Neural architecture search for time series

MSc in Artificial Intelligence and Data Analytics
Final Presentation

Student name: Christoforidis Aristeidis
Supervisor: Margaritis Konstantinos

HELLENIC REPUBLIC
UNIVERSITY OF MACEDONIA

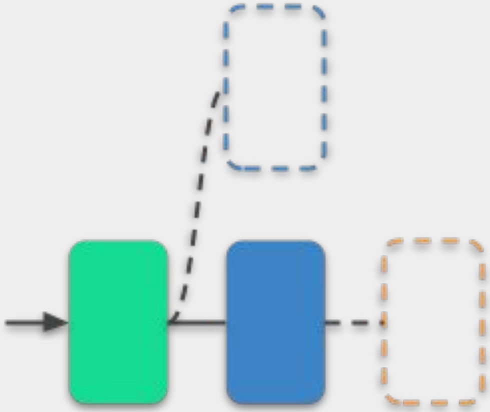
Background

Neural architecture search(NAS) is a research domain concerned with automated neural network design techniques and algorithms. NAS is analyzed into **search space design** and **optimization methods**.

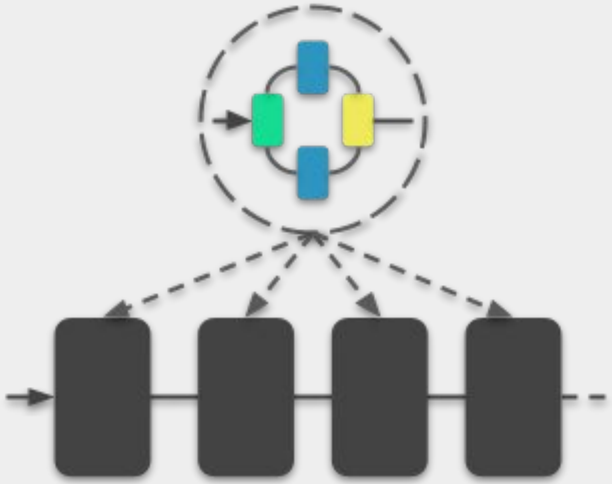
Search space design describes the process of defining the set of networks that can be created by a NAS algorithm.

Optimization methods dictate how the search is conducted and how the process discovers new networks.

Proposed search space design approaches

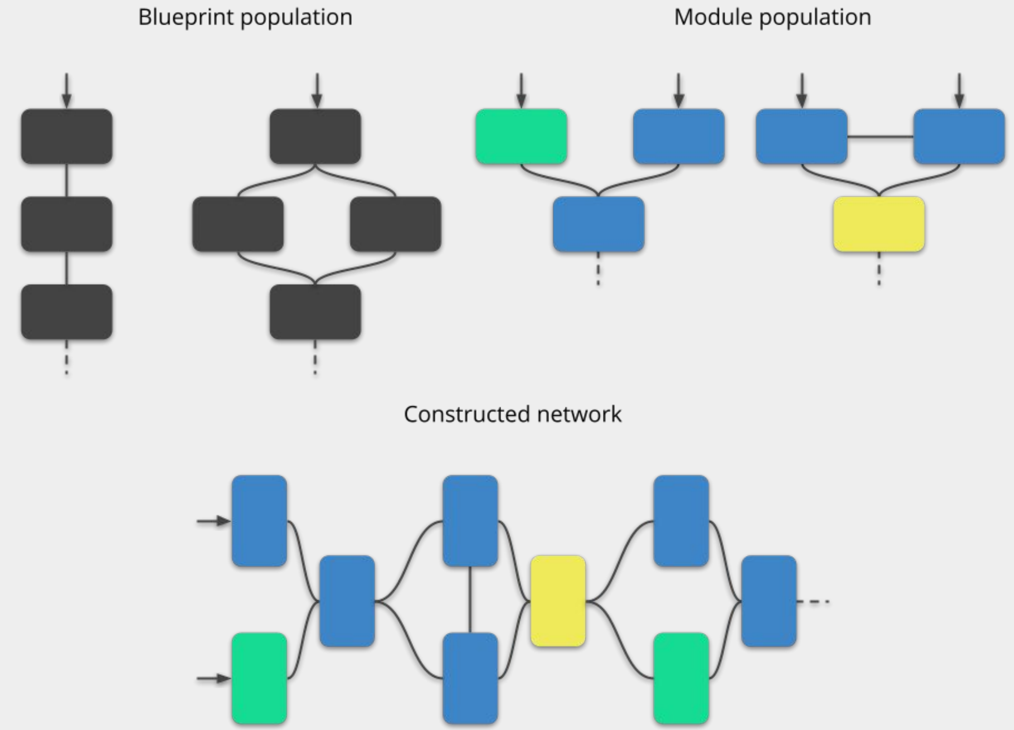


Global(Macro) search space



Micro search space

Proposed search space design approaches



Hierarchical search space

Proposed optimization methods

Evolutionary algorithms: Evolve a population of candidate topologies to discover better performing networks. Candidates topologies are trained and validated to assess their fitness.

Reinforcement learning: Train an agent to construct deep neural networks by iteratively adding layers and making connections between them. The agent's loss function incorporates the accuracy of the constructed network on the given dataset.

