Mini-benchmarking approach to optimize evolutionary methods of Neural Architecture Search

"Moscow State University"

K. G. Khamitov[1], N.N.Popova[2]

[1]phd student

[2]assoc. prof. faculty Computational Mathematics and Cybernetics

Russian Supercomputing Days Moscow

Table of contents

Intro

The number of hyperparameters in modern ANN models constantly grows. It means that even if we use computing nodes with modern Tesla GPU's (V100/A100), it is still not enough for this amount of parameters neither inference nor training. Nowadays, many researchers utilize already pre-trained ANN models as a starting point, since it allows to decrease time to production and many analytical experiments. NAS – is one of the most popular methods designed to deal with such automatic tuning problems, but it requires a lot of computational power even using large-scale clusters. Here we present one approach to reducing the required amount of computational power reducing NAS search space if your task implies applying NAS methods to similar datasets or types of networks and problems.

Why it's so actual

- Reducing analytical works for networks that consumes several types of input(images, video, text).
- Obtaining high precision, using resource constraints in several use-cases(like mobile devices, network applications)

NAS methods

In this research we take into consideration two popular NAS methods

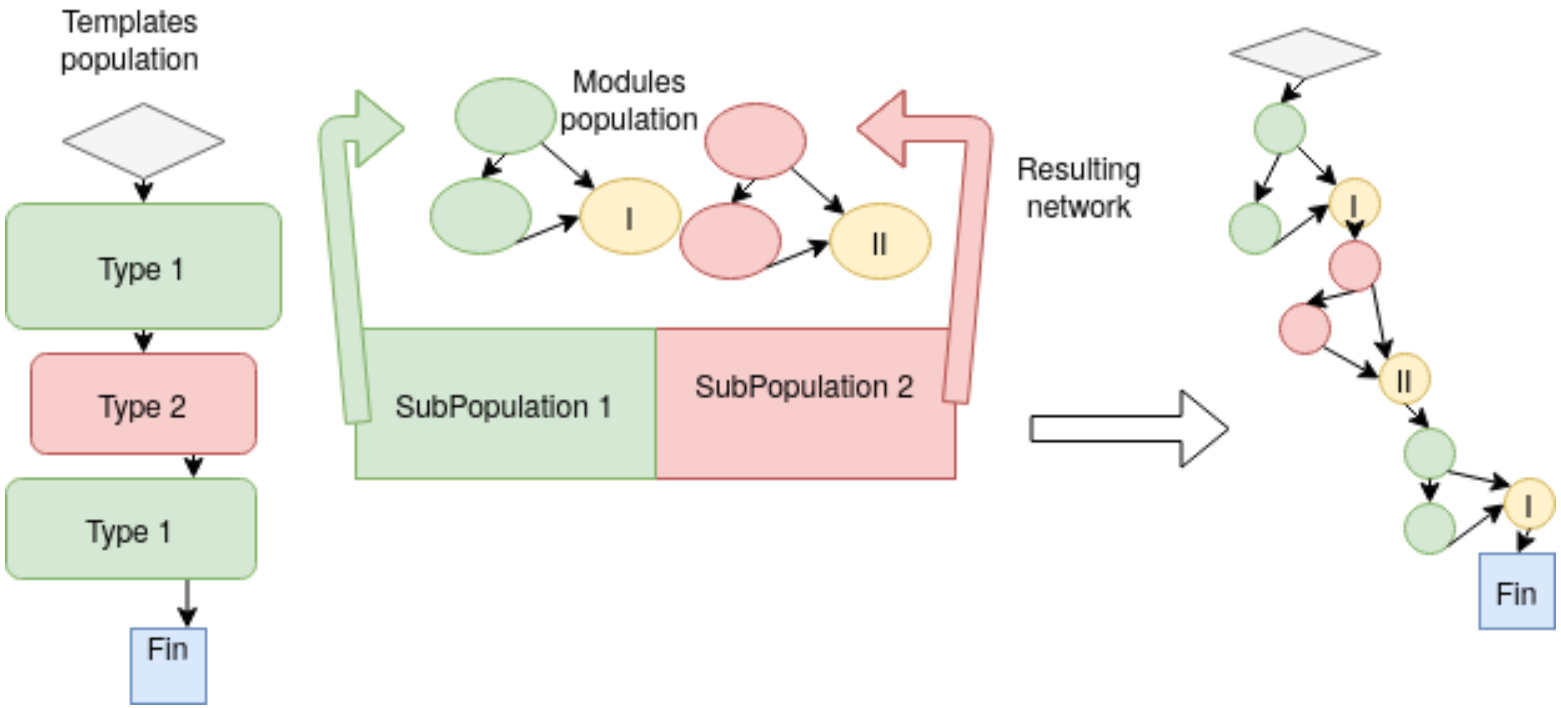
- CoDeepNEAT
- PetriDish

CoDeepNEAT

The CoDeepNEAT(CoEvolution DeepNEAT)[1] is a rather popular NAS method based on utilizing ideas of evolution methods to the NAS problems.

[1] Miikkulainen R., Liang J., Meyerson E., et al. Evolving deep neural networks // Artificial Intelligence in the Age of Neural Networks and Brain Computing. Elseiver 2019.

