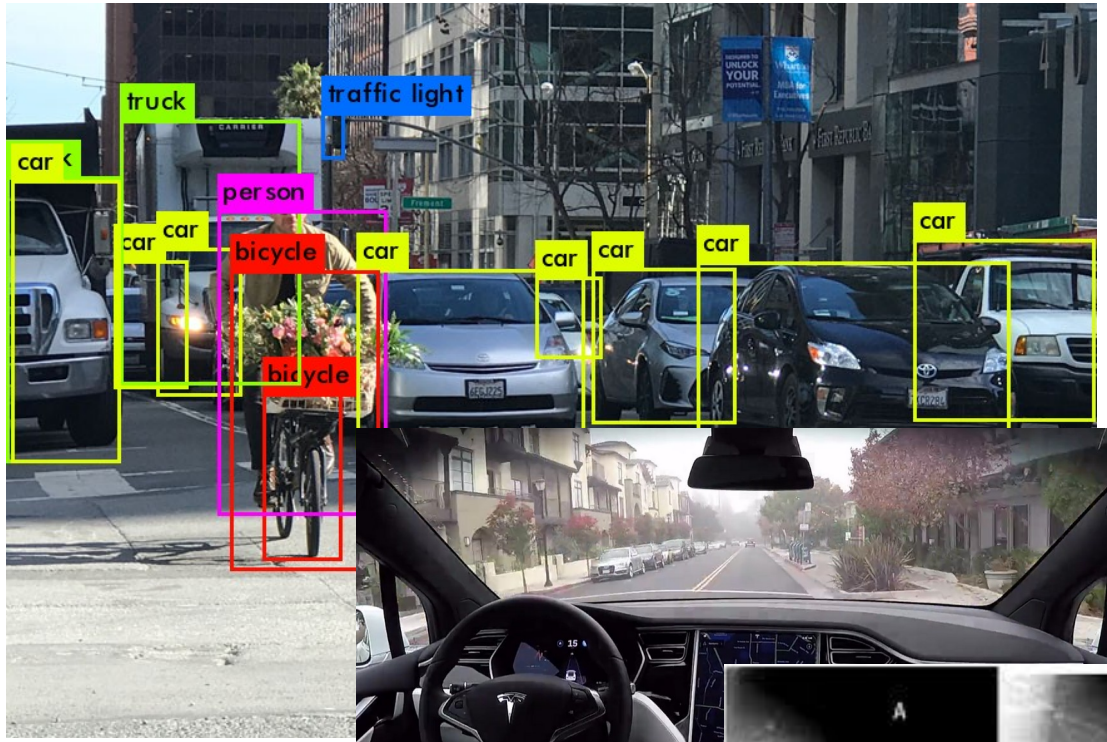
CNNGen: Beyond Accuracy - A Generator and Benchmark for Energy-Aware Neural Architecture Search

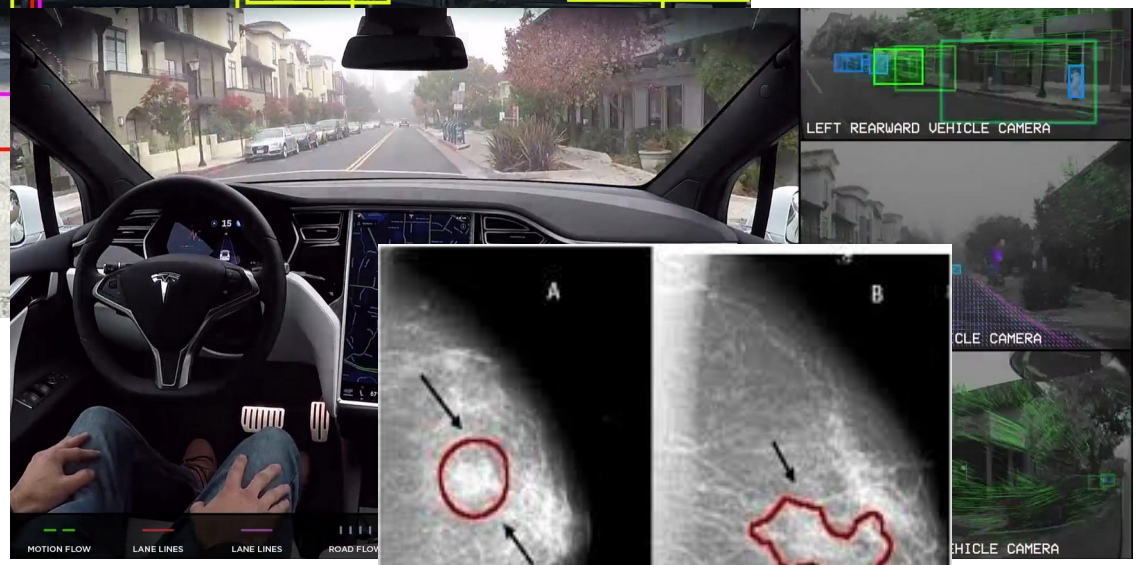
Antoine Gratia

DiverSE Coffee

Myself

- Antoine GRATIA
- PhD student
- University of Namur Belgium
- Supervisors
  - Gilles Perrouin (Unamur)
  - Pierre-Yves Schobbens (Unamur)


truck
traffic light
car
person
car
car
bicycle
car
car
car
car
car
bicycle

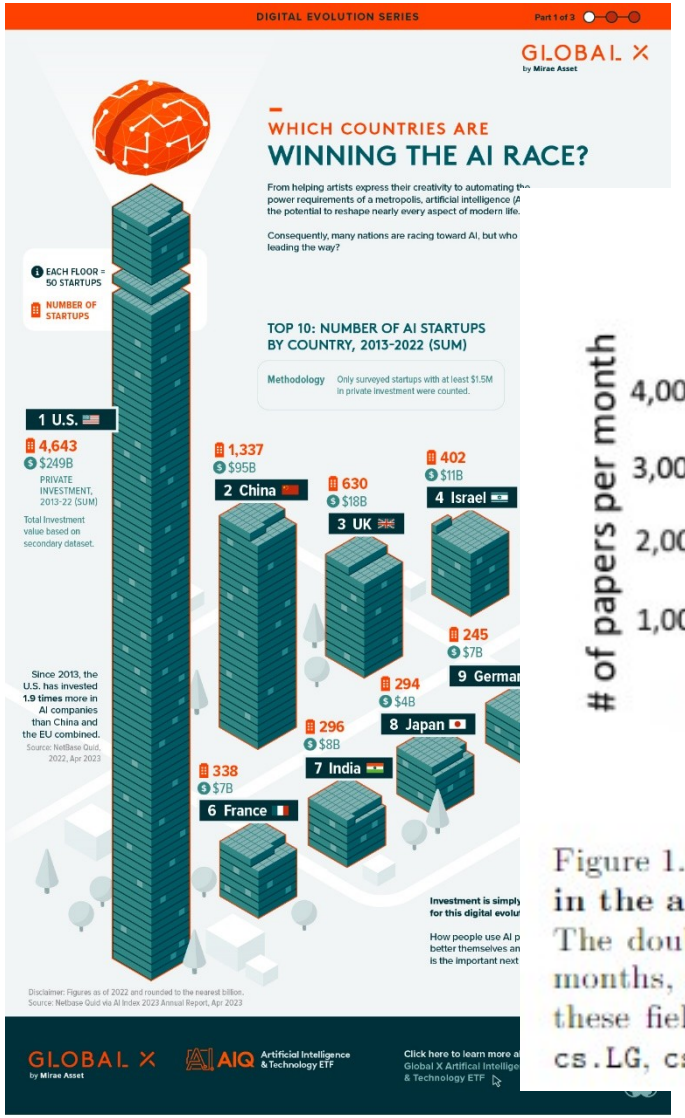

LEFT REARWARD VEHICLE CAMERA
MOTION FLOW
LANE LINES
LANE LINES
ROAD FLOW

A
B


"Hey, Si
Say "Ok Google",
then ask.

