Elements of Machine Learning & Data Science

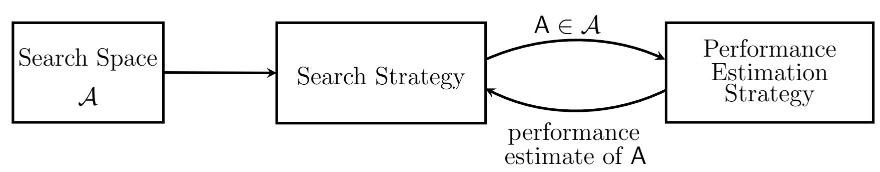
Winter semester 2023/24

Automated Machine Learning (3)

Maria Anastacio & Holger Hoos

Key concepts covered last class:

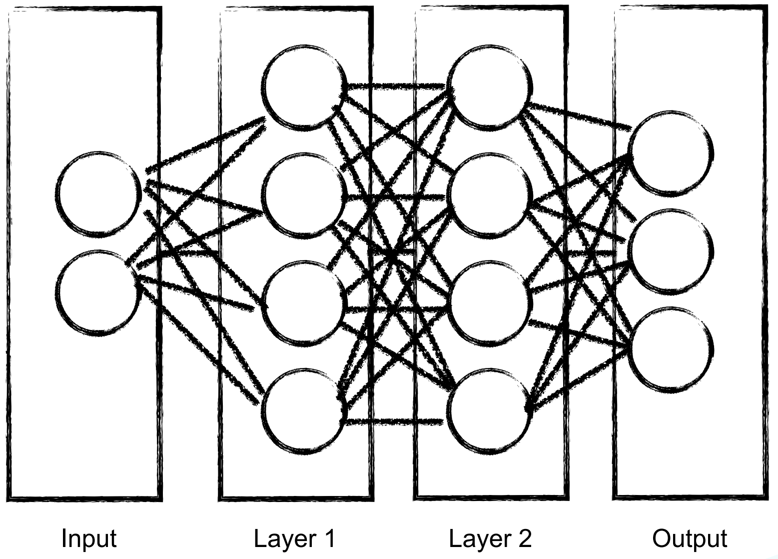
- Hyper Parameter Optimisation
- Random / Grid
- Bayesian Optimisation
- Multi Fidelity Bandit Approach



Source: Elsken et al. (2019)

brief recap

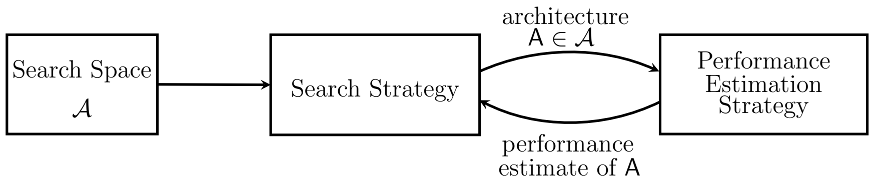
A quick reminder of Neural Networks:



Reminder

Key questions:

- How to describe the architecture of a Neural Network?
- What would be / how to find a better architecture?
- How to cheaply estimate the performance of a network?



Source: Elsken et al. (2019)

In this case our search space is A

Preparation for today:

Remind yourself of your lecture regarding Neural Networks.
Read the paper :
"Towards Automatically-Tuned Neural Networks", *Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, Frank Hutter*
Proceedings of the Workshop on Automatic Machine Learning, PMLR 64:58-65, 2016.
(https://proceedings.mlr.press/v64/mendoza_towards_2016.html)

Focus in particular on the following questions

- In table 1, which hyperparameters correspond to the ones optimised by HPO as seen in the previous class? Which ones correspond to the architecture (the structure) of the network?

What's the fundamental difference?

- Why did they limit the number of layer to 6 at most?

- Which main challenge can you see when searching the architecture of a network?

- Do you think that their search space covers all possible neural networks? If no, what is missing?

