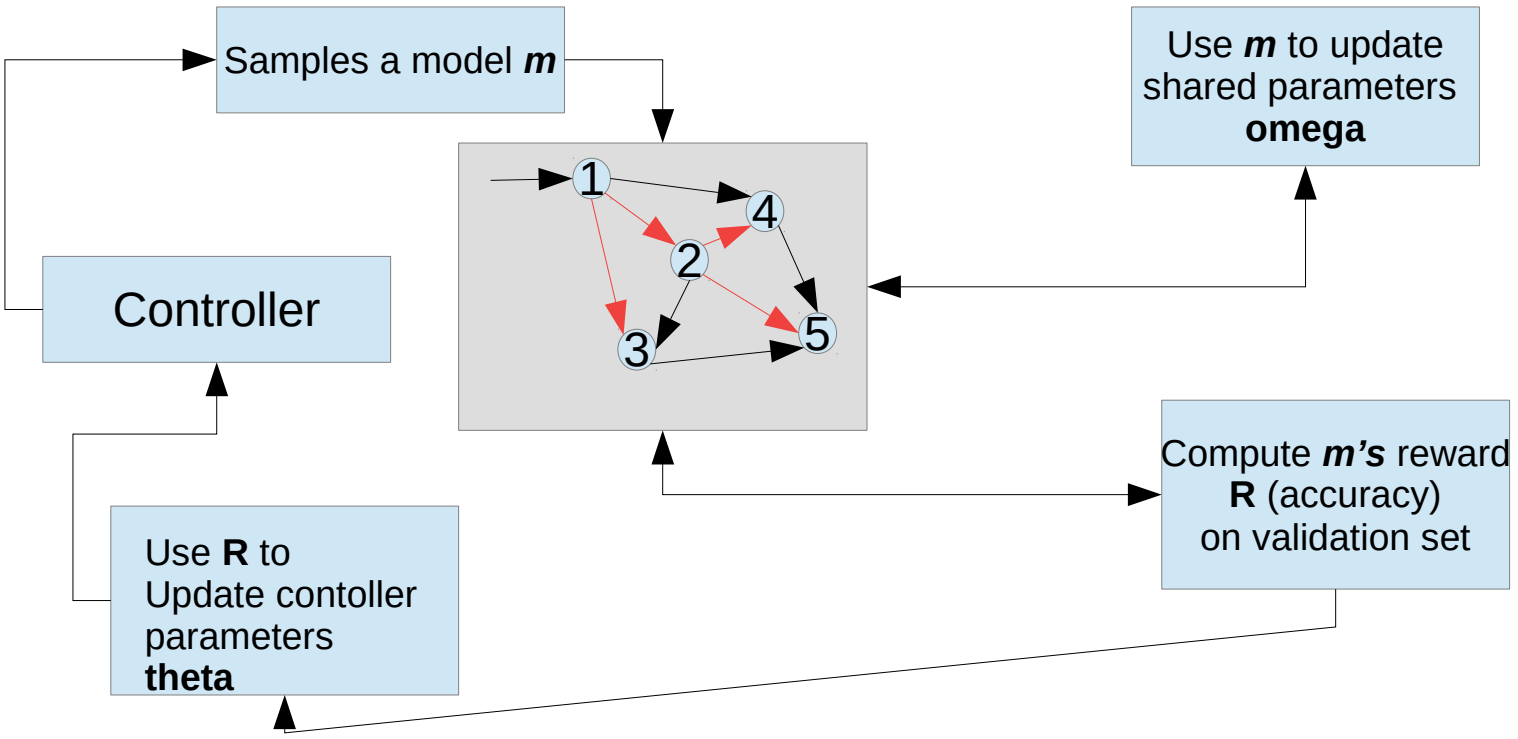




Efficient Neural Architecture Search via Parameter Sharing : Training

- There are two sets of learnable parameters: the parameters of the controller LSTM, denoted by θ, and the shared parameters of the child models, denoted by ω.
- The training procedure of ENAS consists of two interleaving phases.
  - Fix the controller policy and train child models using SGD on **ω** to minimize the expected loss function.
    - Loss function is **cross-entropy loss L(m;ω)** computed on mini-batch of training data, model **m** sampled using parametrized stochastic policy **π(m;θ)** (*turns out any sampled model can be used to update the gradient) then perform SGD(**batch_size**, **m**)
  - Fix the **ω** and update the policy paramters **θ** aiming to maximise expected reward using Monte Carlo Policy Gradient (REINFORCE) described in slide 7.

Efficient Neural Architecture Search via Parameter Sharing : Flow



Efficient Neural Architecture Search via Parameter Sharing : Pros & Cons

- ✔ Achieves state of the art results on PenTreebank datasets and almost beats NAS on CiFAR.
- ✔ Required Nvidia GTX 1080i for 16 hours for training PenTreeBank dataset.
- ✗ Cannot search through optimizers in theory as it will require freezing of shared parameters.
- ✗ Suffers with disdavantages of policy gradient method.