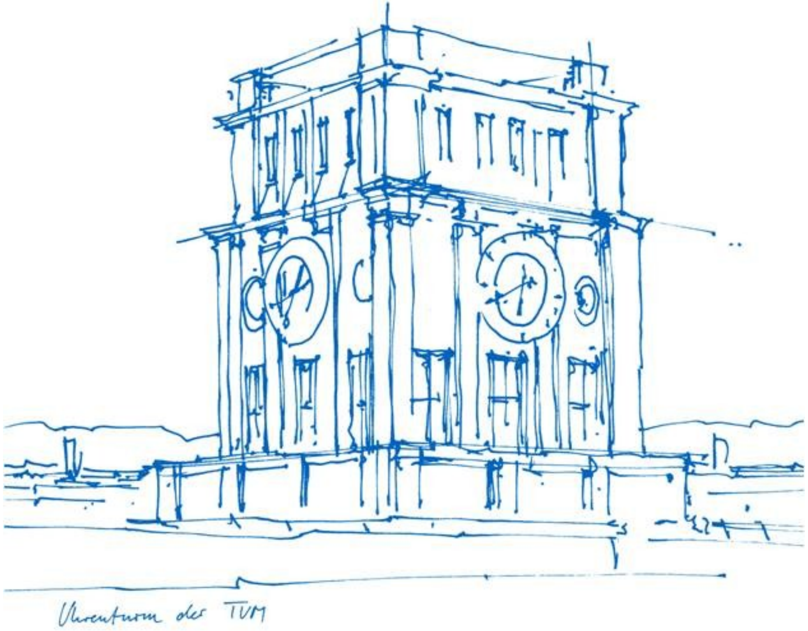
Modern Approaches for Heuristic Algorithms in Network Architecture Searches (NAS) and tinyML

Daniel Duclos-Cavalcanti

Advisor:

M.Sc. and M.Eng. Alex Hoffman



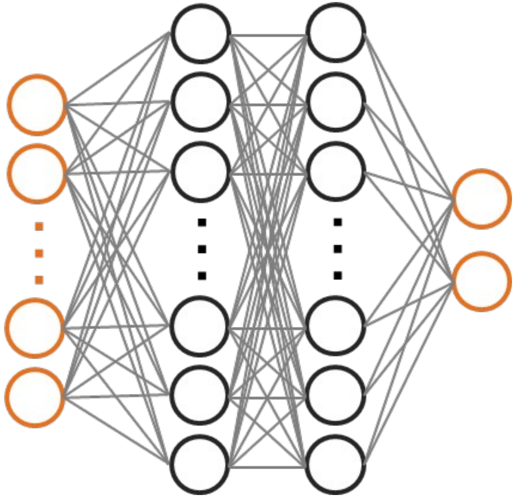
Wissenschaftliches Seminar VLSI Entwurfsverfahren

TUM SoSe 2022

Contents

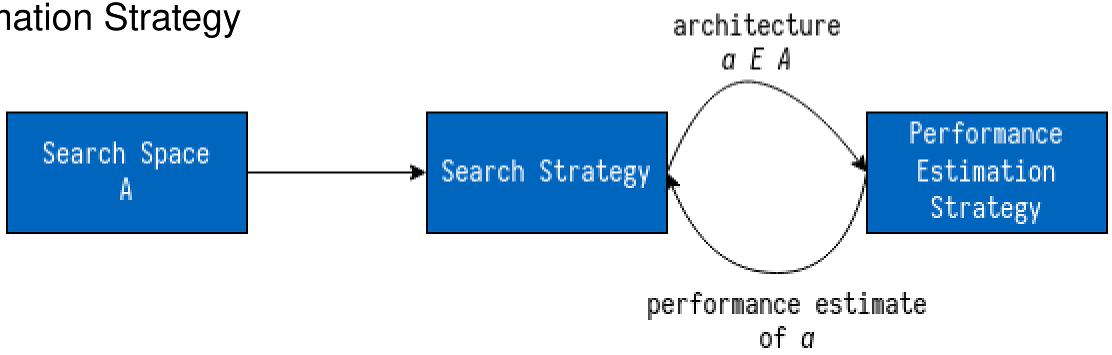
Introduction

- Deep Learning
  - Groundbreaking technology
  - Algorithms complexity increase
  - Architecture engineering and domain expertise
- Neural Architecture Search (NAS)



