Placeholder Title

Derek Koleber
derekkoleber@gmail.com

August 30, 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 models that predict performance based on learning curves or architecture topologies. In this paper, the efficacy of purely a statistical methodology for accelerating the search time of an evolutionary-algorithm-oriented architecture search is assessed. This methodology uses two different windowed measures of relative architecture preformance to predict final relative performance. Results suggest that such methodology does not enable acceleration via early stopping, but does improve search time by using a more efficent performance metric than directly measuring model performance.

1 Introduction

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

Works within the field have taken many different approaches to reducing search time. One common approach is the use of a surrogate predictor model, which allows the prediction of an architecture's performance prior to training completion, or sometimes prior to training altogether. This, however, comes with the downside of the predictor model requiring training data from the search space prior to being able to act in the search space, creating a chicken-or-the-egg problem. While there are solutions to getting around this dilemma, this work seeks to remove the need for the surrogate model entirely.

1.1 Neural Architecture Search Background

Neural architecture search, generally defined as the automation of the design of neural network architectures, has made leaps and bounds within recent years. The field was pioneered in *Neural Architecture Search with Reinforcement Learning* [16] as a single-objective search for an image classifier architecture with an unbounded search space, which achieved SOTA on Cifar-10. Since then, the field progressed significantly, all the while diversifying the approaches and goals for neural architecture search.

NASNet [17] pushed the field forward in the direction of scalability, showing that a bounded search space that could be stacked in the form of cells produced architectures for Cifar-10 which also scaled to perform well on ImageNet.

Some works have innovated in the direction of the search mechanism, moving away from a reinforcement learning driver, toward evolution-based [10] or differentiable architecture search [8].

Other works have investigated alternative search objectives, such as implementing platform-aware searches [12] [13], or multi-objective searches [5] which explores the Pareto fronts including network performance and network size.

While the majority of recent works have remained within the image classification space, some have broken out of image classification and computer vision altogether to find solutions to language modelling, like *The Evolved Transformer* [11].

1.2 Related Work

A major goal within the nerual architecture search field has been the reduction of search time. The reduction of search time has been achieved through approaches like parameter sharing [9] [14], the use of smaller performance proxies [15], or performance preditors. PNAS used a reinforcement-

learning-based surrogate model which predicted *relative* network performance based on cell topology to drastically accelerate search. Similarly, *Peephole* [2] predicts network performance based on network topology with a LSTM agent. A notable shortcoming of these approaches is the fact that the surrogate model requires retraining between different cell topologies.

Rather than relying on cell topology to predict performance prior to training, some works attempt to predict performance based on learning curves. This was initially done with the objective of hyperparameter optimization [3] with a weighted probabilistic learning curve model. This was built upon with the addition of Bayesian models [6] and v-SVRs [1]. Common among these approaches that utilize the learning curve for performance prediction is that they are used to predict model performance directly, rather than predicting a metric that correlates with model performance.

Direct prediction of network performance is not always necessary. As discussed in PNAS [7], the important quality of a predictor is the ability to rank models in same order as they would be ranked if ordered according to their true performances. This allows for the selection of the best models irrespective of their actual performances.

A shortcoming of many surrogate model approaches is the fact that the surrogate model must be trained prior to or during the architecture search. PNAS [7] proposes a clever solution to this which allows the surrogate to train quickly at the beginning of the search, while scaling as the search progresses. 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 space to be representative of the later, larger search space in a way that maintains the relvance of the surrogate.

2 Approach

All surrogate models come with the downside of requiring some amount of data to be used for training prior to being able to effectively predict performance. In this section, a method with which intrinsic properties of a population can be utilized to predict relative performances, is discussed.

Two ways to represent relative performances of a population of candidates are the following 'performance metrics':

- *Raw ranks*: Given candidates trained to the same epoch, candidates can be evaluated at epoch up to and including the final epoch to determine their relative rankings based on performances at that epoch.
- *Z-scores*: Similar to raw ranks, candidates can be evaluated at an epoch according to the z-scores of their performances at that epoch. Z-scores have the added benefit over raw ranks of providing a better indication of outliers in clusters, as is intrinsic with interval data in comparison to ordinal data. Some of the meaningfulness of z-scores is lost due to the fact that model performances do not appear to fall in a normal distribution for at least one of the search spaces used later in this work, NASBench 201 [4]. In this space, final performances of models trained to 200 epochs on Cifar10 result in an Anderson-Darling test statistic of 2717.199 without a Box-Cox transform, and 718.514 with, strongly indicating that model performances do not fall within a normal distribution. Nonetheless, z-scores still provide more information about relative model performances than raw ranks.

To be explicit, 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.

Search time is accelerated by early stopping at epochs specified by scaling the total number of epochs T by different *prediction epoch scalars* in E^s,

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

This produces the actual *prediction epochs* at which final model performance rank will be predicted, E^a. A number of epochs up to the prediction epoch are used to predict final performance. The number of epochs which are used are specified by a *prediction window* which is determined by scaling the prediction epoch by a *prediction window scalar* from W^s

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

This determines actual prediction windows W^a. Within each window, performance metrics defined by M can be used to calculate relative performances at each epoch within the window.

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

As discussed before, these performance metrics include z-score measurements ZM's, and raw rank measurements

RM's. Performance metrics are averaged over the window to produce final performance predictions for the population P.

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

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

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

where

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

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

3 Experiments

3.1 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.

3.1.1 Experiment Details

Three populations of trained models from two different architecutre search spaces are sampled. The first two populations are sampled subsets of the architecture space 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 work. 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 [17]. 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*. 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 prediction 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 prediction 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).

3.1.2 Analysis

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

Earlier stopping result in higher new rank error, and lower Spearman coefficients. New rank error and Spearman ccoefficients converge to levels that show clear trends that decrease and increase respectively with the prediction epoch scalar. 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 early stopping, average rank error appears to converge to $\frac{1}{3}$. This value is signicant since it's the expected distance between two points ran-

domly 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 early stopping, 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**

3.2 Evolutionary Search Experiments

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 [10] is used to conduct a search with a fixed time budget.

3.2.1 Experiment Details

This experiment will use the same population configuration and evolutionary algorithm used in Amoebanet [10], where the population consists of 100 candidates at all times. An initial population of 100 is established. Then, at each selection step, 25 candidates are randomly selected; the best of which produces a mutated offspring. After the addition of the new, mutated offspring, the oldest candidate in the population is removed to maintain a population of 100.

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 smallest prediction epoch scalar exploring the entire NASBench201 search space.

Results for this experiment's grid search are displayed in Figures **???**. Additionally, Figure 9 shows the actual performance of the candidate predicted to be the best within the population for each grid search configuration, in respect to the number of models evalued so far.

3.2.2 Analysis

Searches with early stopping only result in better models at low time budgets. As shown in Figure 10, searches with early stopping only outperform normal searches early on within the allotted time budget. This provides an answer to the primary focus of this experiement: Using a statistical approach for early stopping and thus early stopping is shown to only provide advantages with low time budgets. In the case of this experiment, a low time budget signifies a buget equal to less than 10

Normal search without early stopping performs better with windows that contain more than the last epoch. With these simulations, normal search performs best with a window over all epochs, rather than only the last epoch which represents the true candidate performance. This suggests that candidates that perform better overall tend to be close to the candidates with the best performances within the search space. This is fairly impactful, since this implies that the traditional method of using final performace to be the judge of a model might not be the optimal metric for determining the best candidate during an evolutionary selection step. This comes with the stipulation that the time budget only allows partial exploration of the search space without early stopping. To determine the impact of a larger time budget on the performance of different windows with normal search, an ablation study will be conducted.

RM outperforms ZM in most cases. In the vast majority of cases, RM results in lower PARE, PNRE, and Spearman Coefficients. This is represented in Table 5, where all measurement ratios with 5

3.3 Ablation Study: Impact of Windows without Early Stopping

This study will be composed of a narrowed version of the evolutionary experiment where normal search is evaluated with an expanded search time budget. Specifically, the search budget will be equal to the expected duration to fully explore NASBench 201 by brute-force. It should be noted that the evolutionary algorithm encompases an identity mechanism which allows already-evaluated candidates to exist within the population. Thus, the algorithm does not provide the guarantee that the entire search space will be explored within the expected duration for full exploration. This caveat is reconciled by the fact that NASBench 201 contains a narrow search space in comparison to search spaces such as the NASNet variant used in this work which is $3.299 \cdot 10^{11}$ times larger. Results for this experiment can be found in ??????

