Search Strategies in NAS

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

Goal of NAS : Update*

- We talked about goal of NAS in our previous presentation as to come up with architectures that achieves state of the art performance.
- Our metric of performance is geneally accuracy (or whatever performance metric we are using for error.)
- But state of the architectures that models come up with can be computationally expensive as well for mobile devices. For e.g an app running on phone or emdedded devices with low memory. So we need hardware aware architectures!
- Performance strategy should include both accuracy as well as model latency[1].

Search Strategies

- Reinforcement Learning Methods
- Gradient Based Methods
- Evolutionary Methods
- RL + Gradient Based Hybrid
- Bayesian Optimization

Point to be noted

- Comparing search startegies is not straightforward as it looks.This very recent (Feb, 2020) published ICLR paper[2] explains it comprehensively. This paper made following observations :
  - Different architectures from same search space perform very similarly.
  - How you train your model has a much bigger impact than the actual architecture chosen.
  - hyperparameters, like the number of cells or theseed itself have a very significant effect. the specific operations themselves have less impact on the final accuracy than the hand-designed macro-structure of the network.
  - All models are not verified on variety of datasets as most of them are just done on CiFAR, PTB datasets.

- So we will try to compare strategies if we can compare else talk about strategies independently and their results.

1) Reinforcement Learning Based

- 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
- Best models – ENAS, MnasNET.

RL Based NAS : Pros & Cons

- .
- ENAS is fastest NAS approach till date if we compare on CiFAR & PTB datasets.
- MnasNet[1] takes model latency into picture.
- ENAS is fast but doesn’t take model latency into picture.
- Non-differentiable discrete search space we cannot directly apply gradient descent methods which makes it complicated to implement as well.
- MnasNET is computationally expensive .

2) Gradient Based methods :

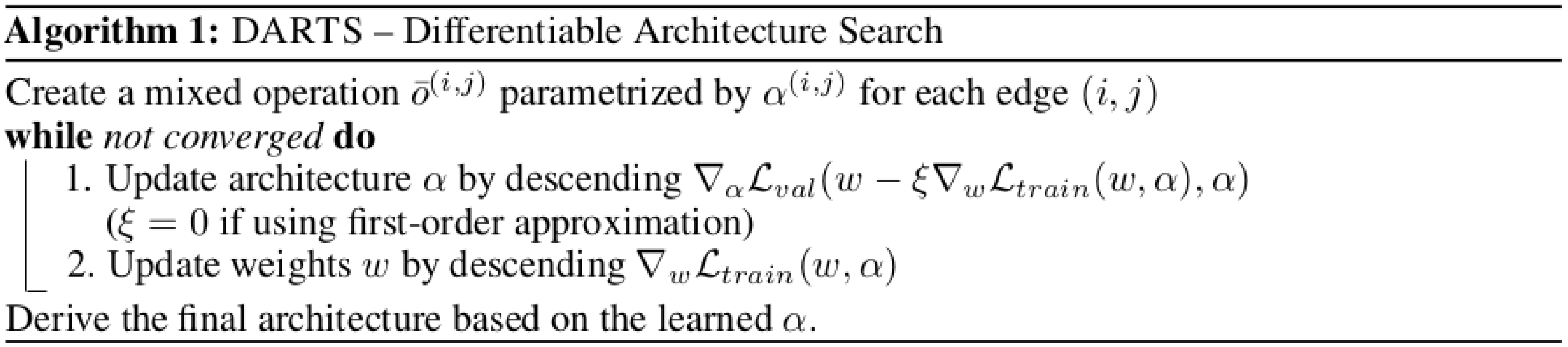
- Instead of searching over a discrete set of candidate architectures, we relax the search space to be continuous, so that the architecture can be optimized with respect to its validation set performance by gradient descent. Best model in this approach is
- Best models - DARTS[3] , ProxylessNAS[4], FBNet[5]
- We will explain the simple DARTS from them as Proxyless, FBNet just does some modifications after doing basic DARTS for taking model latency into picture.
- Continuous relaxation to enable direct gradient-based optimization: instead of fixing a single operation o_i (e.g., convolution or pooling) to be executed at a specific layer applied to input x_i **.**
- To make the search space continuous, we relax the categorical choice of a particular operation to a softmax over all possible operations $o^{(i,j)}$

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$$

  - operation mixing weights for a pair of nodes (i,j) are parametrized by alpha(i,j) of dimension |O|.