Context

- AI Race for Performance



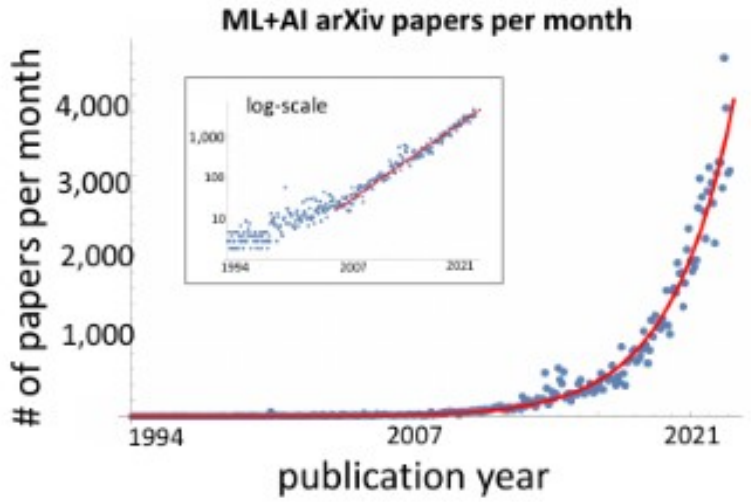
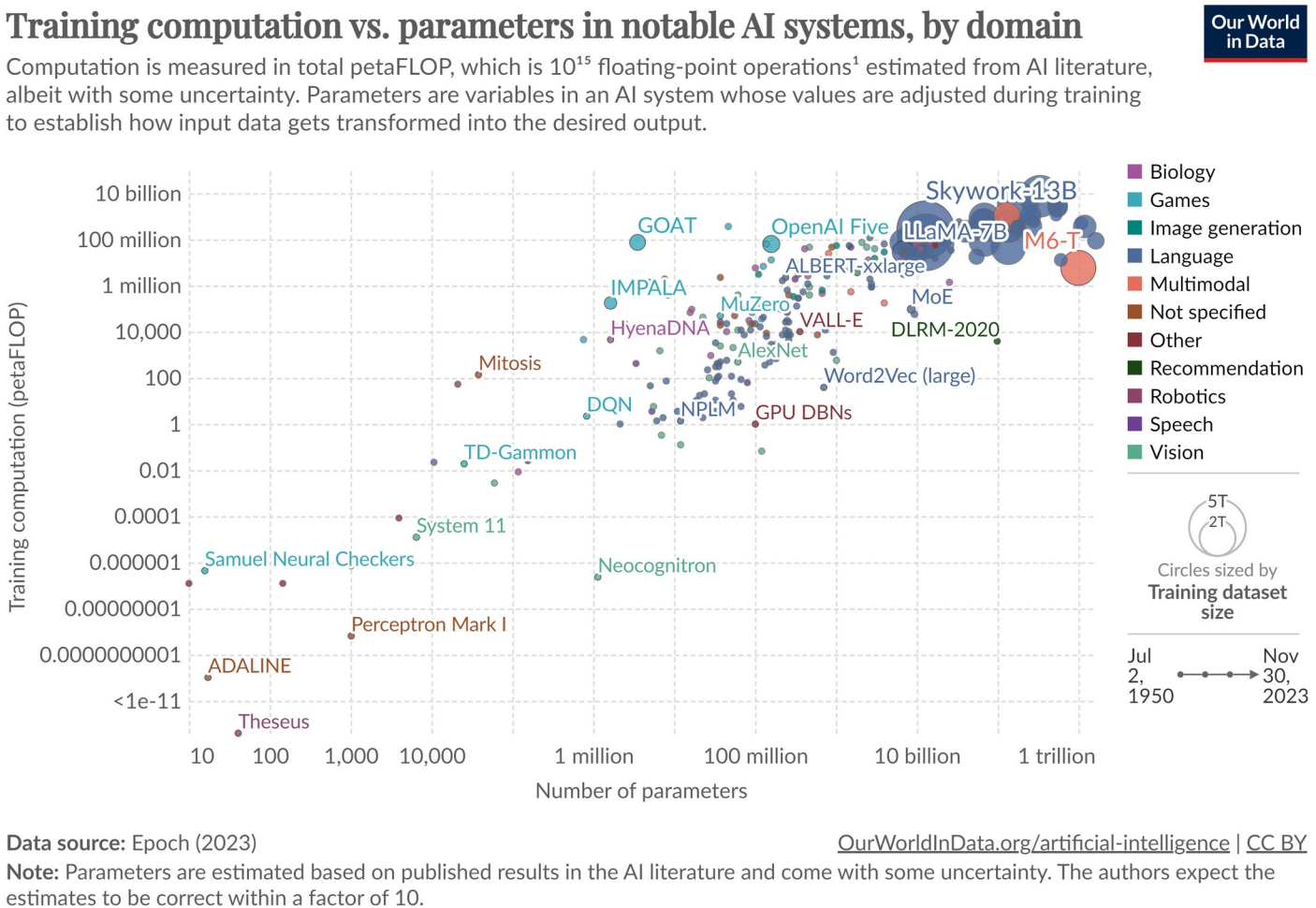


Figure 1. **Number of papers published per months in the arXiv categories of AI grow exponentially.** The doubling rate of papers per months is roughly 23 months, which might lead to problems for publishing in these fields, at some point. The categories are `cs.AI`, `cs.LG`, `cs.NE`, and `stat.ML`.

Energy

- LLaMA-65B cost 173 tCO2eq
- "AI me to the moon… Carbon footprint for training GTP-3 same as driving to our natural satellite and back"



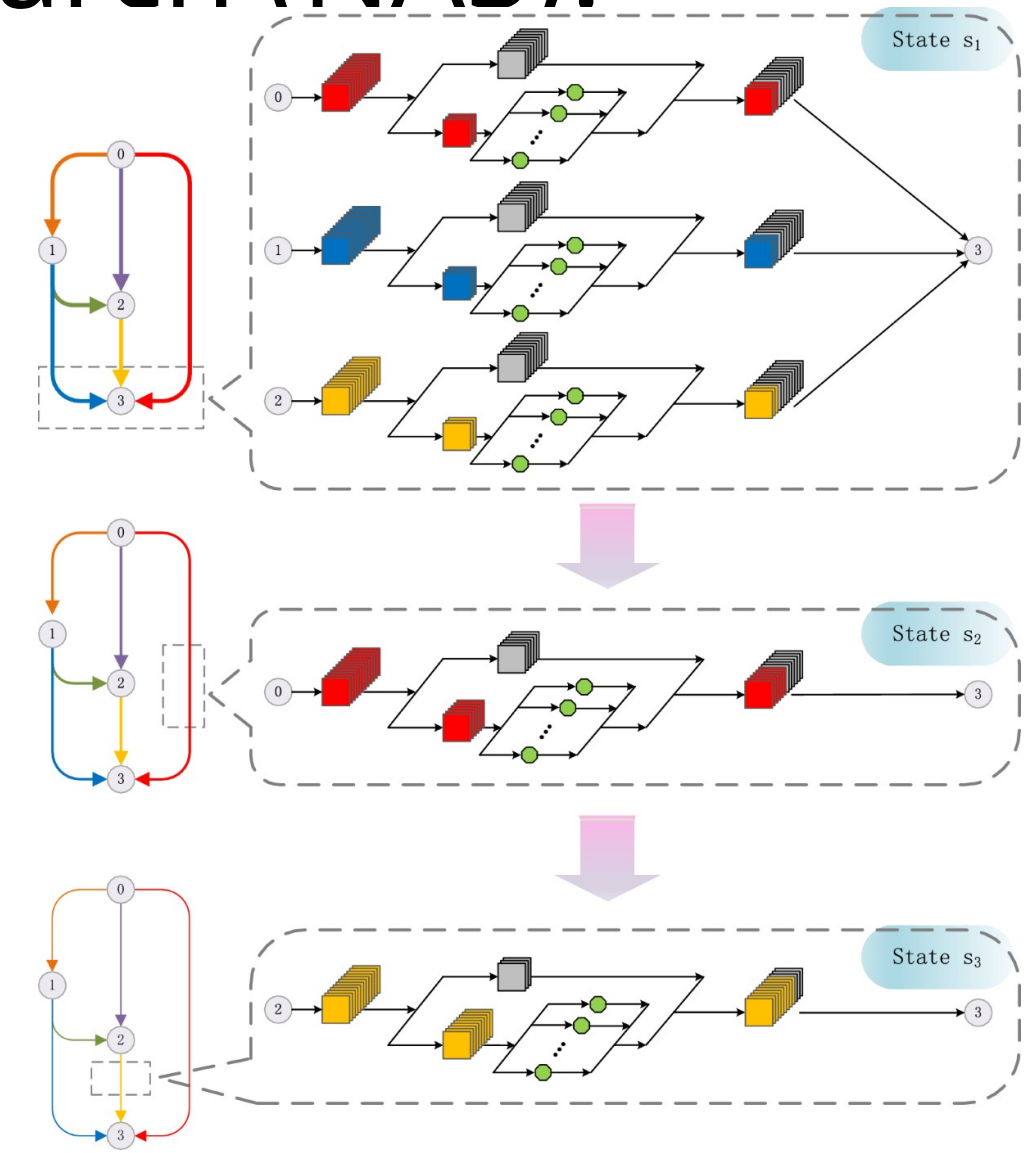
Objectives

Select architectures balancing performance & energy cost

- Model the search space
- Explore it (sampling)
- Energy cost and Performance Optimisation

Neural Architecture Search (NAS)?

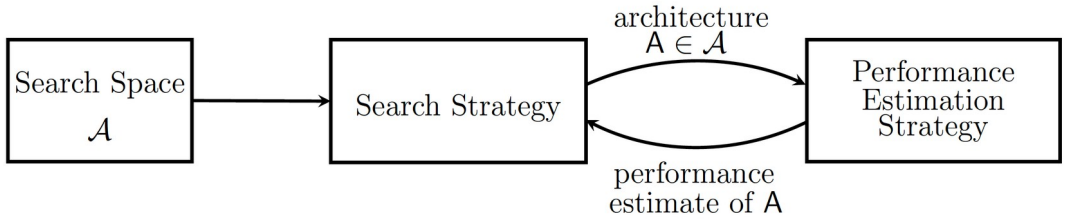
- Design neural network architectures automatically
- Find the optimal for a given task
- Exploring a predefined space (evolutionary algorithms, gradient-based optimisation,...)