3.3.1 Analysis

Besides the very initial time steps in each search, searches with a window of 1 are shown to result in higher performances at faster rates in comparsion to all other window sizes. Searches using only the last epoch's performance can be to be slowly converging toward the same performances, but expanding the search time budget further loses meaninfulness since the search space would at that point be better explored by brute force anyway. The impact of windows on normal search will thus need further evaluation on larger search spaces and larger time budgets in a future study. Regardless, these results suggest that search is at least initially accelerated by the use of a window over the entirity of trained epochs.

4 Discussion and Future Work

The primary contribution of this work is the demonstration that the use of a proxy metric for performance within evoluationary architecture search results in higher performances at fixed time budgets in comparison to direct performance measurement, within the NASBench 201 search space. If, in future work, this method is shown to carry over to different search spaces and macro-architectures, this provides a simple means to further improve search speed. Additionally, this work demonstrates that performance prediction at early stops using two different statistical measures does not provide meaningful search acceleration.

As discussed in the ablation study, future directions include the verification of the primary contribution over larger search spaces and search time budgets. Additionally, the investigation into factors influencing the normality of population performance distrubtions across different search spaces could yield opportunities for approaches that rely on the normality of population distributions. This is of course contingent on the absence of the possiblity for vanishing gradients among other similar problems, which otherwise would lead to inevitable clusters within the population.

References

[1] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Practical neural network performance prediction for early stopping.

[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] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian nerual networks. URL https://ml.informatik.uni-freiburg.de/papers/17-ICLR-LCNet.pdf.

[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] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