One-shot methods: Use weight sharing schemes in a hypergraph of neural operations that is trained in segments and discover the optimal topology in a restricted search space.

Bayesian optimization: Use an estimation model to predict the performance of a topology and an acquisition function (the above methods can be incorporated here) in order to sequentially design new topologies.

Common issues in neural architecture search

- Global search spaces are hard to conduct search due to the seer number of available network topologies.
- Constrained search spaces are often too restrictive and obstruct the process of discovering new performant architectures.
- Network expansion techniques are usually slow - evaluating a network after a single change in a layer or connection is computationally expensive.

Initial idea

- State of the art networks are usually built by repeating architectural segments(e.g. InceptionNet).
- Instead of distinguishing between the micro and macro architecture of a network, it is possible to use a hierarchical representation to represent a topology in multiple intermediate-level modules of varying complexity.
- This can potentially allow for a faster network creation and expansion, where networks evolve by adopting large well performing segments.

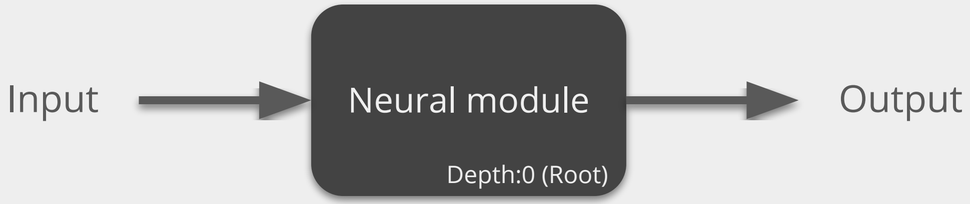
Hierarchical network representation

Core concept: represent each neural network as a dynamic hierarchical graph.

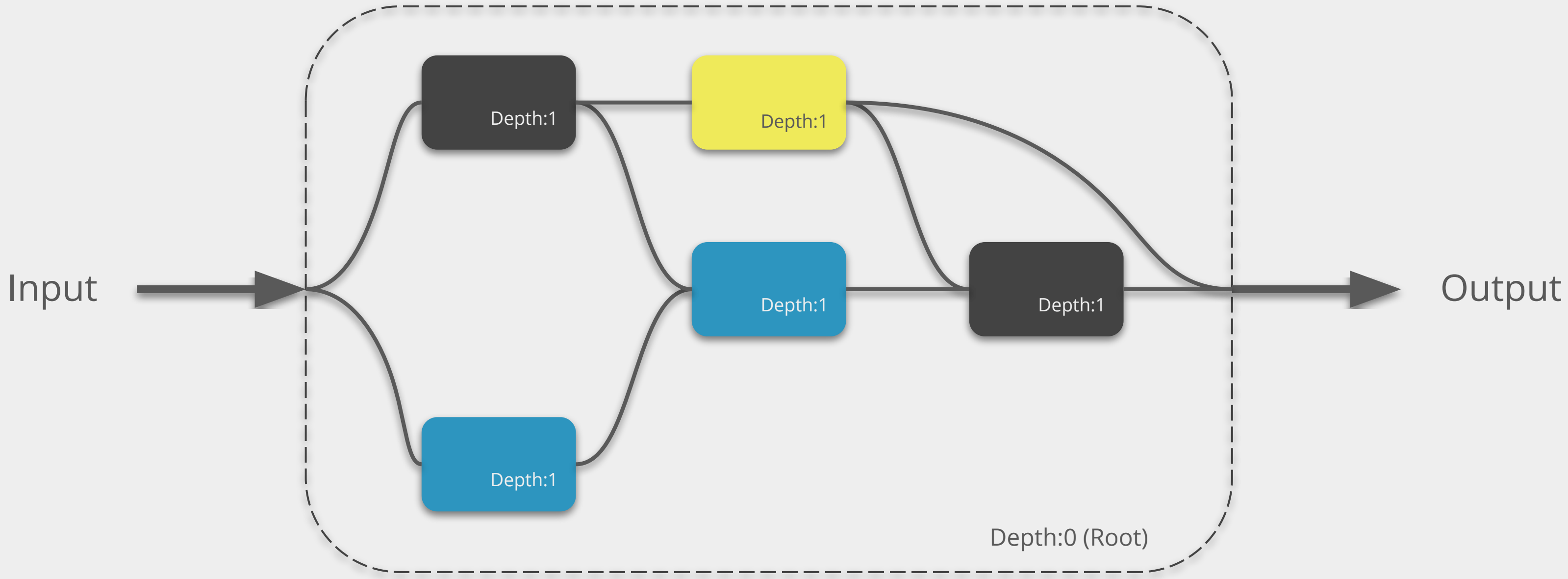
Neural module: a fundamental data structure for representing computational graphs and neural layers.

- Each node in the computational graph is another neural module.
- The top level of the dynamic hierarchical graph is just one neural module, representing the whole network.
- When all nodes are fully expanded, the resulting graph contains only neural layers.
- Networks can be created and modified using custom evolution mechanics.