NAS - General Worflow

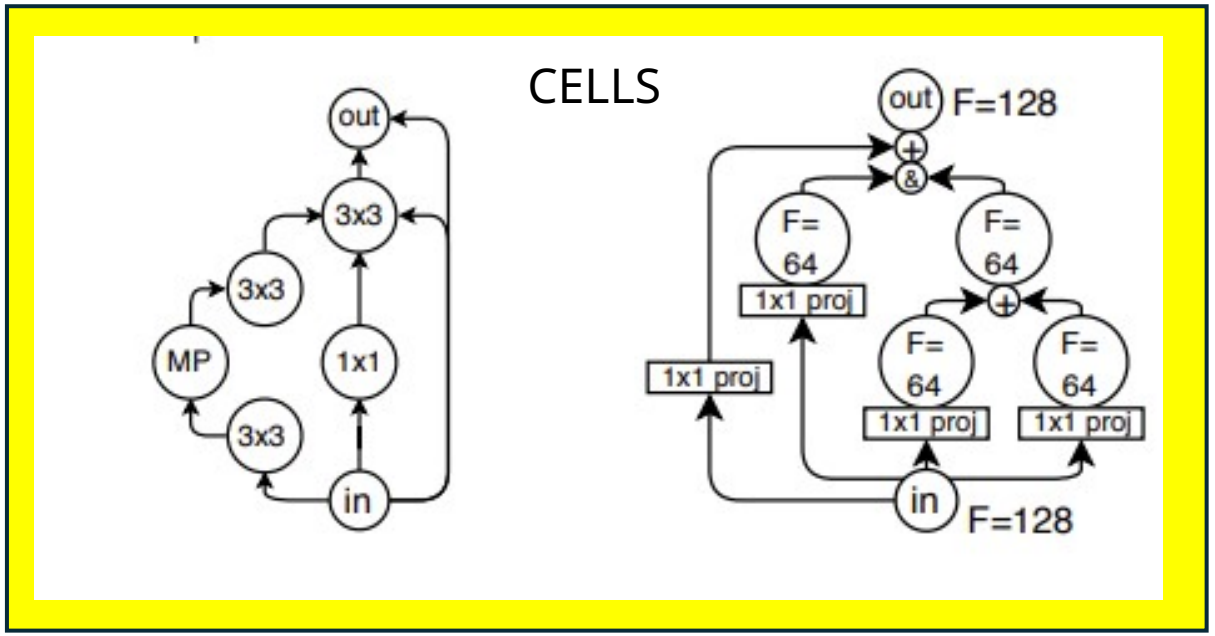
- Explores a search space
- Evaluates each architecture (train, prediction, ...)
- Estimates performance metric (accuracy, efficiency,...)

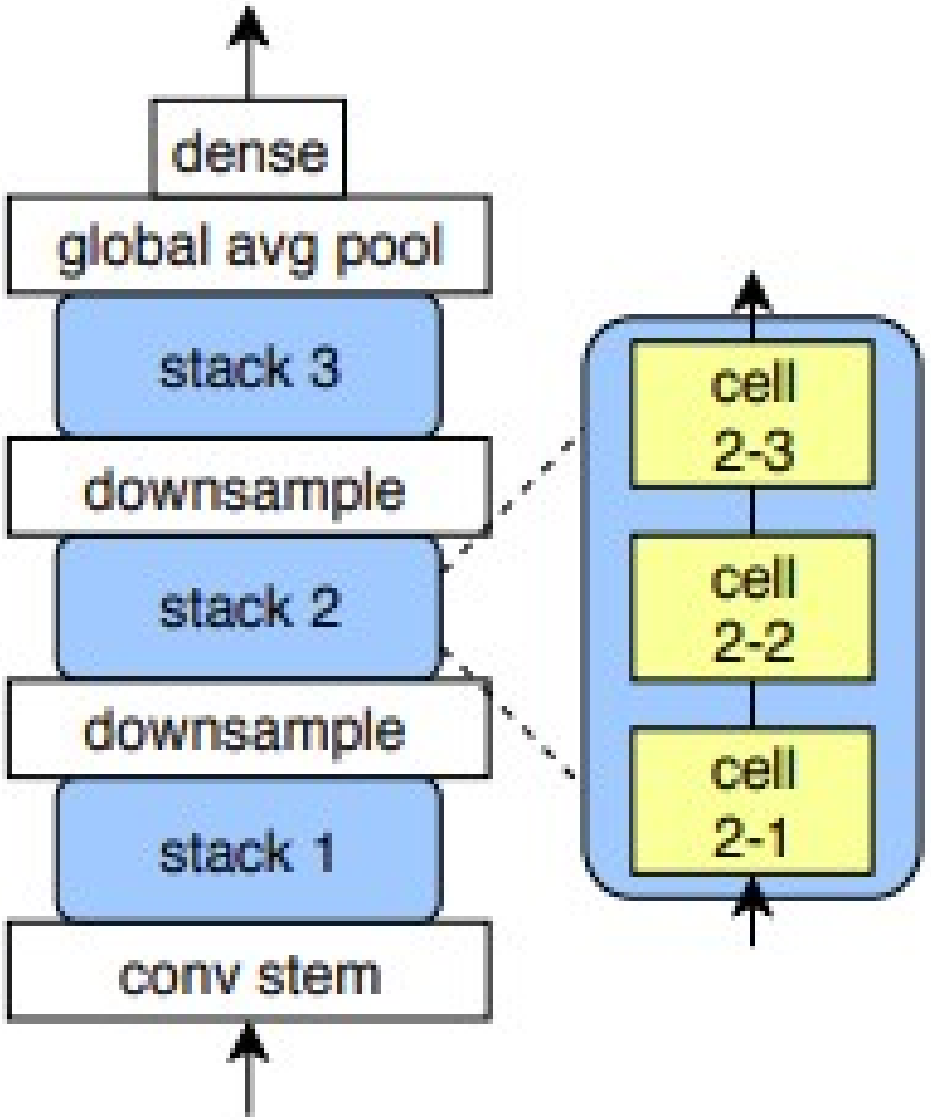
Goal: Find the optimal architecture for a given task



NAS – Example : NASBench-101

- Espace predifined
- 423K unique architectures
- Max 108 epochs





State Of The Art

Auto-Keras: An Efficient Neural Architecture Search System (Haifeng Jin - 06/2018)

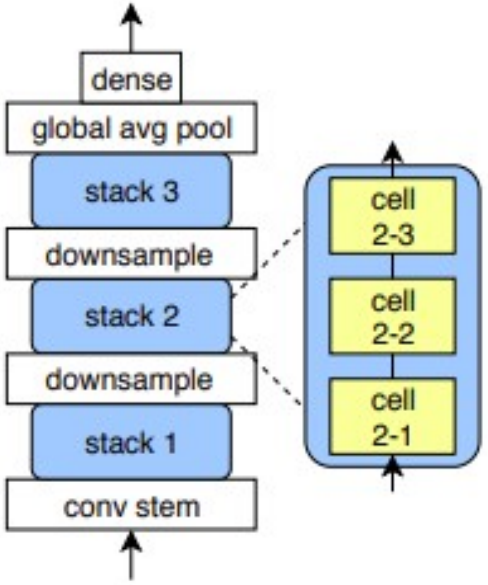
NATS-Bench: Benchmarking NAS Algorithms for Architecture Topology and Size (Xuanyi Dong – 01/2021)

NAS-Bench-101: Towards Reproducible Neural Architecture Search (Chris Ying – 02/2019)

Construction of Hierarchical Neural Architecture Search Spaces based on Context-free Grammars (Simon Schrodi - 02/2024)

EA-HAS-Bench: Energy-Aware Hyperparameter and Architecture Search Benchmark (Shuguang Dou - 02/2023)