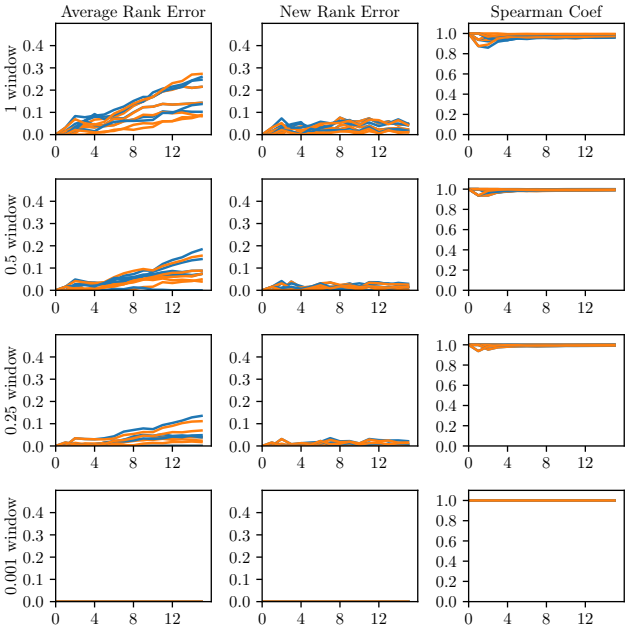

Figure 1: NASNet Simulation with 1x Acceleration

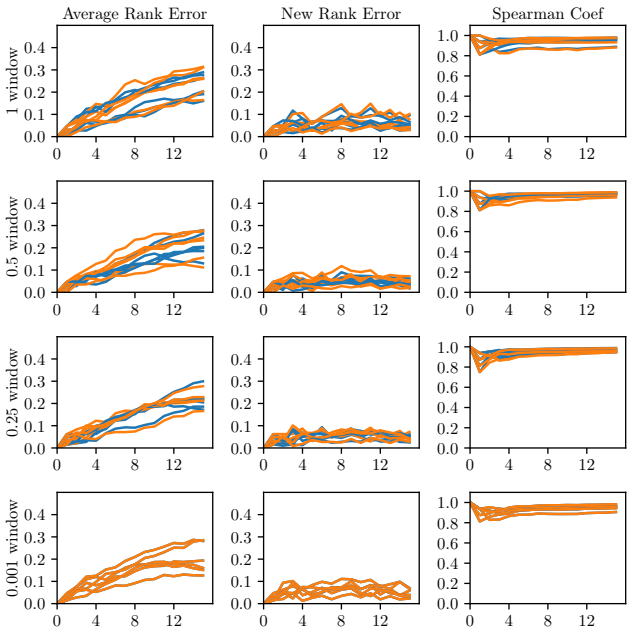

Figure 2: NASNet Simulation with 2x Acceleration

A Appendix

A.1 Stochastic Search Results

A.1.1 NASNet Simulations

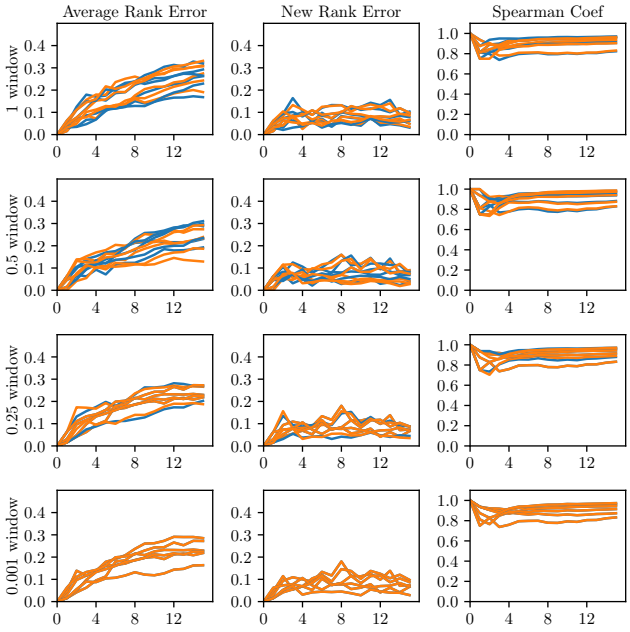

Figure 3: NASNet Simulation with 4x Acceleration

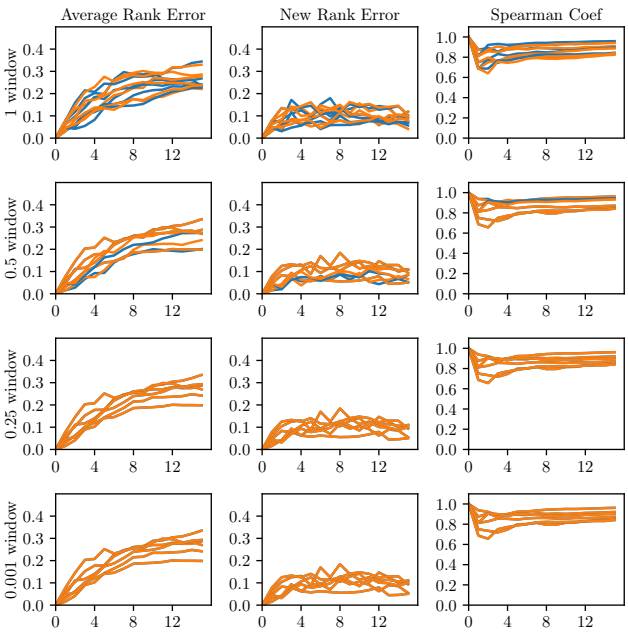

Figure 4: NASNet Simulation with 8x Acceleration

A.1.2 NASBench201 Simulations

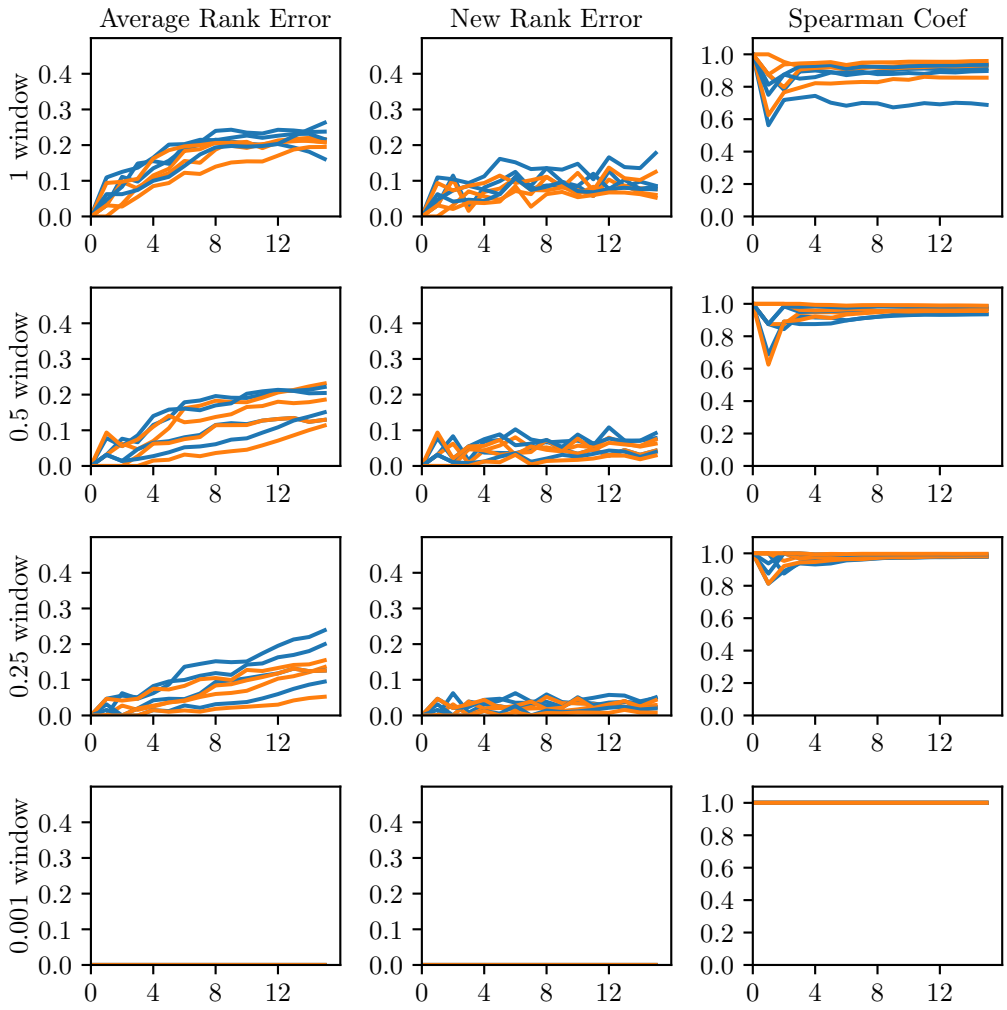

Figure 5: NASBench201 Simulation with 1x Acceleration

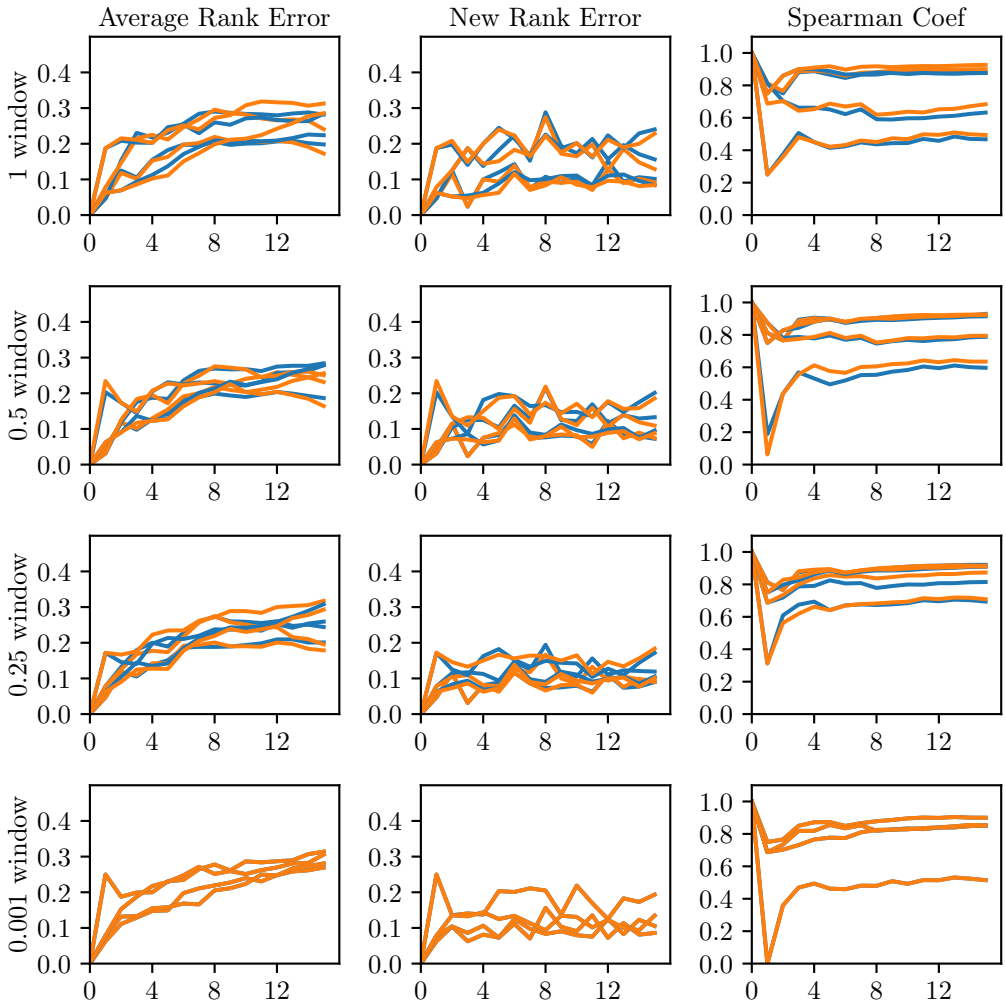

Figure 6: NASBench201 Simulation with 2x Acceleration

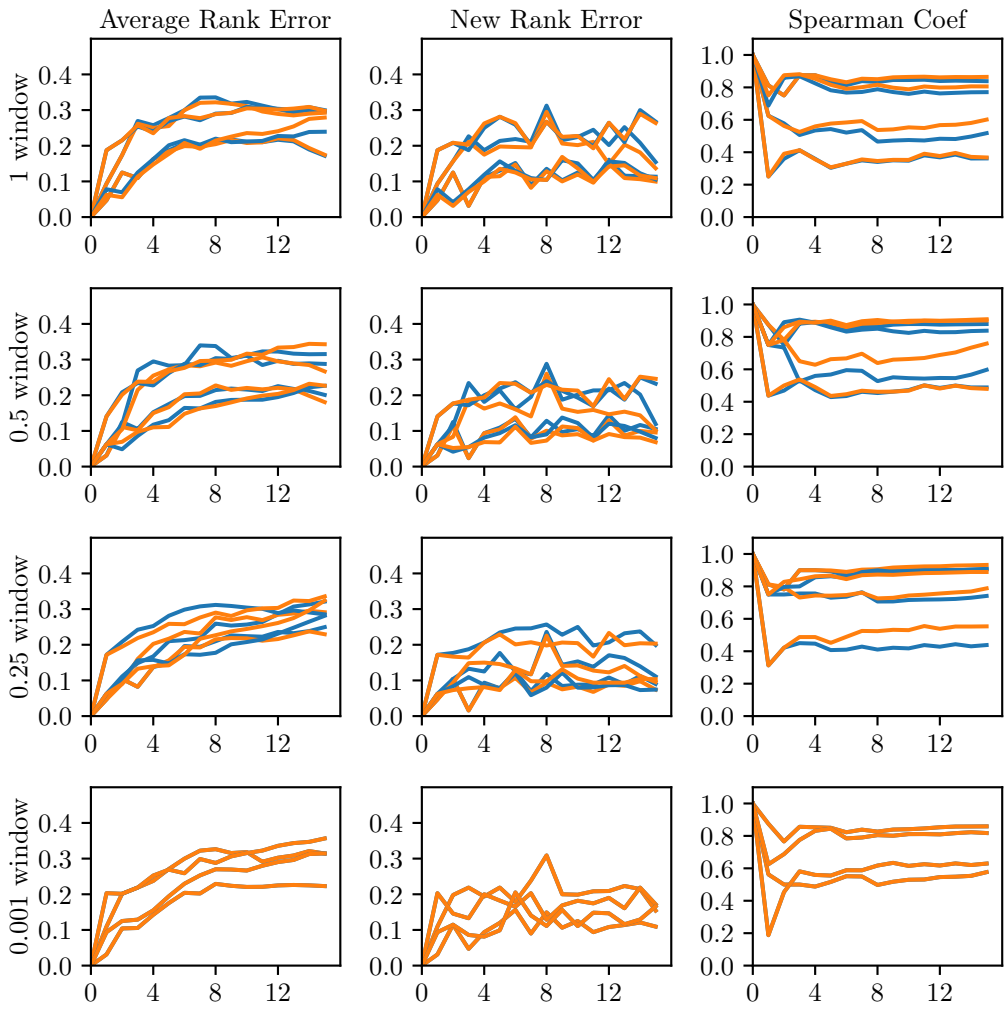

Figure 7: NASBench201 Simulation with 4x Acceleration

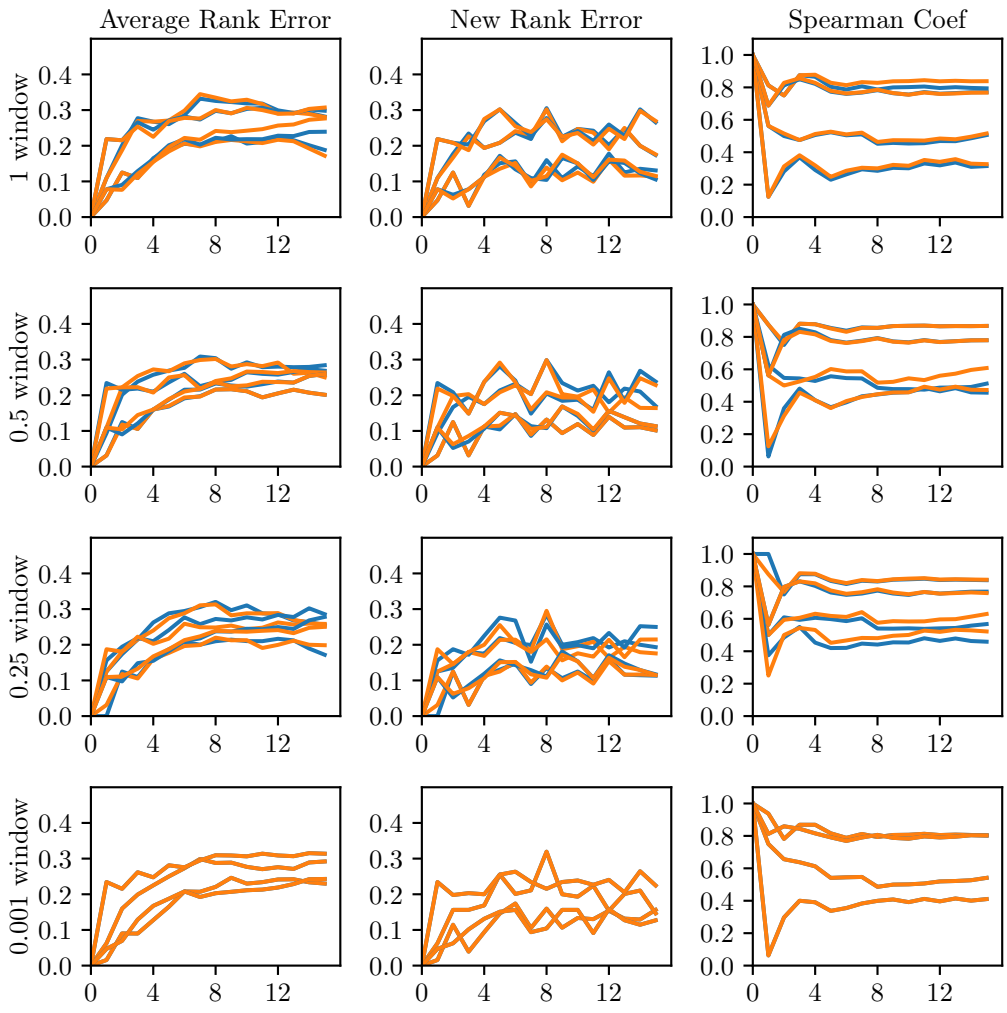

Figure 8: NASBench201 Simulation with 8x Acceleration

A.1.3 Simulation Points of Convergence

Table 1: NASNet Points of Convergence

PES	WS	ZM PARE	RM PARE	PARE Ratio	ZM PNRE	RM PNRE	PNRE Ratio	ZM Spr	RM Spr	Spr Ratio
1	1	0.173	0.141	1.229	0.036	0.028	1.256	0.975	0.981	0.994
1	0.5	0.086	0.073	1.172	0.016	0.012	1.295	0.993	0.994	0.999
1	0.25	0.041	0.042	0.984	0.007	0.007	0.983	0.997	0.997	1
1	0.001	0	0	0	0	0	0	1	1	1
0.5	1	0.216	0.236	0.917	0.059	0.063	0.935	0.948	0.945	1.002
0.5	0.5	0.201	0.207	0.97	0.043	0.047	0.917	0.972	0.967	1.005
0.5	0.25	0.212	0.212	1	0.05	0.053	0.943	0.963	0.957	1.006
0.5	0.001	0.199	0.199	1	0.055	0.055	1	0.947	0.947	1
0.25	1	0.248	0.266	0.932	0.077	0.08	0.966	0.92	0.92	1
0.25	0.5	0.248	0.218	1.142	0.075	0.067	1.13	0.918	0.924	0.993
0.25	0.25	0.231	0.231	1.001	0.077	0.077	1.011	0.91	0.912	0.998
0.25	0.001	0.229	0.229	1	0.079	0.079	1	0.91	0.91	1
0.125	1	0.257	0.261	0.984	0.093	0.097	0.959	0.891	0.885	1.007
0.125	0.5	0.257	0.264	0.975	0.09	0.093	0.972	0.896	0.892	1.004
0.125	0.25	0.268	0.268	1	0.096	0.096	1	0.885	0.885	1
0.125	0.001	0.268	0.268	1	0.096	0.096	1	0.885	0.885	1

Table 2: NASBench201 16p Points of Convergence