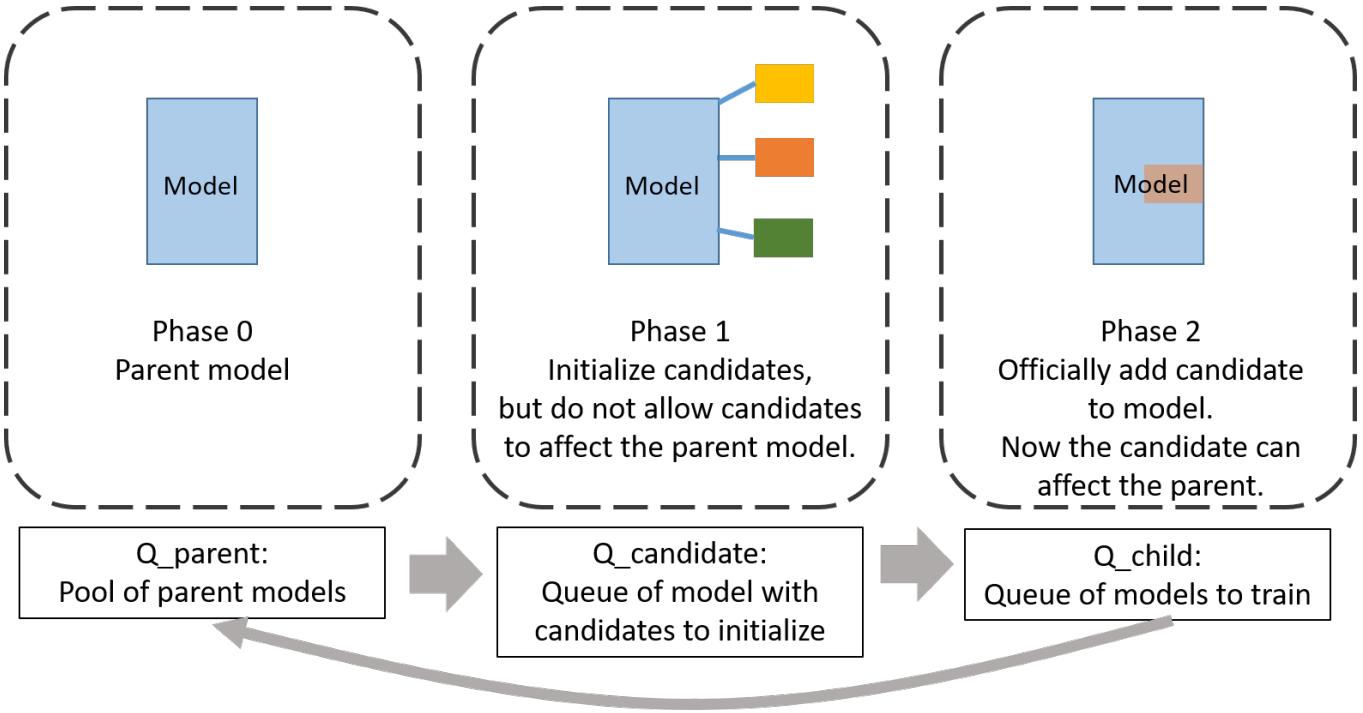
CoDeepNEAT

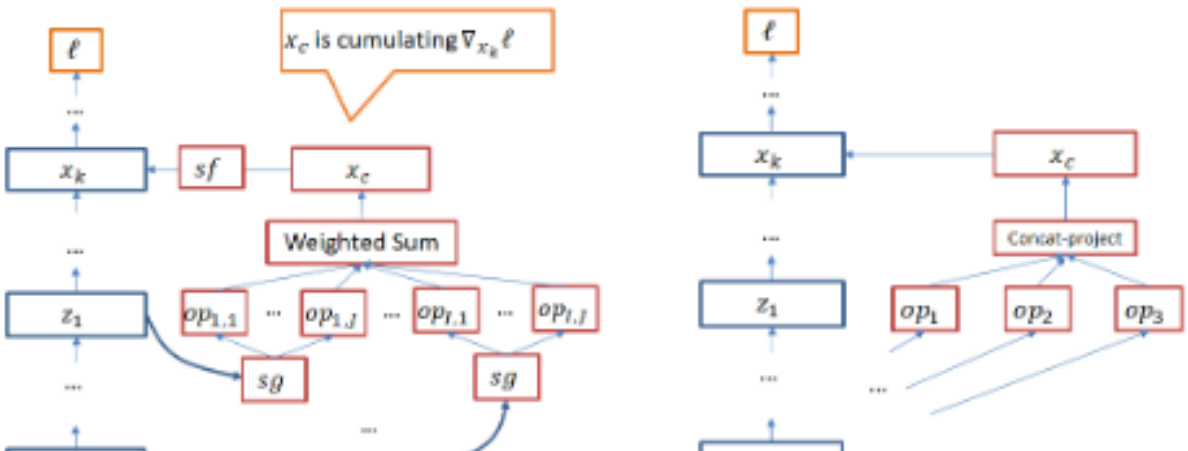

Templates population
Type 1
Type 2
Type 1
Fin
Modules population
I
II
SubPopulation 1
SubPopulation 2
Resulting network
I
II
I
Fin

PetriDish

The Petridish coevolution method is based on "Forward architecture search"for CNNs and Fully-connected blocks. It allows training the networks with modifications using special stop-forward layers.
Hu, Hanzhang, John Langford, Rich Caruana, Saurajit Mukherjee, Eric Horvitz, and Debadeepta Dey. "Efficient forward architecture search."arXiv preprint arXiv:1905.13360 (2019).

PetriDish


Model
Phase 0
Parent model
Model
Phase 1
Initialize candidates,
but do not allow candidates
to affect the parent model.
Model
Phase 2
Officially add candidate
to model.
Now the candidate can
affect the parent.
Q_parent:
Pool of parent models
Q_candidate:
Queue of model with
candidates to initialize
Q_child:
Queue of models to train


x_c is cumulating $\nabla_{x_k}\ell$
ℓ
x_k
sf
x_c
Weighted Sum
z_1
$op_{1,1}$
$op_{1,J}$
$op_{I,1}$
$op_{I,J}$
sg
sg
ℓ
x_k
x_c
Concat-project
z_1
op_1
op_2
op_3

Reasons of parallel implementation

- Large number of HPO parameters in modern tasks for transfer-learning and reapplying existing model.
- Parallel nature of evolutionary approaches .

Table of contents

Mini-benchmarking approach for improving NAS evolution convergence

- Perform mini-benchmarks with high variation of problems and types of ANNs.
- Using previously obtained results and best-synthesized models from the previous iteration, creating a set of constraints on search space in NAS and HPO-tune tasks.

List of problems for Mini-benchmarking DB.

Our modified mini-benchmarking dataset is based on NAS-BENCH-101 and NAS-HPO-BENCH, with some particular RNN-based tasks like UBFC-RPPG, CoLa. In order to increase variation in the search space and cover many aspects of different architectures.