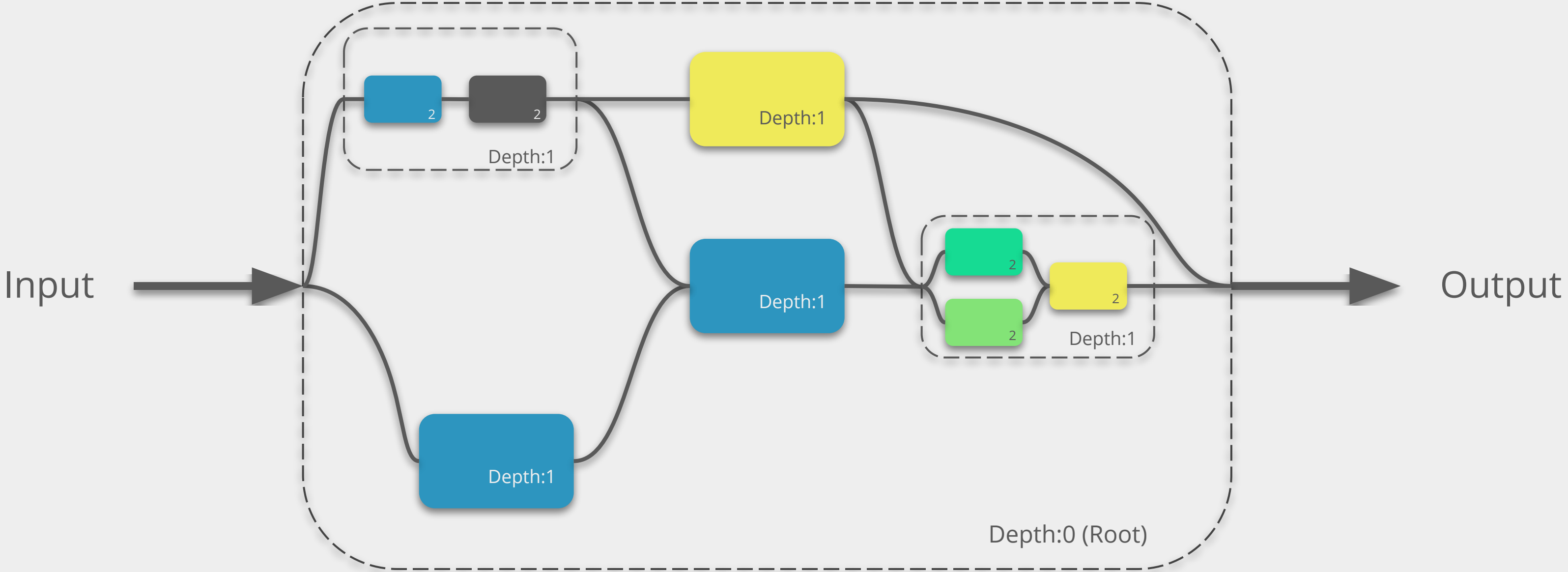
Representation example



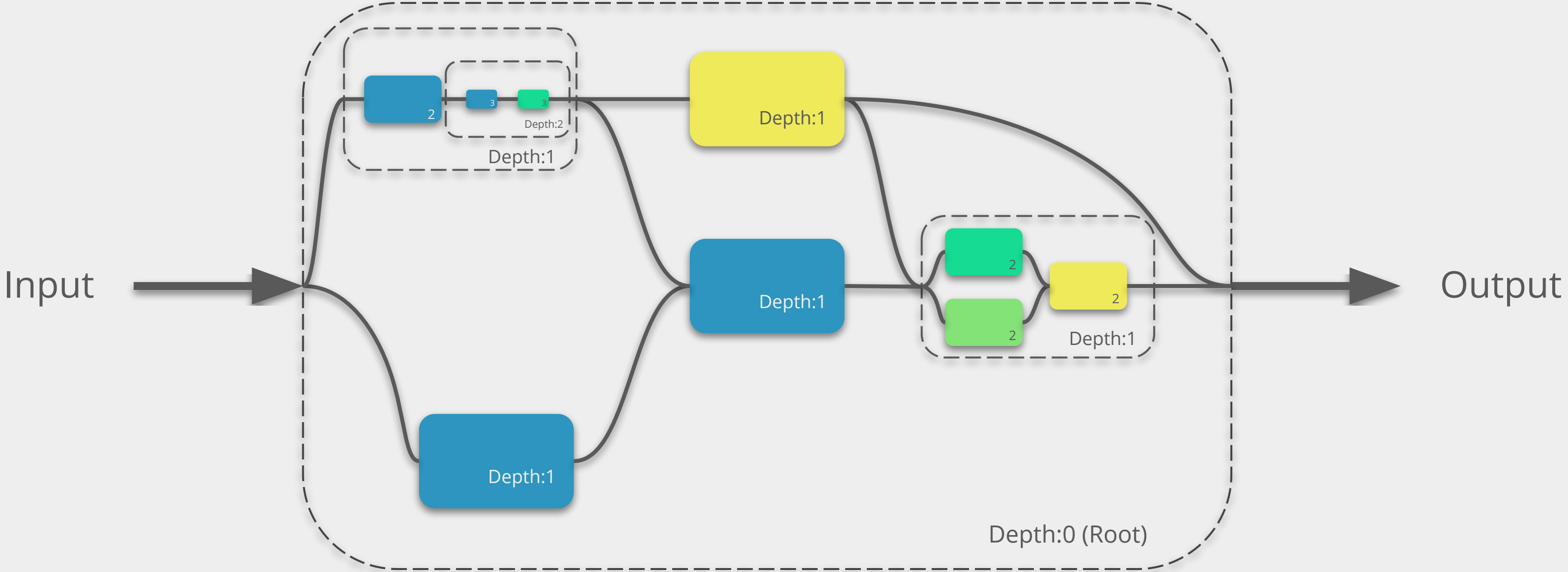
Representation example



Representation example



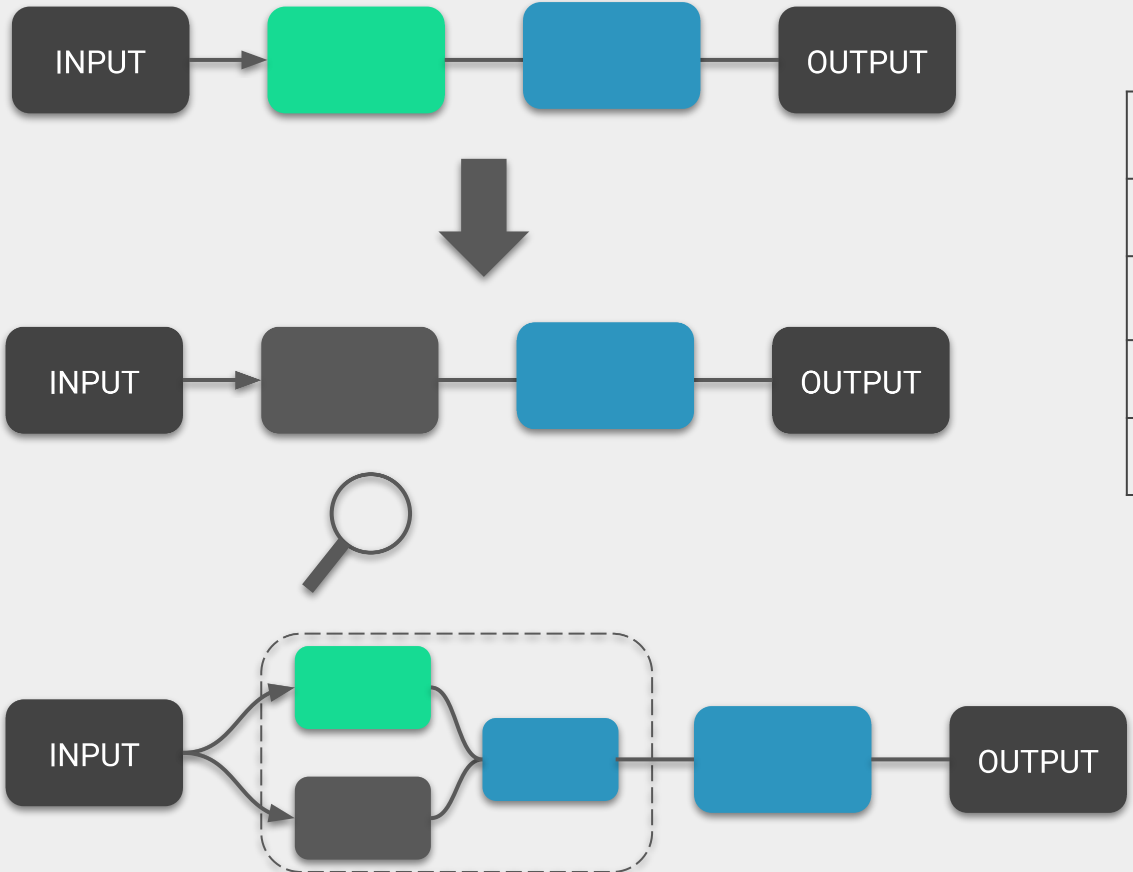
Representation example



Evolution

1) Initialize a **notable modules list**. This originally contains the allowed neural layers.
2) Create the population. Each network has one node, which is a neural module sampled(weighted sampling based on fitness) from the notable modules list. At the start, all networks will essentially have one neural layer.
3) Perform mutation.
    a) Node mutation: a random neural node is selected and replaced with a random neural graph. Each of the nodes on the new graph is assigned a neural module from the notable modules list.
    b) Edge mutation: An edge is added to a randomly selected neural graph of the network.
4) Evaluate modified networks.
5) Update notable modules list.
6) Delete low accuracy networks from the population and replace them with new networks.