PES	WS	ZM PARE	RM PARE	PARE Ratio	ZM PNRE	RM PNRE	PNRE Ratio	ZM Spr	RM Spr	Spr Ratio
1	1	0.221	0.206	1.074	0.104	0.084	1.237	0.858	0.919	0.933
1	0.5	0.171	0.155	1.1	0.058	0.048	1.215	0.956	0.97	0.986
1	0.25	0.15	0.107	1.396	0.035	0.025	1.38	0.983	0.986	0.996
1	0.001	0	0	0	0	0	0	1	1	1
0.5	1	0.244	0.258	0.945	0.151	0.136	1.108	0.71	0.747	0.952
0.5	0.5	0.25	0.232	1.078	0.121	0.116	1.05	0.803	0.818	0.982
0.5	0.25	0.247	0.246	1.006	0.111	0.11	1.002	0.833	0.851	0.979
0.5	0.001	0.278	0.278	1	0.12	0.12	1	0.779	0.779	1
0.25	1	0.258	0.264	0.977	0.181	0.168	1.081	0.618	0.656	0.942
0.25	0.5	0.259	0.262	0.989	0.159	0.138	1.145	0.691	0.753	0.918
0.25	0.25	0.273	0.282	0.969	0.136	0.128	1.063	0.742	0.783	0.947
0.25	0.001	0.296	0.296	1	0.16	0.16	1	0.714	0.714	1
0.125	1	0.257	0.263	0.979	0.187	0.182	1.023	0.591	0.609	0.972
0.125	0.5	0.25	0.246	1.016	0.17	0.16	1.061	0.65	0.676	0.962
0.125	0.25	0.253	0.243	1.041	0.175	0.161	1.087	0.655	0.684	0.958
0.125	0.001	0.266	0.266	1	0.172	0.172	1	0.634	0.634	1

Table 3: NASBench201 100p Points of Convergence

PES	WS	ZM PARE	RM PARE	PARE Ratio	ZM PNRE	RM PNRE	PNRE Ratio	ZM Spr	RM Spr	Spr Ratio
1	1	0.309	0.303	1.021	0.088	0.069	1.269	0.912	0.945	0.965
1	0.5	0.293	0.291	1.01	0.051	0.044	1.159	0.967	0.976	0.991
1	0.25	0.264	0.264	1.001	0.028	0.025	1.13	0.987	0.992	0.996
1	0.001	0	0	0	0	0	0	1	1	1
0.5	1	0.316	0.313	1.012	0.113	0.105	1.082	0.857	0.879	0.975
0.5	0.5	0.315	0.316	1	0.096	0.09	1.057	0.901	0.91	0.99
0.5	0.25	0.314	0.314	1.002	0.092	0.088	1.046	0.907	0.913	0.993
0.5	0.001	0.324	0.324	1	0.11	0.11	1	0.874	0.874	1
0.25	1	0.319	0.318	1.004	0.13	0.123	1.054	0.811	0.833	0.974
0.25	0.5	0.315	0.315	1	0.116	0.111	1.043	0.852	0.865	0.985
0.25	0.25	0.314	0.311	1.011	0.112	0.107	1.046	0.862	0.873	0.987
0.25	0.001	0.321	0.321	1	0.14	0.14	1	0.79	0.79	1
0.125	1	0.32	0.317	1.009	0.142	0.138	1.027	0.776	0.79	0.982
0.125	0.5	0.317	0.32	0.992	0.125	0.122	1.029	0.829	0.837	0.991
0.125	0.25	0.321	0.32	1.003	0.13	0.126	1.03	0.821	0.83	0.989
0.125	0.001	0.323	0.323	1	0.143	0.143	1	0.779	0.779	1