NAS - Problems



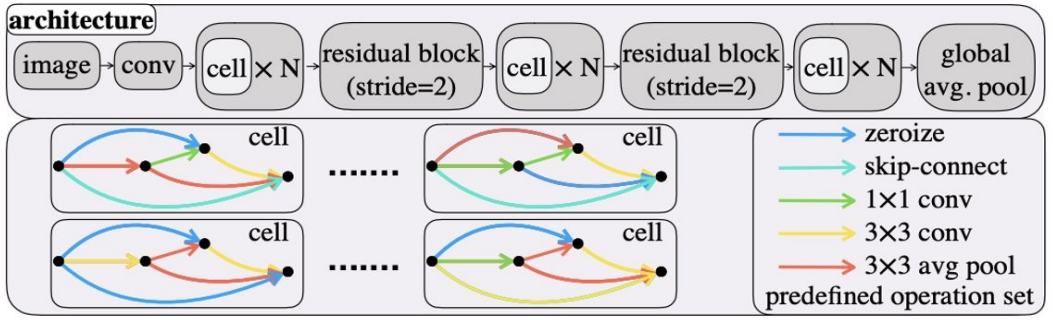
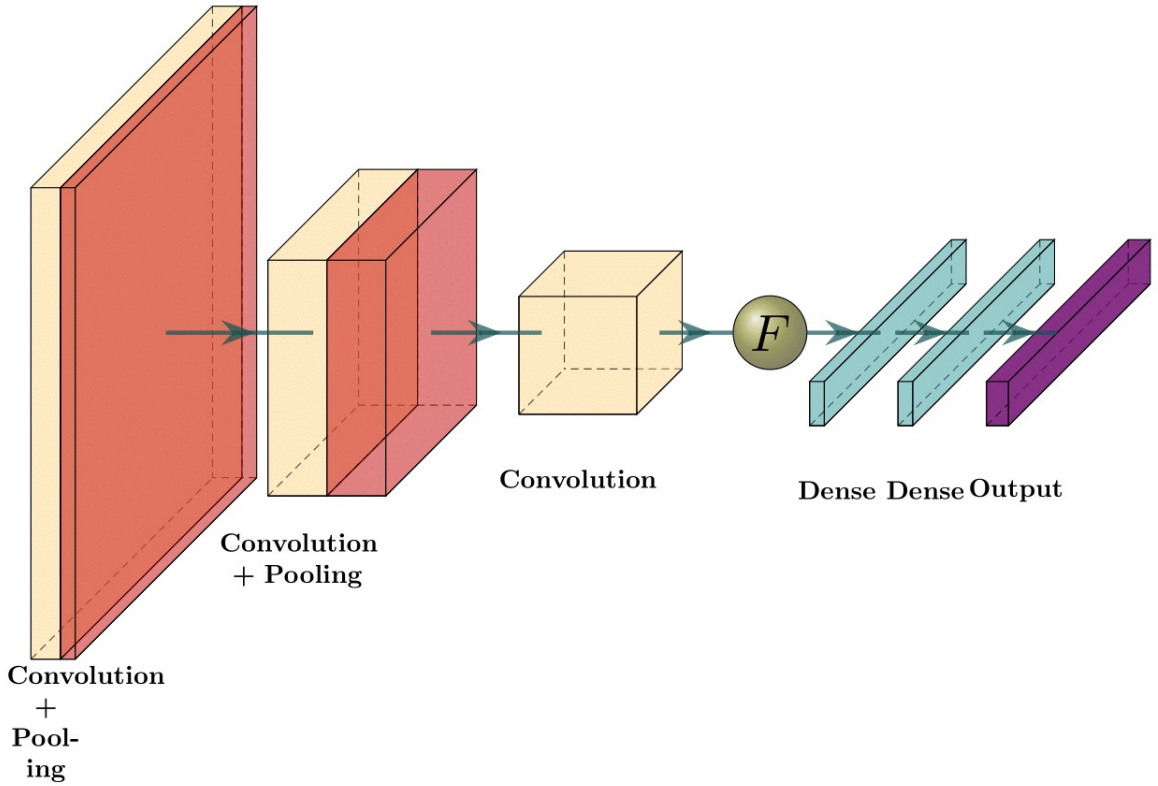


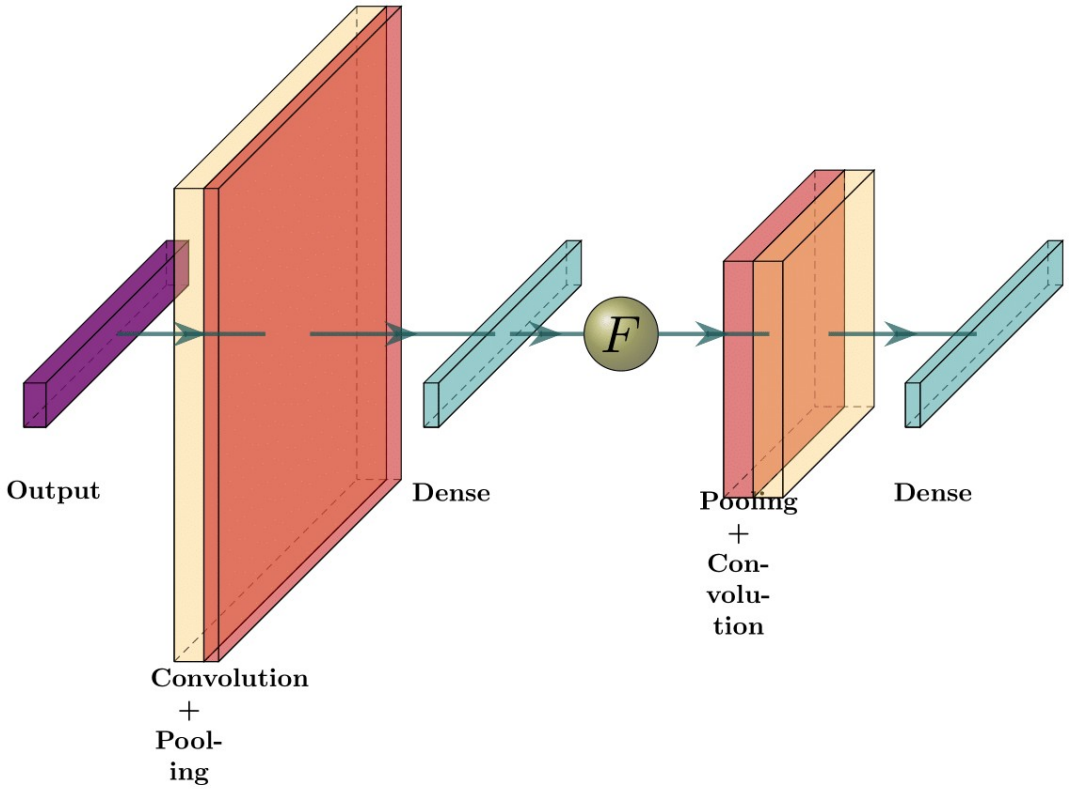
Figure 1: **Top**: the macro skeleton of each architecture candidate. **Bottom-left**: examples of neural cell with 4 nodes. Each cell is a directed acyclic graph, where each edge is associated with an operation selected from a predefined operation set as shown in the **Bottom-right**.

Pre-selection (architecture)

Variability (limited)

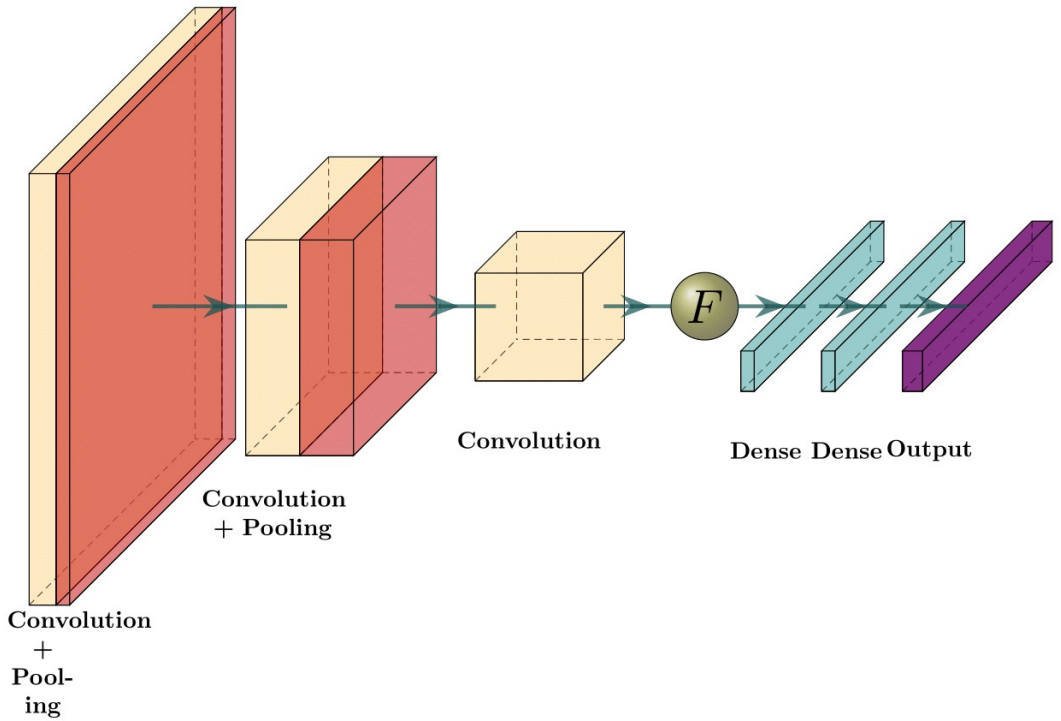
Need for (re)training (or pre-trained)


Convolution
+
Pool-
ing
Convolution
+ Pooling
Convolution
F
Dense
Dense
Output


Output
Convolution
+
Pool-
ing
Dense
F
Pooling
+
Con-
volu-
tion
Dense

Example

input	conv max_pooling conv max_pooling conv	flatten	dense dense	output



Why?

```
input conv max_pooling conv max_pooling conv flatten dense dense output
```

