Introduction to AutoML

Tejaswini Pedapati

IBM Research

11 May 2023

Agenda

- What is AutoML?
- Neural Architecture Search
- Hyper parameter Optimization

Outline

1. What is AutoML?
2. Neural Architecture Search
3. Hyperparameter Optimization

Deep Learning Workflow

Preprocessing

- Normalization
- Standardization
- Categorical data
- Missing data

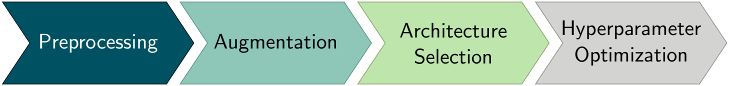
Deep Learning Workflow

- Random crops
- Flipping
- Cutout/mixup
- Other image manipulations (contrast, random noise, etc.)

Deep Learning Workflow

- Architecture selection: search vs. existing architectures
- Use pretrained architectures
- Constraint consideration: parameters, inference time

Deep Learning Workflow



- Learning rate
- Weight decay
- Dropout rates

Deep Learning Workflow

- Prune the network without significant loss in accuracy

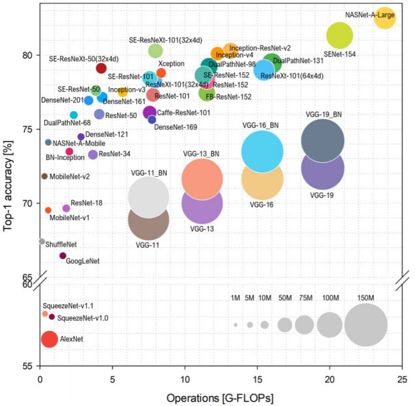
Deep Learning Workflow

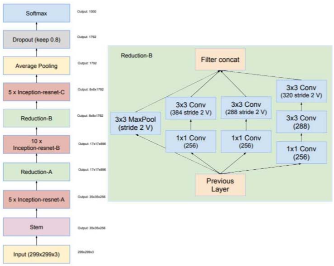
AutoML

- Lot of choices at every stage
- Explore them intelligently
- Automated machine learning is the process of automatically searching for the right pipeline to solve a particular task.
- Usage scenarios: No-code, good baseline

Outline

Architecture Design for Image Tasks





Innovations: Deeper networks, auxiliary classifiers, skip connections, bottlenecks, convolution stacking, global average pooling and many more

Images taken from Simone Bianco et al. "Benchmark Analysis of Representative Deep Neural Network Architectures". In: *IEEE Access* 6 (2018), pp. 64270–64277, Christian Szegedy et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* 2017, pp. 4278–4284

Outline

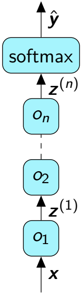
Search Spaces

- Neural architecture search space: subspace of all possible neural architectures.
- The limitation to a subspace allows for considering
  - human expert knowledge,
  - specific task (e.g. mobile architectures)
- We distinguish two types of search spaces:
  - global search space
  - cell-based search space

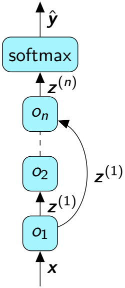
Outline

Simple Search Spaces

The global search space does not enforce repetitive operation pattern across the architecture.



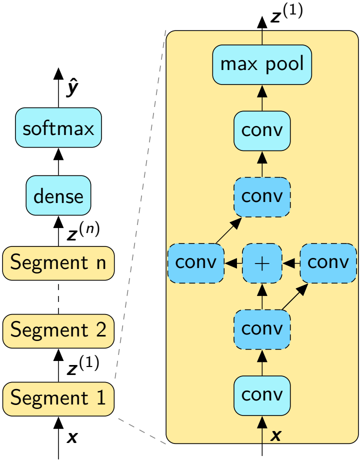
(a) Chain-structured [1]



(b) with skips [14]

Architecture Templates

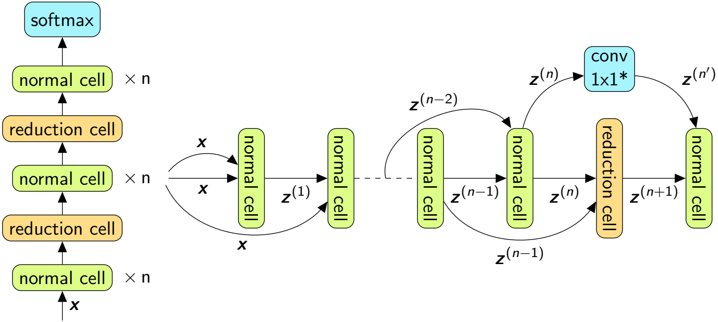
Architecture templates can be used to constrain the search space.