A.1.4 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.2 Aging Tournament Search

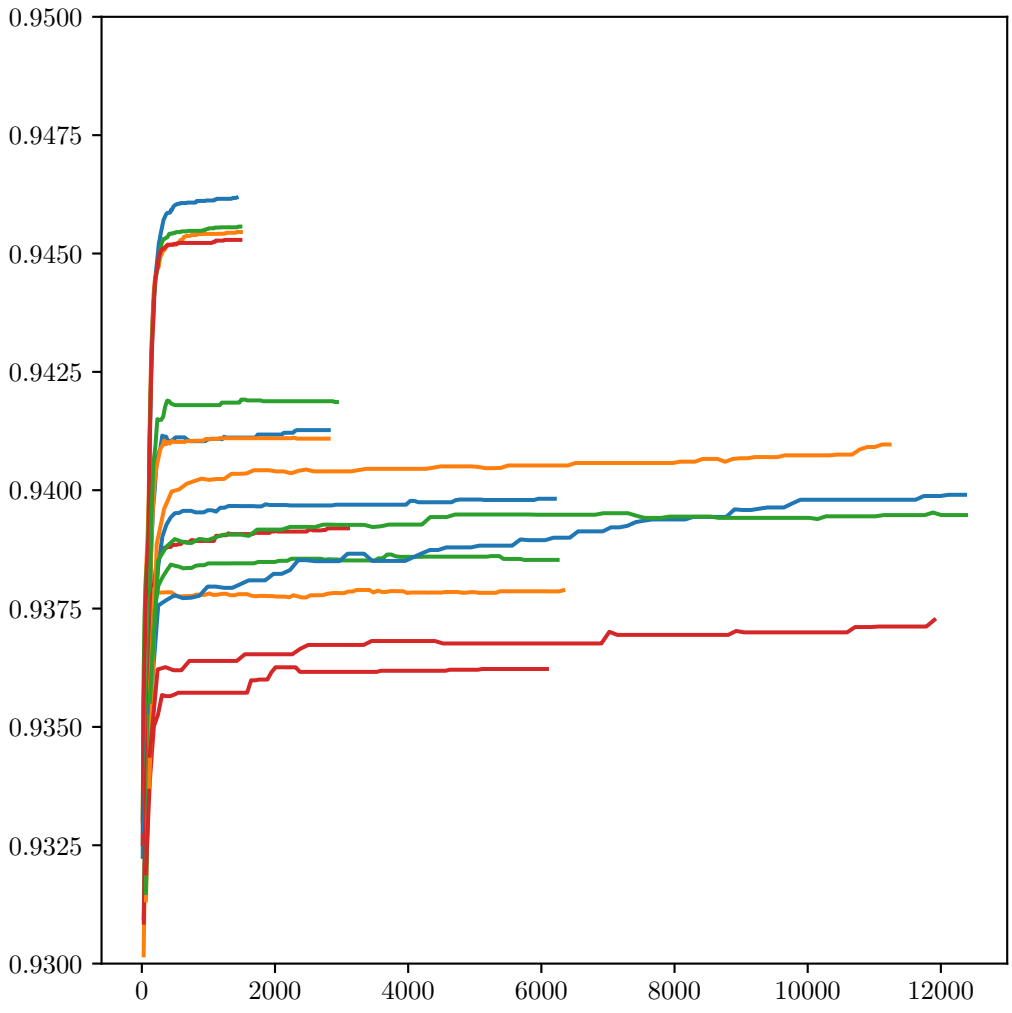

Figure 9: NASBench201 Simulation Performances vs History Size

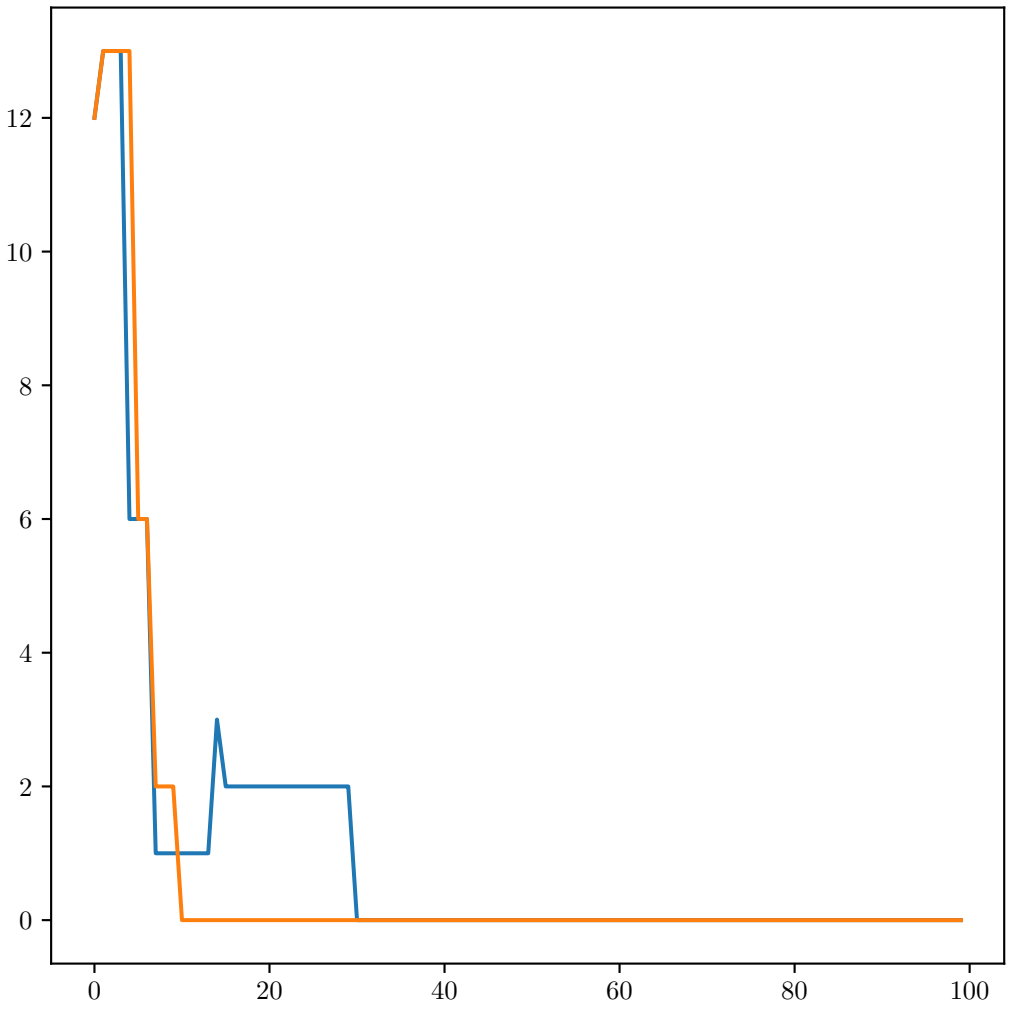

Figure 10: NASBench201 Simulation Performances vs Time Slice

Table 5: NASBench201 Points of Convergence

PES	WS	ZM PARE	RM PARE	PARE Ratio	ZM PNRE	RM PNRE	PNRE Ratio	ZM Spr	RM Spr	Spr Ratio
1	1	0.088	0.086	1.019	0.088	0.089	0.984	0.92	0.924	0.996
1	0.5	0.059	0.06	0.984	0.052	0.054	0.969	0.953	0.96	0.993
1	0.25	0.057	0.053	1.063	0.052	0.049	1.065	0.965	0.969	0.996
1	0.001	0	0	0	0	0	0	1	1	1
0.5	1	0.148	0.144	1.027	0.155	0.147	1.057	0.766	0.778	0.985
0.5	0.5	0.137	0.138	0.988	0.144	0.142	1.015	0.815	0.794	1.027
0.5	0.25	0.137	0.137	1.003	0.139	0.144	0.968	0.797	0.769	1.037
0.5	0.001	0.181	0.178	1.017	0.195	0.192	1.017	0.66	0.668	0.988
0.25	1	0.176	0.162	1.088	0.185	0.173	1.068	0.659	0.724	0.911
0.25	0.5	0.188	0.174	1.077	0.204	0.184	1.108	0.613	0.673	0.911
0.25	0.25	0.182	0.173	1.049	0.199	0.185	1.078	0.687	0.705	0.975
0.25	0.001	0.21	0.21	1	0.227	0.227	1	0.573	0.573	1
0.125	1	0.183	0.176	1.038	0.196	0.187	1.046	0.68	0.697	0.976
0.125	0.5	0.182	0.174	1.048	0.196	0.182	1.077	0.687	0.7	0.982
0.125	0.25	0.19	0.178	1.069	0.205	0.188	1.089	0.648	0.694	0.934
0.125	0.001	0.192	0.192	1	0.202	0.202	1	0.621	0.621	1