Domain Specific Language

- CNN architectures' grammar
- Constraints support
- Validity check

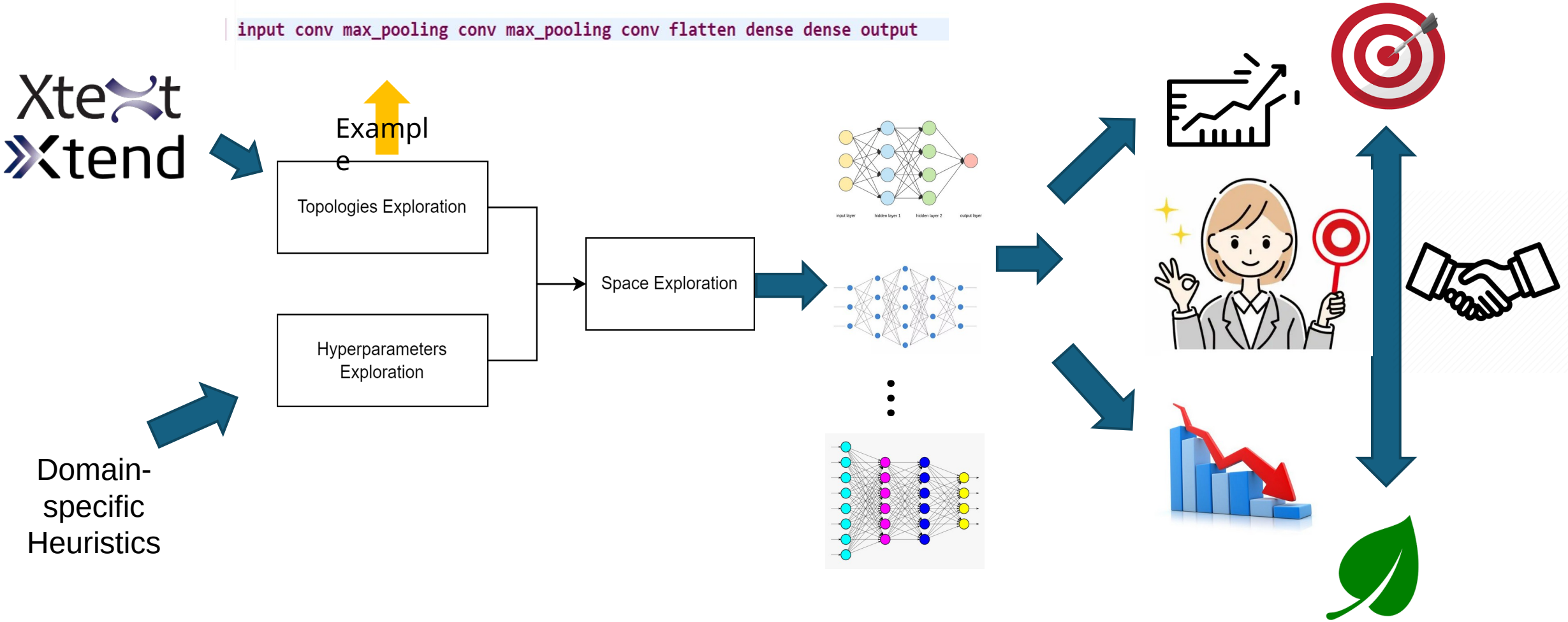
Xtext

Xtend

How did we find the grammar rules ?

- Based on domain knowledge
- Based on SOTA

CNNGEN


input conv max_pooling conv max_pooling conv flatten dense dense output
Xtext
Xtend
Example
Topologies Exploration
Hyperparameters Exploration
Space Exploration
Domain-specific Heuristics

Selecting Candidate Architectures

Search strategies

- Random search
- Evolutionary algorithms

Select model to train

- Performance/energy prediction
- Deep Metric Learning
- Embedding based on grammar
- Distance (Levenshtein)
- Graph

Selecting Candidate Architectures

Search strategies

- Random search
- Evolutionary algorithms (Future work)

Select model to train

- **Performance/energy prediction**
- ~~Deep Metric Learning~~
- ~~Embedding based on grammar~~
- ~~Distance (Levenshtein)~~
- ~~Graph~~

State of the Art Performance Prediction NAS Overview

- **N-shot** evaluation method: The number of trained architectures >= the number of searched architectures. All searched architectures are trained, and the focus is on accelerating their training to reduce runtime compared to Traditional Evaluation Method (TEM).
  - Downscaled dataset methods => reduce the dataset scale
  - Downscaled model methods => focus on reducing the model size
  - Network morphism and learning curve extrapolation => reduce the number of epochs

State of the Art Performance Prediction NAS Overview

- N-shot evaluation method
- **Few-shot** evaluation method: The number of trained architectures is less than the number of searched architectures, but greater than one. The runtime is naturally less than TEM due to fewer trained architectures.
  - Performance predictor
  - Population memory

State of the Art Performance Prediction NAS Overview

- N-shot evaluation method
- Few-shot evaluation method
- **One-shot** evaluation method: Only one architecture is trained, resulting in a runtime lower than TEM. The focus is on training a single architecture.
  - Path-based methods
  - Gradient-based methods

State of the Art Performance Prediction NAS Overview

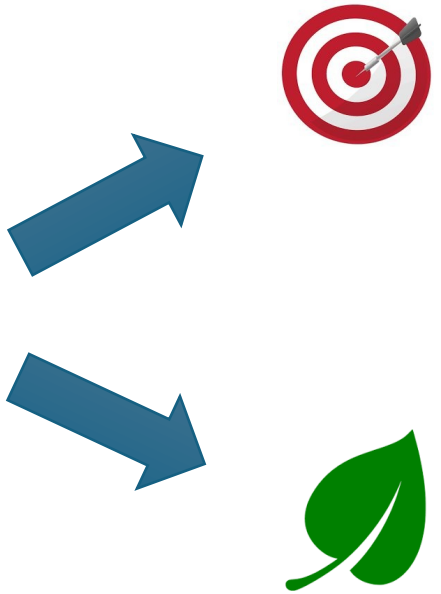
- N-shot evaluation method
- Few-shot evaluation method
- One-shot evaluation method
- **Zero-shot** evaluation method: No architectures are trained, resulting in an extremely low cost. This method requires no training.
  - Parameter-level methods
  - Architecture-level methods

State of the Art Performance Prediction NAS Overview

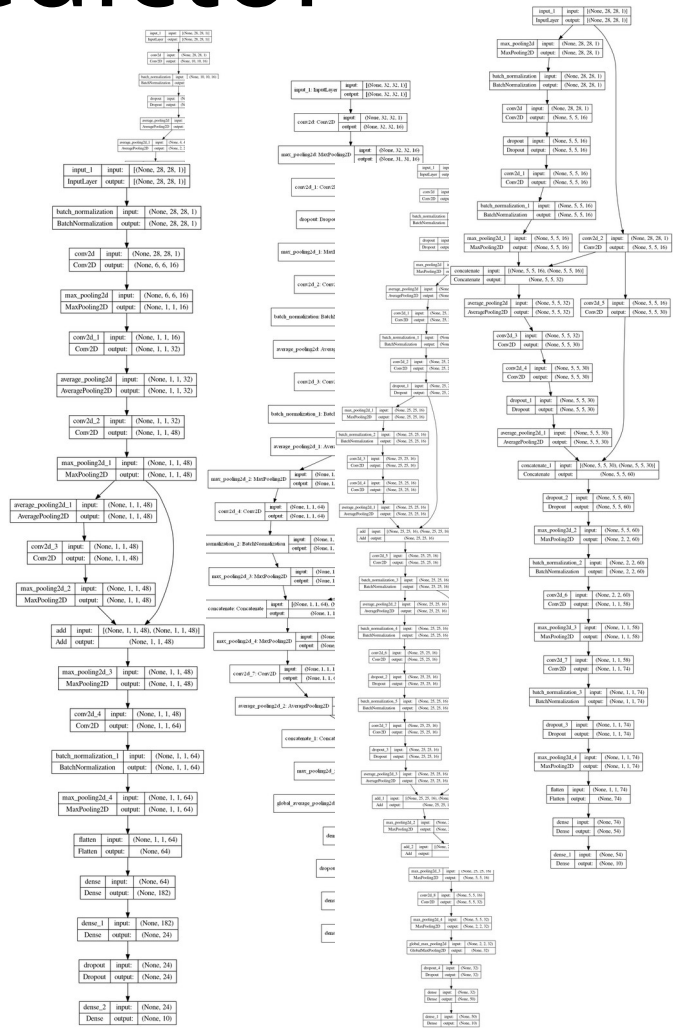
- N-shot evaluation method:
- **Few-shot evaluation method**
  - **Performance predictor**
  - Population memory
- One-shot evaluation method
- Zero-shot evaluation method

