Neural Architecture and Feature Search for Predicting the Ridership of Public Transportation Routes

Afiya Ayman[1], Juan Martinez,
Philip Pugliese[3], Abhishek Dubey[2], Aron Laszka[1]

[1]University of Houston, [2]Vanderbilt University, [3]Chattanooga Area Regional Transportation Authority

SMARTCOMP'22
June 21, 2022

This material is based upon work sponsored by the National Science Foundation under grants CNS-1952011, CNS-2029950, and CNS-2029952 and by the Department of Energy under Award Number DE-EE0009212.

Motivation

- Accurately predicting the occupancy of a scheduled transit-vehicle trip is crucial
- Higher prediction accuracy can be achieved by fine-tuning the hyper-parameters of machine-learning models for each transit route
- Designing a predictor for each route-direction combination is laborious
  - Requires time and effort from machine learning experts
- We introduce a *Randomized Local Hyper-parameter Search* to fine tune the hyper-parameters and predictor variables of a deep neural network

Research Questions

RQ1: Does fine-tuning the architecture and features for a specific task improve performance?

RQ2: How much impact does the starting architecture of the randomized local search have on the end results?

RQ3: How well does the optimized architecture of one task perform when trained for other tasks?

Data & Prediction Problem

Data:

- Automatic Passenger Count: Recordings on boarding and alighting events
- Weather: temperature, humidity, etc. based on location and time

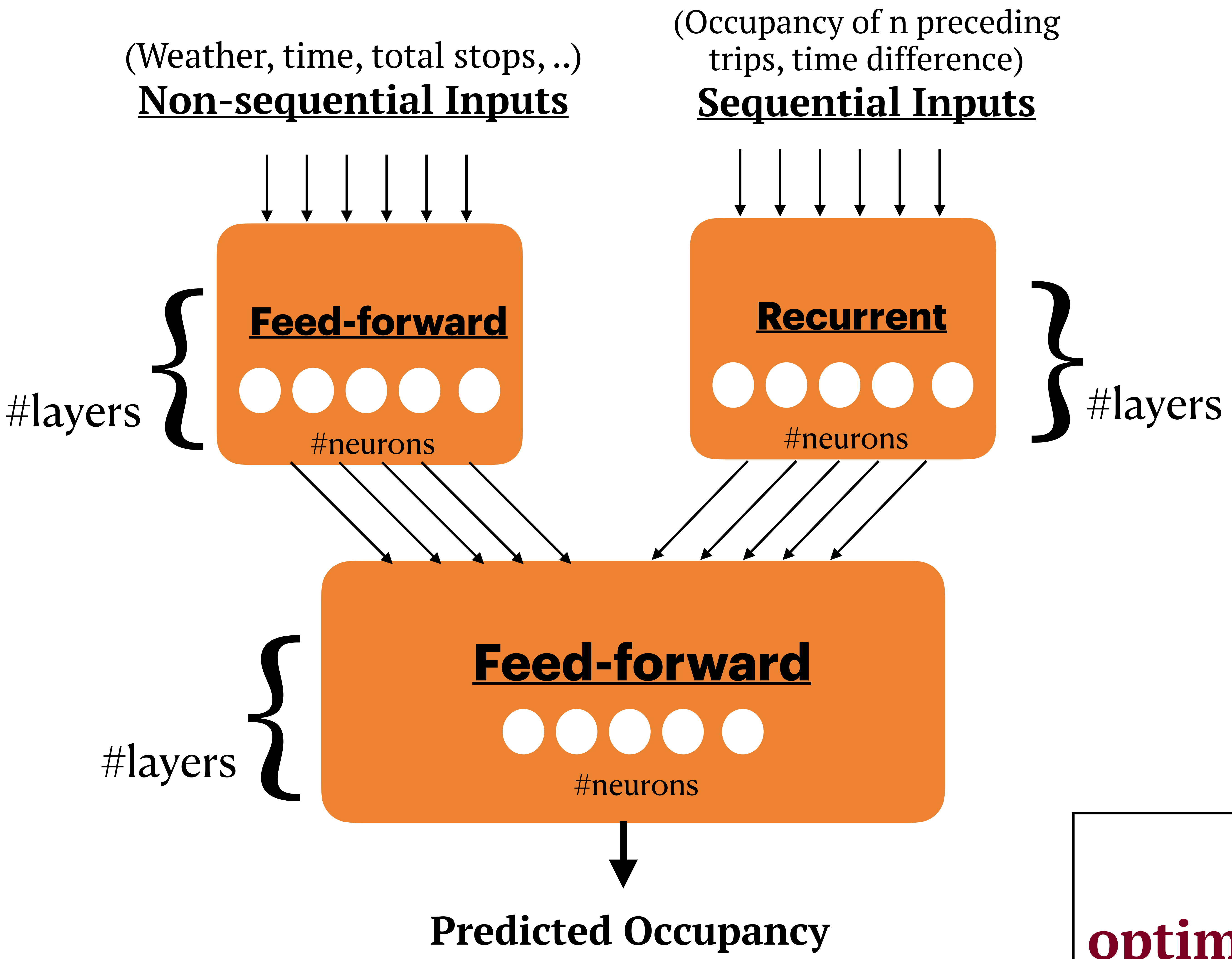
- Input: Aggregated information about each trip in a particular route-direction integrated with weather

