Placeholder Title

Derek Koleber
derekkoleber@gmail.com

August 23, 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

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.

2 Experiements

2.1 Stochastic Search Experiments

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.

2.1.1 Experiment Details

Three populations of trained models from two different architecutre search spaces are sampled. The first two populations are sampled subsets of the architectures available in NASBench201 [4]. Each of these populations contains four subpopulations. The first population, *NASBench201 100p*, has subpopulations consisting of 100 models each, the size of which is chosen to mirror the working population size used in aging tournament selection experiments later in this paper. The second population, *NASBench201 16p*, has subpopulations consisting of 16 models each, the size of which is chosen to mirror the number of models in the subpopulations of the third population. The third population, *NASNet 16p*, is generated four different hyperparemeter configurations within a search space similar to that of NASNet [20]. The subpopulations for each hyperparameter configuration consists of 16 models, each trained for 16 epochs. The purpose of this population is to be comparable in population size to the second population, but with significantly more network parameters. **WHAT ARE THESE NETWORK PARAMETER COUNTS?** Specific implementation details of this search space are provided in the appendix.

Each subpopulation P undergoes a grid search, evaluating 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 incrementally. 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, proportaional average rank error (*PARE*), proportional rank error for the newest model (*PNRE*), and the Spearman Coefficient for the simulated 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 1, 2, 3, and 4 in Appendix for simulation results. Additionally, the correlation between new model's predicted rank and the corresponding PNRE is measured for each simulation, and results are averaged across simulation iterations and subpopulations. Correlation results over evaluation metric tracked tightly, and these results were also invariant over different prediction window scalars, so results were averaged over these dimensions as well, for rank/PNRE correlations mapping directly to prediction epoch scalars (Table 4).

2.1.2 Analysis

Unless stated otherwise, only NASBench201 100p will be analyzed, since its increased population size decreases potential variance in measured properties.

Higher acceleration rates result in higher new rank error, and lower Spearman coefficients. New rank error and Spearman ccoefficients converge to levels that show clear trends that increase and decrease respectively 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. Independent of acceleration rate, average rank error appears to converge to $\frac{1}{3}$. This value is signicant since it's the expected distance between two points randomly sampled from two uniformly distributed variables over $[0, 1]$. The predicted and actual rankings, when divided by the simulated population size, are two such uniformly distributed variables, thus it would also be expected for rankings that are predicted randomly to have a PARE near $\frac{1}{3}$. However, the rank predictions causing such a PARE 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. **HYPOTHESIZE WHY THIS MIGHT BE, implications for evo**

Prediction based on zscores and prediction based on raw ranks produce virtually the same results. In all prediction epoch scalar/prediction window scalar combinations, ZMs and RMs have insignifcant marginal differences, with no clear trends of one being more effective than the other. In fact, the two are definitionally the same for populations trained to a number of epochs X when

$$1 >= XE^sW^s = W^a$$

PNRE has an overall weak-to-moderate positive correlation with the predicted new rank. 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. Across all prediction epoch scalars providing acceleration, the correlation between PNRE and predicted new rank is 0.437, indicating a weak-to-moderate positive correlation. This can be interpreted optimistically to suggest that, in an evolution context, the degree of exploration will be bolstered outside of the exploration facilitated by the evolutionary algorithm alone. Pessimistically, this 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 from the population.

Model size appears to impact performance prediction accuracy. As mentioned before, NASNet 16p is comprised of models which all are larger than their NASBench201 16p counterparts. With NASNet simulations, Spearman coefficients remain consistenly higher and new rank error levels off at a lower value in comparison to NASBench201 simulations. Nonetheless, NASNet simulations show an average rank error trajectories similar to the trajectories that NASBench201 simulations play out. **HYPOTHESIZE**

2.2 Aging Search Experimentats

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-performing candidates in a population, the given evaluation metrics might perform less

reliably. To determine the reliability of the evaluation metrics in the context of an evolving population, NASBench201 is used as a search space over which aging tournament selection [11] is used to conduct a search with a fixed time budget.

2.2.1 Experiment Details

This experiment will use the same population configuration as used in Amoebanet [11], where the working population consists of 100 candidates. At each selection step, 25 candidates are randomly selected; the best of which produces a mutated offspring. Similar to the previous experiment, a grid search is conducted over E^s, W^s, and M. Each combination is simulated 64 times, **and the same measurements are taken at each step, in addition to the actual performance of what is predicted to be the best candidate within the population at each step.** Each simulation is given a time budget equal to $max(E^s)ND_\mu$, where N is the size of the NASBench201 search space, and D represents the NASBench201 training durations. This is intended to allow the possiblity of only the fastest acceleration exploring the entire NASBench201 search space.