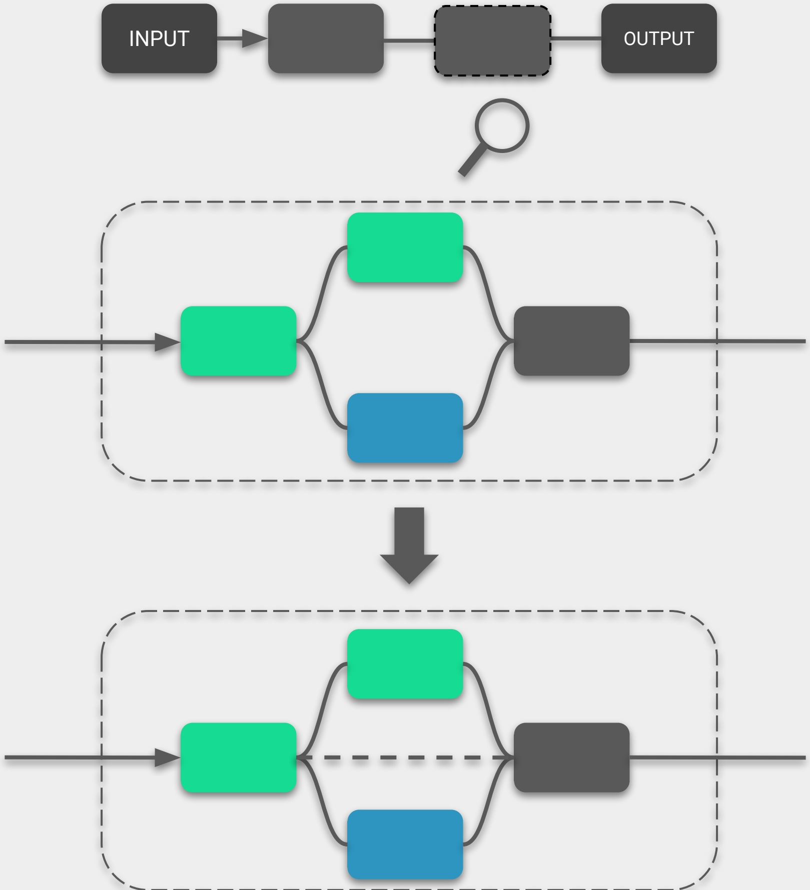
Node mutation



Notable modules list

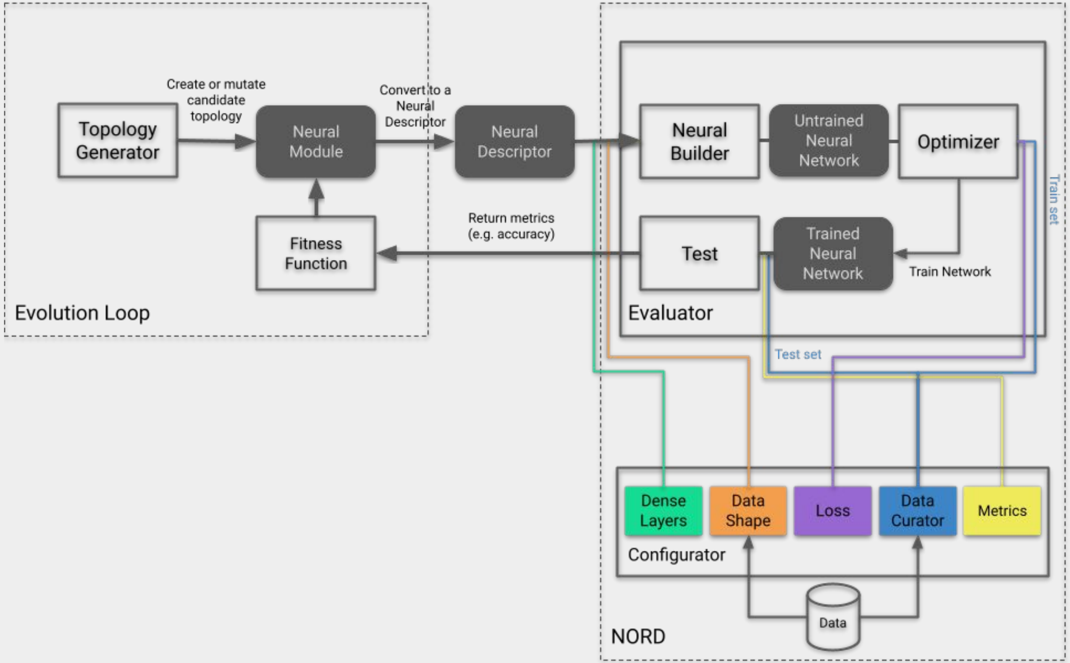
Neural module	Average fitness
CONV_3.N	0.29
POOL_2.M	0.18
CONV_1.H	0.23
ABSTRACT_MODULE	0.45

Edge mutation



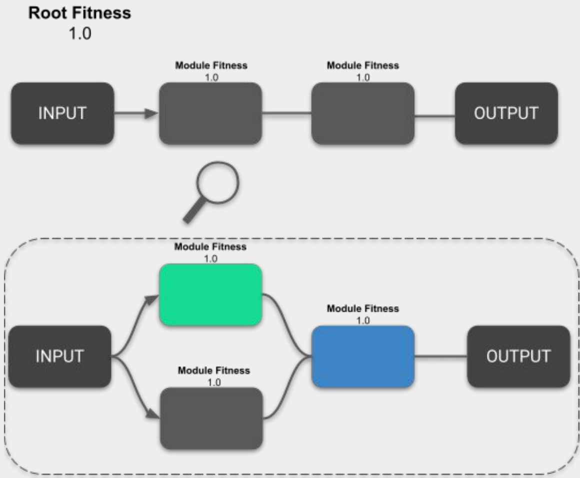
Evaluating topologies with NORD(Neural Operations Research and Development)

NORD is a research framework that simplifies the process of developing NAS algorithms by abstracting the process of designing and evaluating neural networks on common benchmark datasets.