Outline

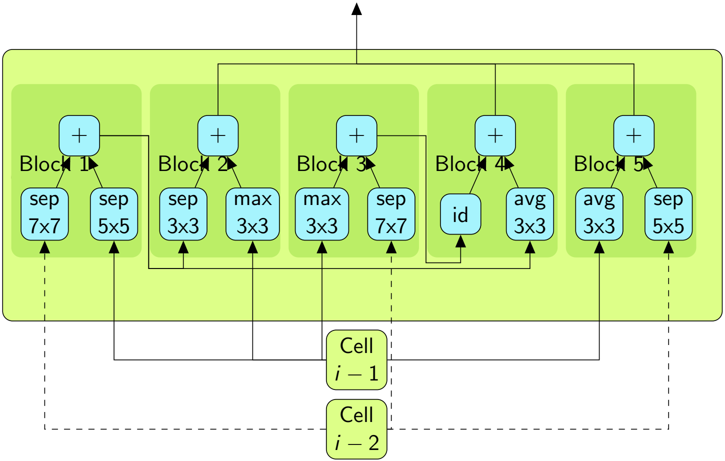
NASNet Search Space

Architectures from a cell-based search space are build by stacking few cells with the same topology.



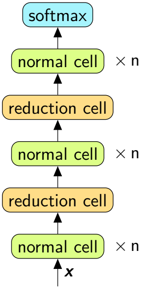
NASNet Search Space

Structure of a cell.

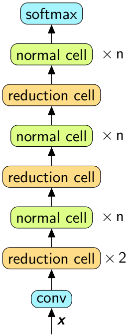


Transferring Architectures

Architectures from cell-based search spaces allow for easy transferability across different datasets.



(a) CIFAR-10



(b) ImageNet

Global vs. Cell-Based Search Spaces

Global Search Space

- Recommended for mobile architectures.

Cell-Based Search Space

- Higher accuracy
- Easier transferability and complexity adaptation

Problem Definition

Machine Learning Problem

$$\Lambda(\boldsymbol{\alpha}, d) = \underset{m_{\boldsymbol{\alpha},\boldsymbol{\theta}} \in M_{\boldsymbol{\alpha}}}{\arg\min}\ \mathcal{L}(m_{\boldsymbol{\alpha},\boldsymbol{\theta}}, d_{\text{train}}) + \mathcal{R}(\boldsymbol{\theta}) \ . \tag{1}$$

- m - machine learning model
- $\boldsymbol{\alpha}$ - neural architecture / hyperparameter configuration
- $\boldsymbol{\theta}$ - model parameters
- d - dataset

HPO/NAS Problem

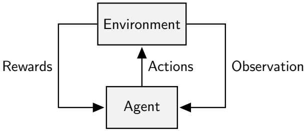
$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha} \in A}{\arg\max}\ \mathcal{O}(\Lambda(\boldsymbol{\alpha}, d_{\text{train}}), d_{\text{valid}}) = \underset{\boldsymbol{\alpha} \in A}{\arg\max}\, f(\boldsymbol{\alpha}) \ . \tag{2}$$

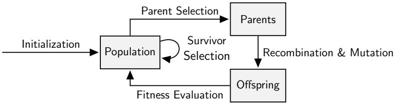
- f - response function

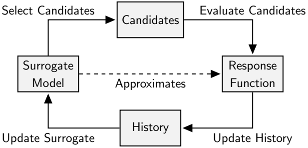
NAS Optimizers

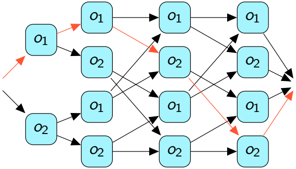
We distinguish several methods that maximize the response function:

- **Reinforcement learning**: learn to sample α that maximize f.
- **Evolutionary algorithms**: evolve α that maximize f.
- **Surrogate model-based optimization**: approximate f by $\hat{f}$ and use it to maximize f.
- **One-shot architecture search**: learn one model and use it to max f.





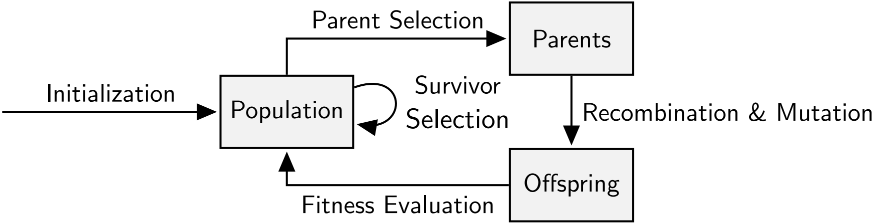




Outline

Evolutionary Algorithms for NAS

1. Select parents from the population for reproduction.
2. Apply recombination and mutation operations to create new individuals.
3. Evaluate the fitness of the new individuals.
4. Select the survivors of the population.