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

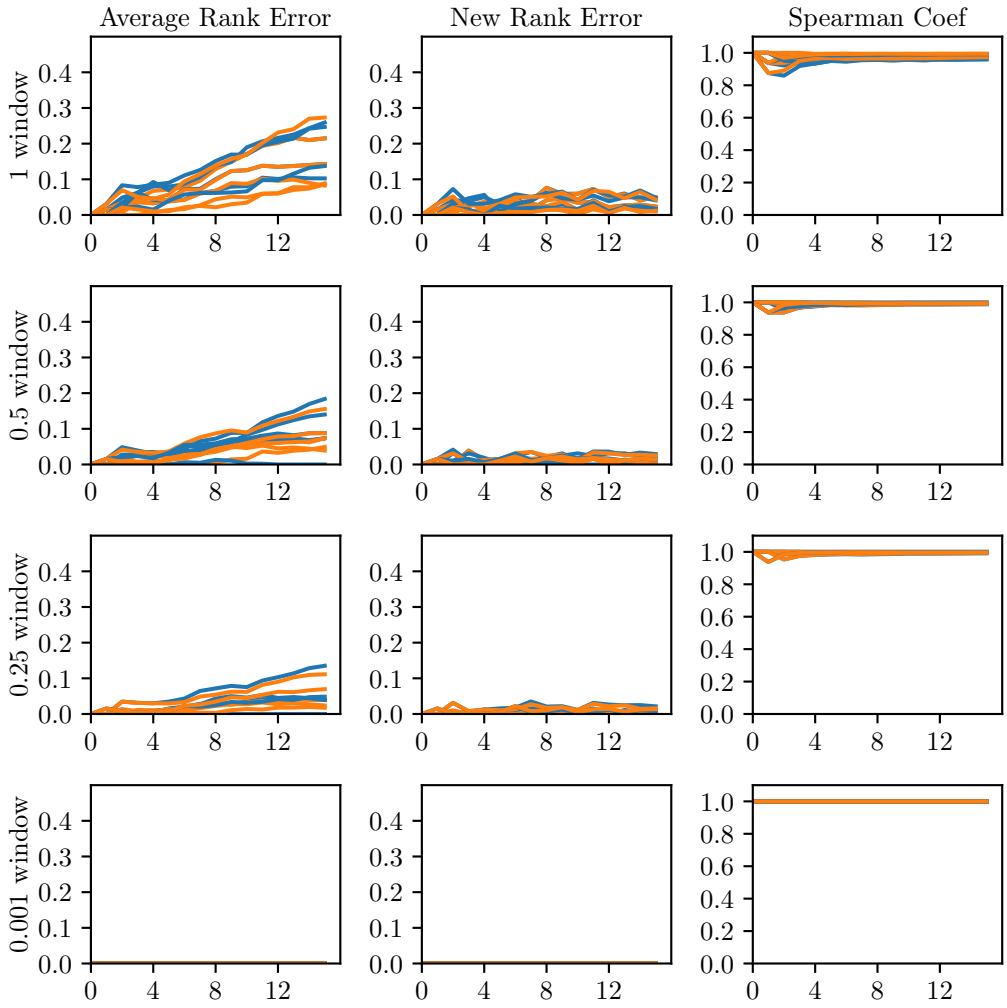

Figure 1: NASNet Simulation with 1x Acceleration

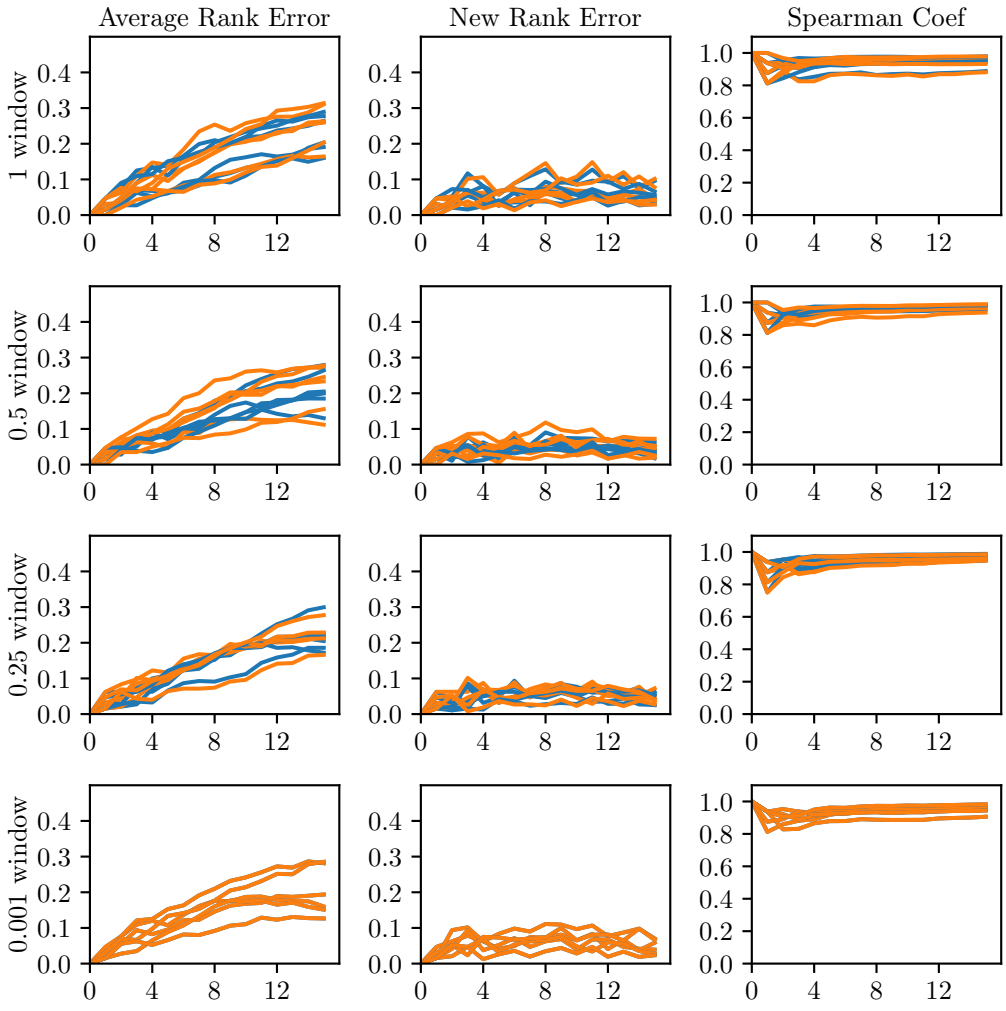

Figure 2: NASNet Simulation with 2x Acceleration

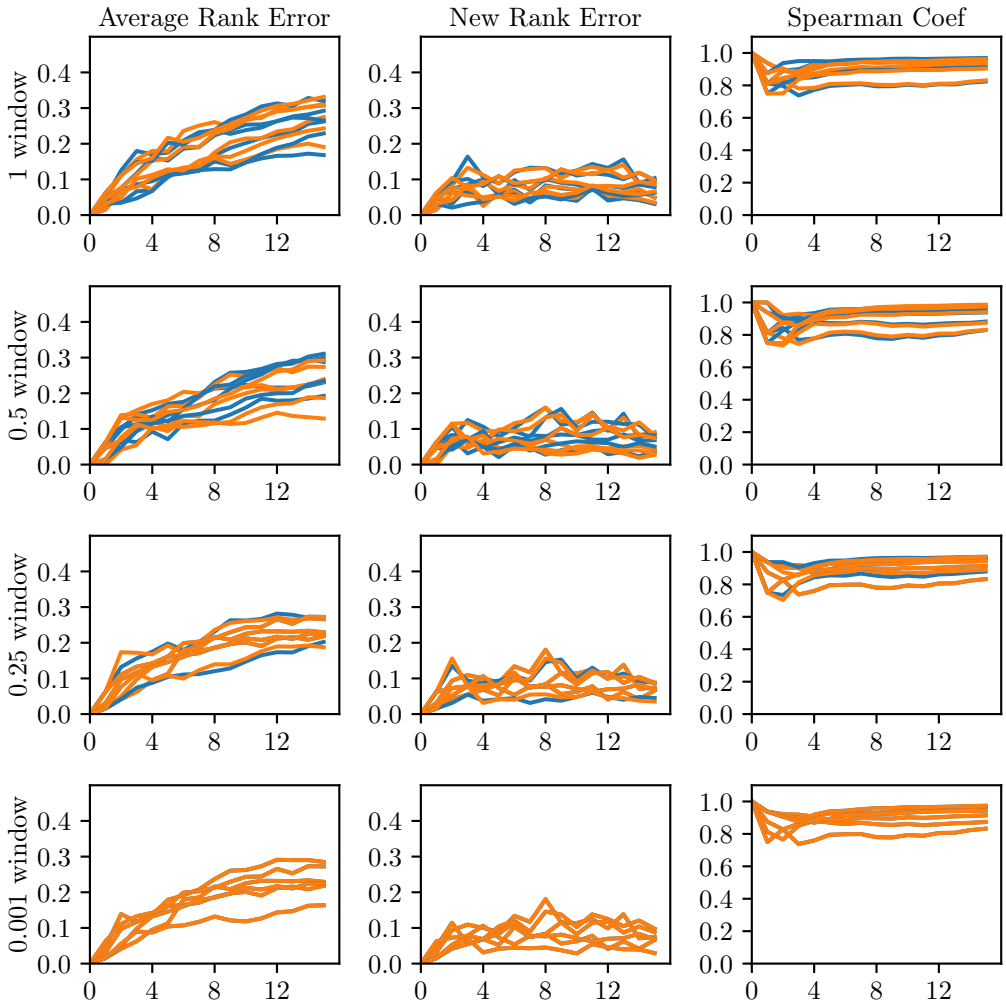

Figure 3: NASNet Simulation with 4x Acceleration

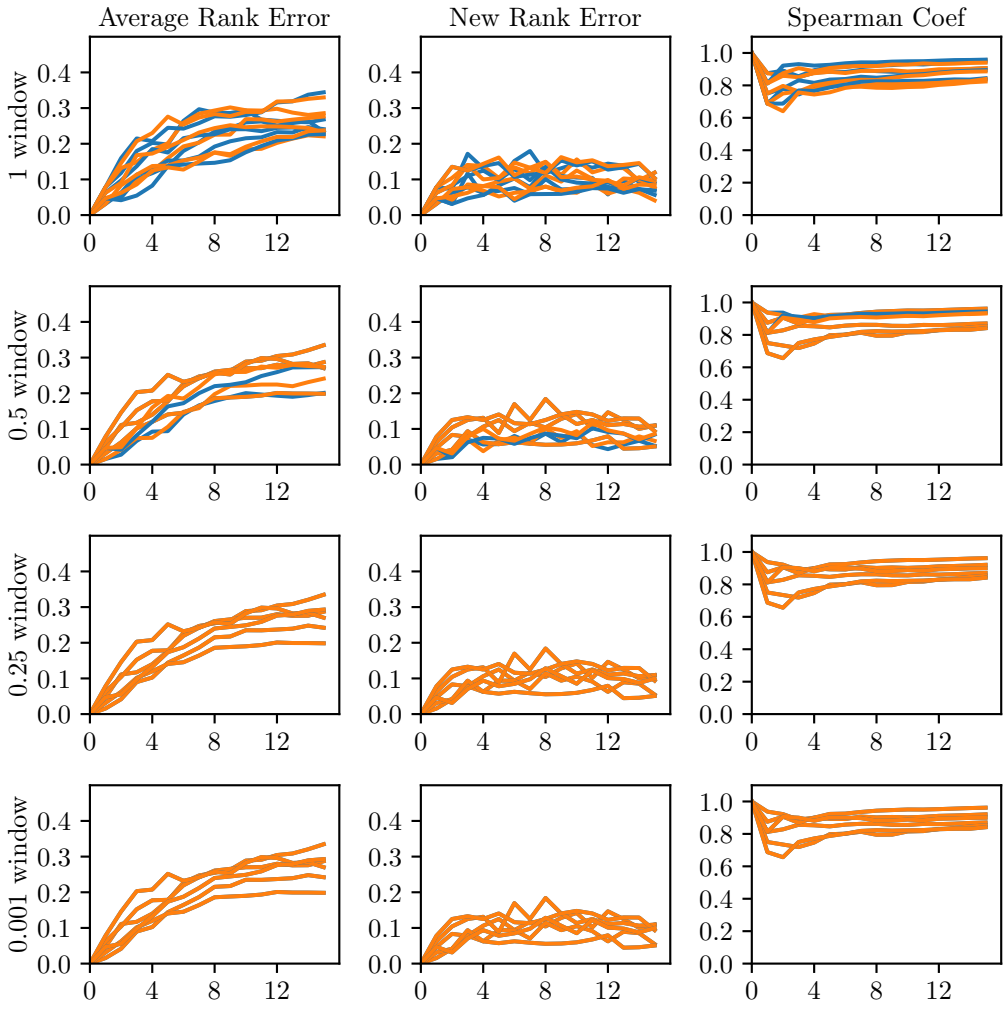