- Then most likely operation is chosen by taking argmax. This will give us the architecture. Then we will train this architecture to optimize its weight w.

DARTS Algorithm



Algorithm 1: DARTS – Differentiable Architecture Search

```
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 $\alpha$.
```

Gradient Based Methods : Pros & Cons

- This model does give us a diffrentiable method for NAS.
- Simpler than RL methods to implement as it does not involve designing any seperate architecture RNN controller.
- ProxylessNAS, FBNet take model latency into consideraton as well
- ProxylessNAS, FBNet take methods approximate network latency using differentiable functions.However, these approximations may fail when the underlying criteria cannot be accurately modeled.[6]

3) Evolutionary Methods

- Evolutionary algorithms evolve a population of models, i.e., a set of (possibly trained) networks; in every evolution step, at least one model from the population is sampled and serves as a parent to generate offsprings by applying mutations to it.
- In the context of NAS, mutations are local operations, such as adding or removing a layer, altering the hyperparameters of a layer, adding skip connections, as well as alteringtraining hyperparameters. After training the offsprings, their fitness (e.g., performance on a validation set) is evaluated and they are added to the population.
- Best Model – AmoebaNet[7]

Amoeba Net

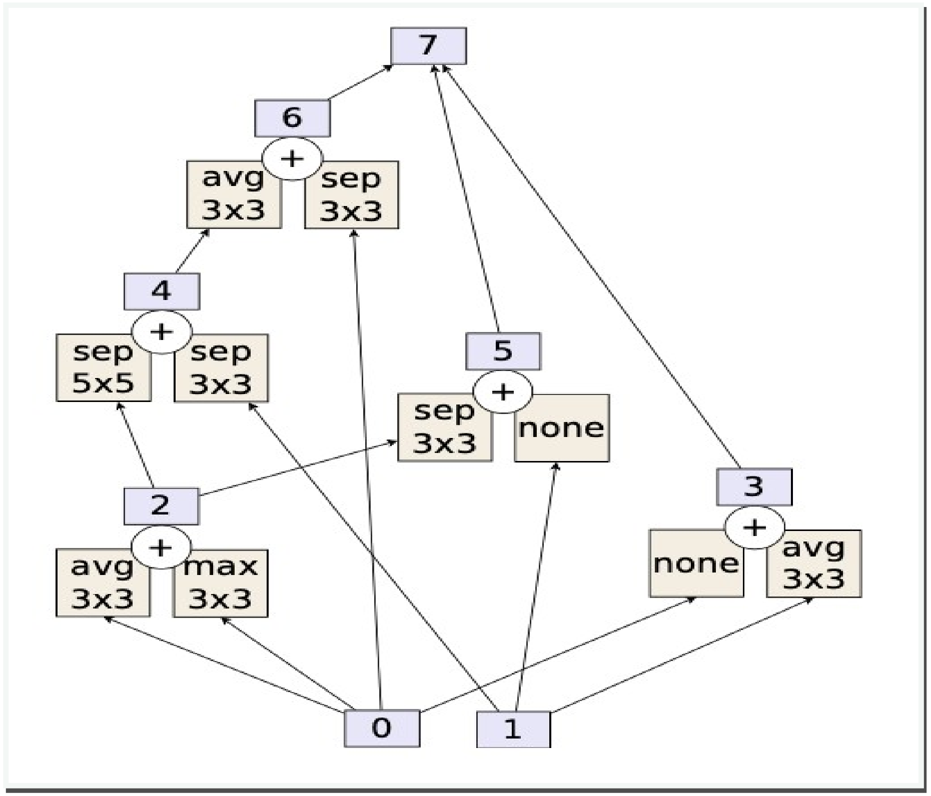
- A image classifier AmoebaNet-A, surpasses surpasses hand-designs[7] .Two additions are made to the standard evolutionary process
  - First, implementing the simplest set of mutations that would allow evolving in the **NASNet** search space.
  - Second, proposing a change to the well established **tournament selection** evolutionary algorithm[*] that refers to **aging** evolution or **regularized** evolution.
  - [*] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *FOGA*, 1991.