Parent/Survivor Selection

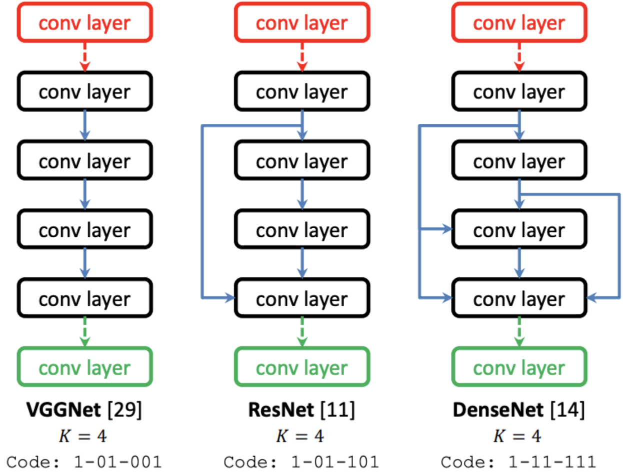
Fitness proportional selection: $p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j}$

Tournament selection:

- Selects k individuals from the population at random
- Selection pressure probability p
- Fittest model is returned with probability p, second fittest model with p(1-p) ... Nth fittest model with $p(1-p)^N$

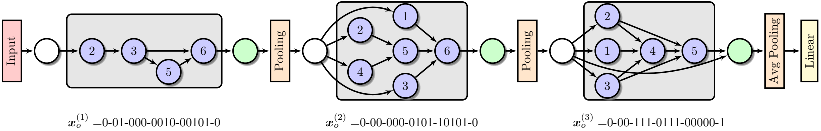
Selection of youngest individuals: Preference given to recently created models

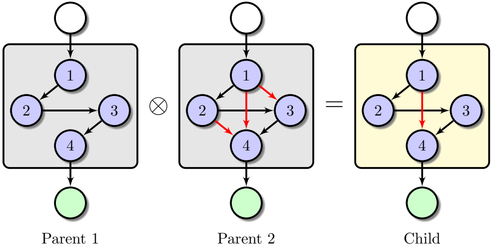
Architecture Encoding



Lingxi Xie and Alan L. Yuille. "Genetic CNN". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer

Recombination / Crossover





(Parent 1) VGG:	1-01-001-0
(Parent 2) DenseNet:	1-11-111-0
(Child) ResNet:	1-01-101-0

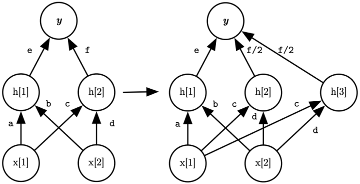
: common
: VGG
: DenseNet

Zhichao Lu et al. "NSGA-NET: A Multi-Objective Genetic Algorithm for Neural Architecture Search". In: *CoRR* abs/1810.03522 (2018)

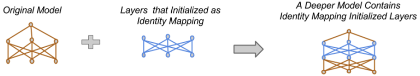
Mutation

- Add/remove layers
- Alter kernel size or number of filters/units
- Add/remove skip connections
- Change layer type

Function preserving Mutations



(a) Net2Wider



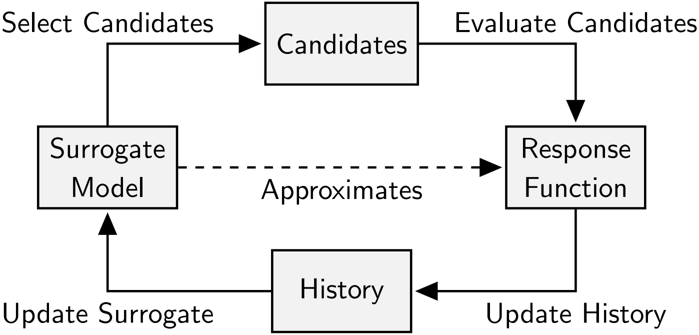
(b) Net2Deeper

Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. "Net2Net: Accelerating Learning via Knowledge Transfer". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016