Figure 4: NASNet Simulation with 8x Acceleration

A.1.2 NASBench201 Simulations

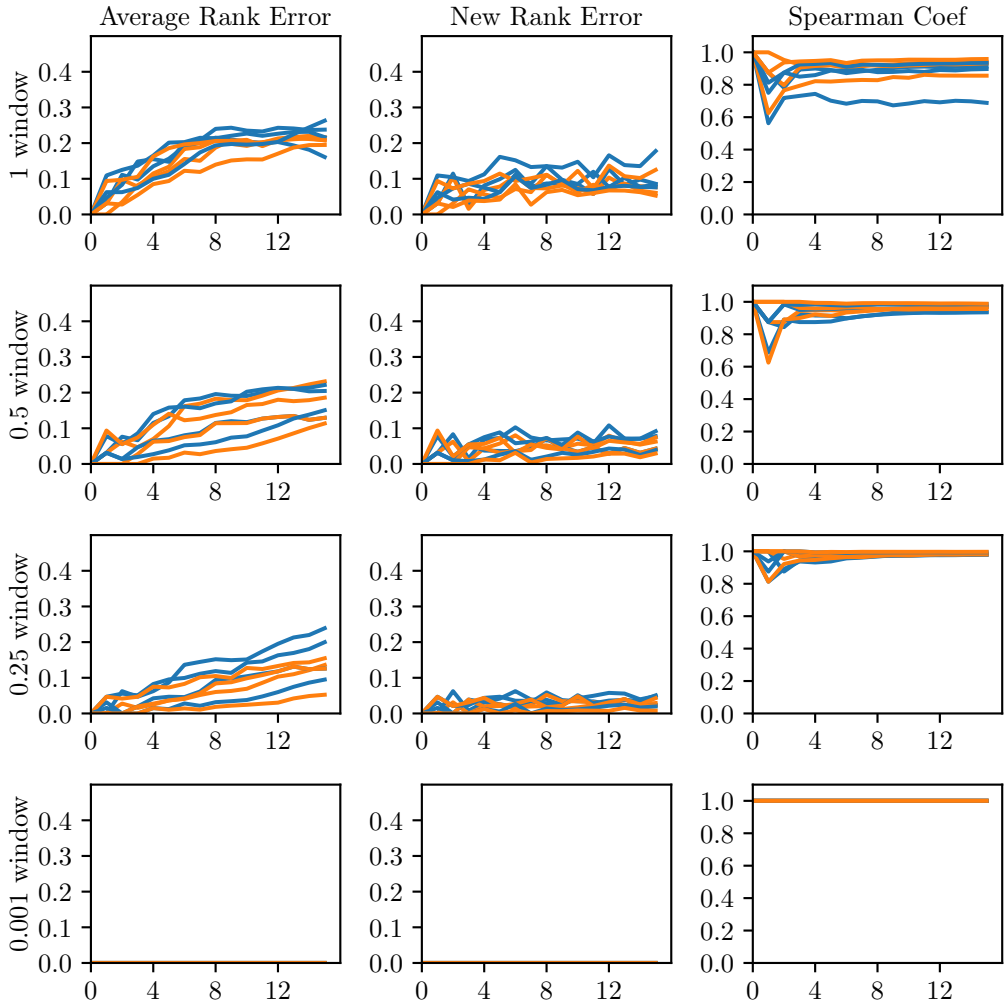

Figure 5: NASBench201 Simulation with 1x Acceleration

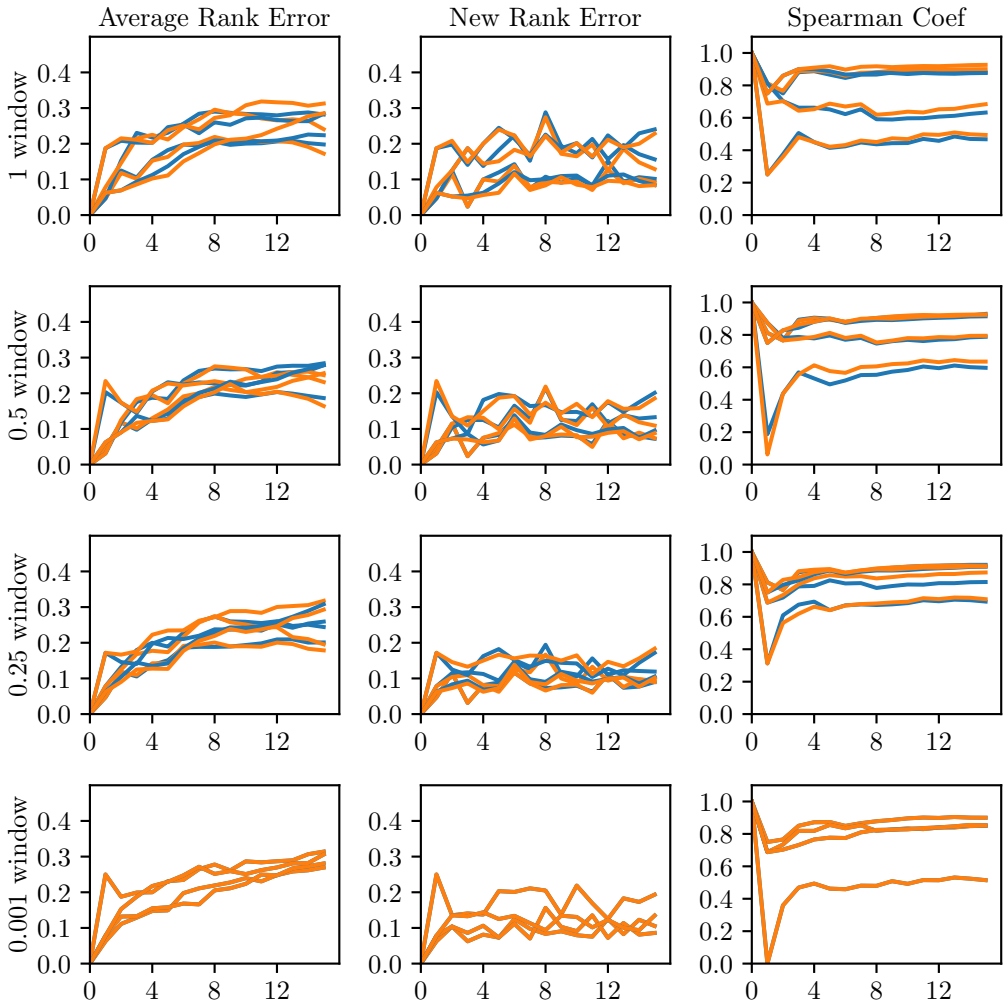

Figure 6: NASBench201 Simulation with 2x Acceleration

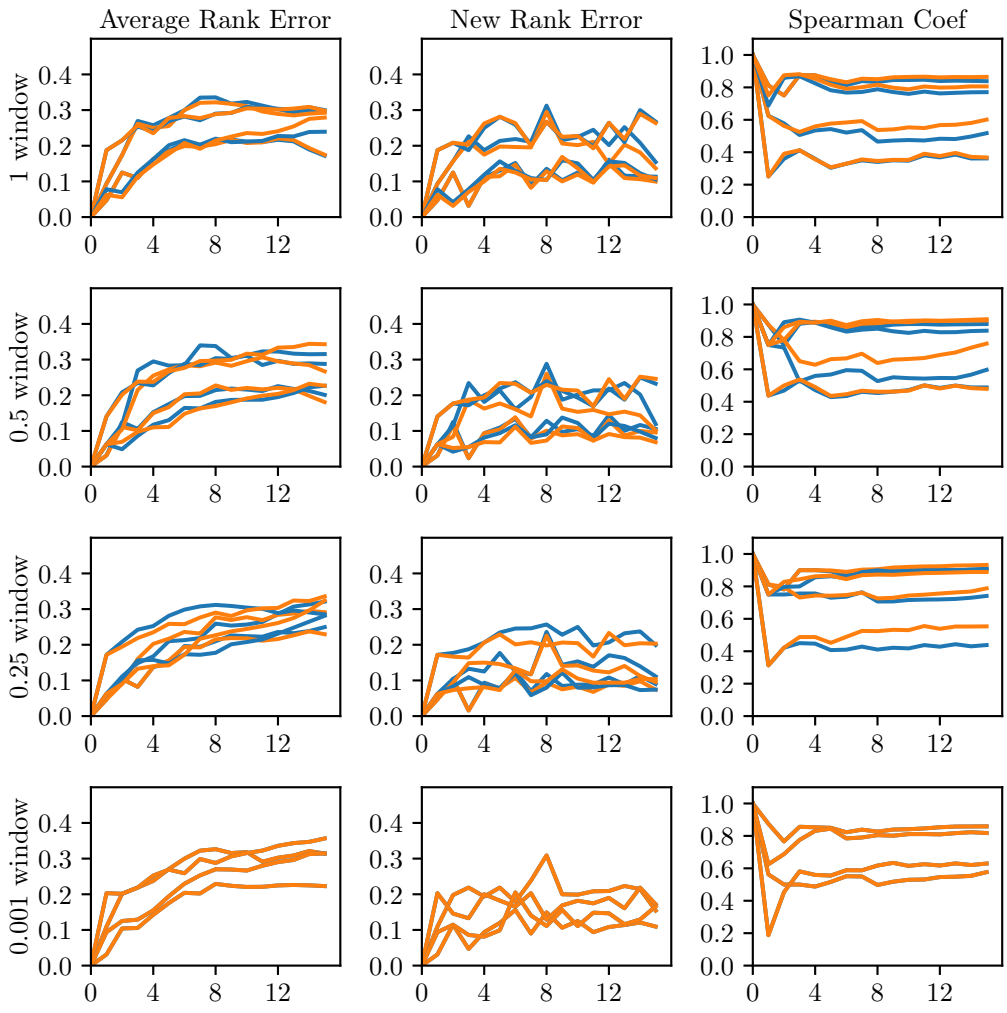

Figure 7: NASBench201 Simulation with 4x Acceleration