- HRM time-series prediction using UBFC-RPPG data.
- Image classification (TinyImageNET, FashionMNIST, ImageNET-16-120).
- Tokenization in NLP Minified CoLa.
- 3D structured search NAS-Bench-HPO-Protein.

For Minified CoLa we use NLTK-based ANN for comparison.

Building up constraints

- Data preparation:
  - Full-NAS evolving without any constraints on the search space.
  - Evolutionary selection certain number of the best model for the particular task from Mini-benchmark.
- Classifier training:
  - Performing DeepNEAT-like encoding, with the corresponding folding level of nodes.
  - Building classification model from DeepNEAT graph representation and corresponding classes from mini-benchmarking stages.

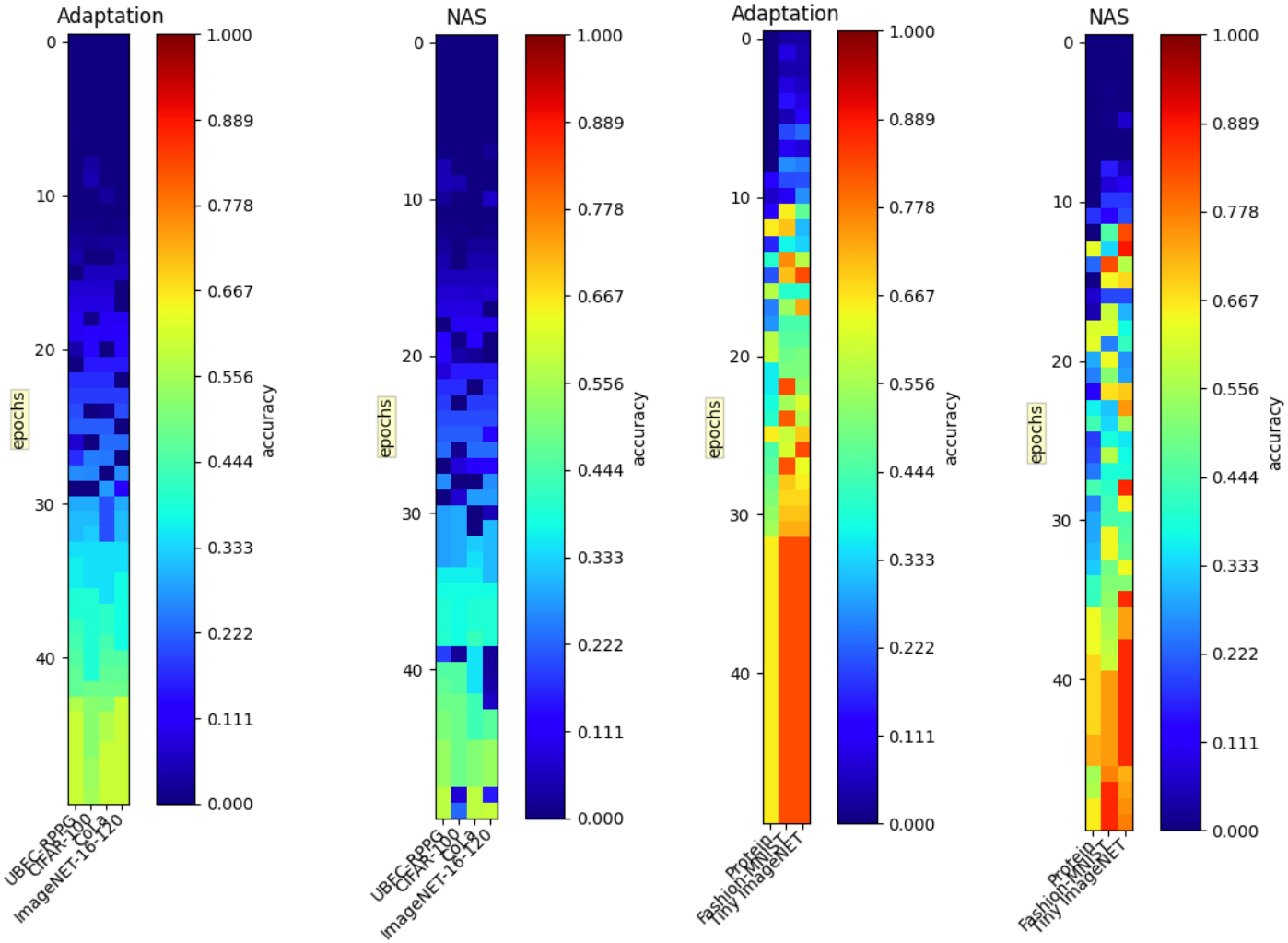
Experiments

- Tuning of constraints of search space for CoDeepNEAT using full dataset described above.
- Performing fine-tuning of constraints of search space for PetriDish using NAS-BENCH-HPO dataset.