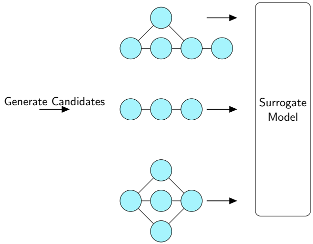
Outline

Surrogate Model-Based Optimization for NAS

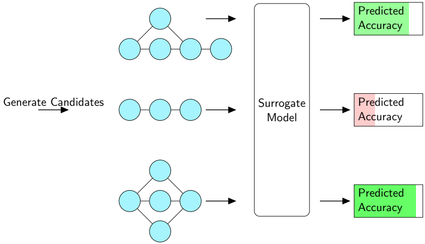
Surrogate model - Regression problem



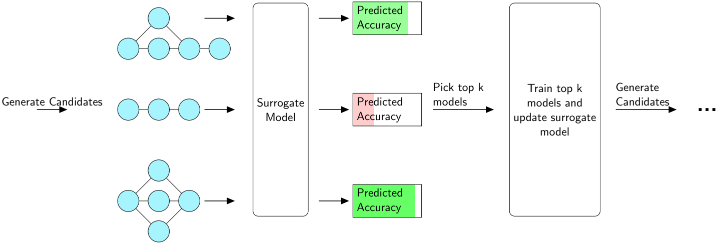
Surrogate Model-Based Optimization



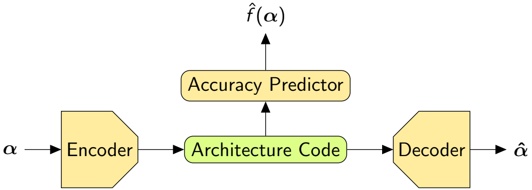
Surrogate Model-Based Optimization



Surrogate Model-Based Optimization



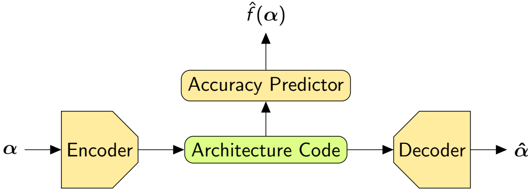
Neural Architecture Optimization



- Jointly learn an auto-encoder and surrogate model ($\hat{f}$)
- Auto-encoder produces the architecture code (h_t)
- Initialization: Trained on 600 random architectures and their validation accuracy
- Augmentation: Symmetric architectures

Renqian Luo et al. "Neural Architecture Optimization". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.* 2018, pp. 7827–7838

Neural Architecture Optimization



- Search: Sample new architectures by taking gradient steps to maximize predicted accuracy with respect to the architecture code.

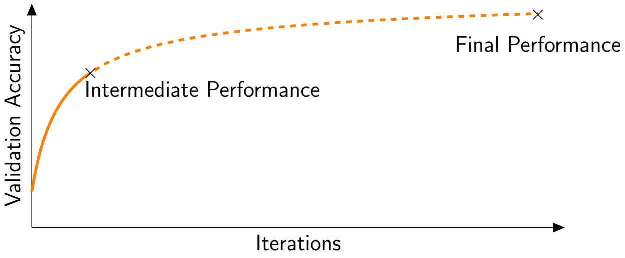
$$h_t' = h_t + \eta * \frac{\partial \hat{f}}{\partial h_t} \tag{3}$$

- Decode architecture codes and evaluate the corresponding architectures.
- Newly generated data will be taken into account to retrain the NAO model.

Outline

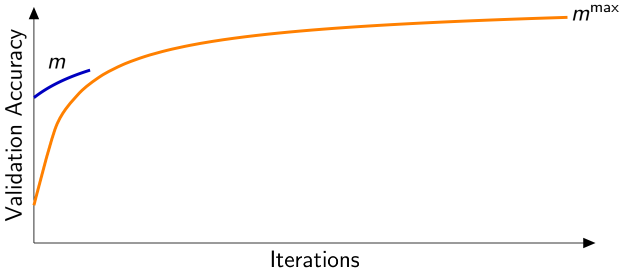
Motivation

- Hyperparameter and neural architecture optimization are computationally expensive.
- Humans monitor the model's learning curve and employ early stopping
- With the rise of AutoML, a system that is able to perform this automatically is desired.



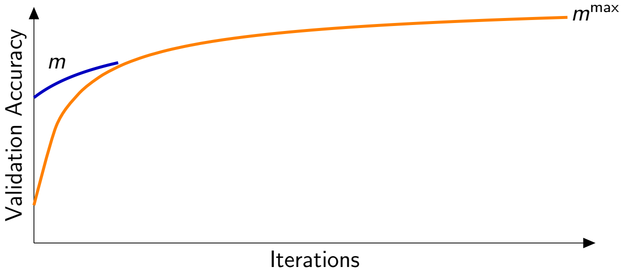
Use Simple Statistics

1. Use median/mean or last value in learning curve to make decision.



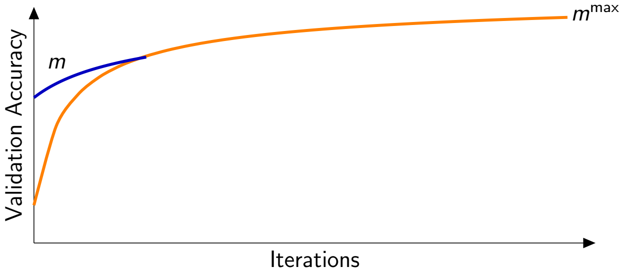
Use Simple Statistics

1. Use median/mean or last value in learning curve to make decision.



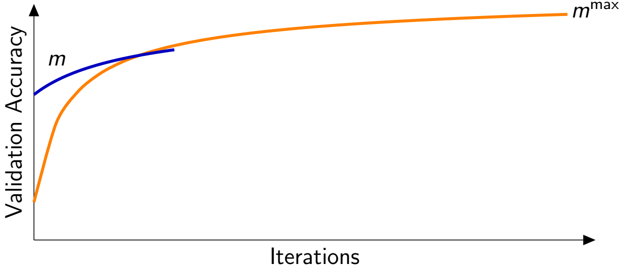
Use Simple Statistics

1. Use median/mean or last value in learning curve to make decision.



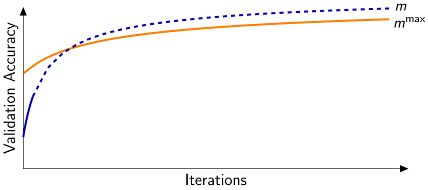
Use Simple Statistics

1. Use median/mean or last value in learning curve to make decision.



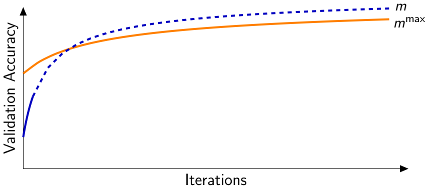
Simple Statistics - Problems

- Late bloomers will not be considered.