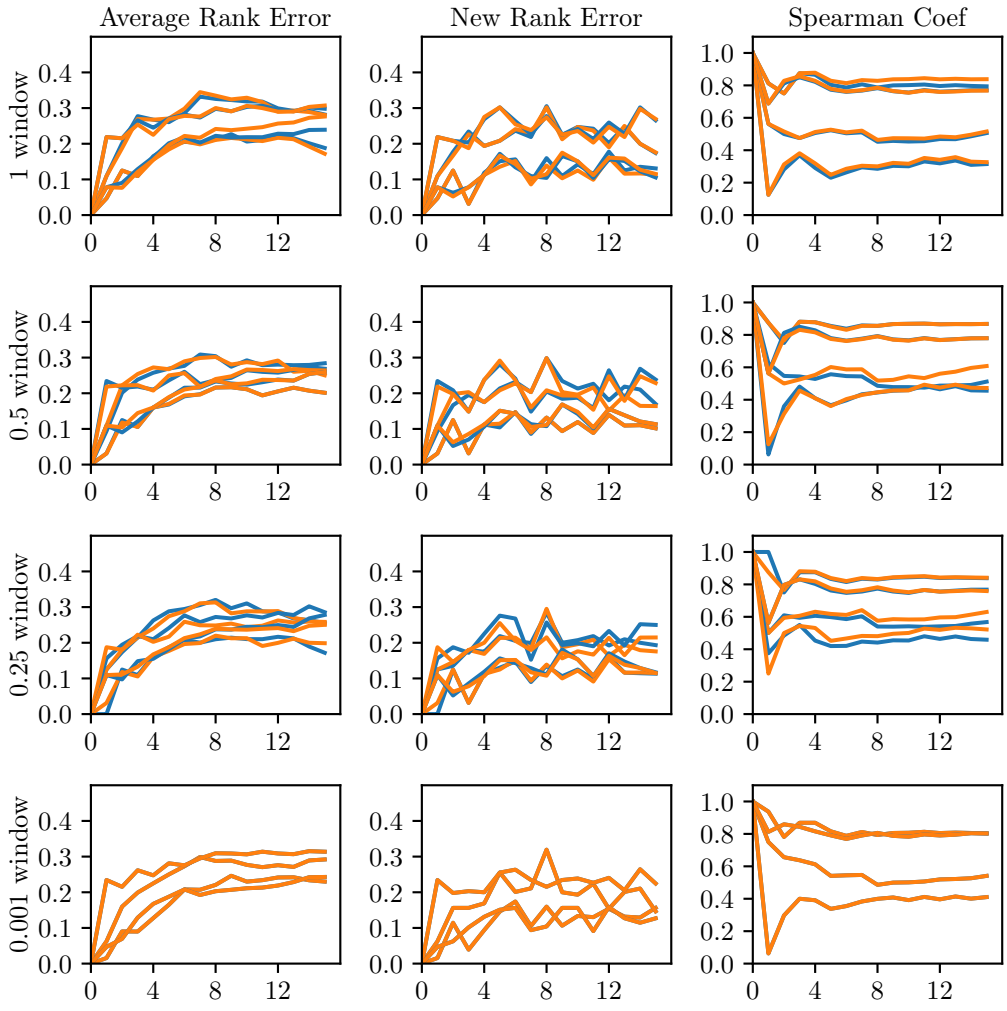

Figure 8: NASBench201 Simulation with 8x Acceleration

A.1.3 Simulation Points of Convergence

Table 1: NASNet Points of Convergence

P E	W	A R E ZM	A R E RM	N R E ZM	N R E RM	S C ZM	S C RM
1	1	0.17	0.14	$3.56 \cdot 10^{-2}$	$2.83 \cdot 10^{-2}$	0.98	0.98