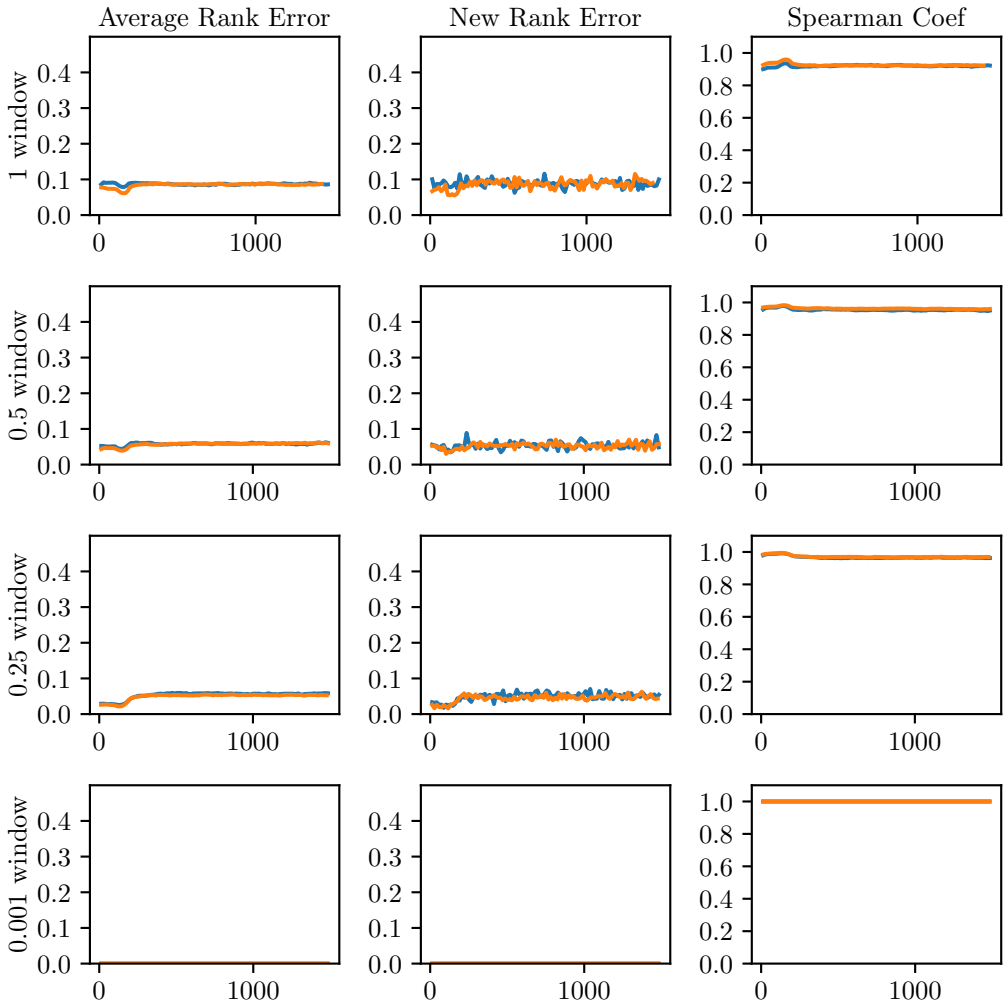

Figure 11: NASBench201 Simulation with No Acceleration

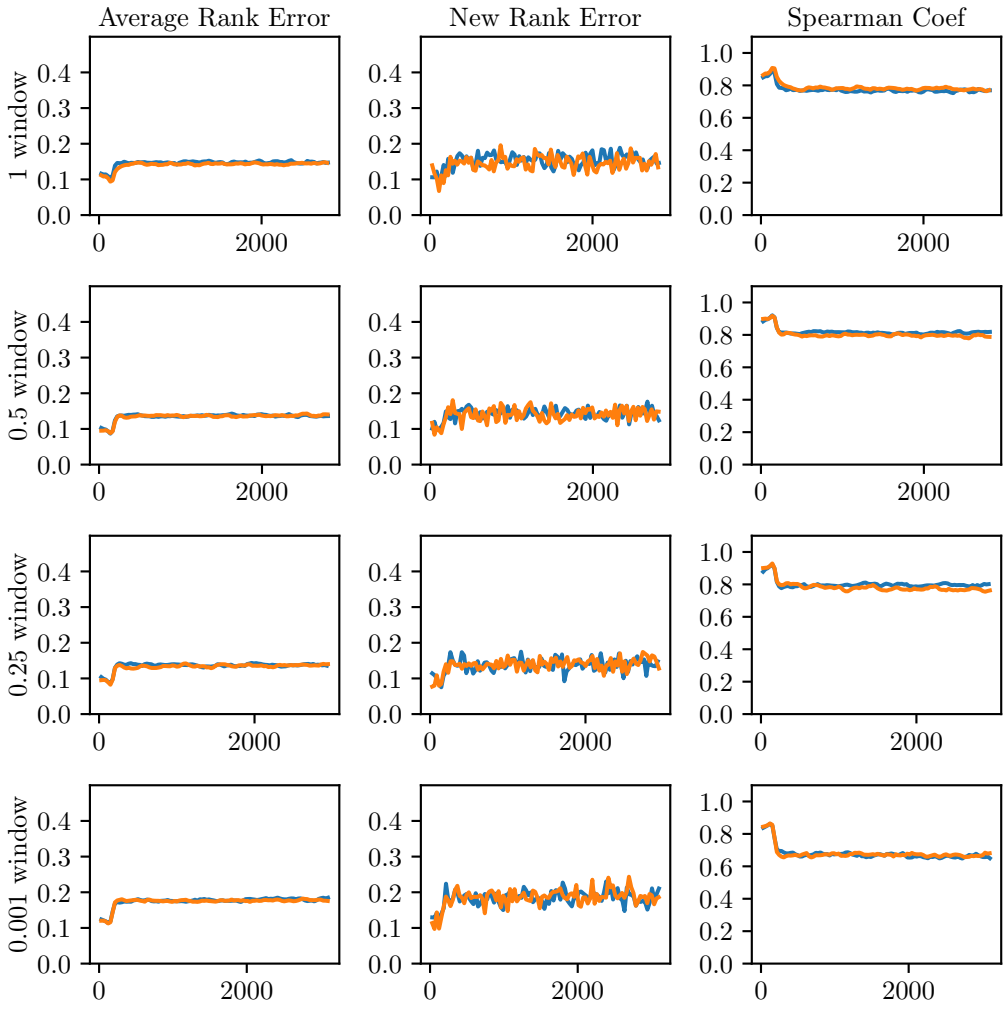

Figure 12: NASBench201 Simulation with 2x Acceleration

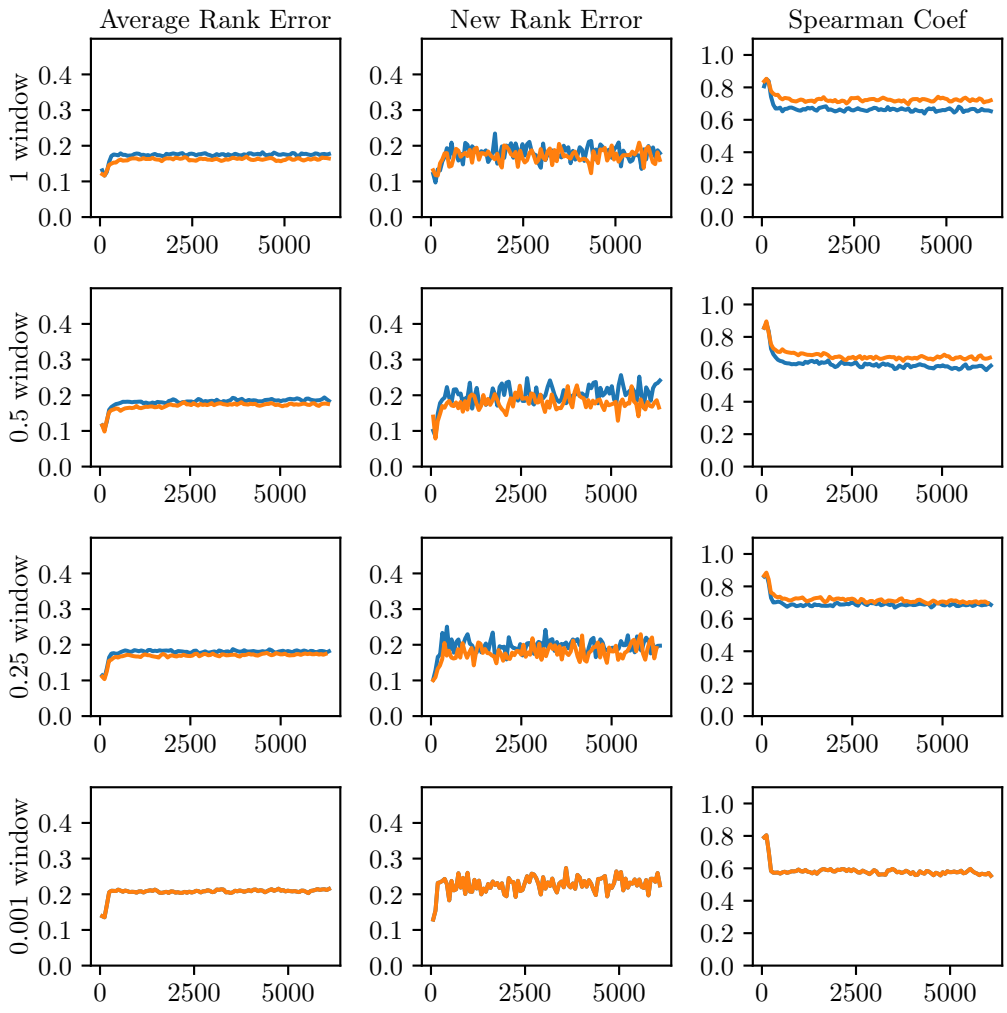

Figure 13: NASBench201 Simulation with 4x Acceleration

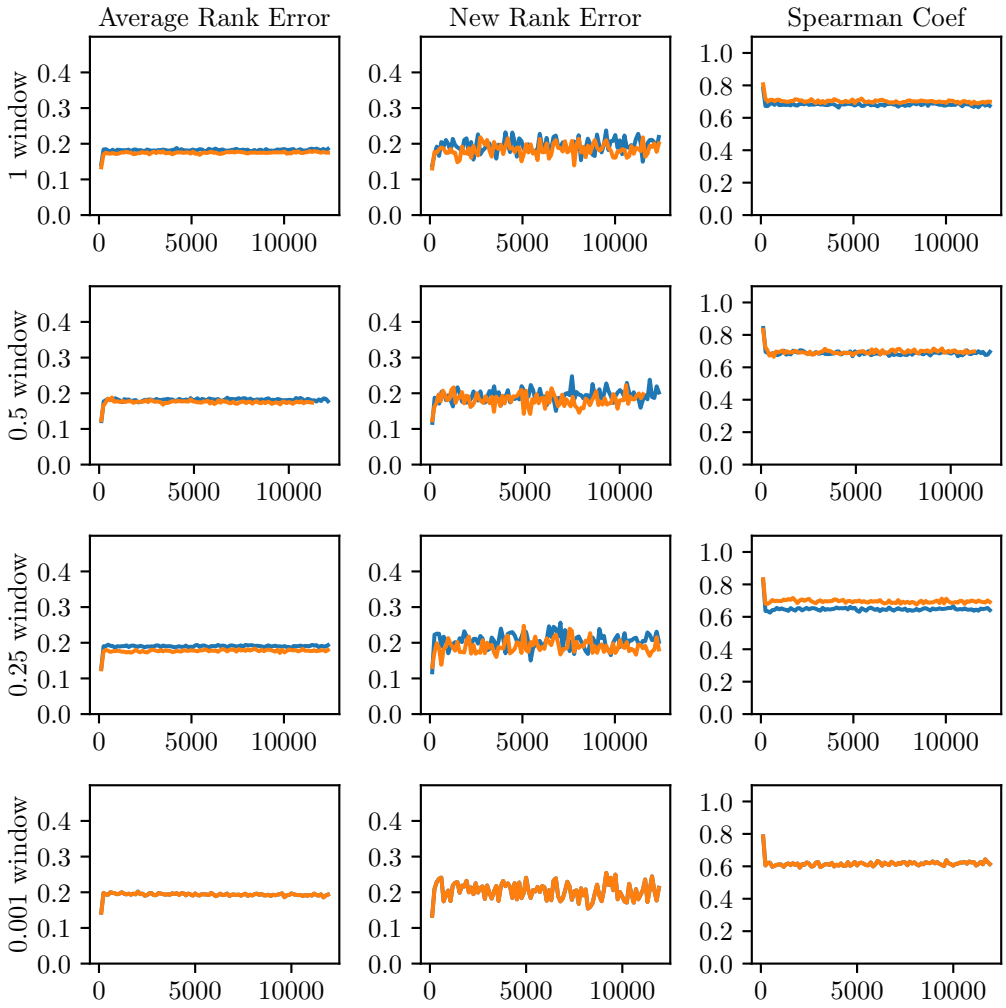

Figure 14: NASBench201 Simulation with 8x Acceleration

A.2.1 Aging Tournament Ablation Study

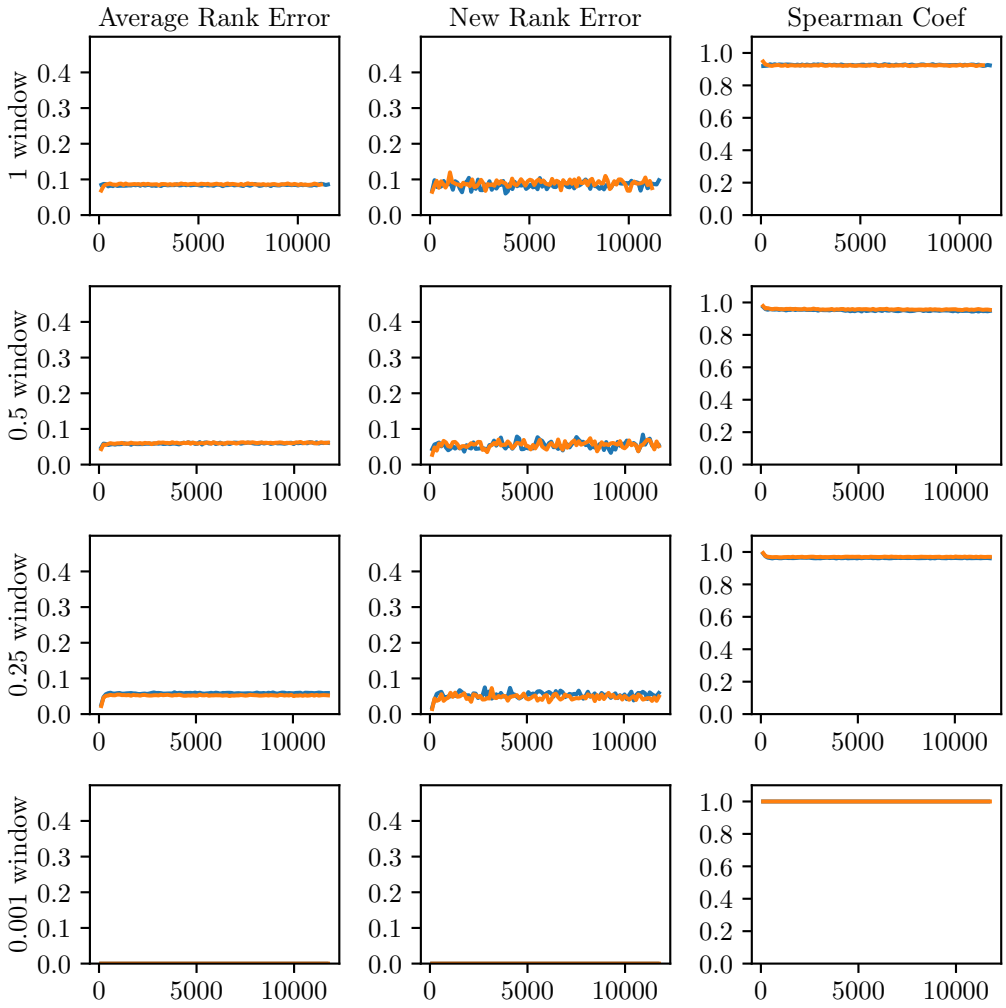

Figure 15: NASBench201 Simulation with No Acceleration

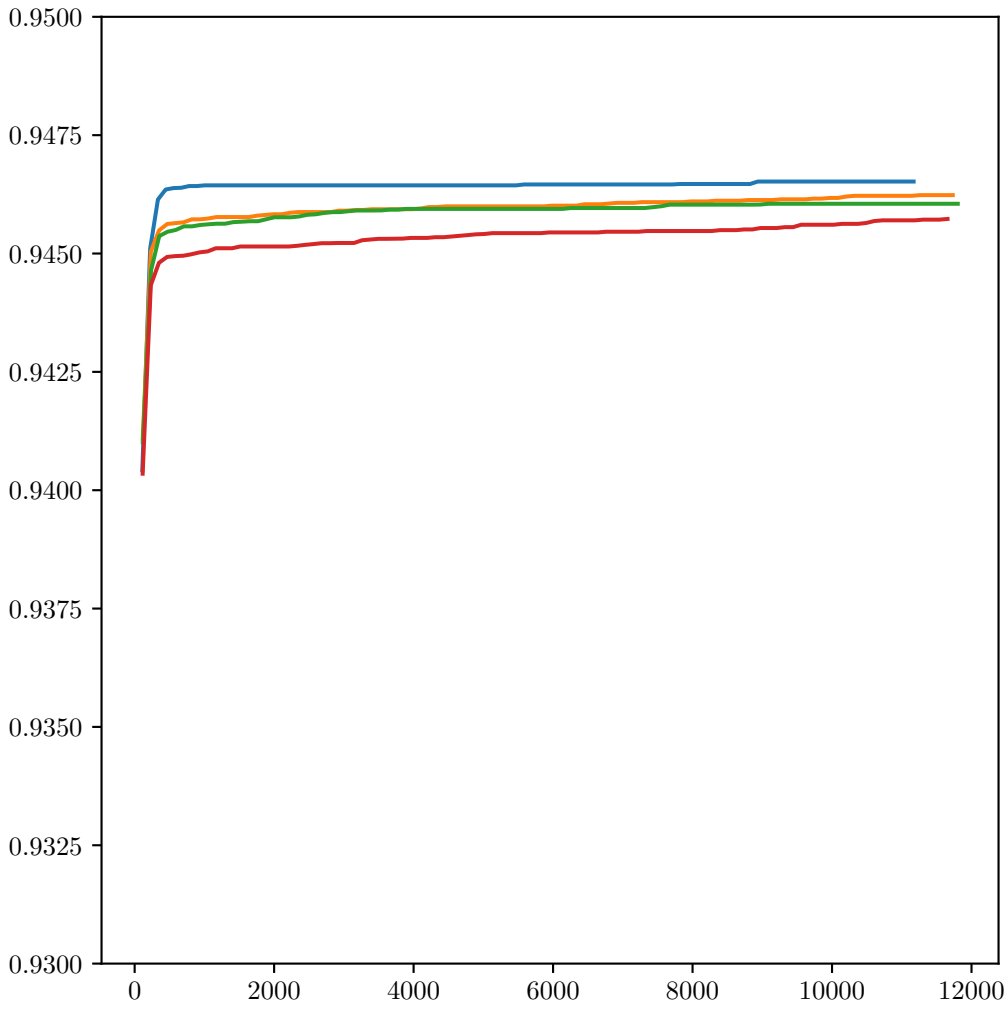

Figure 16: NASBench201 Simulation with No Acceleration

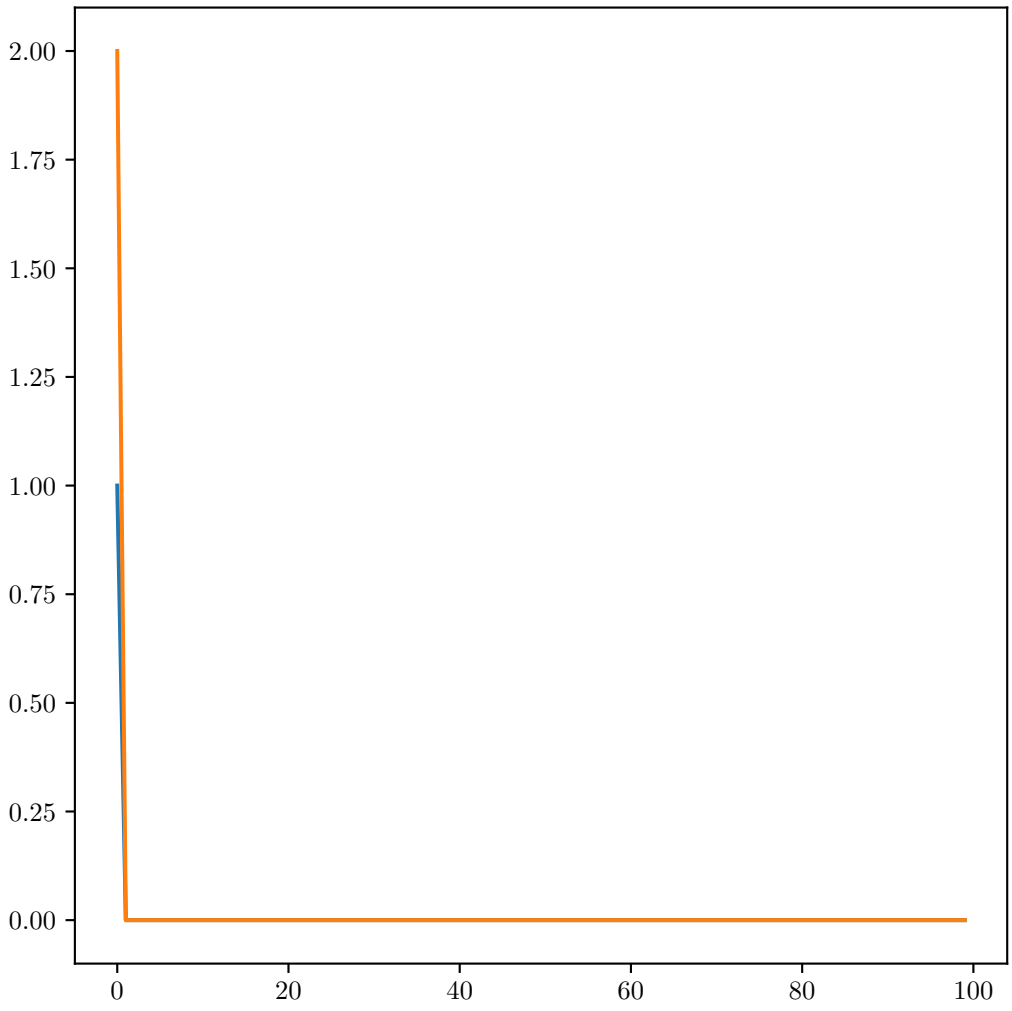