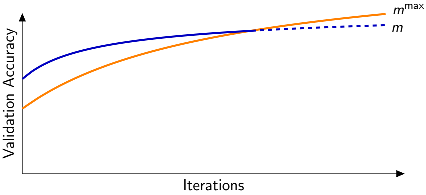


Simple Statistics - Problems

- Late bloomers will not be considered.



- Quick learners will be considered unnecessarily long.



Learning Curves Prediction

- Given a partial learning curve, predict the final performance.
- Use this prediction to estimate $p\left(m > m^{\max}\right)$.
- Terminate all runs with $p\left(m > m^{\max}\right) \leq \delta$

Learning Curves Ranking

- Proposing to predict $p\left(m > m^{\max}\right)$ directly.
- Defining the probability that m_i is better than m_j as

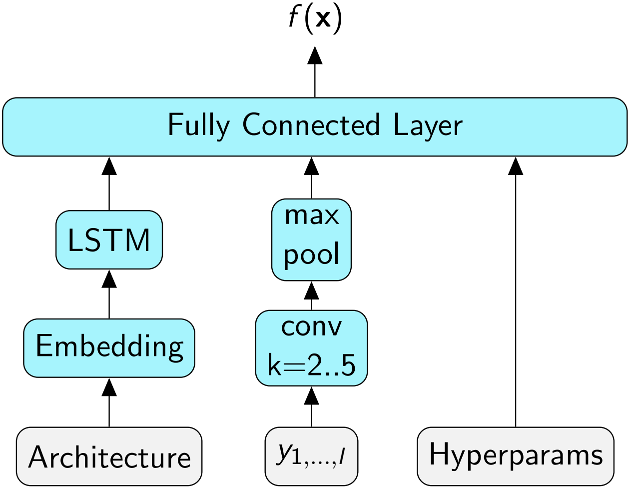
$$p(m_i > m_j) = \hat{p}_{i,j} = \frac{e^{f(\mathbf{x}_i)-f(\mathbf{x}_j)}}{1 + e^{f(\mathbf{x}_i)-f(\mathbf{x}_j)}}. \tag{4}$$

- Minimize the cross-entropy loss

$$\sum_{i,j} -p_{i,j} \log \hat{p}_{i,j} - (1 - p_{i,j}) \log(1 - \hat{p}_{i,j}) \tag{5}$$

Martin Wistuba and Tejaswini Pedapati. "Learning to Rank Learning Curves". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 12-18 June 2020, Vienna, Austria*. 2020

Modelling f


$f(\mathbf{x})$
Fully Connected Layer
LSTM
max pool
Embedding
conv k=2..5
Architecture
$y_{1,\ldots,l}$
Hyperparams

Learning Curves Ranking with Meta Learning

- Learning requires data which is not available.

Learning Curves Ranking with Meta Learning

- Learning requires data which is not available.
- Solution 1: Do not learn, only consider given partial learning curve.

Learning Curves Ranking with Meta Learning

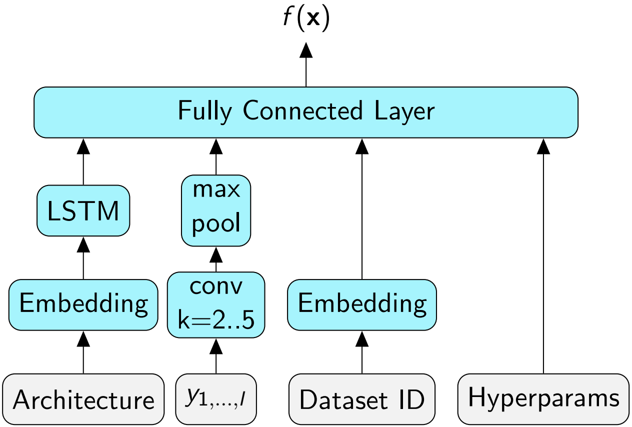
- Learning requires data which is not available.
- Solution 1: Do not learn, only consider given partial learning curve.
- Solution 2: First collect sufficient learning curves and then train your model.

Learning Curves Ranking with Meta Learning

- Learning requires data which is not available.
- Solution 1: Do not learn, only consider given partial learning curve.
- Solution 2: First collect sufficient learning curves and then train your model.
- Proposal: Use meta learning to reduce this problem.