1	0.5	$8.57 \cdot 10^{-2}$	$7.31 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$	$1.24 \cdot 10^{-2}$	0.99	0.99
1	0.25	$4.15 \cdot 10^{-2}$	$4.21 \cdot 10^{-2}$	$7.12 \cdot 10^{-3}$	$7.24 \cdot 10^{-3}$	1	1
1	$1 \cdot 10^{-3}$	0	0	0	0	1	1
0.5	1	0.22	0.24	$5.86 \cdot 10^{-2}$	$6.26 \cdot 10^{-2}$	0.95	0.95
0.5	0.5	0.2	0.21	$4.35 \cdot 10^{-2}$	$4.74 \cdot 10^{-2}$	0.97	0.97
0.5	0.25	0.21	0.21	$4.98 \cdot 10^{-2}$	$5.29 \cdot 10^{-2}$	0.96	0.96
0.5	$1 \cdot 10^{-3}$	0.2	0.2	$5.48 \cdot 10^{-2}$	$5.48 \cdot 10^{-2}$	0.95	0.95
0.25	1	0.25	0.27	$7.71 \cdot 10^{-2}$	$7.98 \cdot 10^{-2}$	0.92	0.92
0.25	0.5	0.25	0.22	$7.52 \cdot 10^{-2}$	$6.66 \cdot 10^{-2}$	0.92	0.92
0.25	0.25	0.23	0.23	$7.74 \cdot 10^{-2}$	$7.66 \cdot 10^{-2}$	0.91	0.91
0.25	$1 \cdot 10^{-3}$	0.23	0.23	$7.86 \cdot 10^{-2}$	$7.86 \cdot 10^{-2}$	0.91	0.91
0.13	1	0.26	0.26	$9.25 \cdot 10^{-2}$	$9.65 \cdot 10^{-2}$	0.89	0.88
0.13	0.5	0.26	0.26	$9.02 \cdot 10^{-2}$	$9.28 \cdot 10^{-2}$	0.9	0.89
0.13	0.25	0.27	0.27	$9.56 \cdot 10^{-2}$	$9.56 \cdot 10^{-2}$	0.88	0.88
0.13	$1 \cdot 10^{-3}$	0.27	0.27	$9.56 \cdot 10^{-2}$	$9.56 \cdot 10^{-2}$	0.88	0.88