Evolutionary Algorithm

- Algorithm starts with a population of models.
- Each of these model has an architecture and each architecture is chosen at random and all the operations are chosen randomly from the list of allowed operations.
- Once we have an initial set of architectures. These all trained and evaluated on the data set.
- We have accuracy for each of them and that is our initial population.
- After this, the algorithm proceeds in cycles.
- At each cycle a worker takes a small subset of population.
- Take the best of the architecture or the best of the models in this subset.
- Make it as parent. (This step is critical because it is the only time in which the algorithm does something that is not random).
- Once the parent is selected, we copy it and mutate it’s architecture to produce the new architecture.
- Then we train and evaluate on the data set to create new model that is called child.
- This child is now put into the population and this will increase the size of the population by 1.
- Kill the oldest individual in the population to keep the population size constant.
- Oldest means the one that has been in the population for the longest time.
- This algorithm is called aging evolution.
- It also has some regularizing properties. So sometimes it is called regularized evolution.
- There are multiple workers, doing the same thing(described in previous slide). So, this algorithm is highly parallelizable.

Mutation

- Mutation is a rule that gives the architecture of the child from that of the parent.
- Diagram is the architecture of the parent.
- A mutation is a small local transformation that is applied and it's a random one
- There are two possible mutations to navigate the NASNet search space:
  - One is that replace one of the operation with another. Pick a random box in this diagram and replace the operation in it with one chosen at random from the list of allowed operations.
  - Another is that replace the hidden state. Take one of these arrows and move its origin to another hidden state so that it will not produce a loop.
- Produce a child from parent we do one of these two.