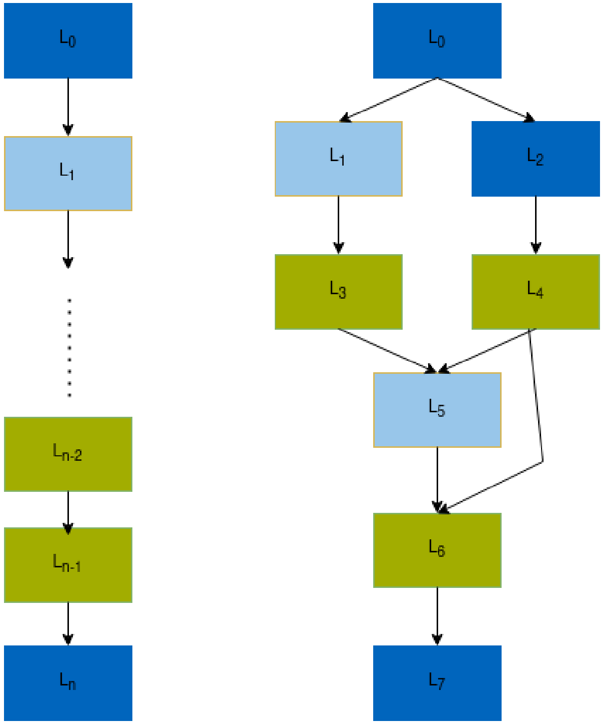
Background – Neural Architecture Search (NAS)

- NAS methods categorized by three dimensions:
  - Search Space
  - Search Strategy
  - Performance Estimation Strategy



NAS – Search Space

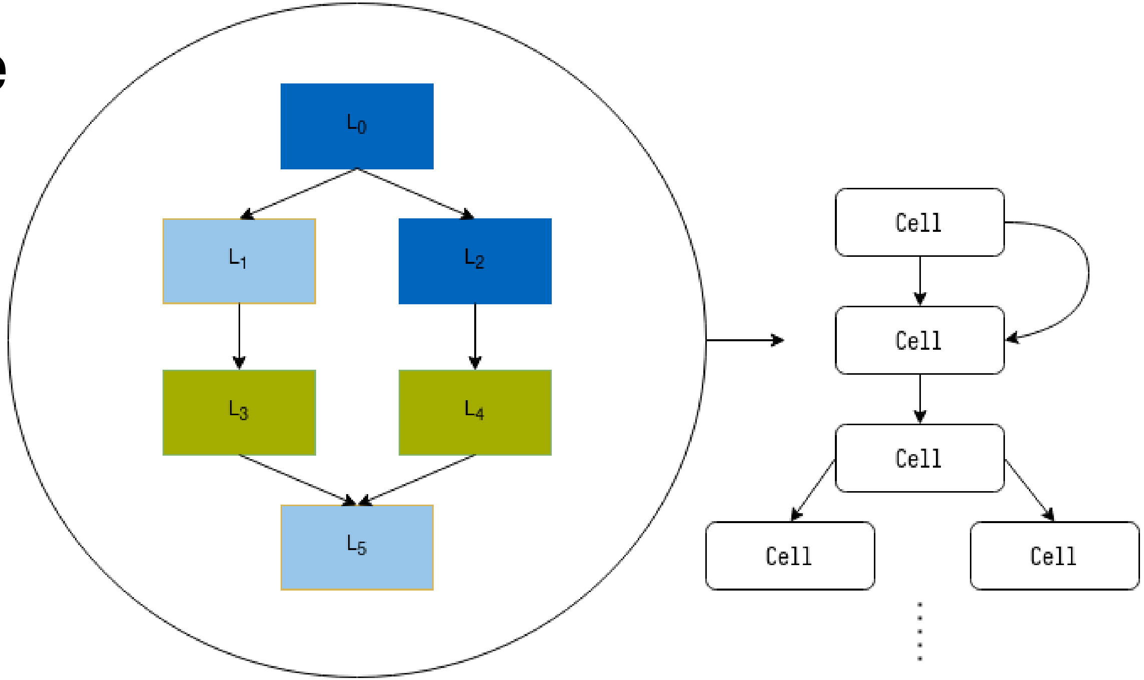
- Optimization problem
- Exploration x Exploitation trade-off
- Chain-Structured NN (a) x Multi-Branch NN (b)



