Placeholder Title

Derek Koleber
derekkoleber@gmail.com

August 22, 2020

Abstract

One of the saught after objectives within neural architecture search is the reduction of search time. Some recent approaches attempt to reduce search time by using surrogate performance metric that predict performance based on learning curves and/or architecture embeddings. In this paper, the efficacy of purely statistical methodology for accelerating the search time of evolutionary algorithm oriented NAS approaches is assessed. Analysis of results suggests that statistical methods do not provide meaningful search time reductions within commonly used NAS search spaces.

1 Introduction

In recent years, various approaches to neural architecture search (NAS) have been successful in discovering neural architectures in a variety of computer vision tasks as well as in language modeling, and INSERT SOMETHING. While many discovered architectures have achieved state-of-the-art performance on popular computer benchmarks, the compute cost of NAS acts as a significant obstacle.

Works within the field have taken many different approaches to reducing search time. Of the most significance to this work is the approach of using a performance surrogate, which allows the prediction of an architecture's performance prior to training completion. The effectiveness of different executions of approach manifest themselves in efficiency improvements all the way from early stopping to [2] architecture analysis without the necessity to train. [16] [8] [2] [18] [11] [1] [7] [5] [20] [3] [10] [16] [9] [17] [19] [12] [6] [13] [14] [15]

1.1 Background and Related Work

For NAS approaches utilizing genetic evolutionary algorithms, the means with which the algorithm determines the most and least fit candidates for mutation and optionally removal hinges upon the ability to determine comparative performance between candidates. The go-to metric for determining the performance of candidates with single-objective models is validation or test accuracy or error. However, recent approaches have forgone ratio variables to measure perforance in favor of ordinal variables [7], since relative performance is the only necessity for common selection strategies like traditional tournament selection and its variations, such as aging selection [11]. In the case of PNAS [7], a surrogate model is used to predict model performance in a way that ranks models approximately as they would be ranked as they would be using actual performance.

1.2 Approach

Using a surrogate model to predict model rank has the downside of training surrogate model to predict in a space that's growing in parallel with the surrogate. PNAS [7] proposes a clever solution to this which allows the surrogate to train over more data during the beginning of search. Its proposed approach is only possible due to the fact that the PNAS search space has the capability to expand its search space in a way that allows the initial, smaller search spaces to be representative of the later, larger search spaces in a way that is meaningful to the surrogate. Such an approach doesn't provide benefits to search spaces that aren't expandable in a manner similar to that of PNAS, which means a surrogate would have to train on architectures representing a complete search space.

Surrogate models are not the only way to predict candidate ranking. Two ways to represent relative performances of a population of candidates are the following 'performance metrics':

- Raw Rank Measurements Based on Candidate Accuracy

Rank measurements are a clear way to represent accuracies as ordinal data. No additional explanation is necessary.

- Z-Scores of Candidate Accuracy

1

Zz-scores of candidate performance can somewhat act as interval data and thus can better represent outliers and clusters in comparison to raw ranks. Z-Scores are typically used with normal distributions. Model accuracies in NASBench 201 [4] provide a complete view of the distributions of accuracies in what could be considered to be a search space representative of common NAS search spaces. An Anderson-Darling test over model test accuracies for Cifar10 (200 epochs) in NASBench201 produces a test statistic of 2717.199 (718.514 after a Box-Cox transform) indicates at all significance levels that model accuracies do not fall into a normal distribution. Nonetheless, z-scores still provide a way to judge relative model performance.

The relevant benefit of ordinal and interval performance metrics is that a population's measurements evaluated at an early epoch are directly comparable to its measurements at a later epoch. This is the basis for the possibility of search time acceleration, since the early performances have the potential to be predictive of final performances.

1.3   Stochastic Search Experimentation

1.3.1   Experiment Details

To measure the potential of the use of z-scores and raw ranks as a predictors of final performance, populations of models are used to emulate a stochastic search that is accelerated by early performance prediction. While stochastic search is a far-from-optimal search algorithm, this experiment's goal is rather to provide insight into the reliability using the two performance predictors to find the best candidates at different population sizes.

Two populations of trained models from two different architecutre search spaces are sampled for this purpose. The first population is generated using a variety of hyper-paremeter combinations within a search space similar to that of NASNet [20]. Implementation details of this search space are provided in the appendix. The second population is a sampled subset of the architectures in NASBench201 [4].

Each population P is evaluated at each combination between elements of E^s, W^s, and M. Each such evaluation at a given combination is referred to as a *simulation*. The set of *prediction epoch scalars* E^s scales t, the total number of epochs per model in P, to produce the actual *prediction epochs* at which final model performance rank will be predicted, E^a. This facilitates early performance prediction and thus the potential for search acceleration. The set of *prediction window scalars* W^s scales E^a to produce the actual *prediction windows* to be used over trained

epochs to be used during evaluation, W^a. The set of *evaluation metric* functions M includes functions for calculating an evaluation metric, given a model, a prediction epoch, and a prediction window. These metric functions include averaged-over-window z-score measurements zm's, and raw rank measurements rm's.

$$E^s = \{1, 0.5, 0.25, 0.125\}$$

$$W^s = \{1, 0.5, 0.25, 0.001\}$$

$$M = \{f_{zm}, f_{rm}\}$$

$$f_{zm}(x, e, w) = \frac{1}{e - w} \sum_{i=e-w}^{e} \frac{x_i - \mu_i}{\sigma_i}$$

$$f_{rm}(x, e, w) = \frac{1}{e - w} \sum_{i=e-w}^{e} r_{x_i}(P)$$

$$predictions(P, e, w, f_{predictor}) = r(map(P, f_{predictor}))$$

$$r(P) = rank\ of\ every\ element\ in\ P$$

$$r_x(P) = rank\ of\ x\ within\ P$$

In each simulation, P is shuffled then evaluated as an incremental subset of the shuffled population. Thus, each simulation evaluates a simulated population as if it were growing to its final size, while using the evaluation metrics to estimate simulated population rankings at each step. This allows a direct comparison between the predicted rankings and actual rankings at each increment. Simlar to PNAS [7], the Spearman Coefficient is chosen to judge the reliability of early prediction of population performance ranks. At the addition of each subsequent model to the simulated population, average rank error, rank error for the newest model (new rank error), and the Spearman Coefficient for the population are measured and used as evaluation metrics. Each simulation is run N times, and evaluation metrics are averaged across simulation iterations. Refer to the Figure 2, 3, 4, and 5 in Appendix for simulation results. Additionally, the correlation between new models' predicted rank error and new models' predicted rank is measured for each simulation, and results are averaged across simulation iterations. These results can be found in Figure **??**.