Considering Meta Learning in our Modelling

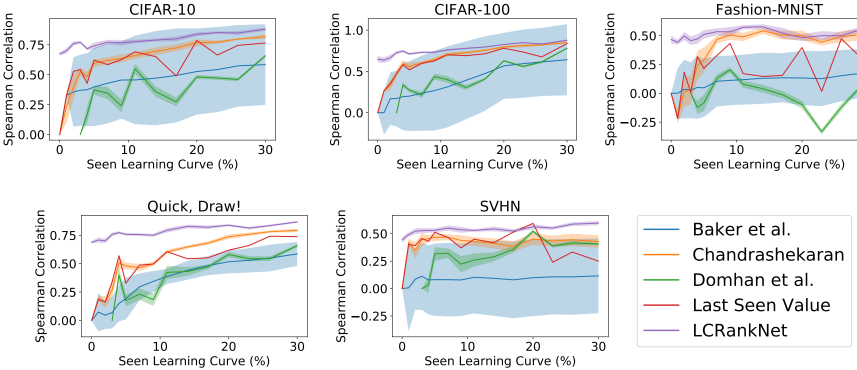
To account for meta learning, an embedding per dataset is added.



Setup

- Experiments are conducted on five different datasets: CIFAR-10, CIFAR-100, Fashion-MNIST, Quickdraw, and SVHN.
- To create the meta-knowledge, 200 architectures per dataset are choosen at random from the NASNet search space (i.e. 1000 unique architectures) and train it for 100 epochs.
- Experiments are conducted in a leave-one-dataset-out cross-validation.

Ranking Performance



Outline

Multi Objective

- Optimize multiple objectives

$$\max_{\boldsymbol{\alpha} \in A} f_1(\boldsymbol{\alpha}), f_2(\boldsymbol{\alpha}), \ldots, f_n(\boldsymbol{\alpha}) . \tag{6}$$

- Objectives such as model size, number of parameters, inference time
- No single optimal solution for MO
- Convert multi-objective to single objective by assigning weights to various objectives using weighted sum, weighted exponential sum or weighted product

$$\max_{\boldsymbol{\alpha} \in A} h\left(\left(f_1(\boldsymbol{\alpha}), f_2(\boldsymbol{\alpha}), \ldots, f_n(\boldsymbol{\alpha})\right), \boldsymbol{w}\right) . \tag{7}$$

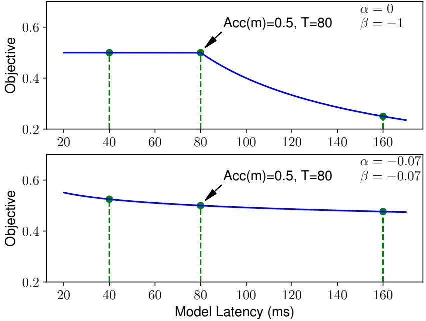
- Pick w according to domain knowledge

MNasNet

$$\underset{m}{\text{maximize}} \quad ACC(m) \times \left[\frac{LAT(m)}{T}\right]^{w} \tag{8}$$

where w is the weight factor defined as:

$$w = \begin{cases} \alpha, & \text{if } LAT(m) \leq T \\ \beta, & \text{otherwise} \end{cases} \tag{9}$$

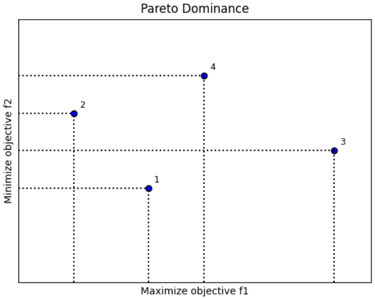


Mingxing Tan et al. "MnasNet: Platform-Aware Neural Architecture Search for Mobile". In: *CoRR* abs/1807.11626 (2018)

Outline

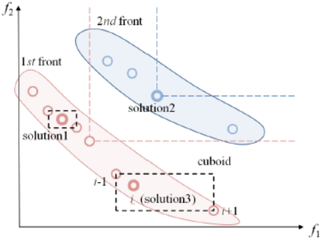
Pareto Front

- A solution $\alpha^{'}$ is said to dominate another solution α if $f_i(\alpha^{'}) \geq f_i(\alpha)$ for all objectives and $f_i(\alpha^{'}) > f_i(\alpha)$ for at least one objective
- A solution is Pareto Optimal if no other solution dominates it.
- All the non-dominated solutions are in the first Pareto front. Solutions in the i-th Pareto front are dominated by solutions in the 1st to i-1-th front



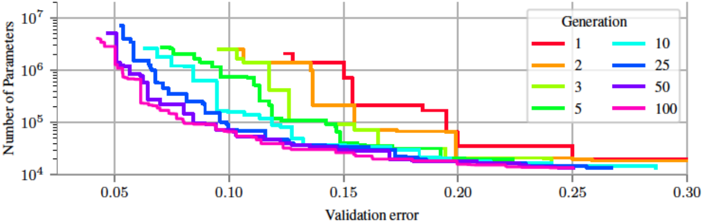
NSGA-II

- Elitist evolutionary algorithm
- Solutions are first ranked into fronts
- Solutions within a front are sorted according to inverse of their crowding distance – measure of the density of solutions in the given solution's neighbourhood.
- Enforces diverse solutions
- (Kim et al.) and (Lu et al.) employ NSGA-II