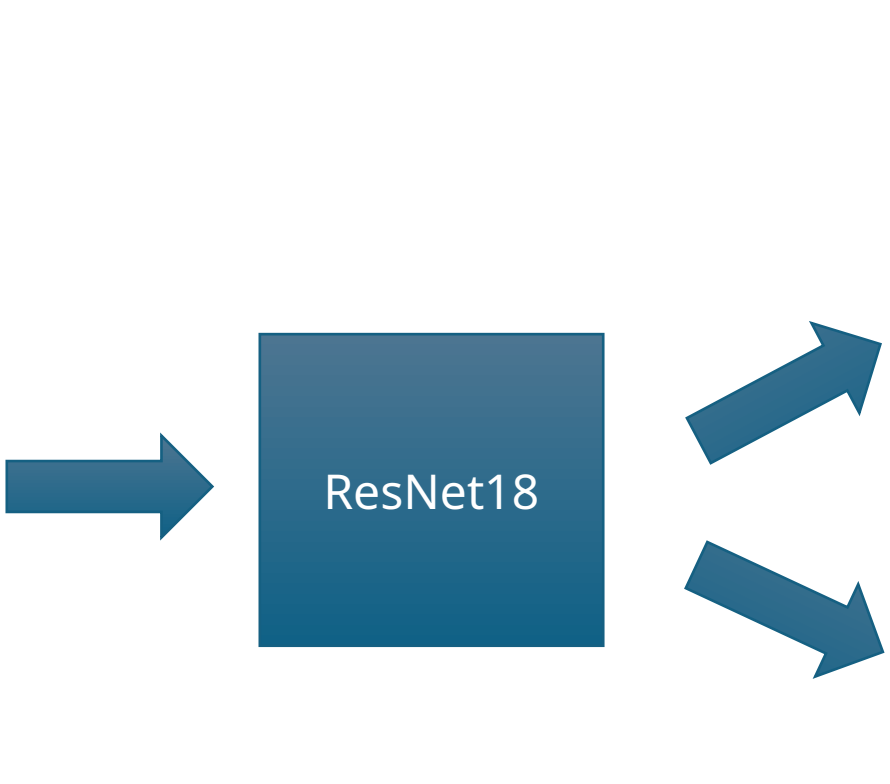
Predictor ¼ : Grammar-Based Predictor

```
input conv max_pooling conv max_pooling conv flatten dense dense output
```

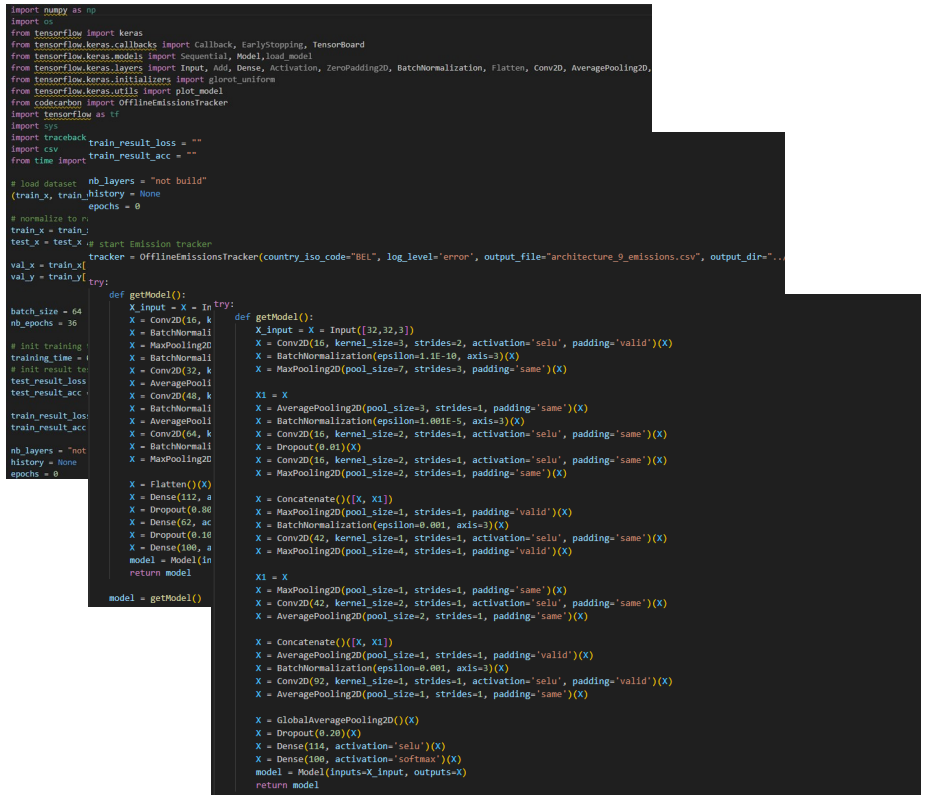


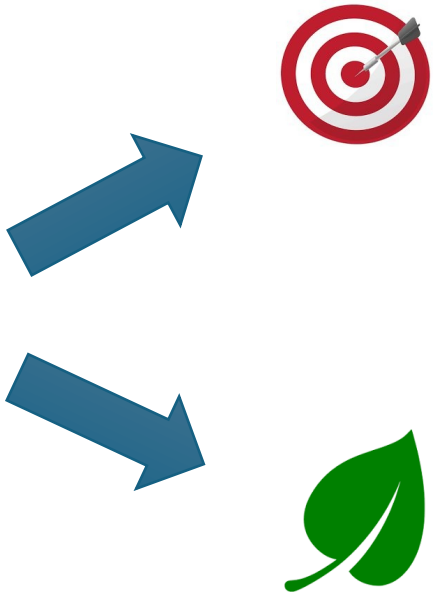
Predictor 2/4 : TensorBoard Visualization P redictor





Predictor ¾ : Code-Based Predictor

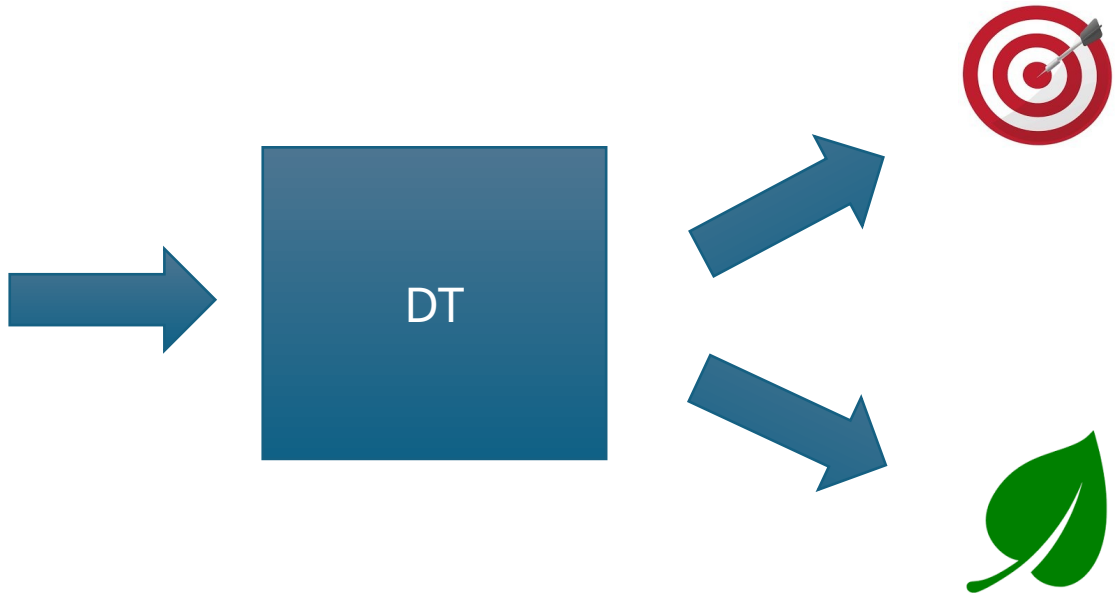




Predictor 4/4 : Decision Tree-Based Predictor

Manually selected features:

- Layers
- Epochs
- FLOP
- Parameters



Experimentation

- Generate 1300 architectures with CNNGen
- Train Dataset (CIFAR-10, CIFAR-100, f-MNIST)
- Architecture Dataset to train predictors (energy & performance)

Results

- Decision Tree-Based Predictor => best result for energy prediction
- Code-Based Predictor => best result for performance prediction

Future works - Neuro-evolution

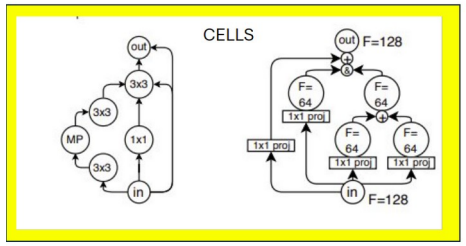
- Automatic optimisation => genetic algorithms
  - JMetal
  - Genetic improvement on grammar instance

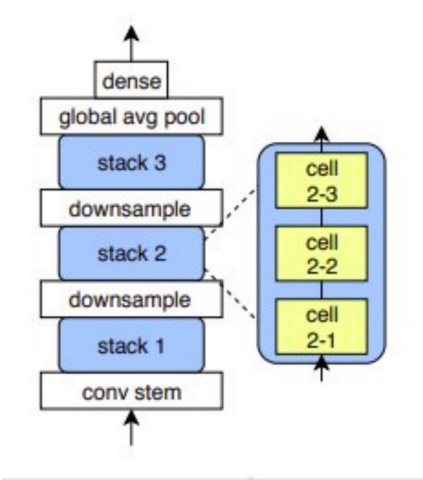
```
input conv max_pooling conv max_pooling conv flatten dense dense output
```