- Target: Predicting maximum occupancy for a future trip on a particular route and in a particular direction
    - based on time and a few recent trips from a model trained on historical data

- Input consists of both non-sequential and sequential features

Non-Sequential Features	Sequential Features
• Total number of stops in a trip • Time - Month, Time of day, Day of week (Monday, Tuesday, …, Sunday) • Weather - Temperature, Windspeed, Visibility, etc.	Maximum and median occupancy of n preceding trips and time difference between them and the future trip

Architecture Template for Occupancy Prediction



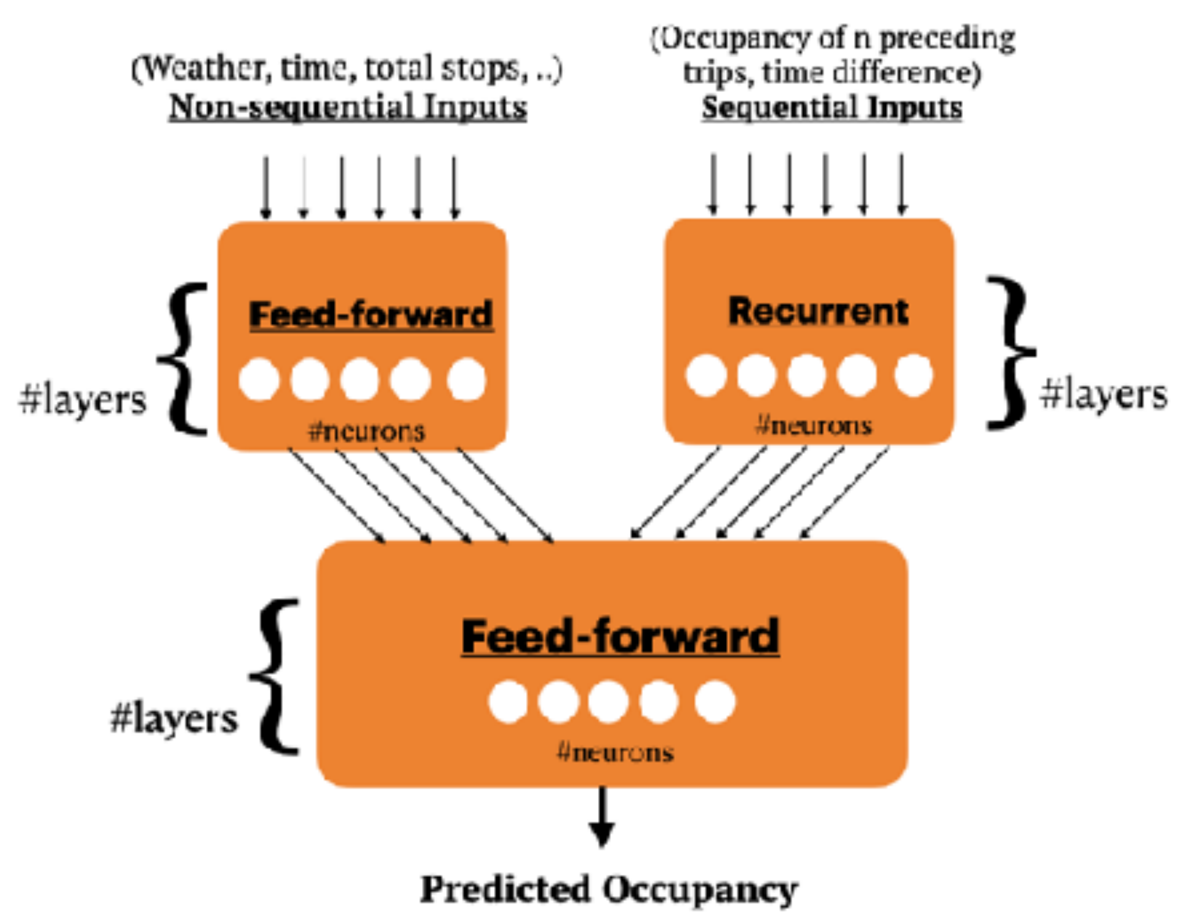
What is the optimal architecture??