Figure 17: NASBench201 Simulation with No Acceleration

A.3 NASNet Population Details

The search space is similar to that of NASNet, with only the modification of possible operations. Possible operations include the following:

- Identity
- Convolution 3x3
- Depthwise Seperable Convolution 3x3
- Depthwise Seperable Convolution 5x5
- Depthwise Seperable Convolution 7x7
- Average Pooling 3x3
- Average Pooling 5x5
- Max Pooling 3x3
- Max Pooling 5x5
- Depthwise Seperable Convolution 1x7 followed by Depthwise Seperable Convolution 7x1

This results in the search space of $(10^5 \cdot (6! - 2!))^2 = 5.155 \cdot 10^{15}$ architectures, in comparison to the NASNet space which has $(13^5 \cdot (6! - 2!))^2 = 7.107 \cdot 10^{16}$ architectures. The NASNet 16p population is comprised of the following six subpopulations specified by their hyperparameters, all of which are trained on Cifar-10. The population is intended to represent an architecture near the base architecture used to evaluate Cifar-10 in NASNet, in addition to a variety of smaller proxies.

Subpopulation	Layers	Normal Cells per Layer	Filters	Epochs	Parameters (μ)
1	3	3	24	16	1766642.5
2	3	3	24	32	1766642.5
3	3	5	24	16	2719517.5
4	3	5	32	16	4735204
4	3	6	32	16	5564662
4	3	6	32	32	5564662

Table 6: NASNet Subpopulation Configurations

UPDATE CHARTS TO INDICATE WHAT ORANGE/BLUE MEANS