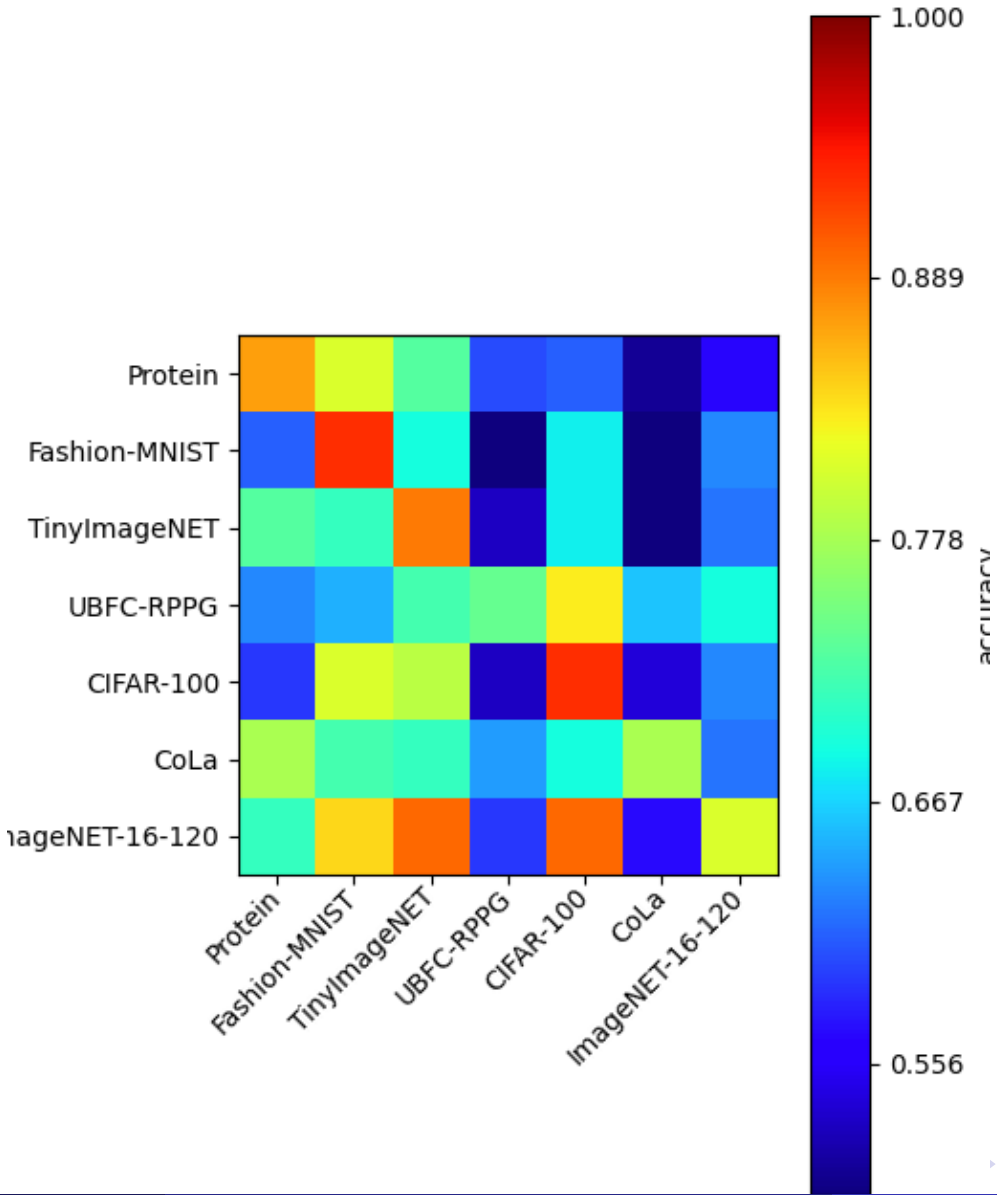
Experimental setup

- 4 x nodes of Polus cluster with 2 x IBM POWER8 processors with up to 160 threads, 256GB RAM and 2 x Nvidia Tesla P100 on each node.