Neural Architecture and Feature Search for Occupancy Prediction

- We propose an *Architecture and Feature Search* to fine tune the feature set and architecture hyper-parameters

- **Objective:** Finding an architecture and set of features A that minimizes the prediction error l_{RMSE} and model complexity:

$$min_{A \in \Omega}(l_{RMSE} + \text{Model Complexity})$$

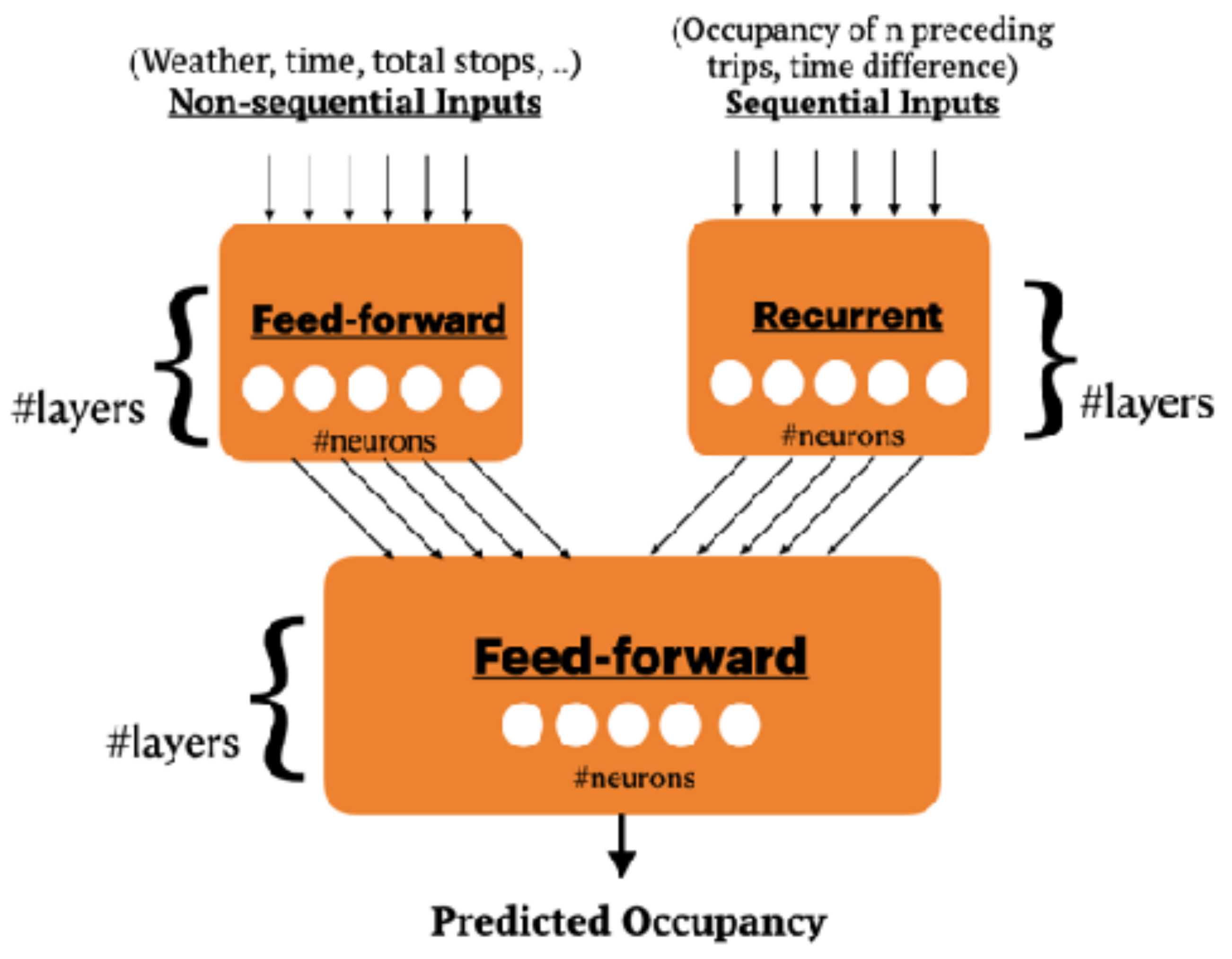


Neural Architecture and Feature Search for Occupancy Prediction

- **Search Space, Ω**

Includes hyper-parameters, $\mathcal{HP}$ for both architecture and feature set

Architecture hyper-parameter, h	Feature hyper-parameter, $\mathcal{F}$
• Number of layers, $\mathcal{L}$ in different modules • Number of neurons, $\mathcal{N}$ in each layer of different modules • Learning Rate, α for the model	• Non-sequential and sequential features to include