Fitness assessment

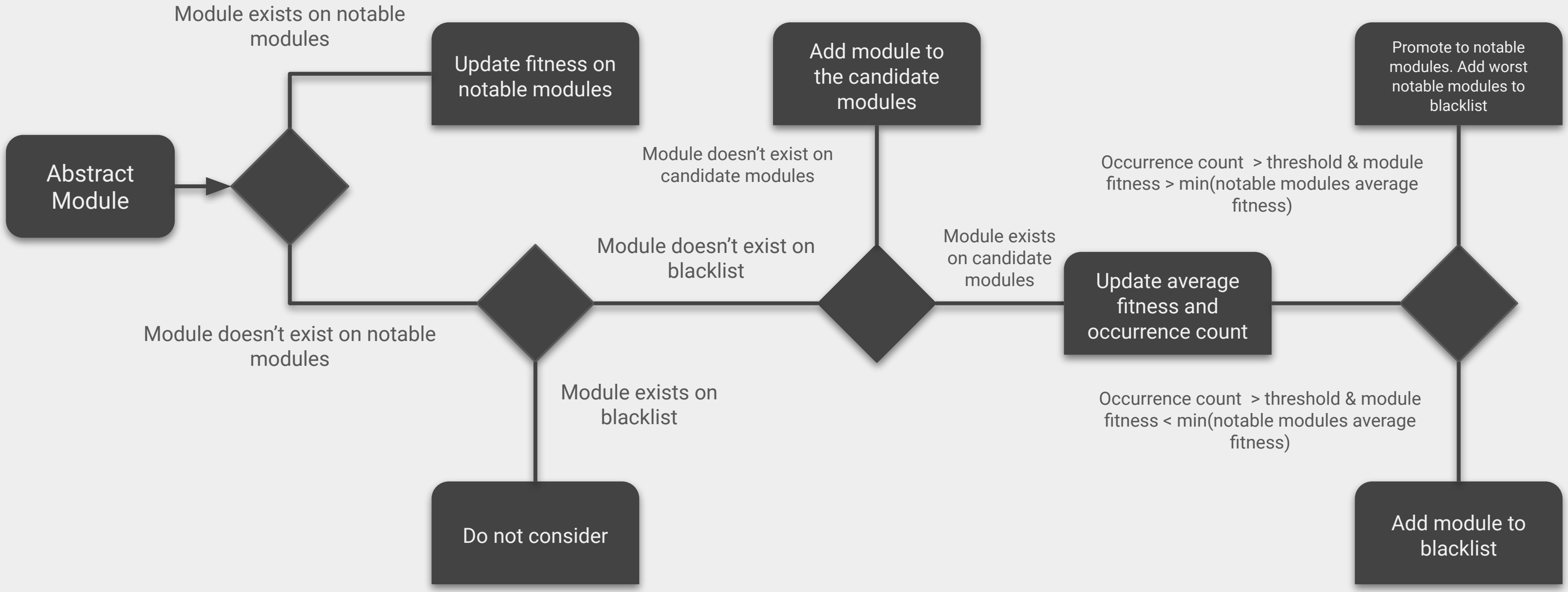
- All modules in a candidate’s hierarchy are assigned the same fitness.
- Modules that perform well in a variety of situations will have higher average fitness values in the candidates & notables lists.
- Well performing higher level modules phase out simpler modules that appear more often because the latter are more likely to occur in networks with poor performance.



Updating notable modules

- Neural modules that already exist on the notable modules list have their average fitness values updated.
- Modules that are not on the notable modules list must first be placed in a candidate modules list. This list holds modules that have been encountered at least once. If a candidate module is encountered more than a minimum amount of times, and its average fitness is high enough, it is promoted to “notable” and enters the notable modules.
- The notable modules list has a fixed size, so when the module limit is exceeded, the weakest modules are deleted.
- Candidate modules have a TTL timer. If they are not encountered the required amount of times before they expire, they are placed in a blacklist.
- The blacklist also contains modules that were removed from the notable modules.

Notable modules update logic



After each generation, candidate modules with expired TTLs are added to the blacklist.

Experiments

The proposed method was tested in 3 different benchmarks:

- 1 time series classification dataset
- 2 image classification datasets

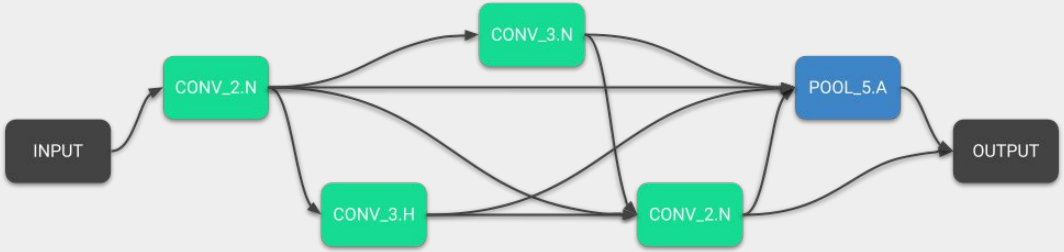
The evolution algorithm builds convolutional neural networks using a variety of convolutional and pooling operations with different properties.