Average accuracy per epoch in CoDeepNEAT in NAS and HPO-tuning modes



Average pairwise accuracy for presented benchmarking db



Best Accuracy at NAS-Bench-101 in NAS и HPO tuning modes

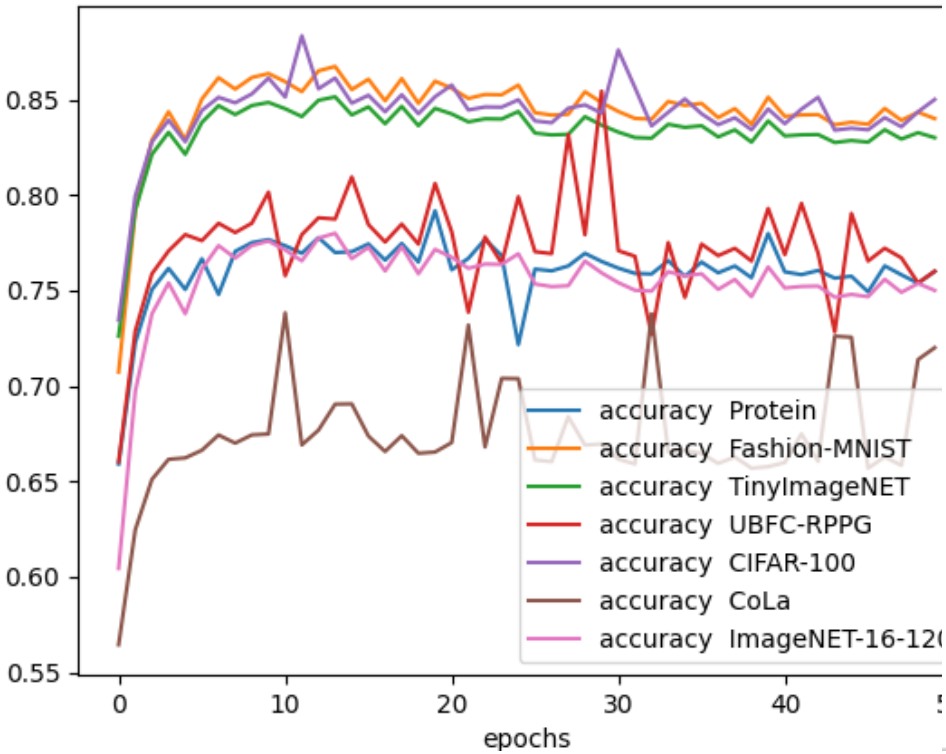


Рис.: Accuracy in NAS mode

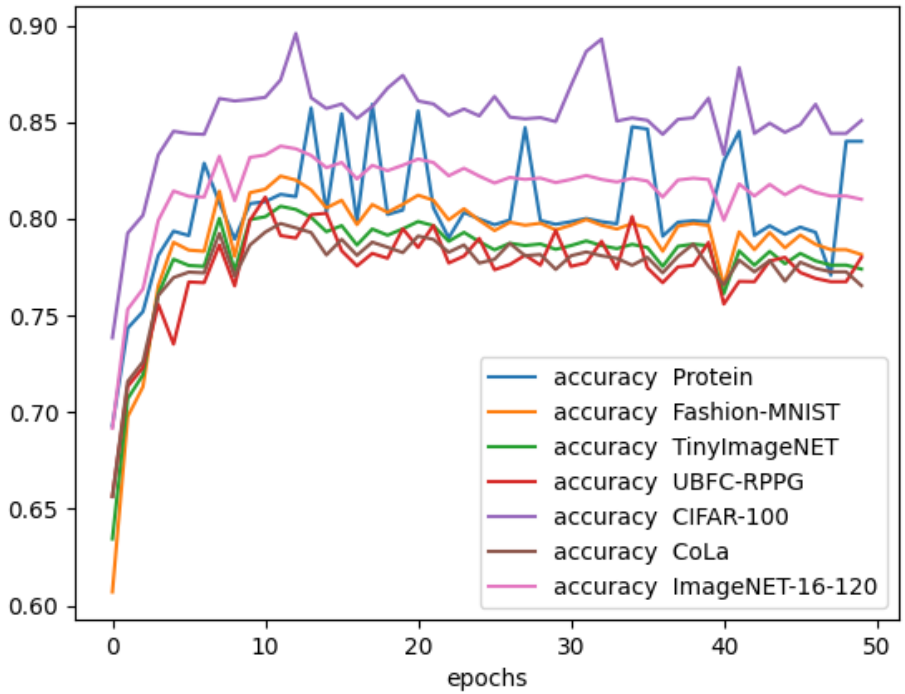
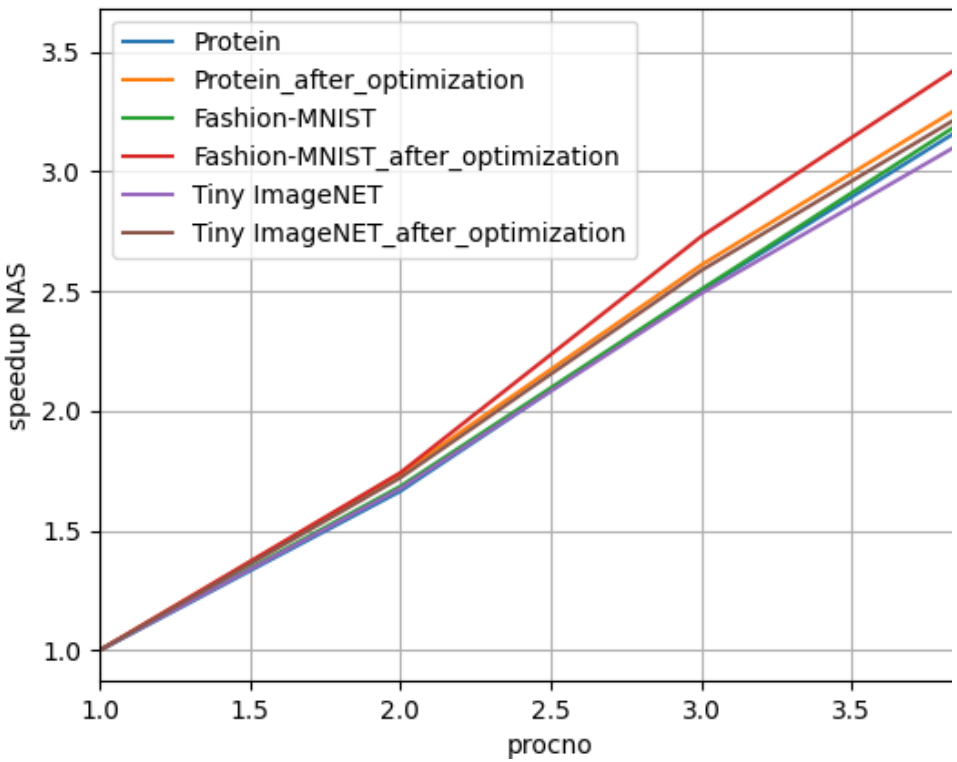
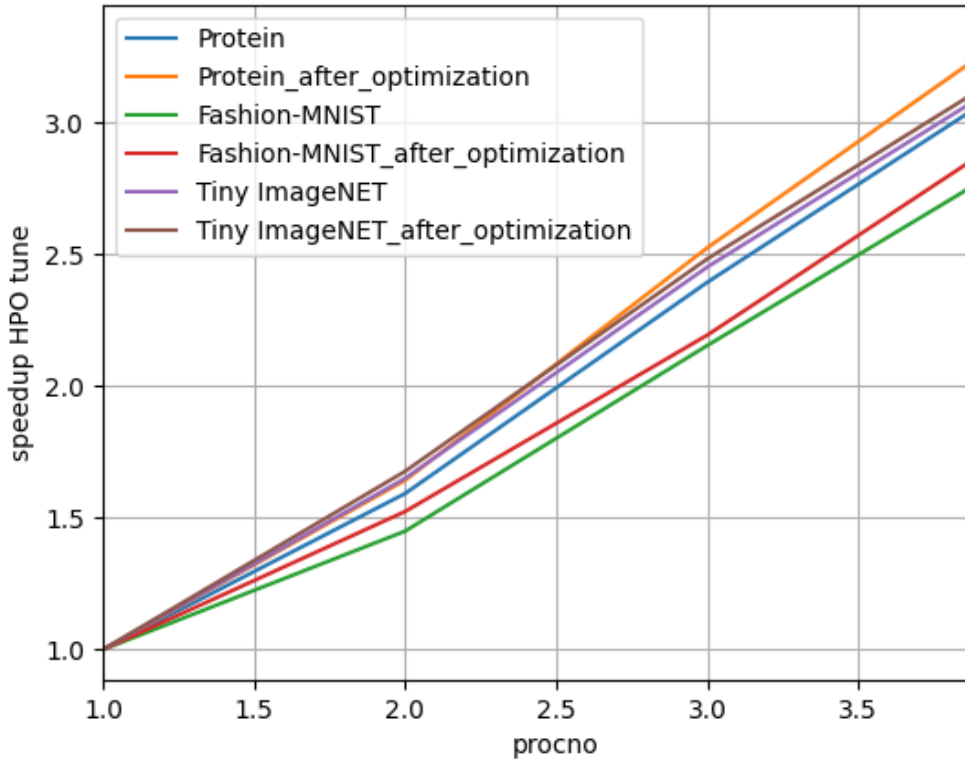