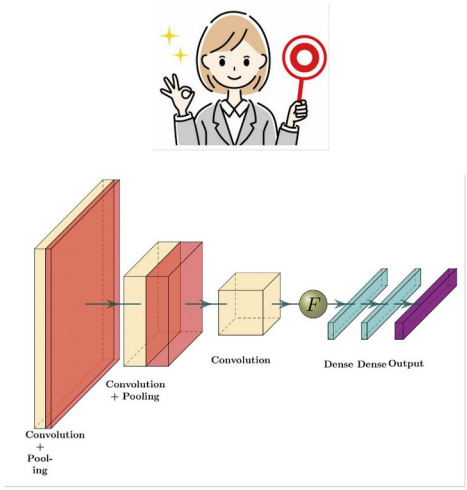
Sum up

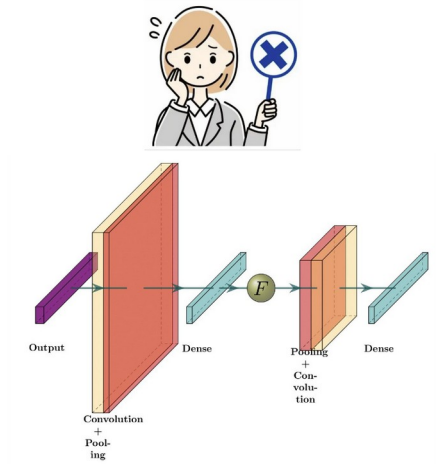
NAS - Example

- Espace predifined
- 423K unique architectures
- Max 108 epochs

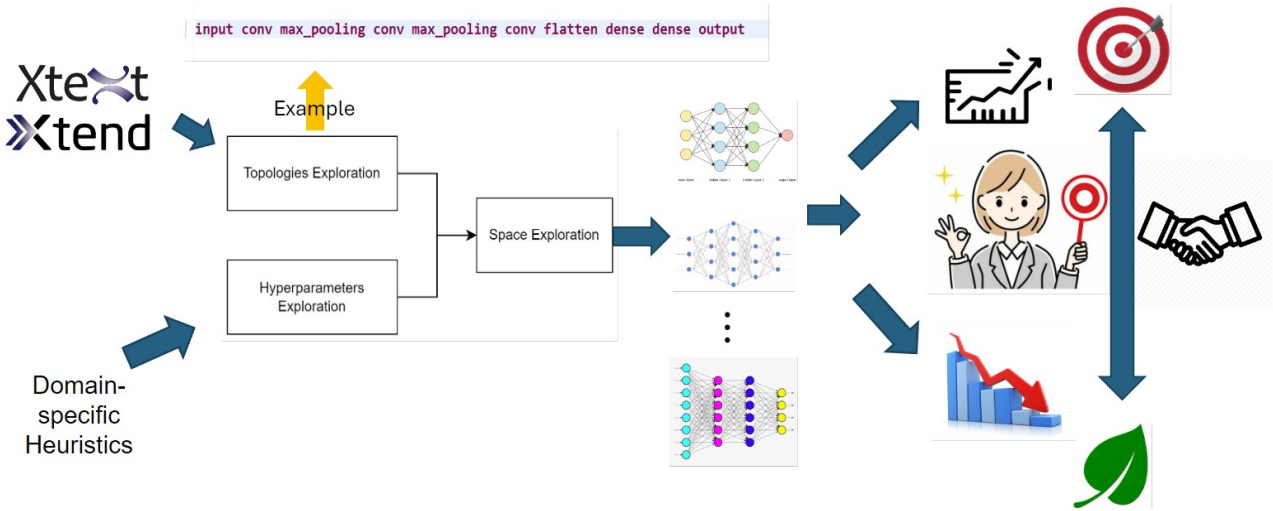








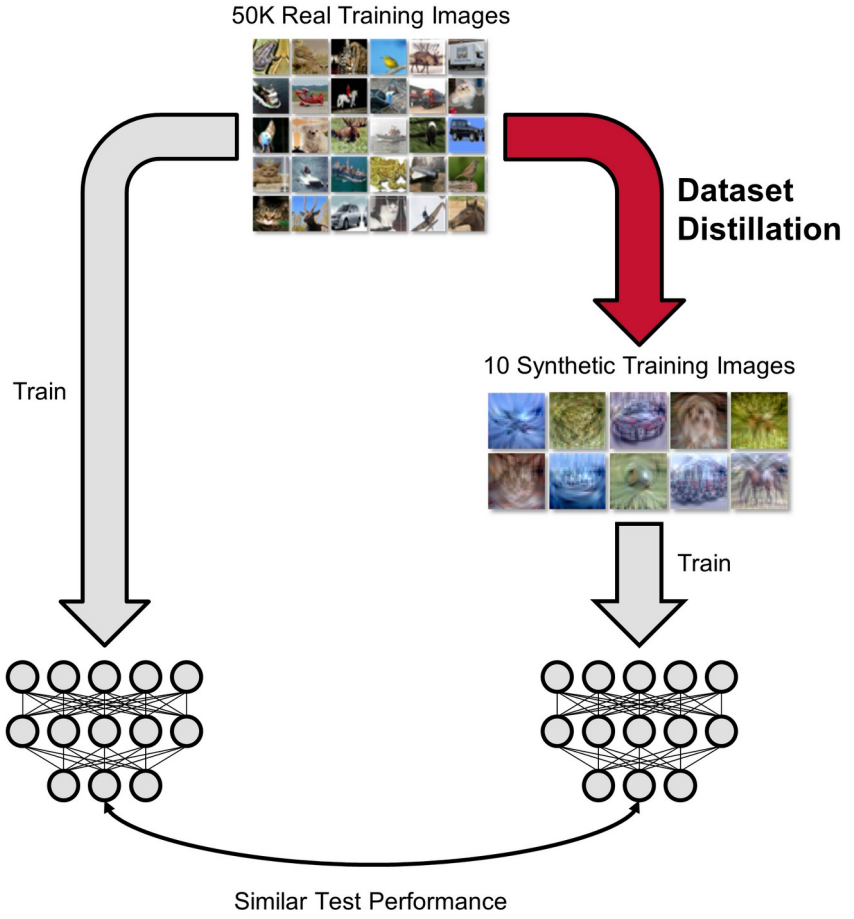
CNNGEN

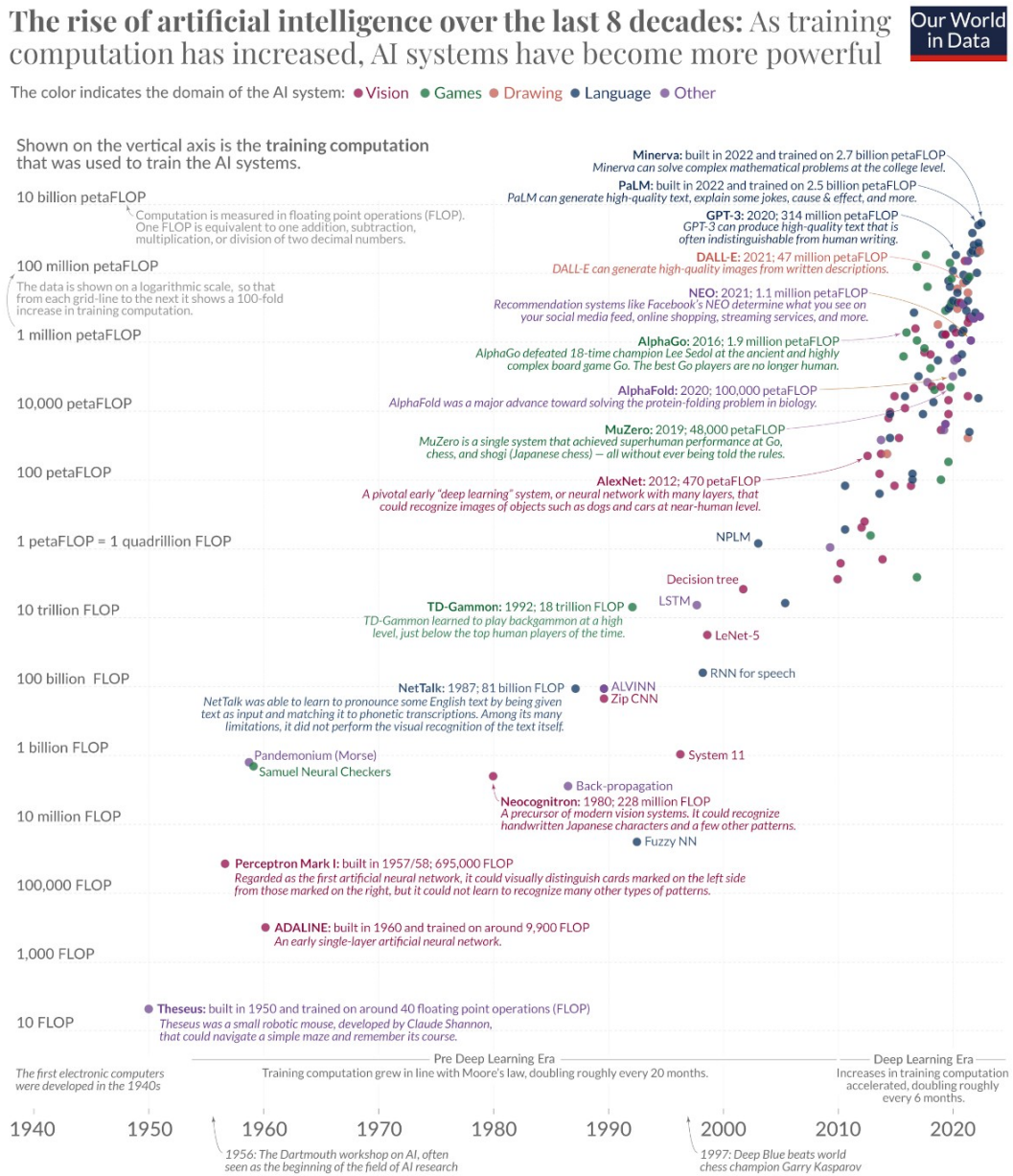


BackUP

Future works - Dataset Distillation

- Create SynD
  - Same performance
  - Less cost
- Benchmark divers DD technics (Energy cost)
- Train selected CNNGen architectures