All tests were conducted in the Google Colaboratory platform. A single NVIDIA T4 GPU was used for the training and evaluation of the candidate networks.

Experiments: Human activity recognition(1/3)

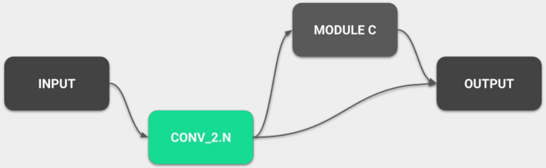
- Problem: classify 7 distinct actions performed by humans using the acceleration data from a chest-mounted sensor for the 3 axes.
- Sampling frequency is 52Hz.
- There are 26 lags of overlap between pairs of consecutive instances.
- Data was collected from 15 participants: 11 are used for training networks and 4 for evaluating network performance.

Experiments: Human activity recognition(2/3)

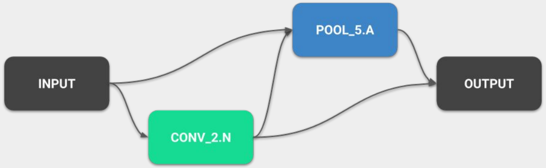


Root module

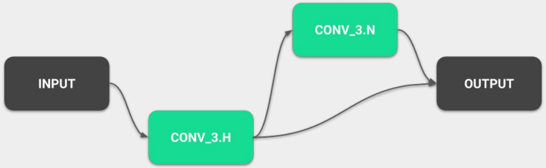
Module A



Module B

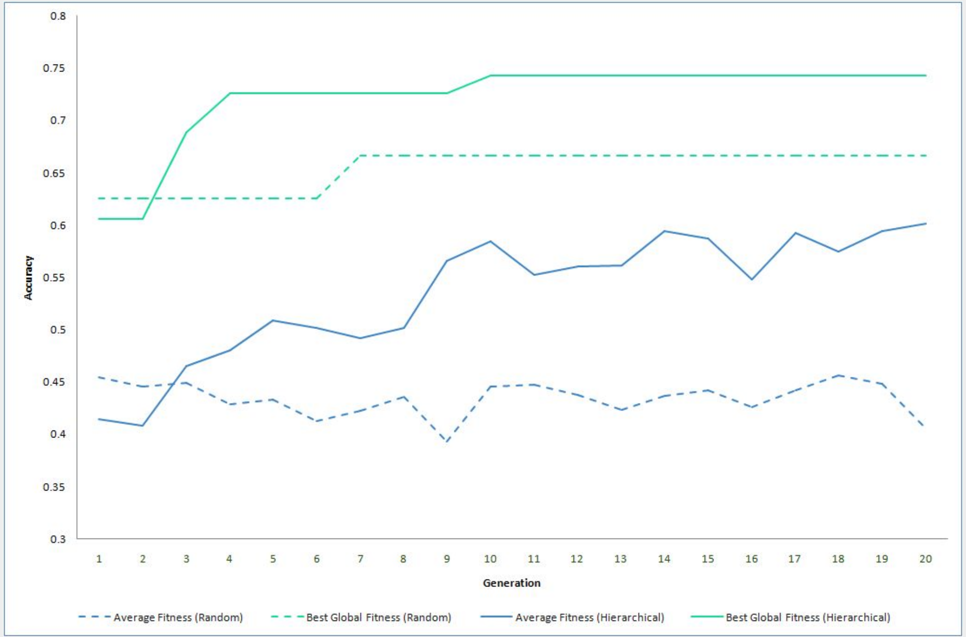


Module C



Experiments: Human activity recognition(3/3)

Best network accuracy: **74.3%**



Experiments: Fashion-MNIST (1/2)

- Problem: classify 28x28 grayscale images of 10 different types of articles of clothing and accessories.
- 60000 samples in train set, 10000 samples on the test set.

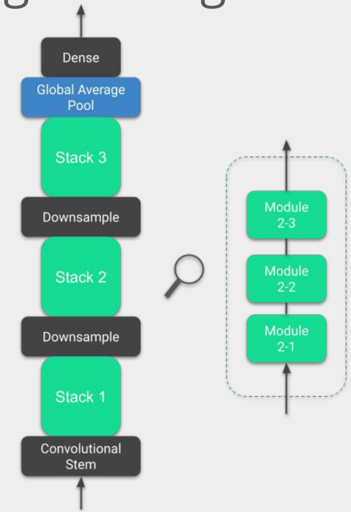
Experiments: Fashion-MNIST (2/2)

Best network accuracy: **93.2%**

Network has 23 operations and 93 connections (too complex to visualize).