- **Randomized Local Search**

Start → Mutation → Evaluation → Accept/ Reject

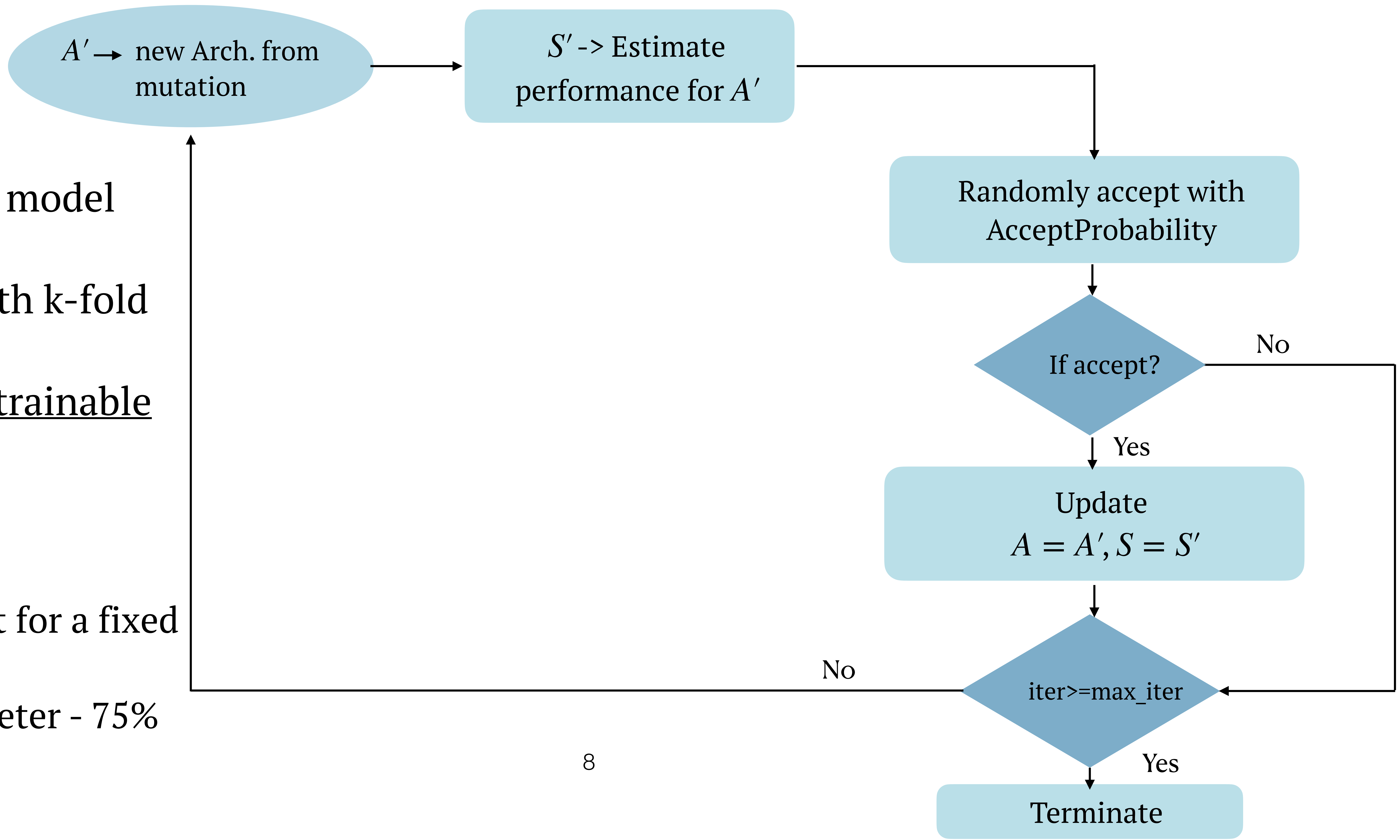
Mutation: Iteratively generate random neighbors in Ω