2

1.3.2 Analysis

Higher acceleration rates result in higher average and new rank error, and lower Spearman coefficients. New rank error and Spearman ccoefficients converge to levels that show clear trend that increases with acceleration rate. This is unsurprising, as these measurements are based on model performance that are further away in terms of epochs to their otherwise final performance. As acceleration rate increases, average rank error comes closer to stabilizing around $\frac{1}{3}$, the expected average error rate for a random population, but the rank predictions causing such a rank error are clearly not random since the Spearman coefficient is far from zero.

New rank error stabilizes at a lower value than average rank error. In all simulations, new rank error appears to stabilize at approximately half the value of the average rank error. Additionally, new rank error stabilizes faster than average rank error, stabilizing near a population of 4 while average rank error begins to stabilize around a population of 12.

Model size appears to impact performance prediction accuracy. The NASNet model population is comprised of models which all are larger than their NAS-Bench201 counterparts. With NASNet simulations, Spearman coefficients remain consistenly higher and new rank error levels off at a lower value in comparison to NAS-Bench201 simulations. Nonetheless, NASNet simulations show an average rank error trajectories similar to the trajectories that NASBench201 simulations play out. **Where do average rank error/new rank error/spearman coeff actually level out to? - take last quarter and average it?**

Prediction based on zscores and prediction based on raw ranks produce virtually the same results. This is more evident with NASBench201 since it draws out to a higher population size. **CORRELATION?**

New rank error has an overall weak-to-moderate positive correlation with the predicted new rank itself. This trend is better observed with NASBench201 simulations, due to the fact that its larger simulated population will produce an innately more accurate representation of such a trend. **HOW DOES THIS CORRELATE WITH INCREASING ACCELERATION RATE?** This trend can be interpreted optimistically to suggest that, in an evolution context, degrees of exploration will be bolstered outside of the exploration facilitated by the evolutionary algorithm alone. Pessimistically, this trend can be interpreted to suggest that suboptimal candidates might be chosen during the selection step of evolutionary

selection algorithms, resulting in slower time to find better candidates or the possibility for better candidates to be spuriously removed.

2 Aging Search Experimentation

Results from stochastic search experimentation are only applicable to an evolutionary algorithm context during the population's development prior to selection mechanisms taking effect, since selection mechanisms generally increase the frequency of higher-performance candidates. With a higher concentration of highly-performaning candidates in a population, the given evaluation metrics might perform less reliably.

testtest

3 Headings: first level

All headings should be lower case (except for first word and proper nouns), flush left, and bold.

First-level headings should be in 12-point type.

3.1 Headings: second level

Second-level headings should be in 10-point type.

3.1.1 Headings: third level

Third-level headings should be in 10-point type.

3.2 Footnotes

Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number[1] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).

Note that footnotes are properly typeset *after* punctuation marks.[2]

3.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction. The figure number and caption always appear after the figure. Place one line space before the figure caption and one line space

---

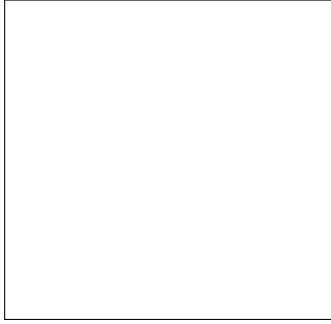[1]Sample of the first footnote.
[2]As in this example.

Figure 1: Sample figure caption.

Table 1: Sample table title

| Part | | |
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

after the figure. The figure caption should be lower case (except for first word and proper nouns); figures are numbered consecutively.

You may use color figures. However, it is best for the figure captions and the paper body to be legible if the paper is printed in either black/white or in color.

3.4 Tables

References

[1] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating Neural Architecture Search using Performance Prediction. *ICLR*, 2018. URL https://arxiv.org/abs/1705.10823.

[2] Boyang Deng, Junjie Yan, and Dahua Lin. Peephole: Predicting Network Performance Before Training. *CoRR*, 2017. URL https://arxiv.org/abs/1712.03351.

[3] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. URL https://ml.informatik.uni-freiburg.de/papers/15-IJCAI-Extrapolation_of_Learning_Curves.pdf.

[4] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. *ICLR*, 2020. URL https://arxiv.org/abs/2001.00326.

[5] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient Multi-objective Neural Architecture Search via Lamarckian Evolution. *ICLR*, 2019. URL https://arxiv.org/abs/1804.09081.

[6] Jason Liang, Elliot Meyerson, and Risto Miikkulainen. Evolutionary Architecture Search For Deep Multitask Networks. 2018. URL https://arxiv.org/abs/1803.03745.

[7] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive Neural Architecture Search. *ECCV*, 2018. URL https://arxiv.org/abs/1712.00559.

[8] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. *ICLR*, 2019. URL https://arxiv.org/abs/1806.09055.

[9] Hieu Pham, Melody Y. Guan, Barret Zoph, and Jeff Dean Quoc V. Le. Efficient Neural Architecture Search via Parameter Sharing. URL https://arxiv.org/abs/1802.03268.