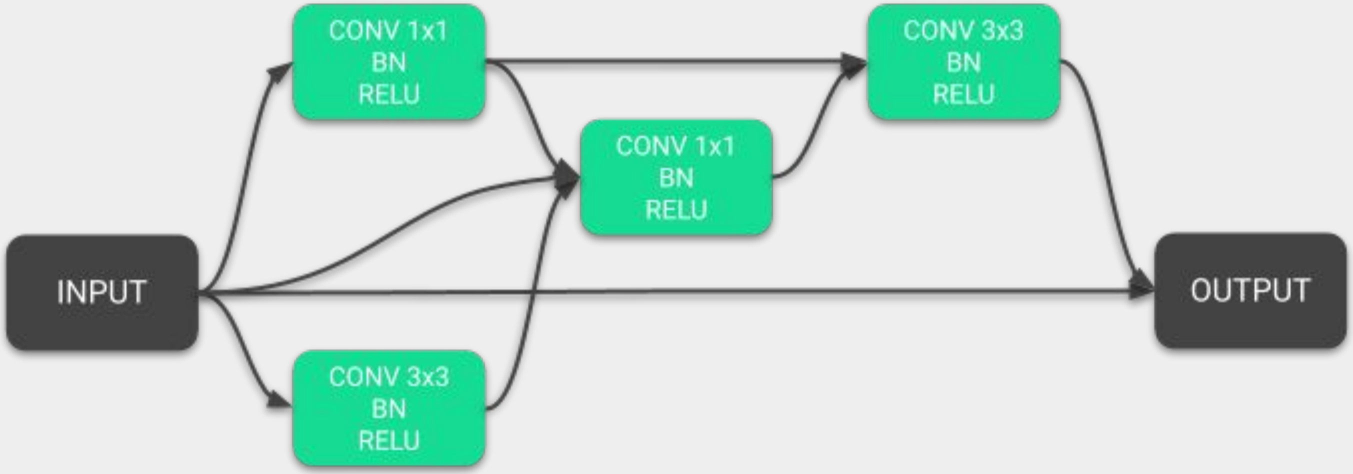
Experiments: NAS-Bench-101(1/2)

- NAS-BENCH-101 is a lookup table that contains the accuracy scores of all network topologies for the CIFAR-10 dataset in a constrained space.
- Network accuracy values can be obtained instantly by providing the structure of a module that has up to 7 nodes and 9 edges using 3 operations(1x1 & 3x3 convolutions, 3x3 max pooling).

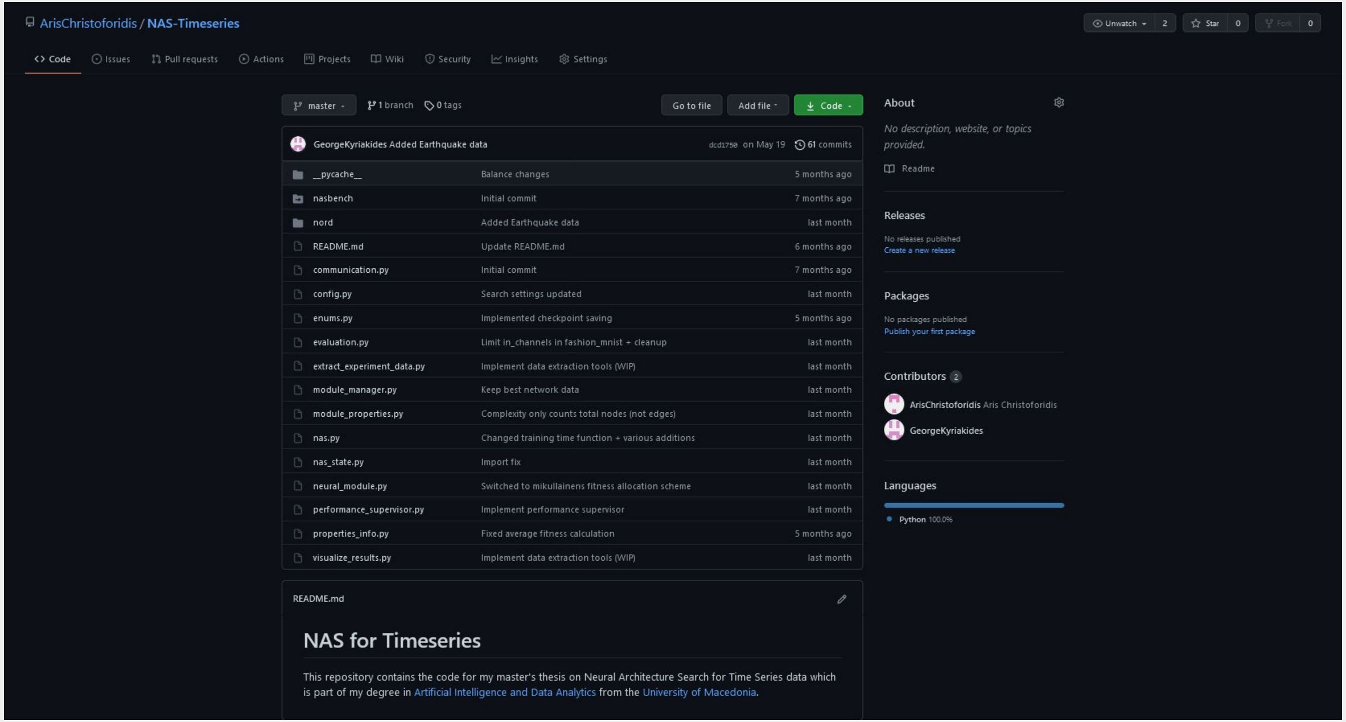


Experiments: NAS-Bench-101(2/2)

Best network accuracy: **94.8%**



Repository



https://github.com/ArisChristoforidis/NAS-Timeseries

Questions?