Pareto Front

Lemonade



Evolution of generations

- Two objectives: One is expensive to evaluate and the other is cheap to evaluate
- Sample candidates according to inverse of kernel density estimator wrt cheap objective (KDE)
- Evaluate candidates on expensive objective

Outline

Problem Definition

Machine Learning Problem

$$\Lambda(\boldsymbol{\alpha}, d) = \underset{m_{\boldsymbol{\alpha},\boldsymbol{\theta}} \in M_{\boldsymbol{\alpha}}}{\arg\min}\ \mathcal{L}(m_{\boldsymbol{\alpha},\boldsymbol{\theta}}, d_{\text{train}}) + \mathcal{R}(\boldsymbol{\theta}) \ . \tag{10}$$

- m - machine learning model
- $\boldsymbol{\alpha}$ - neural architecture / hyperparameter configuration
- $\boldsymbol{\theta}$ - model parameters
- d - dataset

HPO/NAS Problem

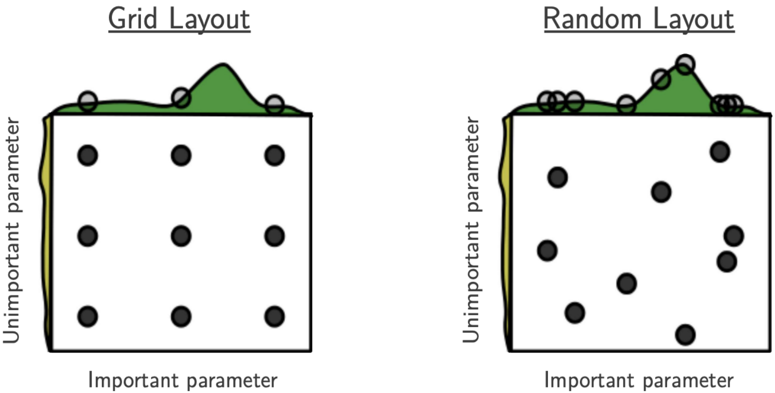
$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha} \in A}{\arg\max}\ \mathcal{O}(\Lambda(\boldsymbol{\alpha}, d_{\text{train}}), d_{\text{valid}}) = \underset{\boldsymbol{\alpha} \in A}{\arg\max}\, f(\boldsymbol{\alpha}) \ . \tag{11}$$

- f - response function

Outline

Grid search and Random search

- Grid search : Explicitly enumerate which values to search for
- Random search: Randomly samples, strong baseline



James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13 (2012), pp. 281–305

Outline

Hyperband

	$s=4$		$s=3$		$s=2$		$s=1$		$s=0$	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

- More resources to promising configurations
- Terminate poor configurations early
- *R*: Total number of Resources
- η: Proportion of configurations that can be discarded

Lisha Li et al. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *Journal of Machine Learning Research* 18 (2017), 185:1–185:52

Hyperband Algorithm

Algorithm 1 Hyperband

Input: R, η (default $\eta = 3$)
  Init: $s_{\max} = \lfloor \log_{\eta}(R) \rfloor$, $B = (s_{\max} + 1)R$
  for $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ **do**
    $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil, \qquad r = R\eta^{-s}$
    $\mathrm{T} = get\_hyperparameter\_configuration(n)$
    for $i \in \{0, \ldots, s\}$ **do**
      $n_i = \lfloor n\eta^{-i} \rfloor$
      $r_i = r\eta^i$
      $L = \{\texttt{run\_then\_return\_val\_loss}(t, r_i) : t \in T\}$
      $T = \texttt{top\_k}(T, L, \lfloor n_i/\eta \rfloor)$

Drawbacks: Random Search, Late Bloomers

Outline

Introduction

- Evaluating each configuration is expensive
- Builds a surrogate model to mimic the response function f
- Intelligently samples the next datapoint
- Black box optimization
- Gives an uncertainty estimate

Gaussian Processes

- BO uses Gaussian Processes
- Gaussian distribution $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$
- A Gaussian processes (GP) model describes a probability distribution over possible functions that fit a set of points.
- In a GP, the joint distribution for any finite collection of inputs x_1 .. x_N is a multivariate Gaussian distribution

$$\begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{pmatrix} \sim N \left[\begin{pmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_N) \end{pmatrix}, \begin{pmatrix} \sigma(x_1, x_1) & \sigma(x_1, x_2) & \dots & \sigma(x_1, x_N) \\ \sigma(x_2, x_1) & \sigma(x_2, x_2) & \dots & \sigma(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ \sigma(x_N, x_1) & \sigma(x_N, x_2) & \dots & \sigma(x_N, x_N) \end{pmatrix} \right]$$

GP cont.