[10] Aditya Rawal, Joel Lehman, Felipe Petroski Such, Jeff Clune, and Kenneth O. Stanley. Synthetic Petri Dish: A Novel Surrogate Model for Rapid Architecture Search. 2020. URL https://arxiv.org/abs/2005.13092.

[11] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized Evolution for Image Classifier Architecture Search. *ICML*, 2017. URL https://arxiv.org/abs/1802.01548.

[12] David R. So, Chen Liang, and Quoc V. Le. The Evolved Transformer. URL https://arxiv.org/abs/1901.11117.

[13] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. *CVPR*, 2019. URL https://arxiv.org/abs/1807.11626.

[14] Ning Wang, Yang Gao, Hao Chen, Peng Wang, Zhi Tian, Chunhua Shen, and Yanning Zhang. NAS-FCOS:

Fast Neural Architecture Search for Object Detection. *CVPR*, 2020. URL `https://arxiv.org/abs/1906.04423`.

[15] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. 2019. URL `https://arxiv.org/abs/1812.03443`.

[16] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic Neural Architecture Search. URL `https://arxiv.org/abs/1812.09926`.

[17] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. CARS: Continuous Evolution for Efficient Neural Architecture Search. URL `https://arxiv.org/abs/1909.04977`.

[18] Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. EcoNAS: Finding Proxies for Economical Neural Architecture Search. 2020. URL `https://arxiv.org/pdf/2001.01233.pdf`.

[19] Barret Zoph and Quoc V. Le. Neural Architecture Search with Reinforcement Learning. URL `https://arxiv.org/abs/1611.01578`.

[20] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. *CVPR*, 2018. URL `https://arxiv.org/abs/1707.07012`.

A Appendix

A.1 Stochastic Search Results

A.1.1 NASNet Simulations
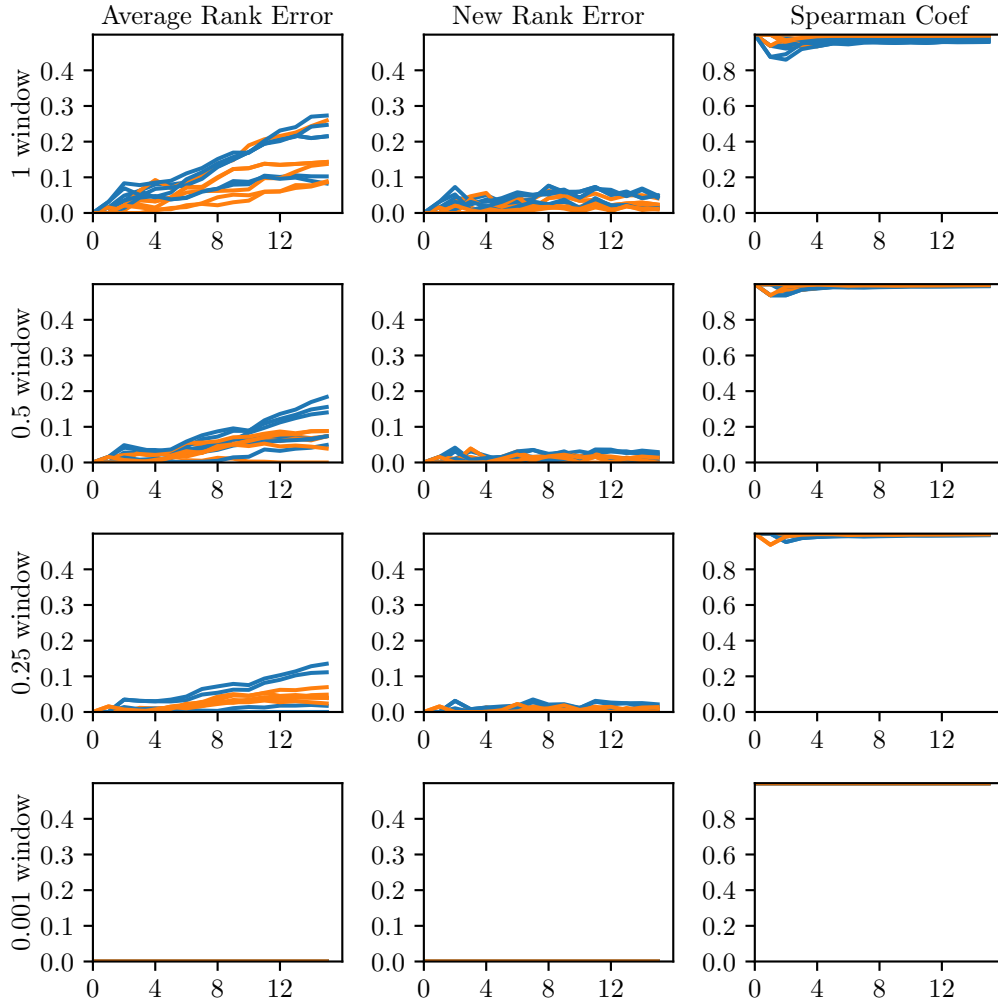
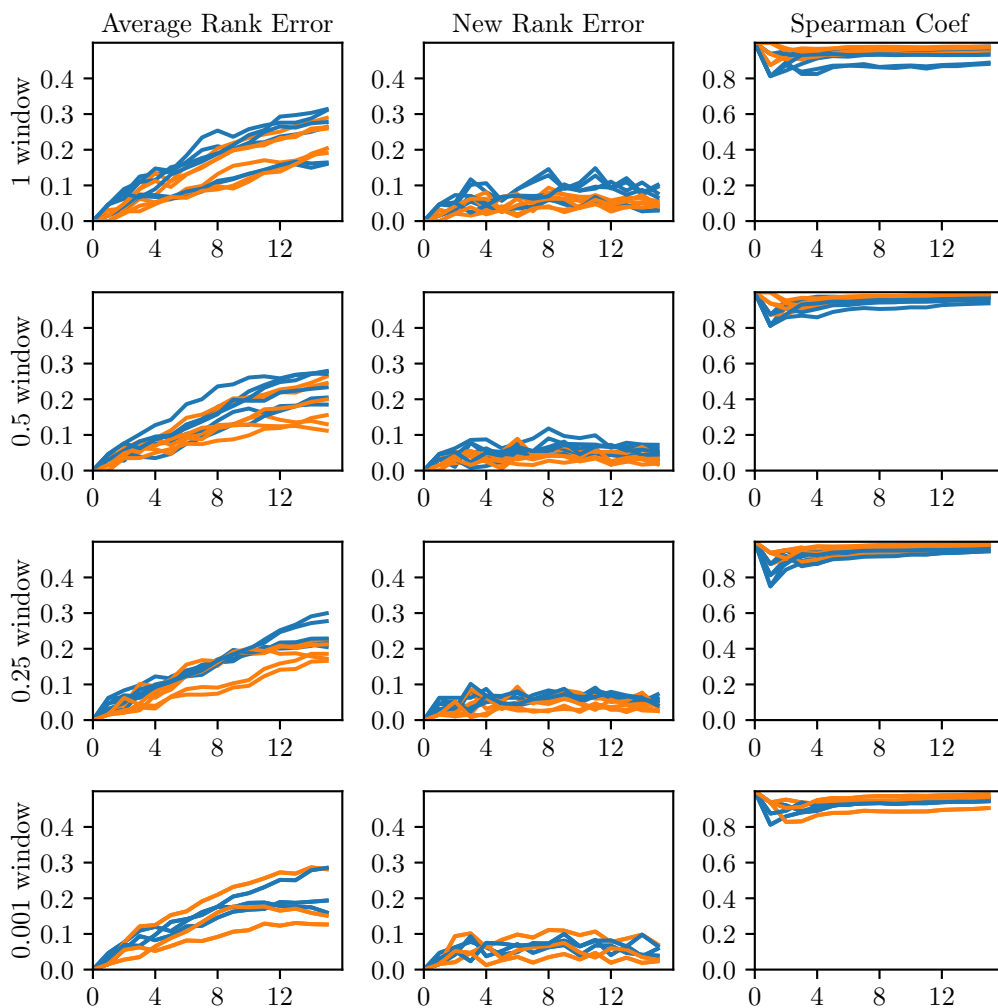