TPS Exercise (T part = done as homework)

1. In table 1, which hyperparameters correspond to the ones optimised by HPO as seen in the previous class? Which ones correspond to the architecture (the structure) of the network?
What's the fundamental difference?

2. Do you think that their search space covers all possible neural networks? If no, what is missing?

3. Which main challenge can you see when searching the architecture of a network?

2min pair, 4min share = 6min

In table 1, which hyperparameters correspond to the ones optimised by HPO as seen in the previous class? Which ones correspond to the architecture (the structure) of the network?

	Name	Range	Default	log scale	Type	Conditional
Network hyperparameters	batch size	$[32, 4096]$	32	✓	float	-
	number of updates	$[50, 2500]$	200	✓	int	-
	number of layers	$[1, 6]$	1	-	int	-
	learning rate	$[10^{-6}, 1.0]$	10^{-2}	✓	float	-
	L_2 regularization	$[10^{-7}, 10^{-2}]$	10^{-4}	✓	float	-
	dropout output layer	$[0.0, 0.99]$	0.5	✓	float	-
	solver type	{SGD, Momentum, Adam, Adadelta, Adagrad, smorm, Nesterov }	smorm3s	-	cat	-
	lr-policy	{Fixed, Inv, Exp, Step}	fixed	-	cat	-
Conditioned on solver type	β_1	$[10^{-4}, 10^{-1}]$	10^{-1}	✓	float	✓
	β_2	$[10^{-4}, 10^{-1}]$	10^{-1}	✓	float	✓
	ρ	$[0.05, 0.99]$	0.95	✓	float	✓
	momentum	$[0.3, 0.999]$	0.9	✓	float	✓
Conditioned on lr-policy	γ	$[10^{-3}, 10^{-1}]$	10^{-2}	✓	float	✓
	k	$[0.0, 1.0]$	0.5	-	float	✓
	s	$[2, 20]$	2	-	int	✓
Per-layer hyperparameters	activation-type	{Sigmoid, TanH, ScaledTanH, ELU, ReLU, Leaky, Linear}	ReLU	-	cat	✓
	number of units	$[64, 4096]$	128	✓	int	✓
	dropout in layer	$[0.0, 0.99]$	0.5	-	float	✓
	weight initialization	{Constant, Normal, Uniform, Glorot-Uniform, Glorot-Normal, He-Normal, He-Uniform, Orthogonal, Sparse}	He-Normal	-	cat	✓
	std. normal init.	$[10^{-7}, 0.1]$	0.0005	-	float	✓
	leakiness	$[0.01, 0.99]$	$\frac{1}{3}$	-	float	✓
	tanh scale in	$[0.5, 1.0]$	$2/3$	-	float	✓
	tanh scale out	$[1.1, 3.0]$	1.7159	✓	float	✓

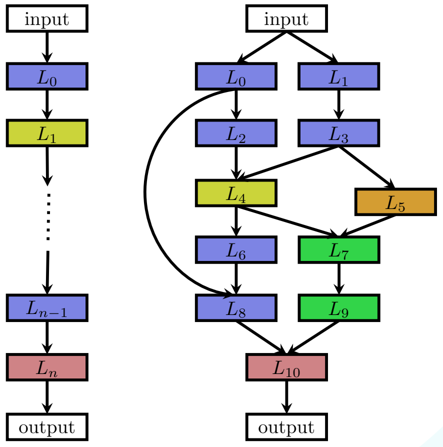
Source: Mendoza et al. (2016)

Share
Point out conditional parameters
Point out the number of layer

Does it cover all possible neural networks? If no, what is missing?