Accept/ Reject: Based on performance, accept it with some random probability

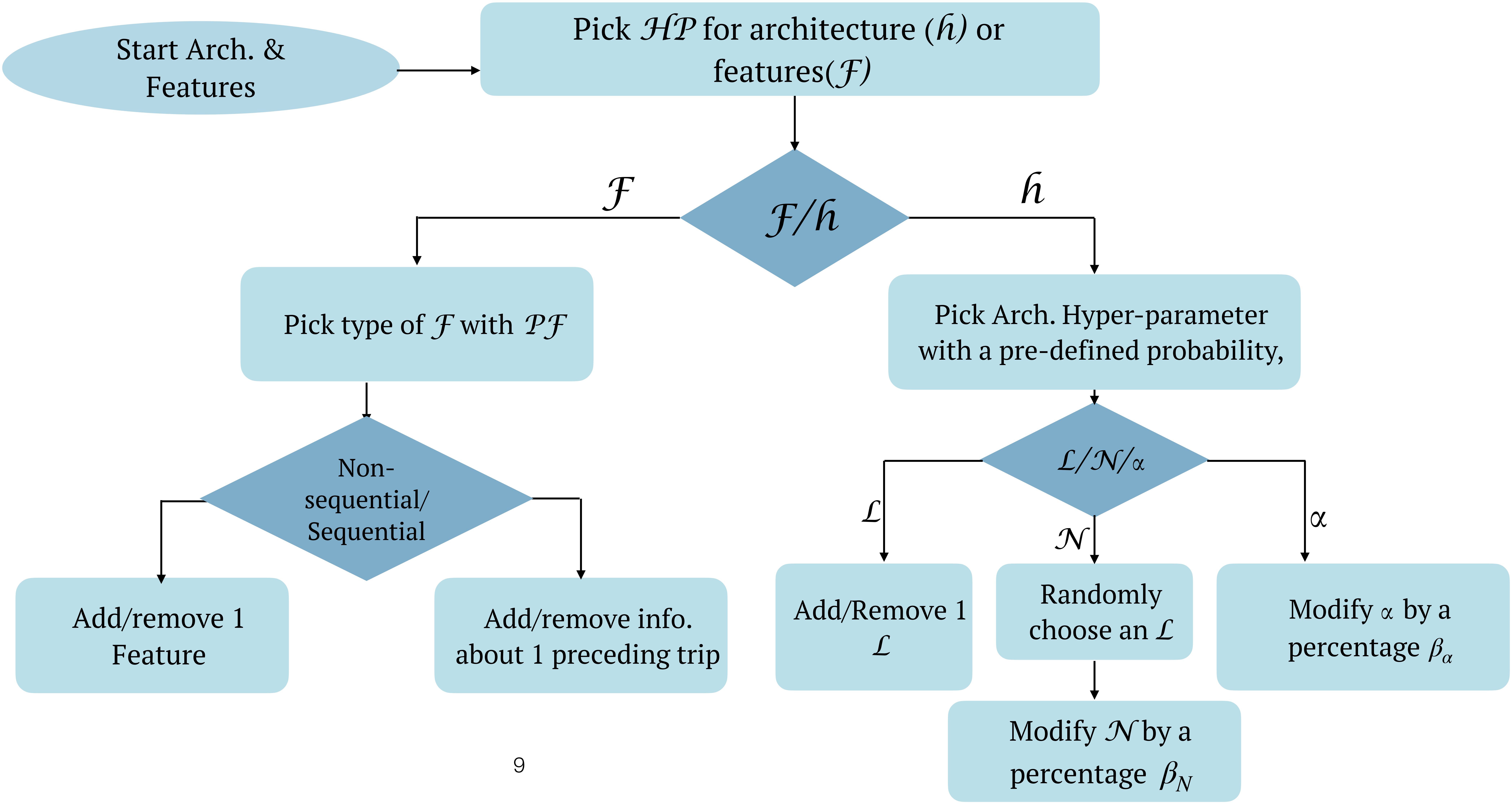
Neural Architecture and Feature Search: Estimating Prediction Loss

- Evaluation is based on both model loss and complexity
  - Loss = <u>RMSE</u> obtained with k-fold cross validation
  - Complexity = <u>number of trainable parameters</u>

- Randomized search will repeat for a fixed number of iteration
  - architecture's hyper-parameter - 75%
  - predictor variables - 25%