Figure 2: NASNet Simulation with 1x Acceleration

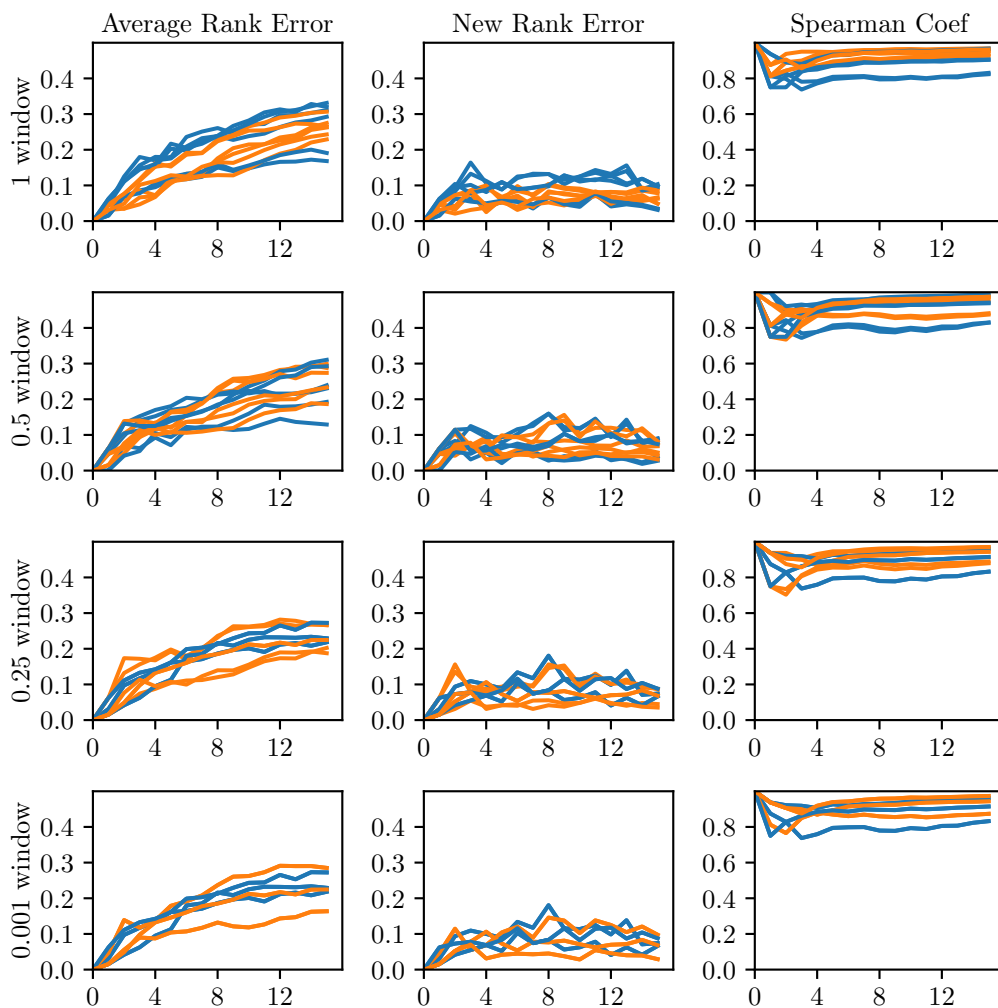Figure 3: NASNet Simulation with 2x Acceleration
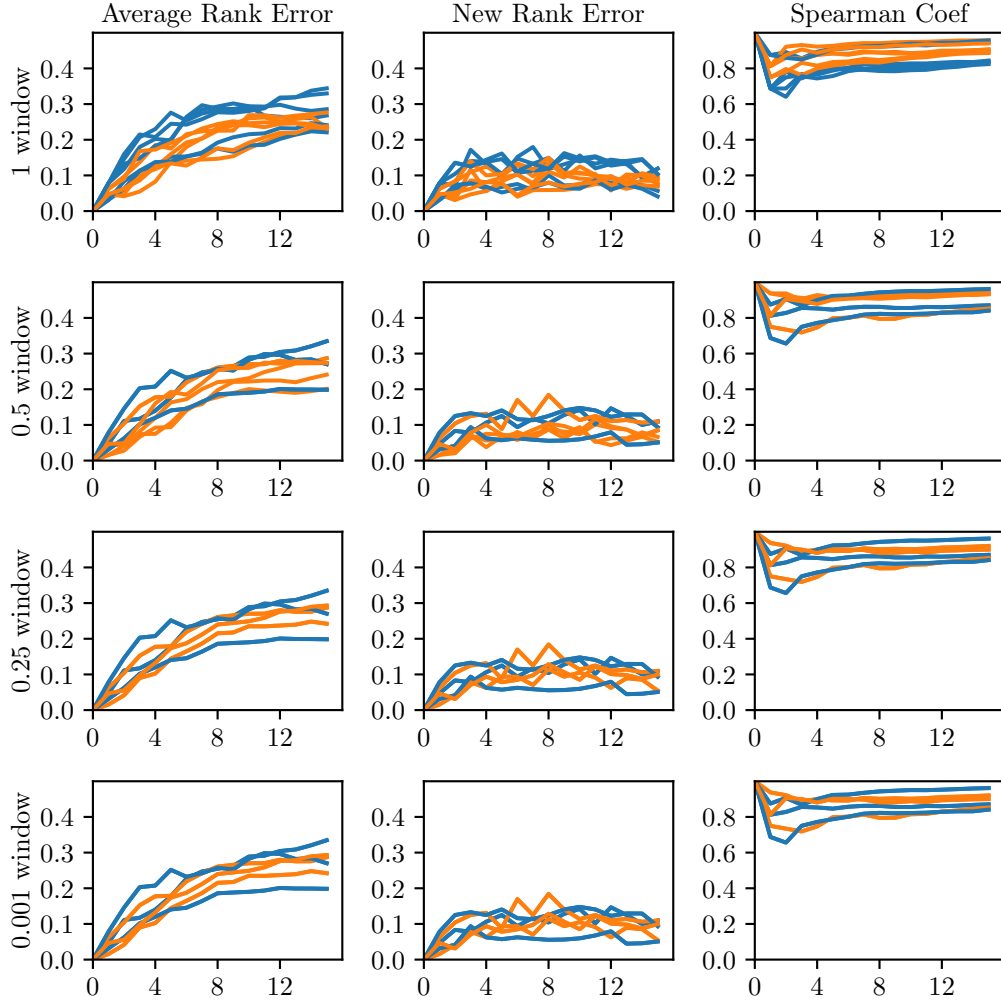