(a) (b)

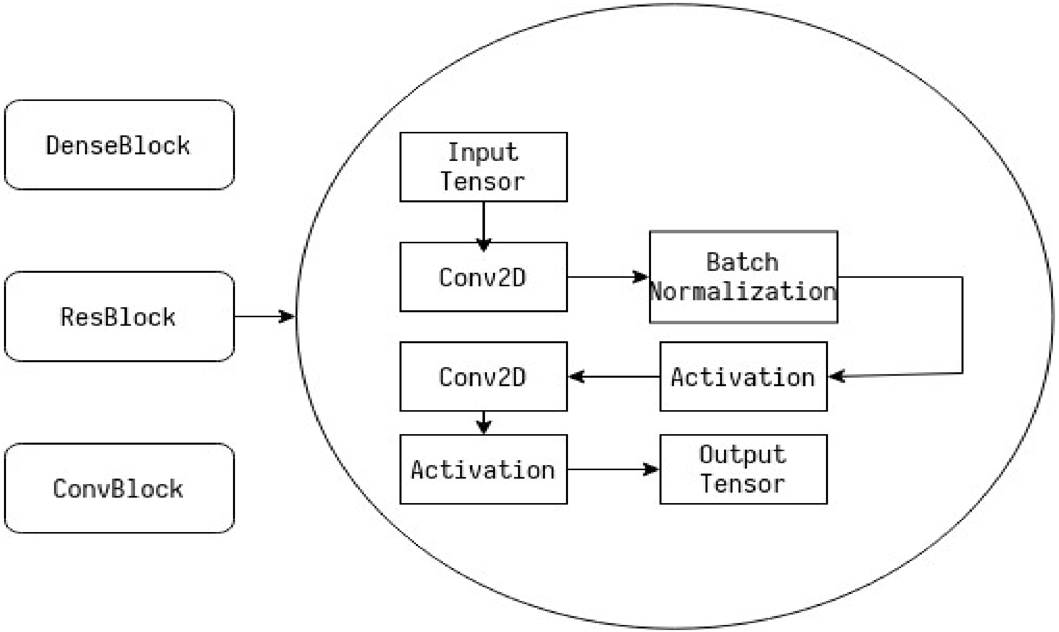
NAS – Search Space

- Optimization problem
- Exploration x Exploitation trade-off
- Cells x Blocks
- Macro x Micro architectures



NAS – Search Space

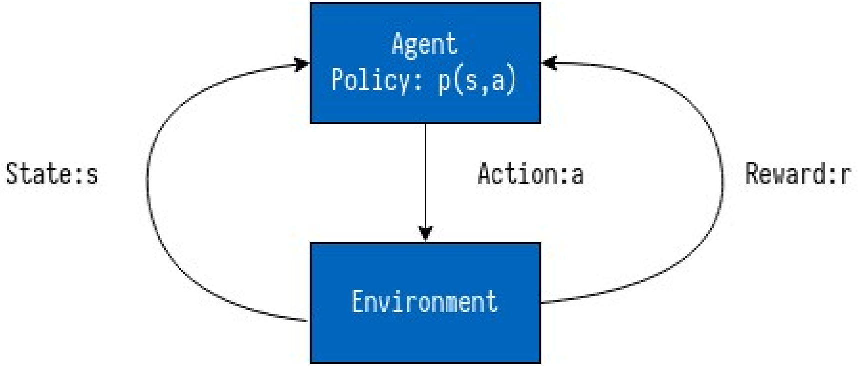
- Optimization problem
- Exploration x Exploitation trade-off
- Cells x Blocks
- Macro x Micro architectures