1. Number of layers
2. Number of unit per layer
3. Operation performed by each layer
4. Parameters of the operations

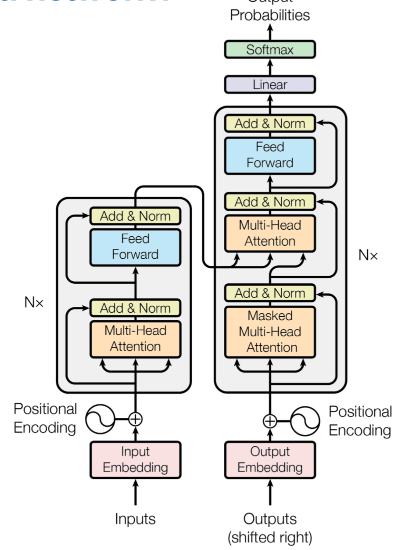
5. Input(s) of each layer



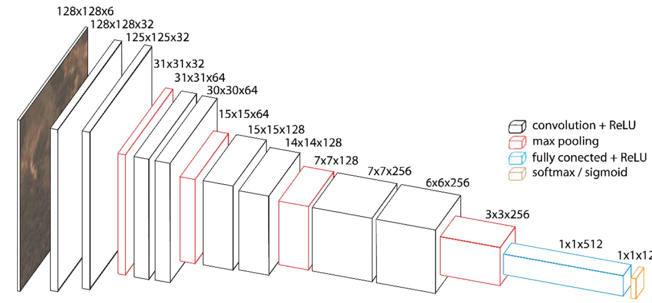
Source: Elsken et al. (2019)

Suming up the search space
Do you know of more complex architectures?
Can you imagine an easy way to have more complex architectures?

Which main challenge can you see when searching the architecture of a network?



Source: Vaswani et al. (2017)

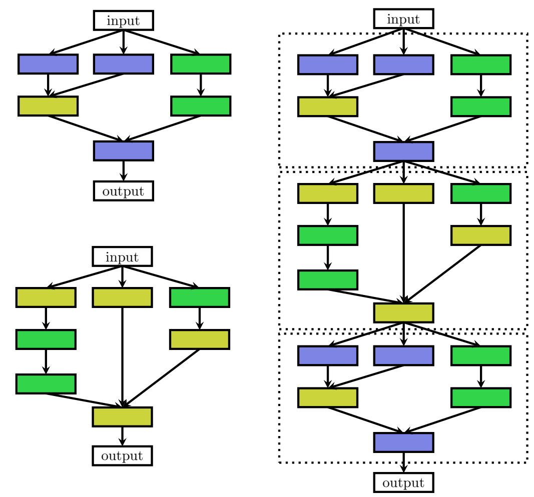


Source: Shendryk et al. (2019)

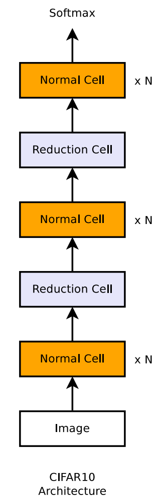
Share challenges
Huge search space
In practice typically 15-20 hidden layers for image recognition, up to hundreds
Example of architectures
Left, transformer, used by LLM
Right CNN, used for image recognition

Focus on CNN, what do you notice?
2 black, one red, 2 convolution, one reduction

Carefully crafting a search space



Source: Elsken et al. (2019)



Source: Zoph et al. (2018)

That's the idea behind the next type of search space
Cells
Optimise cells put them together

Limitation : you need to know the high level structure of the network

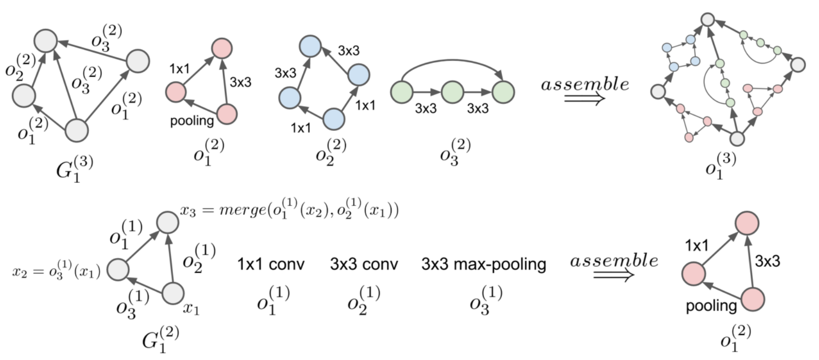
TPS Exercise

You have your 2 possible search spaces:
- separate layers with their hyper parameters
- cells of layers inside a given arrangement

How could you combine them to be more expressive than each separately?