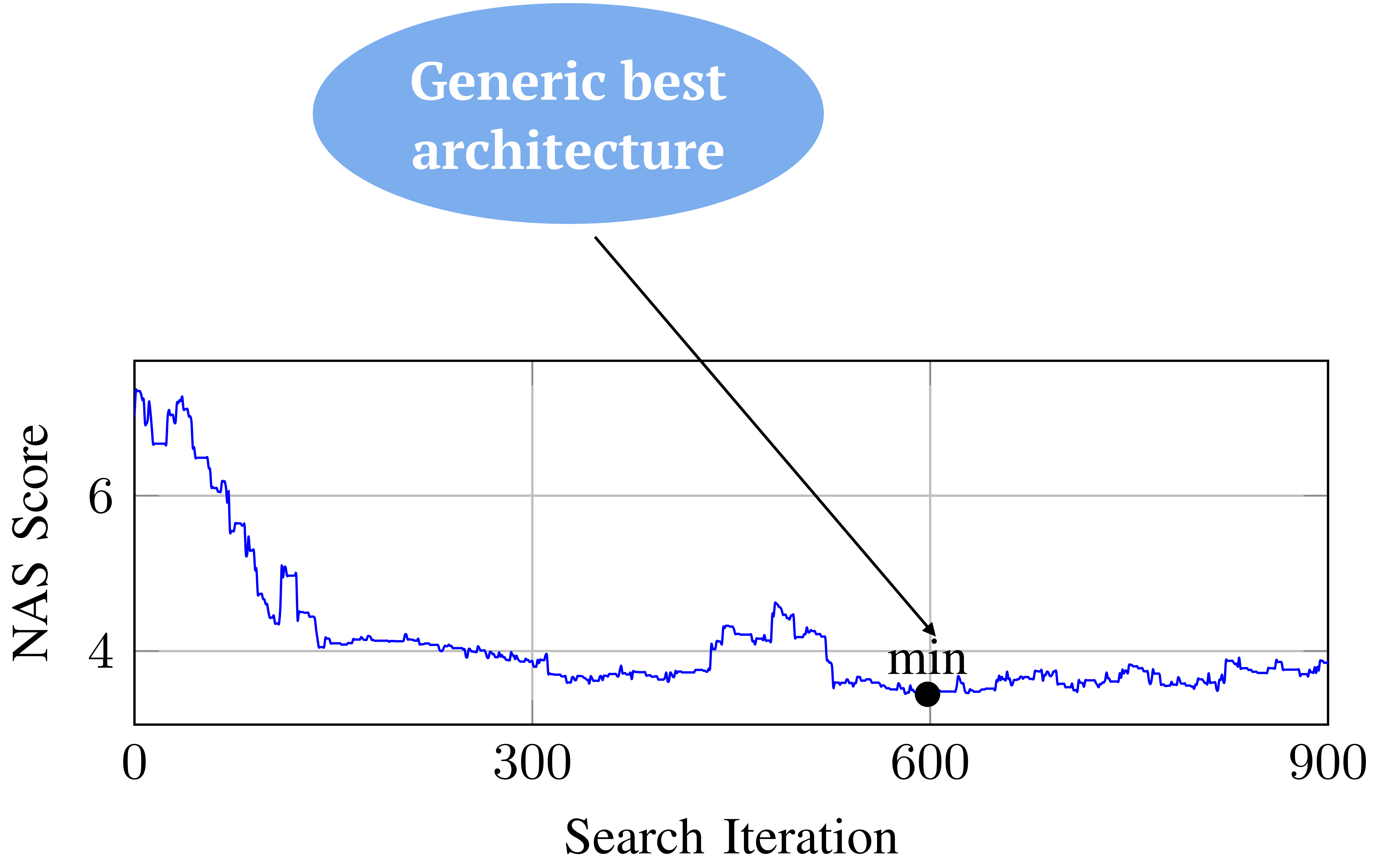
Neural Architecture and Feature Search: Single Mutation Step



Experimental Setup

Dataset

- APC data for Chattanooga, TN
    - Trips in total 23 routes in both direction
- Dataset timespan: 2 years (2019-2021)
- Algorithm is evaluated on 10 diverse tasks, i.e., route-direction combination
    - considering number of trips, average occupancy, variance, etc.