Figure 4: NASNet Simulation with 4x Acceleration
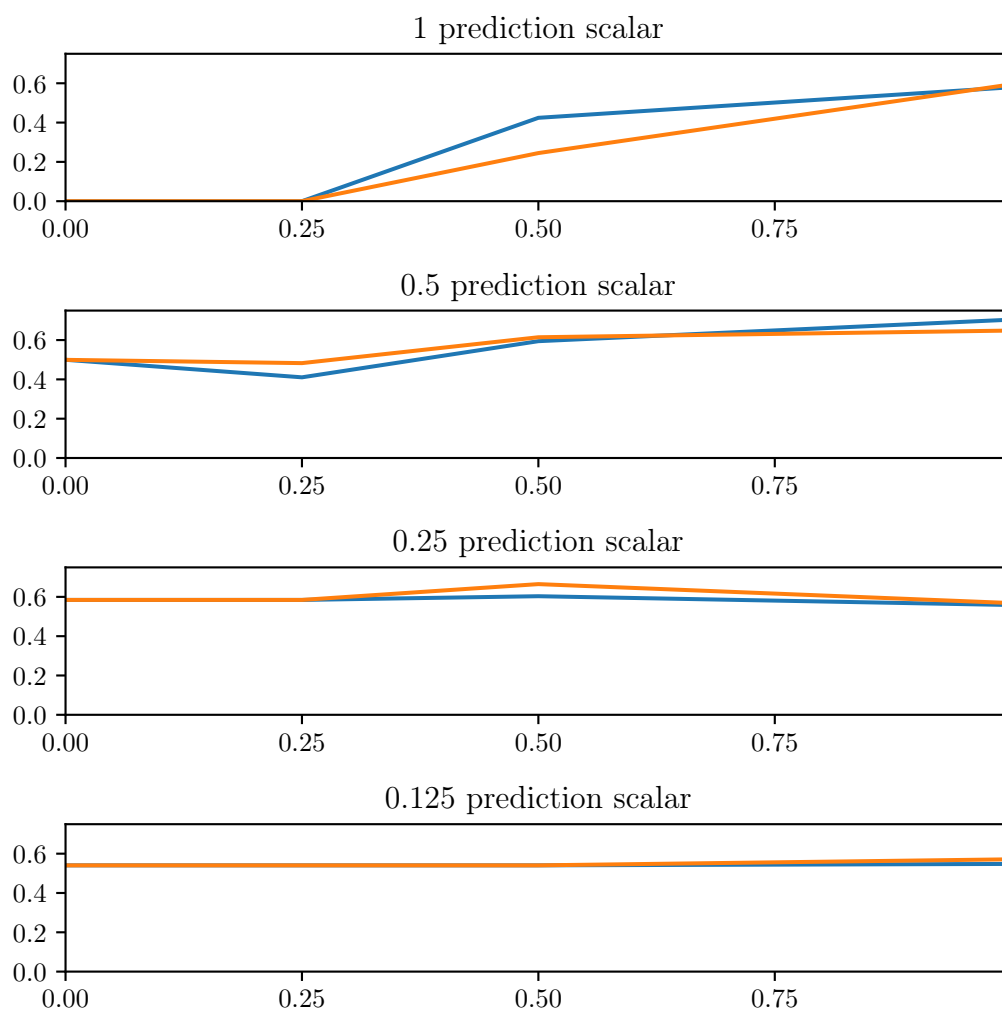
Figure 5: NASNet Simulation with 8x Acceleration