2min think, 2min pair, 3min share = 7min

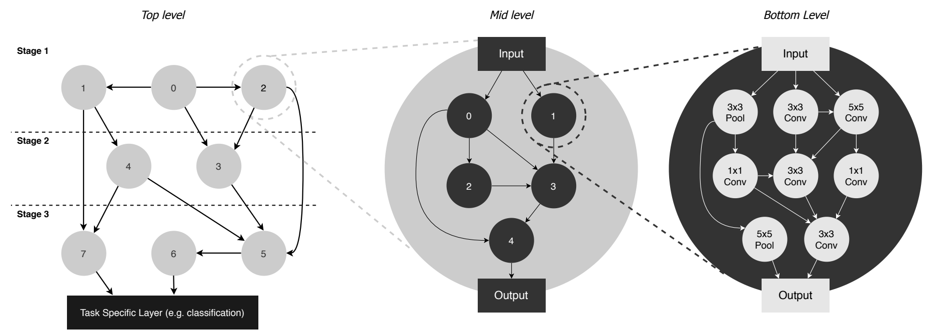
How could you combine them to be more expressive than each separately?



Source: Liu et al. (2018)

Share
Hierarchical search space
From small sets of operations, to cells, to network

How could you combine them to be more expressive than each separately?



Source: Ru et al. (2020)

Other example, similar idea

TPS Exercise

Which strategies could you use to optimise your architecture within those search spaces?

- Random
- Bayesian optimisation
- Gradient descent
- Evolutionary algorithm
- Monte Carlo tree search
- Reinforcement Learning

1min think, 2min pair, 2min share = 5min

TPS Exercise

Which strategies could you use to optimise your architecture within those search spaces?

- Random
- Bayesian optimisation
- Gradient descent
- Evolutionary algorithm
- Monte Carlo tree search
- Reinforcement Learning

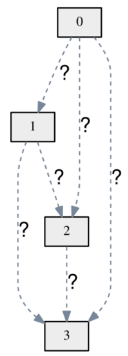
TPS Exercise

Which strategies could you use to optimise your architecture within those search spaces?

- Random
- Bayesian optimisation
- Gradient descent
- Evolutionary algorithm
- Monte Carlo tree search
- Reinforcement Learning

Which strategies could you use to optimise your architecture within those search spaces?

Random



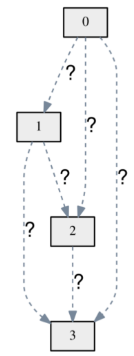
(a)

- Sample uniformly at random
- Keep the best found architecture

Random, actually working pretty well

Which strategies could you use to optimise your architecture within those search spaces?

Bayesian optimisation



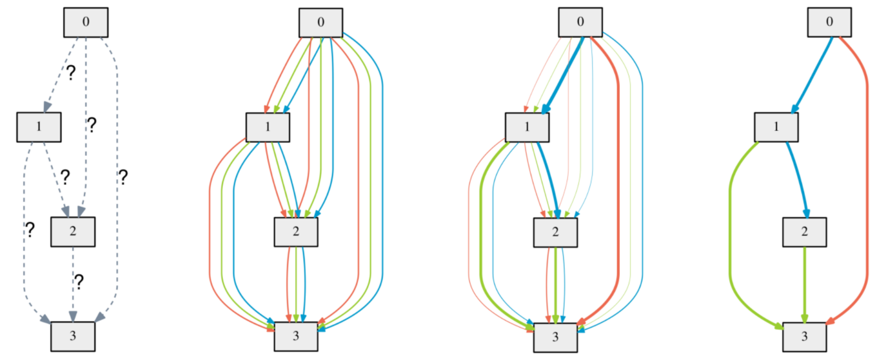
(a)

- Sample uniformly at random
- Learn a surrogate model to know which works best
- Sample more using Bayesian optimisation

Bayesian optimisation, BOHB based on it implemented in come of current autoML systems

Which strategies could you use to optimise your architecture within those search spaces?