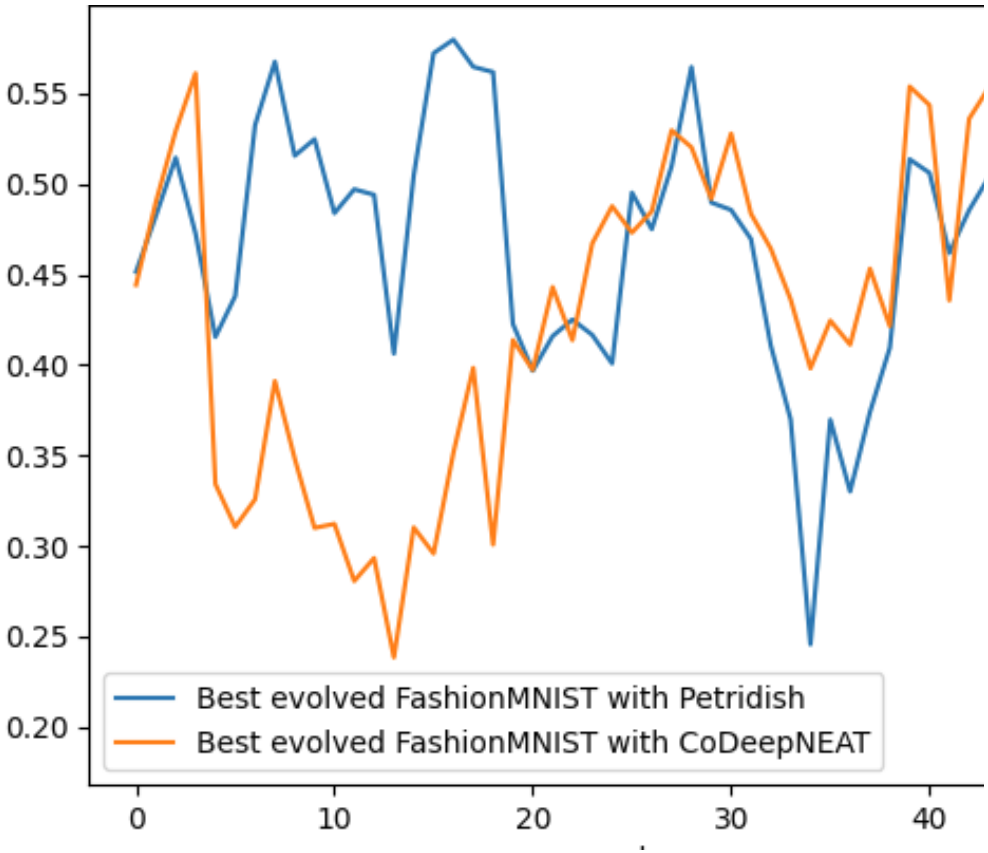
Рис.: Accuracy in HPO tuning mode

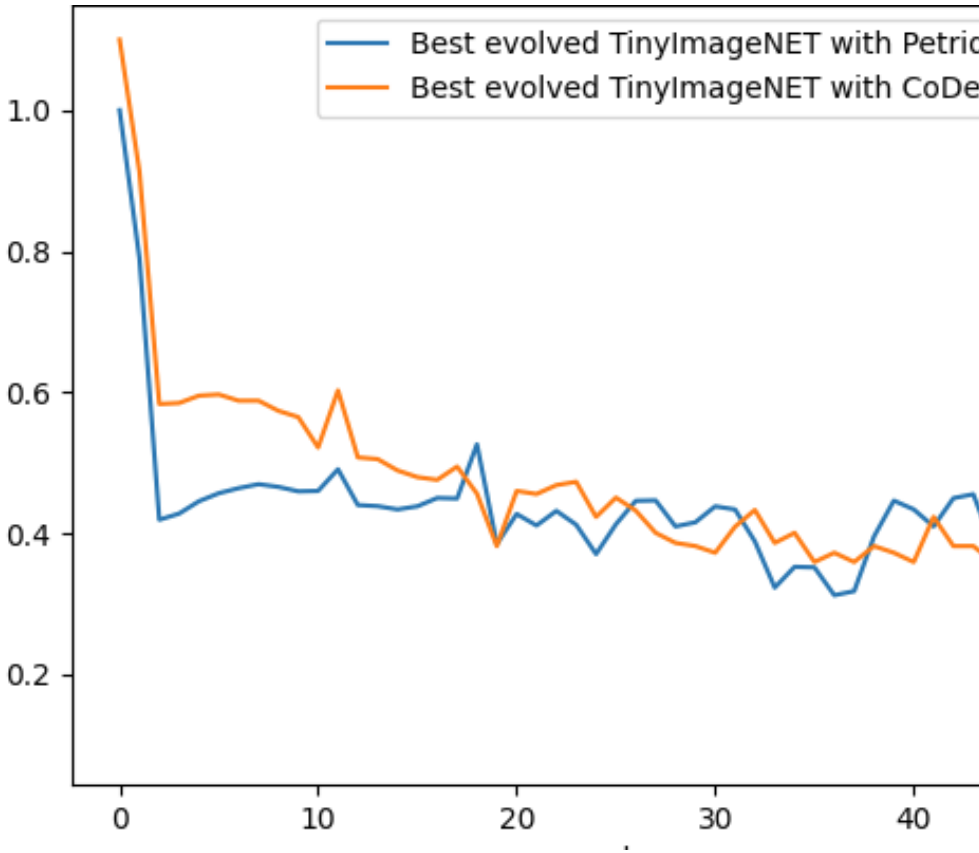
CoDeepNEAT speedup variation with new methods of testing.