Figure 6: NASNet Rank Error Correlations with Rank
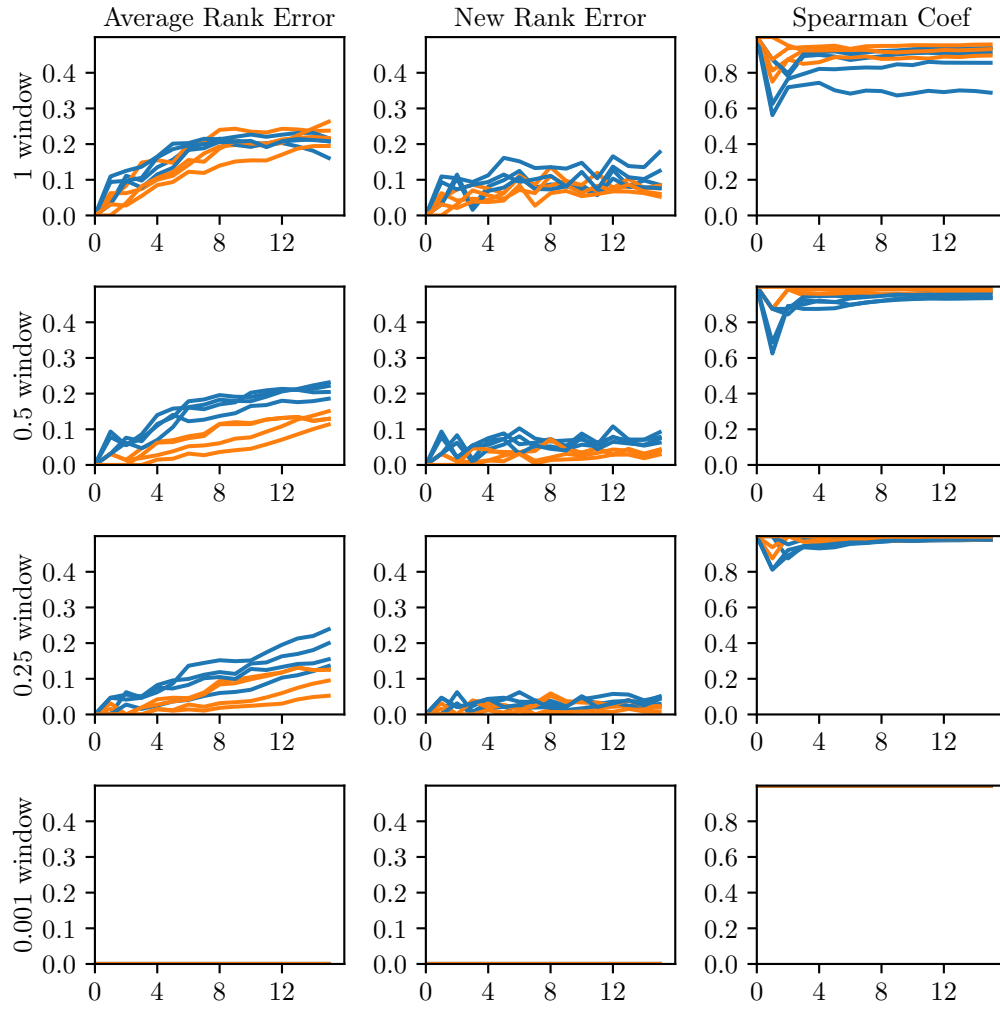
A.1.2 NASBench201 Simulations

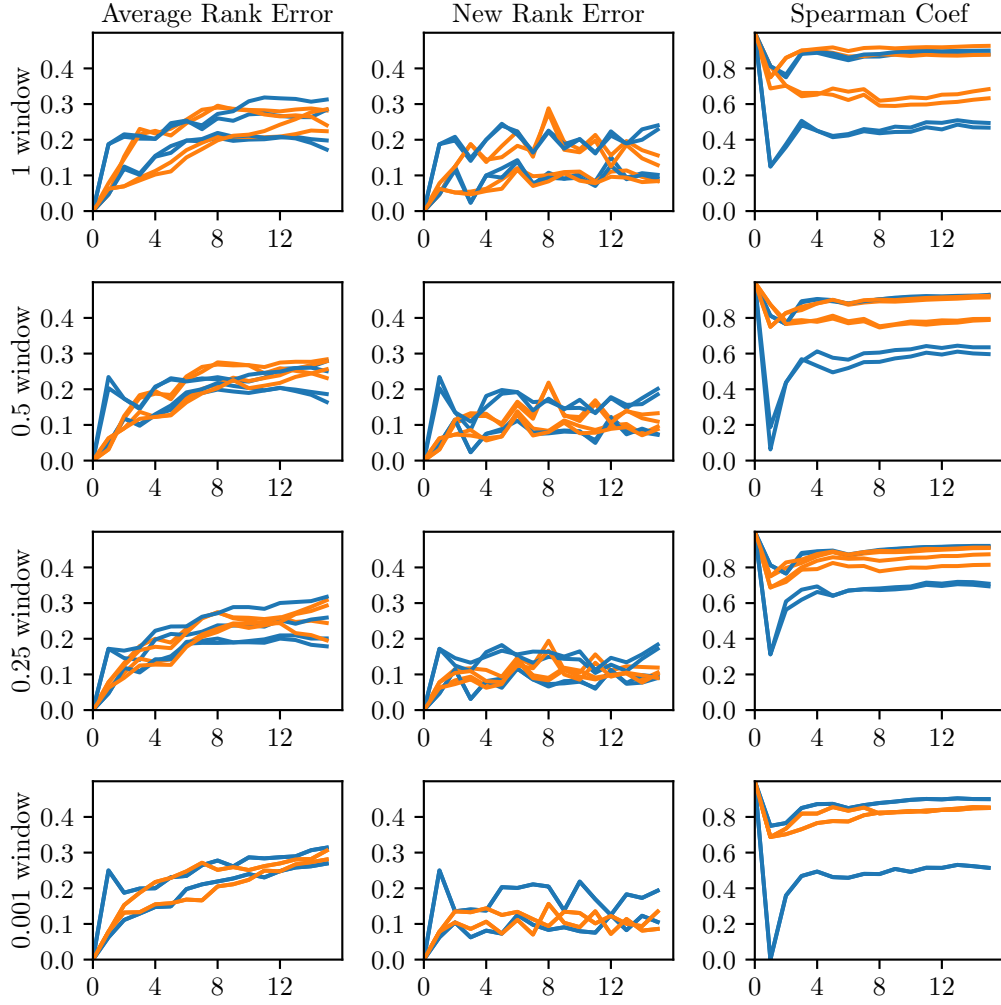Figure 7: NASBench201 Simulation with 1x Acceleration