Gradient descent



(a) (b) (c) (d)

Source: Liu et al. (2019)

Gradient Descent, DARTS, optimise the type of operation at the same time as the weights.

TPS Exercise

Training every architecture during the search process would be very costly.

How to lower the cost of deciding which generated architecture to keep?

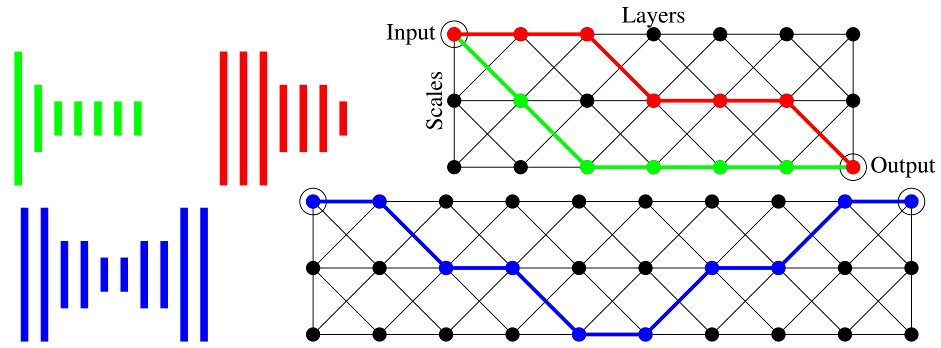
2min think, 2min pair, 3min share = 7min

How could you lower the cost of deciding which generated architecture to keep?

- Low number of Epoch
- Share weights between similar networks (Weight Sharing)
- Optimise architecture and weights together (DARTS)
- Train a surrogate model
- Train one super network (One-shot model)

- low number of epoch, can use learning curves for prediction

How could you lower the cost of deciding which generated architecture to keep?

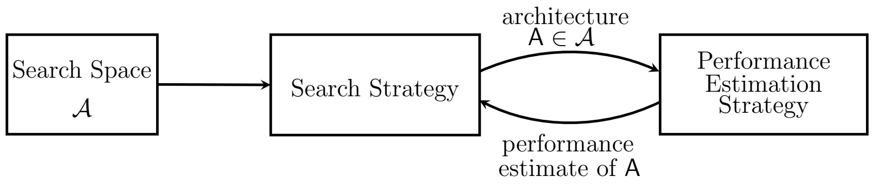


Source: Saxena and Verbeek (2016)

One Shot Model
Train a super network, give you an idea of the performance of a subnetwork.
Estimate not super good but still it works

Key concepts covered today:

- Search space : macro, cell-based, hierarchical
- Search strategies : random, Bayesian optimisation, gradient descent
- Performance estimation strategies : partial training, surrogate model, one-shot model



Source: Elsken et al. (2019)

Reference for figures:

Mendoza et al. (2016)	Towards Automatically-Tuned Neural Networks
Saxena and Verbeek (2016)	Convolutional Neural Fabrics
Vaswani et al. (2017)	Vaswani et al. (2017)
Liu et al. (2018)	Hierarchical Representations for Efficient Architecture Search
Zoph et al. (2018)	Learning Transferable Architectures for Scalable Image Recognition
Elsken et al. (2019)	Neural Architecture Search: A Survey
Liu et al. (2019)	DARTS: Differentiable Architecture Search
Shendryk et al. (2019)	Deep learning for multi-modal classification of cloud, shadow and land cover scenes in PlanetScope and Sentinel-2 imagery
Ru et al. (2020)	Neural Architecture Generator Optimization

If you want to go further:

In depth explanation of DARTS	towardsdatascience.com
Python libaries	Auto-PyTorch, AutoKeras