The rise of artificial intelligence over the last 8 decades: As training computation has increased, AI systems have become more powerful
Our World in Data
The color indicates the domain of the AI system: Vision Games Drawing Language Other
Shown on the vertical axis is the training computation that was used to train the AI systems.
10 billion petaFLOP
Computation is measured in floating point operations (FLOP). One FLOP is equivalent to one addition, subtraction, multiplication, or division of two decimal numbers.
100 million petaFLOP
The data is shown on a logarithmic scale, so that from each grid-line to the next it shows a 100-fold increase in training computation.
1 million petaFLOP
10,000 petaFLOP
100 petaFLOP
1 petaFLOP = 1 quadrillion FLOP
10 trillion FLOP
100 billion FLOP
1 billion FLOP
10 million FLOP
100,000 FLOP
1,000 FLOP
10 FLOP
Minerva: built in 2022 and trained on 2.7 billion petaFLOP
Minerva can solve complex mathematical problems at the college level.
PaLM: built in 2022 and trained on 2.5 billion petaFLOP
PaLM can generate high-quality text, explain some jokes, cause & effect, and more.
GPT-3: 2020; 314 million petaFLOP
GPT-3 can produce high-quality text that is often indistinguishable from human writing.
DALL-E: 2021; 47 million petaFLOP
DALL-E can generate high-quality images from written descriptions.
NEO: 2021; 1.1 million petaFLOP
Recommendation systems like Facebook's NEO determine what you see on your social media feed, online shopping, streaming services, and more.
AlphaGo: 2016; 1.9 million petaFLOP
AlphaGo defeated 18-time champion Lee Sedol at the ancient and highly complex board game Go. The best Go players are no longer human.
AlphaFold: 2020; 100,000 petaFLOP
AlphaFold was a major advance toward solving the protein-folding problem in biology.
MuZero: 2019; 48,000 petaFLOP
MuZero is a single system that achieved superhuman performance at Go, chess, and shogi (Japanese chess) — all without ever being told the rules.
AlexNet: 2012; 470 petaFLOP
A pivotal early "deep learning" system, or neural network with many layers, that could recognize images of objects such as dogs and cars at near-human level.
NPLM
Decision tree
LSTM
TD-Gammon: 1992; 18 trillion FLOP
TD-Gammon learned to play backgammon at a high level, just below the top human players of the time.
LeNet-5
RNN for speech
NetTalk: 1987; 81 billion FLOP
NetTalk was able to learn to pronounce some English text by being given text as input and matching it to phonetic transcriptions. Among its many limitations, it did not perform the visual recognition of the text itself.
ALVINN
Zip CNN
Pandemonium (Morse)
Samuel Neural Checkers
System 11
Back-propagation
Neocognitron: 1980; 228 million FLOP
A precursor of modern vision systems. It could recognize handwritten Japanese characters and a few other patterns.
Fuzzy NN
Perceptron Mark I: built in 1957/58; 695,000 FLOP
Regarded as the first artificial neural network, it could visually distinguish cards marked on the left side from those marked on the right, but it could not learn to recognize many other types of patterns.
ADALINE: built in 1960 and trained on around 9,900 FLOP
An early single-layer artificial neural network.
Theseus: built in 1950 and trained on around 40 floating point operations (FLOP)
Theseus was a small robotic mouse, developed by Claude Shannon, that could navigate a simple maze and remember its course.
The first electronic computers were developed in the 1940s
Pre Deep Learning Era
Training computation grew in line with Moore's law, doubling roughly every 20 months.
Deep Learning Era
Increases in training computation accelerated, doubling roughly every 6 months.
1940
1950
1960
1970
1980
1990
2000
2010
2020
1956: The Dartmouth workshop on AI, often seen as the beginning of the field of AI research
1997: Deep Blue beats world chess champion Garry Kasparov
The data on training computation is taken from Sevilla et al. (2022) – Parameter, Compute, and Data Trends in Machine Learning.
It is estimated by the authors and comes with some uncertainty. The authors expect the estimates to be correct within a factor of two.
OurWorldinData.org – Research and data to make progress against the world's largest problems.
Licensed under CC-BY by the authors
Charlie Giattino, Edouard Mathieu, and Max Roser

State of the Art Performance Prediction NAS Overview

- N-shot evaluation method: The number of trained architectures is greater than or equal to the number of searched architectures. All searched architectures are trained, and the focus is on accelerating their training to reduce runtime compared to Traditional Evaluation Method (TEM).
  - Downscaled dataset methods => reduce the dataset scale
  - Downscaled model methods => focus on reducing the model size
  - Network morphism and learning curve extrapolation => reduce the number of epochs
- Few-shot evaluation method: The number of trained architectures is less than the number of searched architectures, but greater than one. The runtime is naturally less than TEM due to fewer trained architectures.
  - Performance predictor
  - Population memory
- One-shot evaluation method: Only one architecture is trained, resulting in a runtime lower than TEM. The focus is on training a single architecture.
  - Path-based methods
  - Gradient-based methods
- Zero-shot evaluation method: No architectures are trained (At = ∅), resulting in an extremely low cost. This method requires no training.
  - Parameter-level methods
  - Architecture-level methods

N-shot evaluation method

- Downscaled dataset methods
  - Reduced Training Time & Accelerated Exploration
    - Using a smaller dataset, the training process becomes quicker and more efficient (beneficial large and complex datasets)
    - Allow to iterate more rapidly in architecture search and experimentation.
- Downscaled model methods
  - Reduced Training Time & Accelerated Exploration
    - Reducing the size of the models the training process becomes faster and more efficient (beneficial large and complex models)
    - Allow to iterate more rapidly in architecture search and experimentation.
- Network morphism
- Learning curve extrapolation

N-shot evaluation method

- Downscaled dataset methods
- Downscaled model methods
- Network morphism
  - Accelerated Training Process
    - Child network can inherit the weights and function of a fully-trained parent network => child network does not need to be trained from scratch and requires fewer epochs to converge
- Learning curve extrapolation
  - Reduced training time
    - Predict the final performance of architectures trained for only a few epochs
  - Incorporation of Partial Learning Curves:
    - Partial learning curves captures valuable information about the architecture's learning progress => accurate predictions of the architecture performance, even with limited training epochs.

Few-shot evaluation method

- Performance predictor
  - Reduced runtime
    - Estimate the performance of architectures in the search space without actually training and evaluating
  - Resource efficiency
    - Build a small architecture set (At) that represents the search space adequately => reduces the number of architectures that need to be trained and evaluated
  - Flexibility and adaptability
    - Updated during the architecture search process, allowing them to incorporate new samples and improve their prediction performance.
    - Add new architectures to the training set and retrain the predictor, making them more flexible and adaptable to changing search spaces.
- Population Memory
  - Accelerated Fitness Evaluation
    - Avoiding the repeated evaluation of the same architecture by reusing architectural information that has appeared in previous populations(as a cache system)

One-shot evaluation method

- Path-based methods
  - Accelerated performance evaluation
    - Instead of training and evaluating each subnet individually, the weights of the subnets are directly extracted from the supernet, and their performance is inferred on a validation dataset
  - Weight decoupling
    - To decouple the weights from the architecture and improve the performance evaluation sampling strategies are employed during the supernet training
    - Optimize the weights of different subnets in a more balanced way, effectively alleviating the weight coupling problem and providing better performance estimation for the subnets.
- Gradient-based methods
  - Decoupling of training and architecture search
    - Separate the training of the supernet and the search for the architectures => allows for more efficient optimization by jointly optimizing the supernet weights and architecture parameters
  - Continuous relaxation of the search space
    - Relax the discrete search space to be continuous => softmax function allows for the continuous representation of the architecture choices
    - Relaxation enables to efficiently search for optimal architectures
  - Challenges
    - Huge memory consumption
    - Poor generalization
    - Performance collapse

Zero-shot evaluation method

- Parameter-level methods
  - Reduced computational cost
    - Require a minibatch of data and a forward/backward propagation pass to calculate saliency indicators for certain parameters.
    - Don't involve training the architecture, the computational requirements are significantly reduced.
  - Utilization of network pruning techniques:
    - Leverage saliency indicators (used in network pruning literature) measure the importance of parameters and help identify unimportant ones
    - Can evaluate the saliency of parameters and score the entire architecture based on the aggregated saliencies.
- Architecture-level methods
  - Time-saving performance evaluation
    - Evaluate the performance of architectures by measuring properties related to architecture performance without training
  - Theoretical basis and indicators
    - Indicators as activation overlap, learnability, gradient behavior, expressivity, synaptic diversity, ... to judge the properties of architectures and rank them
- Inaccurate => performance fluctuated dramatically among different tasks.