Loss in NAS-Bench-101 at best networks during Petridish and CoDeepNEAT tuning





Variation of accuracy with DeepNEAT folding numbers

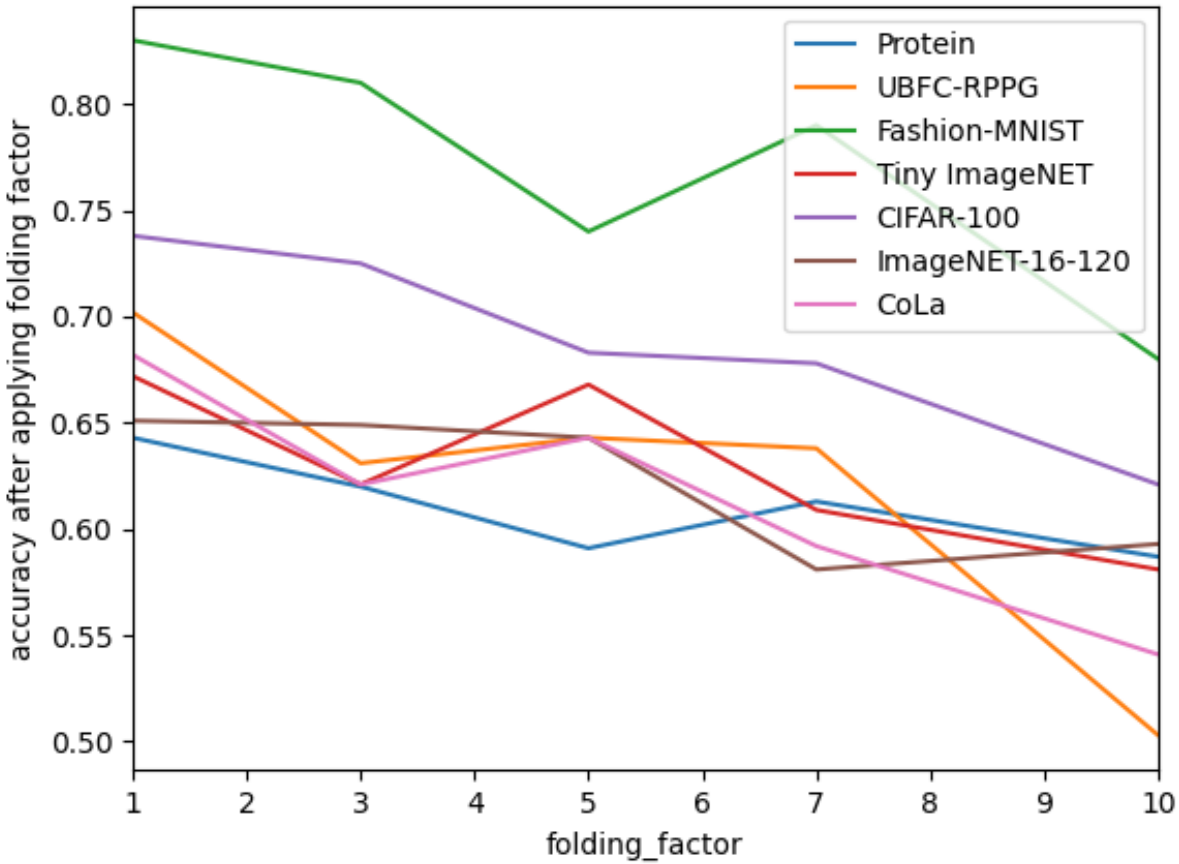
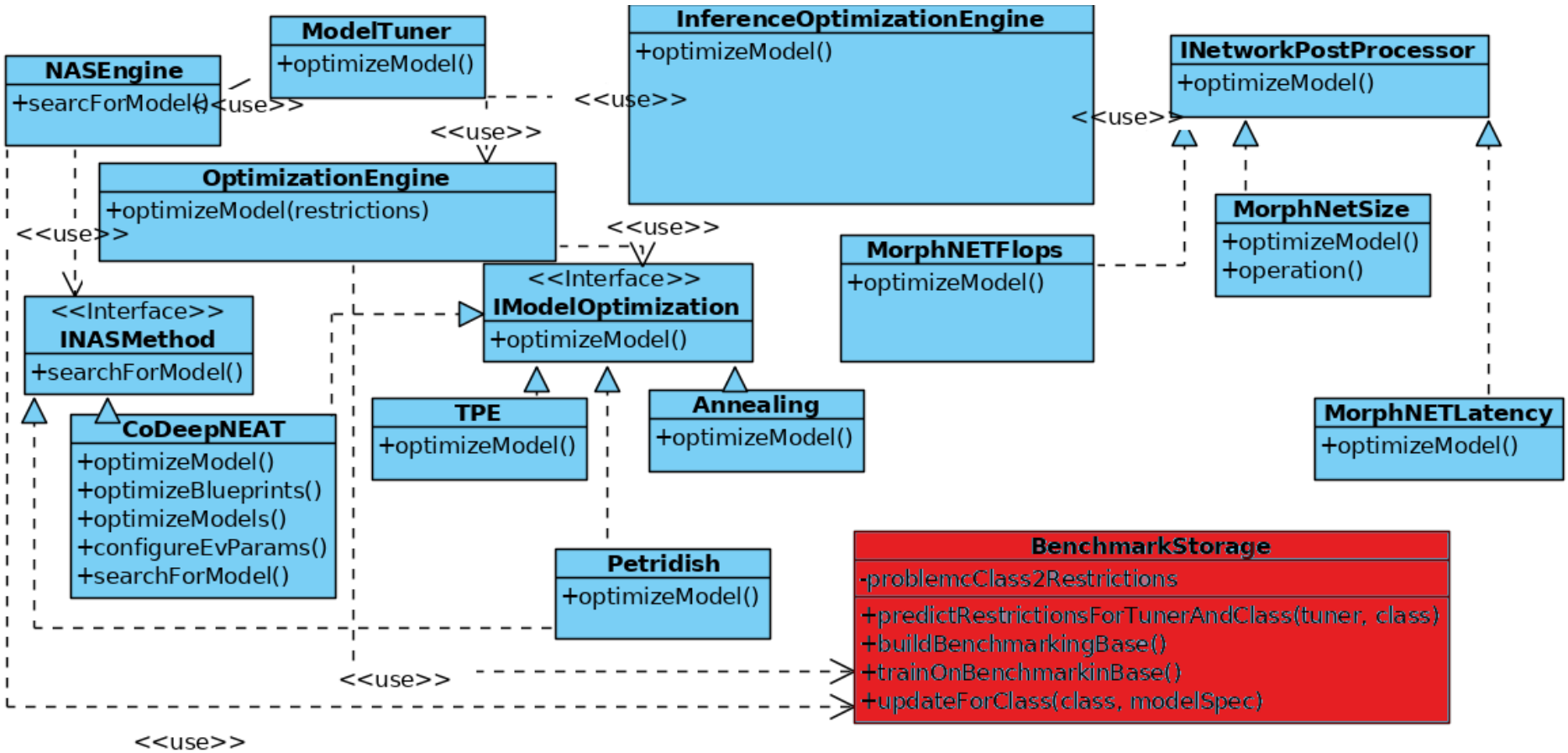


Рис.: Accuracy depends on folding numbers in DeepNEAT

New scheme of Semi-automatic tool for parallel NAS



Implementation and integration

- Implementing mini-benchmarking approaches for CoDeepNEAT/PetriDish using NNI.
- Adapting reclassification algorithm for reusing data information during reclassification.