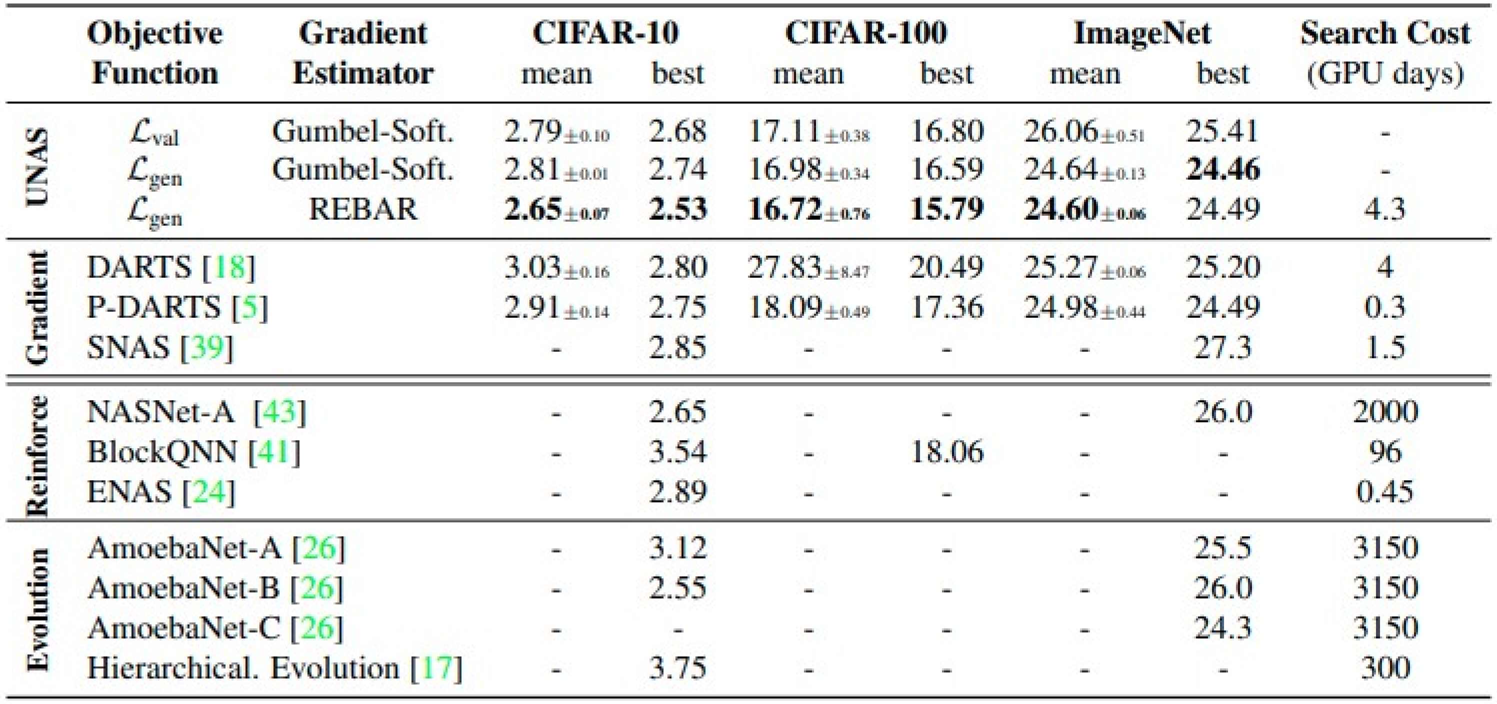
Pros & Cons

- Similar results as RL methods.
- Computationally expensive.
- Does not take model latency into picture as well.

4) RL + Gradient Methods Hydrid : UNAS[6](still in proceedings*)

- DNAS (diffrentiable NAS or simple gradient based) models can only optimize differentiable loss functions in search, and they require an accurate differentiable approximation of non-differentiable criteria.
- RL based methods can optimize non-diffrentiable loss functions but are computationally expensive if we want model latency as well to be considered.
- UNAS*[6] brings best of both worlds.
  - enables us to search for architectures using both differentiable objective functions & and non-differentiable functions (e.g., network latency).
  - Gradient estimation in UNAS is equal to the estimations obtained by RL-based frameworks that operate on discrete variables.

Comparison



	Objective Function	Gradient Estimator	CIFAR-10 mean	CIFAR-10 best	CIFAR-100 mean	CIFAR-100 best	ImageNet mean	ImageNet best	Search Cost (GPU days)
UNAS	$\mathcal{L}_{\text{val}}$	Gumbel-Soft.	$2.79_{\pm 0.10}$	2.68	$17.11_{\pm 0.38}$	16.80	$26.06_{\pm 0.51}$	25.41	-
	$\mathcal{L}_{\text{gen}}$	Gumbel-Soft.	$2.81_{\pm 0.01}$	2.74	$16.98_{\pm 0.34}$	16.59	$24.64_{\pm 0.13}$	**24.46**	-
	$\mathcal{L}_{\text{gen}}$	REBAR	$\mathbf{2.65}_{\pm 0.07}$	**2.53**	$\mathbf{16.72}_{\pm 0.76}$	**15.79**	$\mathbf{24.60}_{\pm 0.06}$	24.49	4.3
Gradient	DARTS [18]		$3.03_{\pm 0.16}$	2.80	$27.83_{\pm 8.47}$	20.49	$25.27_{\pm 0.06}$	25.20	4
	P-DARTS [5]		$2.91_{\pm 0.14}$	2.75	$18.09_{\pm 0.49}$	17.36	$24.98_{\pm 0.44}$	24.49	0.3
	SNAS [39]		-	2.85	-	-	-	27.3	1.5
Reinforce	NASNet-A [43]		-	2.65	-	-	-	26.0	2000
	BlockQNN [41]		-	3.54	-	18.06	-	-	96
	ENAS [24]		-	2.89	-	-	-	-	0.45
Evolution	AmoebaNet-A [26]		-	3.12	-	-	-	25.5	3150
	AmoebaNet-B [26]		-	2.55	-	-	-	26.0	3150
	AmoebaNet-C [26]		-	-	-	-	-	24.3	3150
	Hierarchical. Evolution [17]		-	3.75	-	-	-	-	300

References

1) MnasNet: Platform-Aware Neural Architecture Search for Mobile, CVPR 2019, Tan, Chen , Pang, Vasudevan,Google Brain

2) NAS EVALUATION IS FRUSTRATINGLY HARD, ICLR 2020 - Yang, Esperance, Carlucci

3) DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH, ICLR 2019 - Liu, Simonyan, Yang

4) ProxylessNAS: Direct neural architecture search on target task and hardware , ICLR, 2019.

5) FBnet: Hardware-aware efficient con-vnet design via differentiable neural architecture search. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019

6) UNAS: Differentiable Architecture Search Meets Reinforcement Learning, Arash Vahdat, Arun Mallya, Ming-Yu Liu, Jan Kautz, NVIDIA (Still in Proceedings)

7) Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized Evolution for Image Classifier Architecture Search. In AAAI, 2019