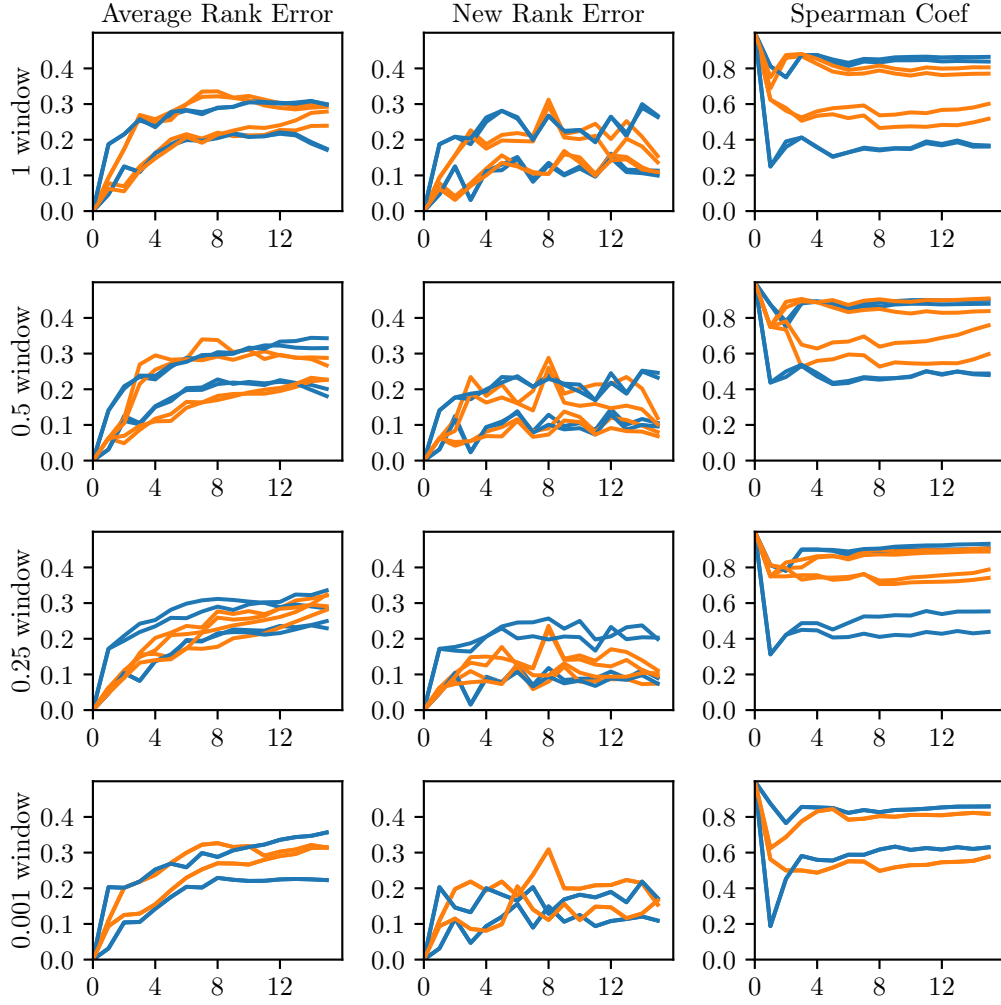Figure 8: NASBench201 Simulation with 2x Acceleration