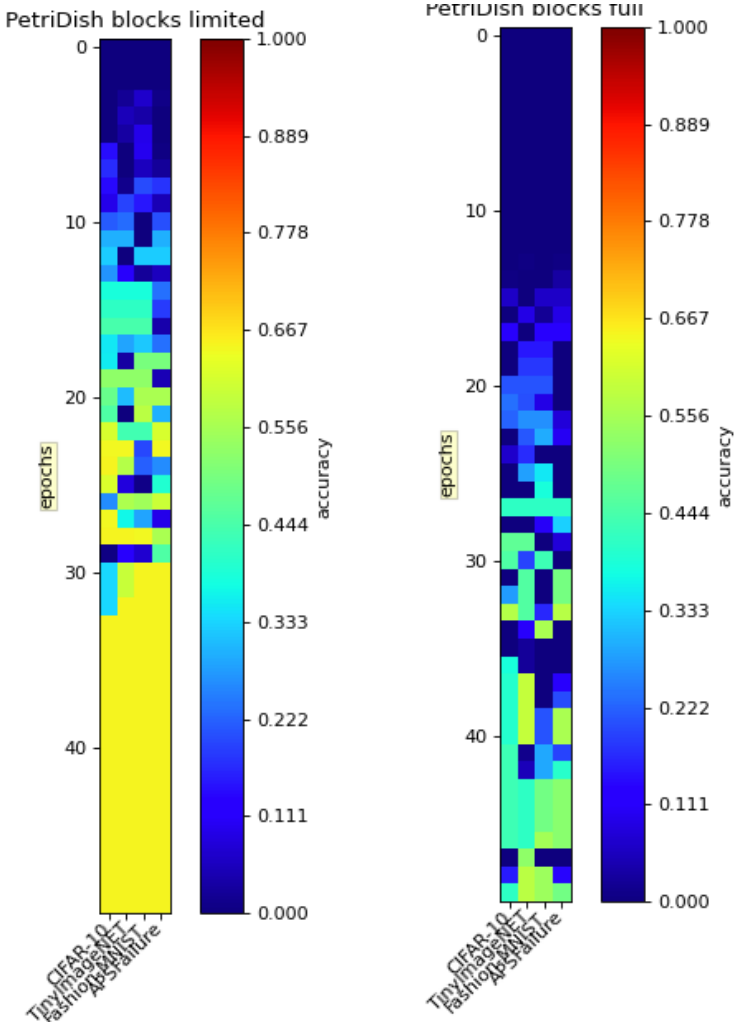
Building generalized configuration sets in PetriDish

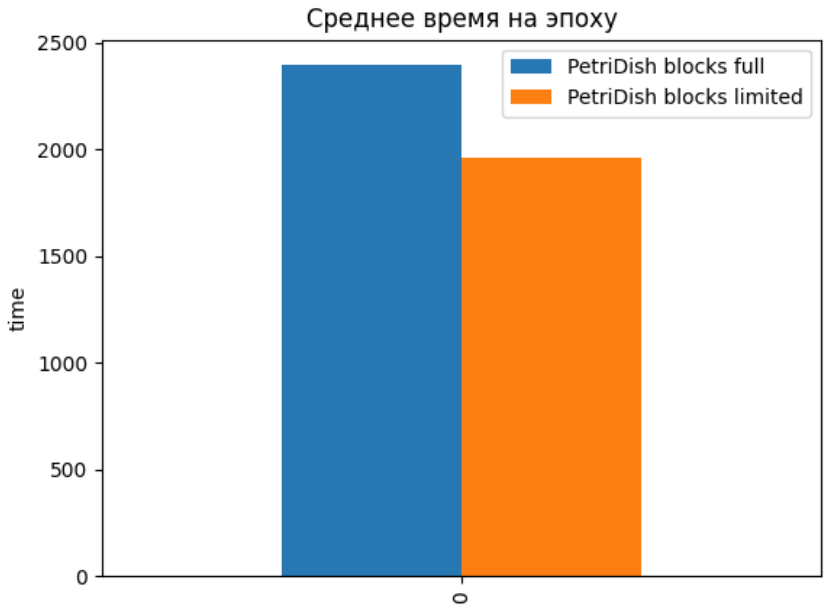
Merging classes (as sets of available blocks) Such merge occurs before performing regrouping and linearization for sets like filter size, blocks/activation-functions availability/certain type of connections

Building generalized configurations sets in PetriDish

- PetriDish full – all types of convolutional layers, full-connected layers, avgpool, maxpool, number groupping layers < 7, GRU, LSTM
- Limited set – only blocks and conncections from TinyImageNET(ResNet34), ImageNET-16-120(Inception-v3) are presented.

NAS-Bench-101: Average accuracy and time per epoch for PetriDish.





Results.

- The proposed method of parallel tuning HPO using the mini-benchmarking approach has been implemented and integrated in semi-automatic tool for HPO tuning/NAS.
- The common dataset for this approaches, covering many types of blocks or typical connections between blocks of ANNs.
- The proposed method on the corresponding dataset allows increasing accuracy up to 7-10% using CoDeepNEAT and 5-7% using PetriDish.
- Using a parallel implementation of proposed methods it's possible to reach 30% running time per epoch reduction with the same level of accuracy in different benchmarks(FashionMNIST).

Thanks for your attention