Table 2: NASBench201 16p Points of Convergence

P E	W	A R E ZM	A R E RM	N R E ZM	N R E RM	S C ZM	S C RM
1	1	0.22	0.21	0.1	$8.4 \cdot 10^{-2}$	0.86	0.92
1	0.5	0.17	0.16	$5.83 \cdot 10^{-2}$	$4.8 \cdot 10^{-2}$	0.96	0.97
1	0.25	0.15	0.11	$3.48 \cdot 10^{-2}$	$2.52 \cdot 10^{-2}$	0.98	0.99
1	$1 \cdot 10^{-3}$	0	0	0	0	1	1
0.5	1	0.24	0.26	0.15	0.14	0.71	0.75
0.5	0.5	0.25	0.23	0.12	0.12	0.8	0.82
0.5	0.25	0.25	0.25	0.11	0.11	0.83	0.85
0.5	$1 \cdot 10^{-3}$	0.28	0.28	0.12	0.12	0.78	0.78
0.25	1	0.26	0.26	0.18	0.17	0.62	0.66
0.25	0.5	0.26	0.26	0.16	0.14	0.69	0.75
0.25	0.25	0.27	0.28	0.14	0.13	0.74	0.78
0.25	$1 \cdot 10^{-3}$	0.3	0.3	0.16	0.16	0.71	0.71
0.13	1	0.26	0.26	0.19	0.18	0.59	0.61
0.13	0.5	0.25	0.25	0.17	0.16	0.65	0.68
0.13	0.25	0.25	0.24	0.17	0.16	0.66	0.68
0.13	$1 \cdot 10^{-3}$	0.27	0.27	0.17	0.17	0.63	0.63

Table 3: NASBench201 100p Points of Convergence

P E	W	A R E ZM	A R E RM	N R E ZM	N R E RM	S C ZM	S C RM
1	1	0.31	0.3	$8.79 \cdot 10^{-2}$	$6.93 \cdot 10^{-2}$	0.91	0.95
1	0.5	0.29	0.29	$5.05 \cdot 10^{-2}$	$4.36 \cdot 10^{-2}$	0.97	0.98
1	0.25	0.26	0.26	$2.77 \cdot 10^{-2}$	$2.45 \cdot 10^{-2}$	0.99	0.99
1	$1 \cdot 10^{-3}$	0	0	0	0	1	1
0.5	1	0.32	0.31	0.11	0.1	0.86	0.88
0.5	0.5	0.32	0.32	$9.55 \cdot 10^{-2}$	$9.04 \cdot 10^{-2}$	0.9	0.91
0.5	0.25	0.31	0.31	$9.19 \cdot 10^{-2}$	$8.78 \cdot 10^{-2}$	0.91	0.91
0.5	$1 \cdot 10^{-3}$	0.32	0.32	0.11	0.11	0.87	0.87
0.25	1	0.32	0.32	0.13	0.12	0.81	0.83
0.25	0.5	0.31	0.31	0.12	0.11	0.85	0.86
0.25	0.25	0.31	0.31	0.11	0.11	0.86	0.87
0.25	$1 \cdot 10^{-3}$	0.32	0.32	0.14	0.14	0.79	0.79
0.13	1	0.32	0.32	0.14	0.14	0.78	0.79
0.13	0.5	0.32	0.32	0.13	0.12	0.83	0.84
0.13	0.25	0.32	0.32	0.13	0.13	0.82	0.83
0.13	$1 \cdot 10^{-3}$	0.32	0.32	0.14	0.14	0.78	0.78

A.2 Rank Error vs Rank Correlations

Population	1x	2x	4x	8x
NASBench201 16p	0.194	0.275	0.342	0.350
NASBench201 100p	0.237	0.423	0.430	0.459
NASNet 16p	0.217	0.402	0.389	0.345

Table 4: Rank Error Correlation with Rank, Averaged over Subpopulation, Prediction Window Scalar, and Evaluation Metric

A.3 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