NAS – Search Strategy

- Reinforcement Learning (RL) - Based
  - Agent's action – generation of a neural architecture
  - Action space – search space
  - Agent's reward – performance estimation

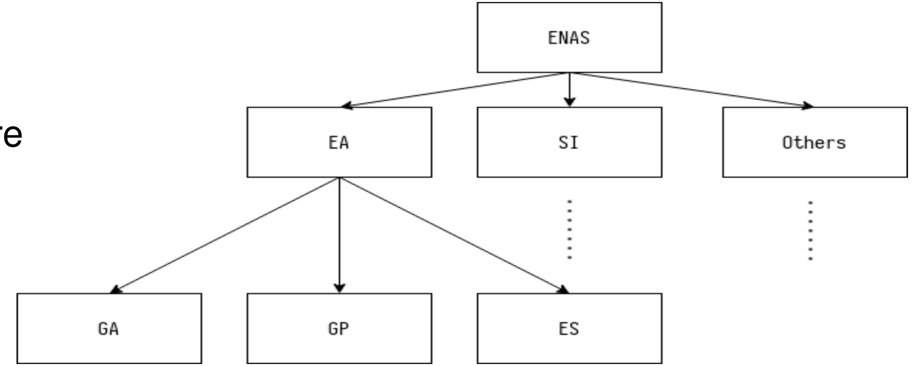
- Evolutionary Computation (EC) - Based
- Gradient-Based



NAS – Search Strategy

- Reinforcement Learning (RL) - Based
  - Agent's action – generation of a neural architecture
  - Action space – search space
  - Agent's reward – performance estimation

- Evolutionary Computation (EC) - Based
- Gradient-Based



NAS – Performance Estimation Strategy

- Bias vs Speed trade-off

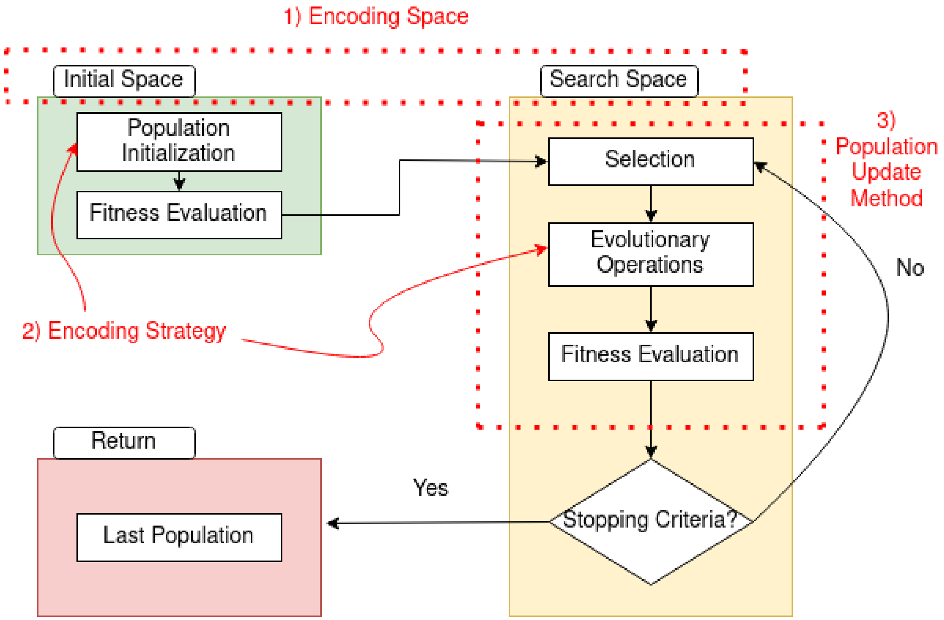
Speed-up Method	**How?**
Lower Fidelity Estimates	Shorter training time, training on subset of the data, training on downscaled data or with downscaled models.
Learning Curve Extrapolation	Performance is extrapolated after just a small number of epochs and then decided upon directly. Hyperparameters to predict promising architectures after partial learning. Extrapolate partial learning curves to predict and eliminate sub-optimal architectures.
Weight Inheritance	Weights from previously trained models passed down to new ones.
Weight Sharing or One-Shot Models	All architectures as subgraphs of a supergraph (one-shot model). Weights are shared between specific architectures. Only one-shot is trained.