Architecture and Feature Search scores for all the tasks combined

Results

RQ1: Task-specific vs Generally Optimized Architecture

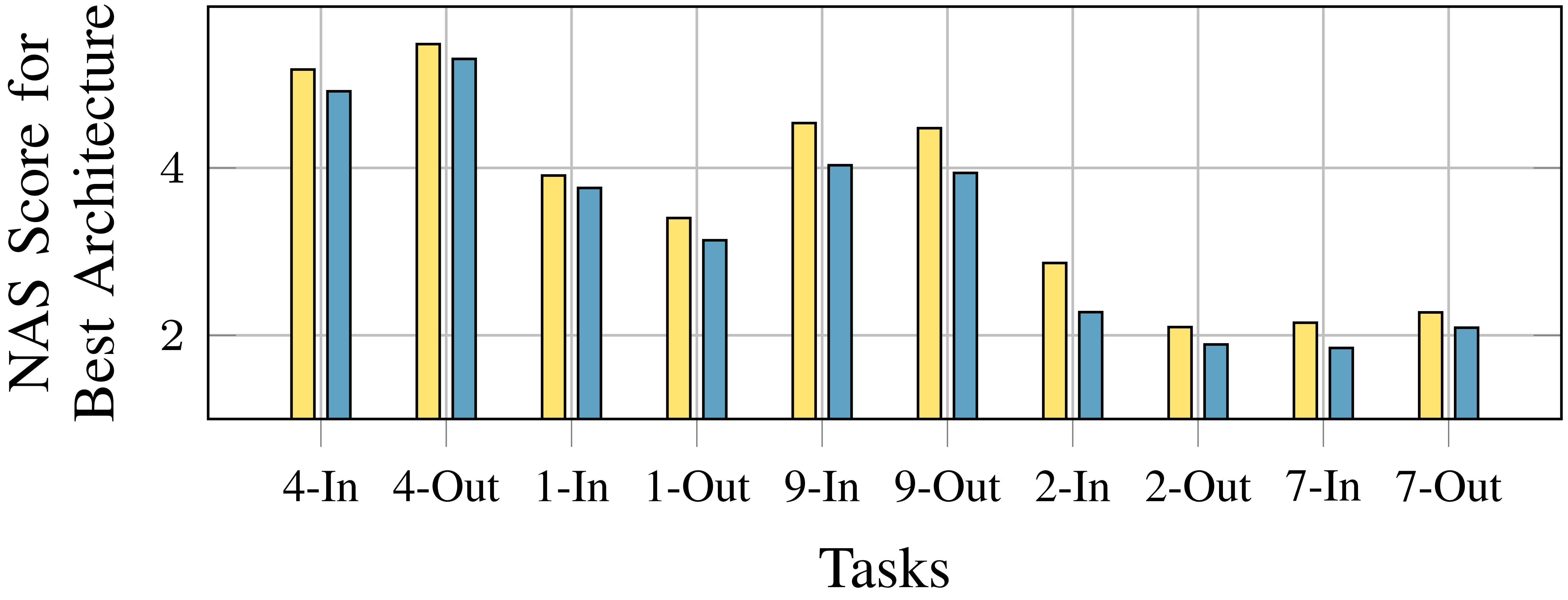


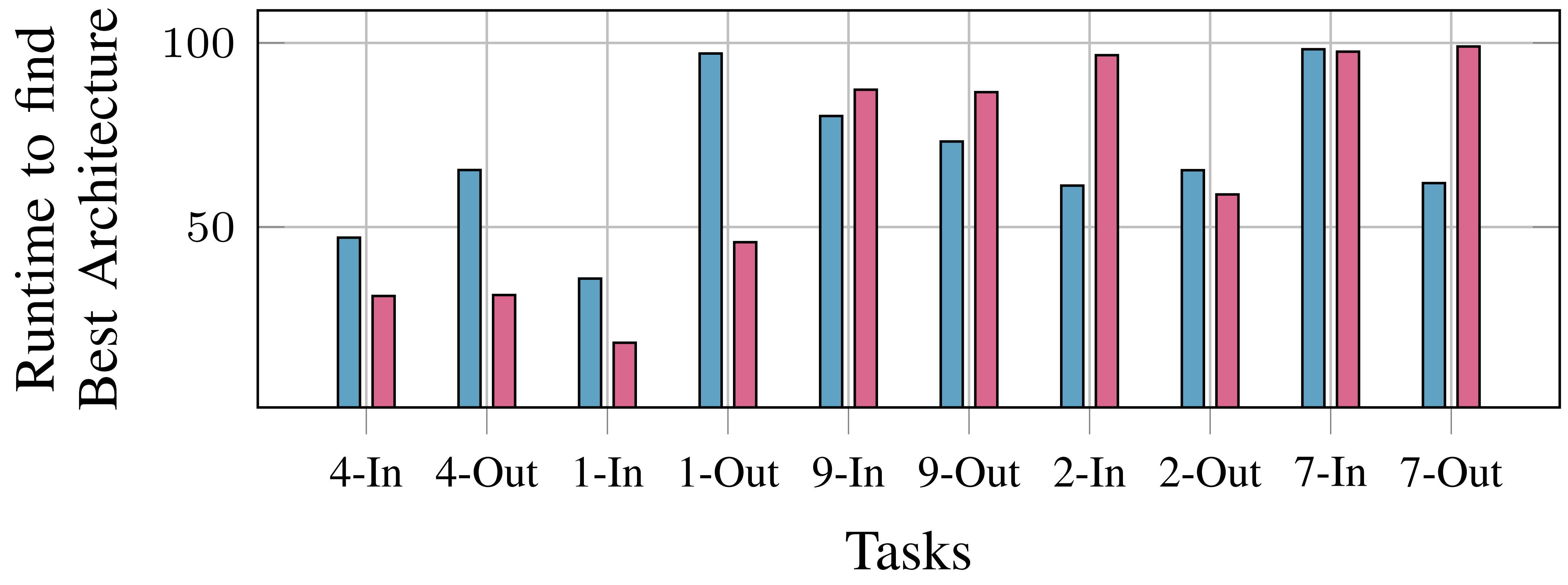
Fig: Comparison between architecture that were found by generic (yellow) and task-specific searches (blue) based on NAS score for each specific task