Covariance matrix

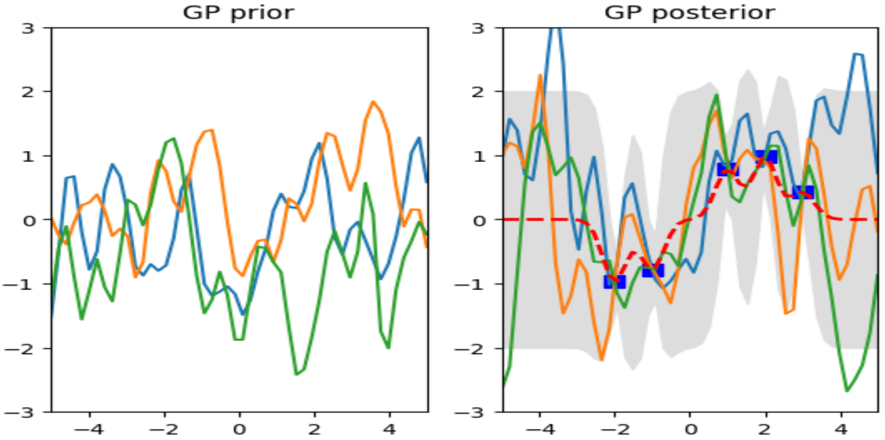
- Similarity between two points
- Modeled using Kernels K(x,x*)
- Some kernel functions are: linear, periodic, RBF, Squared exponential

Posterior:

- Bayes theorem $p(f'|D) = \frac{p(D|f')p(f')}{p(D)}$
- for new observed data x*, updated mean and covariance:

$$\mu_* = k_*^T (K + \sigma^2 I)^{-1} y$$
$$\Sigma_* = k_{**} - k_*^T (K + \sigma^2 I)^{-1} k_x$$

Prior and Posterior



code for this plot:
https://www.cs.ubc.ca/∼nando/540-2013/lectures/gp.py

Baysian Optimization

Algorithm 2 Bayesian Optimization

Input: initial observations D_N, Gaussian process prior on f', acqFn
  for i in N+1 ... K **do**
    Identify $x_i = argmax_x$ acqFN(x|$D_{1:i-1}$)
    $y_i = f(x_i)$
    $D_{1:i} = D_{1:i-1}, (x_i, y_i)$
    Update posterior, μ_*, σ_* on D
  return Either $argmax_x$ $D_{1:k}$ or $argmax_x$ μ_*

Some Acquisition functions

Probability of Improvement(PI):

- P(f(x) $> f^{+} \epsilon|D)$
- Might get stuck in local optima

Excepted Improvement(EI):

- Considers the magnitude of improvement
- Most commonly used

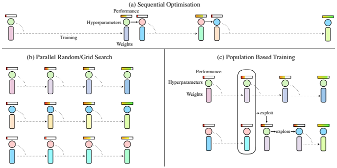
Upper Confidence Bound(UCB):

- $UCB = \mu(x) + \alpha\sigma(x)$
- Exploration vs exploitation

Outline

Population Based Training

Explore: Perturbing hyperparameters of the current model
Exploit: Terminate if performing poorly and update weights and hyperparameters of a more performant model



Max Jaderberg et al. "Population Based Training of Neural Networks". In: *CoRR* abs/1711.09846 (2017)

Conclusion

- EA and Surrogate model based NAS
- Learning curve ranking
- Multi-objective NAS
- Various HPO algorithms
- Code in RAY

Thank you for your attention.

Survey Paper: https://arxiv.org/abs/1905.01392

References I

[1] Bowen Baker et al. "Designing Neural Network Architectures using Reinforcement Learning". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

[2] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13 (2012), pp. 281–305.

[3] Simone Bianco et al. "Benchmark Analysis of Representative Deep Neural Network Architectures". In: *IEEE Access* 6 (2018), pp. 64270–64277.

References II

[4] Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. “Net2Net: Accelerating Learning via Knowledge Transfer”. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016.

[5] Max Jaderberg et al. “Population Based Training of Neural Networks”. In: *CoRR* abs/1711.09846 (2017).

[6] Ye-Hoon Kim et al. “NEMO : Neuro-Evolution with Multiobjective Optimization of Deep Neural Network for Speed and Accuracy”. In: *AutoML Workshop at ICML 2017*. 2017.

[7] Lisha Li et al. “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization”. In: *Journal of Machine Learning Research* 18 (2017), 185:1–185:52.

References III

[8] Zhichao Lu et al. "NSGA-NET: A Multi-Objective Genetic Algorithm for Neural Architecture Search". In: *CoRR* abs/1810.03522 (2018).

[9] Renqian Luo et al. "Neural Architecture Optimization". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.* 2018, pp. 7827–7838.

[10] Christian Szegedy et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* 2017, pp. 4278–4284.

[11] Mingxing Tan et al. "MnasNet: Platform-Aware Neural Architecture Search for Mobile". In: *CoRR* abs/1807.11626 (2018).

References IV

[12] Martin Wistuba and Tejaswini Pedapati. "Learning to Rank Learning Curves". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 12-18 June 2020, Vienna, Austria*. 2020.

[13] Lingxi Xie and Alan L. Yuille. "Genetic CNN". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 1388–1397.

[14] Barret Zoph and Quoc V. Le. "Neural Architecture Search with Reinforcement Learning". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.