State-of-the-Art – NAS Overview

- Reinforcement Learning (RL) - Based
  - Costly, hundreds even thousands of GPU days
- Gradiant Descent - Based
  - Faster and efficient
  - NAS and gradient-based not completely compatible
- Evolutionary Computation (EC) - Based
  - Easily applicable to non-convex optimization problems
  - Competetive models from trivial initial conditions
  - No human interaction

Evolutionary NAS (ENAS)

- Encoding Space
- Encoding Strategy
- Population Update



ENAS Algorithm Flowchart

ENAS – Encoding Space

- Initial Space + Search Space
- Initialization
  - Trivial Conditions
  - Rich Initialization
  - Random Initialization

- Bias vs Speed

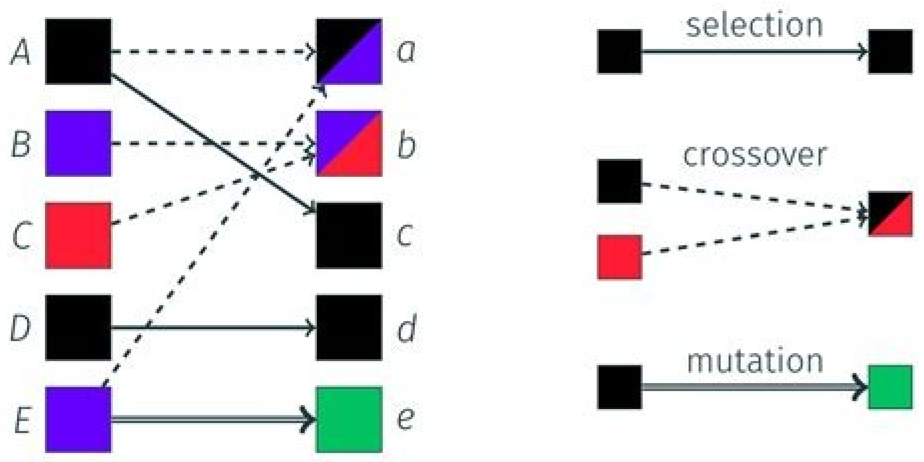
ENAS – Encoding Strategy

- How network architectures are encoded into individuals
- Fixed-length vs variable length
- Easier implementation vs greater discovery freedom

ENAS – Population Update Method

- Selection
  - Elitism
  - Discard worst or oldest
  - Roulette
  - Tournament Selection

- Evolutionary Operators
  - Mutation
  - Crossover



Source: https://minapecheux.com/website/2018/09/15/a-peek-at-genetic-algorithms/

State-of-the-Art – tinyML and NAS

- TinyML
  - Port AI algorithms to constrained devices
  - Training, pruning, quantizing and hardware design
  - Not optimal, mutual influence of HW and algorithm
- McuNET
  - TinyEngine and TinyNAS
  - Optimizes search space according to memory, energy consumption, storage and latency
- MicroNETs
  - Tailored differentiable NAS based on DARTS
  - Regularization term, balance between eFlash memory x SRAM x accuracy

Conclusions

- NAS is a complex optimization problem with great promise.
- NAS can be used to optimize architecture search for multi-objective goals.
- This proves useful within the tinyML paradigm:
  - Energy consumption
  - Latency
  - Memory Usage
  - Storage

Questions