Results

RQ2: Starting Architecture of Task-Specific Search

Lower = better

Runtime

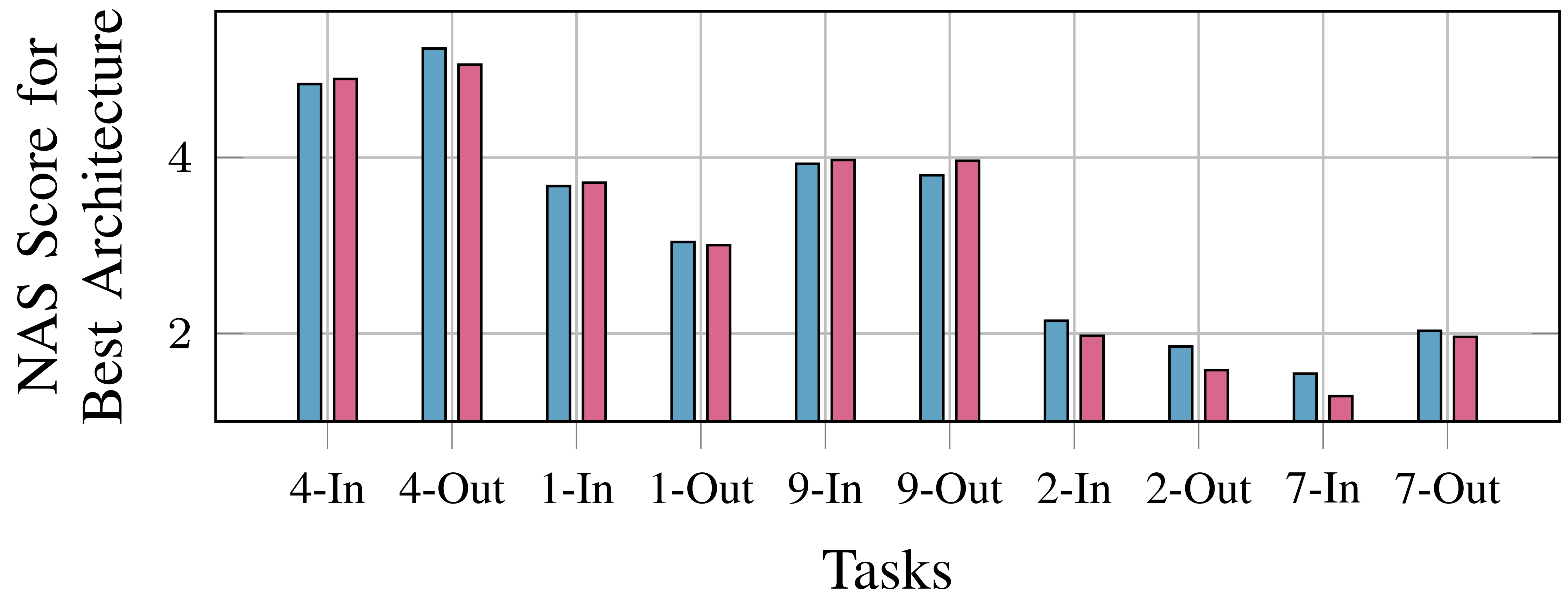


Runtime for finding the optimal architecture for different tasks from-

- Hand-designed start architecture (blue)
- Best generic architecture (purple)

Hand Designed avg. 68.6%
Optimized avg. 65.3%

NAS Score



NAS scores attained for specific task by searches from -

- Hand-designed architecture (blue)
- Best generic architecture (purple)

Hand Designed avg. 3.21
Optimized avg. 3.14

Results

RQ3: Comparison among Architectures Optimized for Specific Tasks

NAS Scores for Models Trained for Various Tasks using Architectures Optimized for Different Tasks

Optimized Arch. \ Task	4 Inbound	4 Outbound	1 Inbound	1 Outbound	9 Inbound	9 Outbound	2 Inbound	2 Outbound	7 Inbound	7 Outbound
4 Inbound	**4.96**	5.30	3.85	3.14	4.66	4.62	2.66	1.94	1.89	2.17
4 Outbound	4.91	**5.09**	3.98	3.14	4.39	4.33	2.68	1.95	1.89	2.19
1 Inbound	5.69	5.81	**3.79**	3.41	4.88	4.48	2.77	2.03	2.07	2.85
1 Outbound	4.94	5.24	3.91	**3.03**	4.37	4.58	2.66	1.94	1.88	2.14
9 Inbound	5.04	5.26	3.95	3.56	**4.52**	4.50	2.75	2.08	1.97	2.37
9 Outbound	5.12	5.42	3.97	3.42	4.64	**4.50**	2.58	2.03	1.96	2.37
2 Inbound	5.06	5.36	3.95	3.18	4.47	4.19	**2.34**	1.57	1.72	2.23
2 Outbound	4.96	5.27	3.81	3.07	4.28	4.21	2.41	**1.72**	1.56	2.15
7 Inbound	5.06	5.26	3.90	3.15	4.17	4.10	2.20	1.69	**1.54**	2.25
7 Outbound	5.58	5.91	3.81	3.38	4.71	4.73	2.64	1.91	1.93	**2.07**
Generic NAS	5.17	5.58	4.02	3.37	4.56	4.61	2.89	2.10	2.17	2.32

Darker red = worse performance
Darker green = better performance
Diagonal cells -> model scores trained for tasks using their corresponding optimized architecture

Conclusion

- Improving prediction accuracy by fine-tuning machine-learning architectures for each transit route in each direction is possible
- We proposed a framework for ***neural- architecture and feature-set search***
    - Alleviates the need for fine-tuning by machine-learning experts
    - Significantly reduces prediction error and model complexity based on real-world data

Thank you for your attention!!

Questions?

Afiya Ayman
afiyaayman.uh@gmail.com

Aron Laszka
alaszka@uh.edu