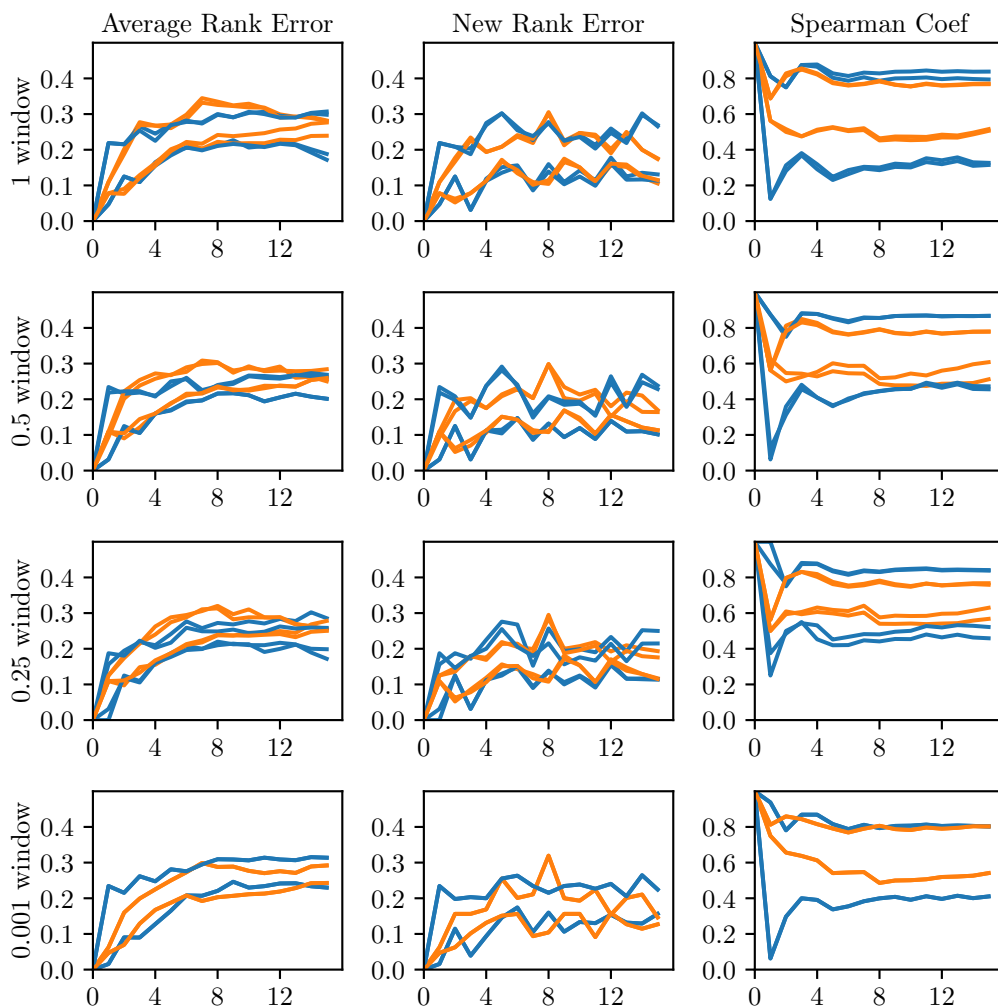Figure 9: NASBench201 Simulation with 4x Acceleration

13

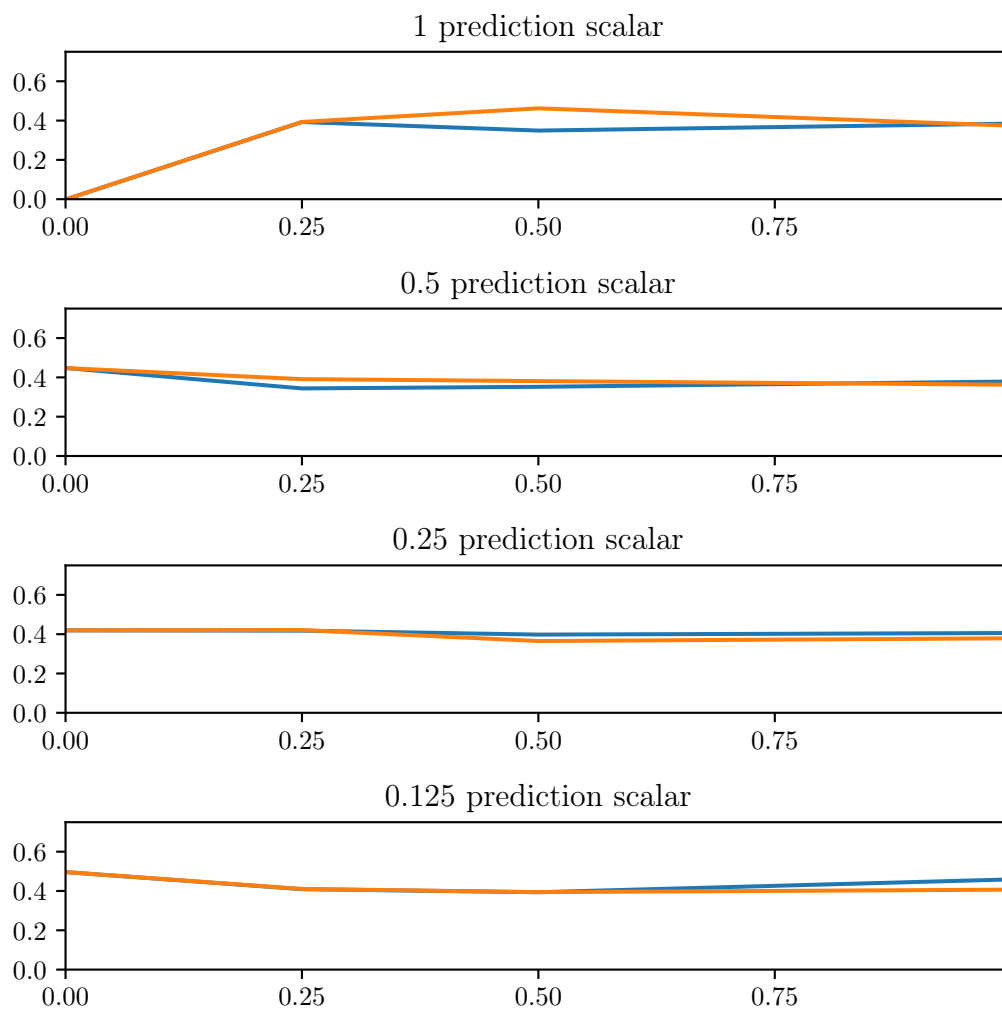Figure 10: NASBench201 Simulation with 8x Acceleration

Figure 11: NASBench201 Rank Error Correlations with Rank

A.2 NASNet Population Details

Using these possible ops

Each evaluated to 16 epochs, how big is population?

hyperparameters are as follows.

UPDATE CHARTS TO INDICATE WHAT ORANGE/BLUE MEANS