EVALUATING RANKING LOSS FUNCTIONS IN PERFORMANCE PREDICTOR FOR NAS

Anonymous authors
Paper under double-blind review

ABSTRACT

Performance evaluation is a critical but compute-intensive procedure in neural architecture search (NAS). To alleviate evaluation costs, performance predictors have been widely adopted to predict architecture performance directly. Recent studies have introduced ranking loss functions into predictors to focus on the architecture rankings instead of absolute accuracy, thus enhancing the ranking ability of performance predictors. Despite the successful application of ranking loss functions, the lack of comprehensive measure metrics and different experimental configurations make a fair comparison among these loss functions a huge challenge. Additionally, some well-known ranking loss functions have not been thoroughly examined in the context of performance predictors. In this paper, we conduct the first study for 11 ranking loss functions containing the existing and the novel ones by comparing their effectiveness in performance predictors under various settings. We find that: (i) The choice of ranking loss function has a major influence on the performance of predictors; (ii) the quality of the architectures searched by the predictor-based NAS methods is closely correlated with the predictor's performance on top-centered rank metrics, rather than traditional metrics like Kendall Tau. We believe these results and insights can serve as recommendations for the optimal loss function to employ in predictors across various search spaces and experimental conditions.

1 INTRODUCTION

Neural architecture search (NAS) aims to automate the design of neural architectures (Elsken et al., 2019) and has been widely applied in various domains (Liu et al., 2018a; Pham et al., 2018; Ghiasi et al., 2019; Wang et al., 2020). One bottleneck of early NAS is the expensive performance evaluation stage, which involves training architectures from scratch (Zoph & Le, 2016; Real et al., 2017). To tackle this issue, performance predictors have been proposed to estimate the final performance of unseen architectures (Luo et al., 2018; Wen et al., 2020; Liu et al., 2021). Specifically, performance predictors can learn the mapping from architecture design to the corresponding performance using only a small number of trained architectures, thus greatly reducing computational costs. However, it remains challenging to accurately estimate the absolute performance of a given architecture at the budget of limited training samples (Xu et al., 2021). To overcome this challenge, recent advancements have proposed a paradigm shift from architecture performance prediction to architecture ranking prediction by replacing the widely-used mean square error (MSE) loss in predictors with pairwise and listwise ranking loss functions (Xu et al., 2021; Zheng et al., 2024; Hwang et al., 2024). This transition allows predictors to easily rank architectures in the correct order and thus facilitate NAS to discover the top-performing architectures.

Although ranking loss functions endow predictors with a better ranking ability, a fair comparison among them is still lacking currently. The results in prior works were mostly obtained under different experimental conditions where predictor types, search spaces, data splits, and hyperparameter tuning vary a lot from each other. Besides, only a few ranking losses have been applied to performance predictors previously because most losses were originally designed for information retrieval. Meanwhile, the relationship between the predictors' performance in rank-based metrics and their results in NAS experiments has not been well investigated.

This naturally raises the following questions: how do pointwise, pairwise and listwise ranking losses compare to each other under consistent conditions? Furthermore, could a ranking loss with excellent performance on ranked-based metrics indicate promising searching results for predictor-based NAS methods?

We answer the questions above by studying 11 ranking loss functions across five search spaces: NAS-Bench-101 (Ying et al., 2019), NAS-Bench-201 (Dong & Yang, 2019), TransNAS-Bench-101 (Duan et al., 2021), NAS-Bench-NLP (Klyuchnikov et al., 2022) and DARTS (Liu et al., 2018b). These ranking losses can be categorized into pointwise, pairwise, listwise, and weighted ones, with five of them used in previous predictors and six implemented newly by ourselves. To ensure a fair comparison between ranking losses, we employ a uniform predictor based on a graph convolution network (GCN) encoder in our experiments. The ranking ability of different ranking losses is assessed from three dimensions: training portion, test portion, and rank-based metrics. We adopt Kendall Tau (Kendall, 1938) and top-centered metrics such as Rel@K and N@K to test their overall ranking performance along with their performance in identifying top-ranked architectures. Then, we evaluate the ability of each ranking loss to facilitate two predictor-based NAS frameworks: predictor-guided random search (Bergstra & Bengio, 2012) and predictor-guided evolutionary search (Real et al., 2019). The relationship between the performance of ranking loss on rank-based metrics and the performance of predictor-based methods is clearly reflected in our experimental results.

Our findings indicate that the choice of ranking loss greatly affects the performance of predictors. For example, ListMLE (Xia et al., 2008) achieves a Kendall Tau of 0.55 for the Autoencoder task in TransNAS-Bench-101-Micro while MSE loss performs poorly with a score of 0.13. We also change the loss function in more complicated predictors for a more suitable one and find significant improvements in ranking performance. In addition, we find that a ranking loss performing well on top-centered metrics is more likely to aid predictor-based NAS in discovering promising architectures. In other words, we disprove the intuition that a predictor with high Kendall Tau is certain to help find well-performing architectures in NAS experiments. We show that a high Kendall Tau only indicates good overall ranking performance instead of a strong ability to recognize top-performing architectures. The latter matters more for NAS because predictor-based NAS methods mainly focus on discovering the optimal architecture in the search space.

We summarize our main contributions as follows:

- We provide the first comprehensive study for a series of pointwise, pairwise, listwise, and weighted ranking loss in performance predictors for NAS across various settings.
- We analyze our experimental results and find two key insights: (i) the importance of ranking loss in performance predictors; (ii) the correlation between performance on top-centered rank metrics and the quality of the architecture discovered by predictor-based NAS.
- We release a framework containing 11 ranking loss functions across 13 NAS tasks, which can serve as recommendations for predictors under different conditions. Note that six of them are adapted from applications in other domains.

2 Related Work

Performance Predictor for NAS. Performance predictors have recently emerged to boost the evaluation efficacy of NAS. Predictors can estimate the final performance of unseen architectures directly, allowing NAS methods to navigate the search space more efficiently. To improve the accuracy and generalization of predictors, many studies focus on extracting diverse features from architectures by opting for complicated models(Wen et al., 2020; Lu et al., 2023), optimizing architecture encoding(Ning et al., 2020; Yan et al., 2021) and incorporating tailored self-supervised tasks(Jing et al., 2022; Zheng et al., 2024). However, the design of loss functions for performance predictors plays an equally significant role but is still rarely explored.

Ranking Loss. Ranking losses, which stem from learning to rank in the field of information retrieval, can be generally categorized into three types: pointwise, pairwise, and listwise (Liu et al., 2009). Pointwise methods treat the ranking problem as a regression task and directly predict the exact relevance degree of each document. Pointwise ranking loss is optimized by minimizing the regret such as the square error (Cossock & Zhang, 2006) on the training data. Pairwise methods

are closer to the nature of 'ranking'. They focus on the relative ranking for a pair of documents and predict which one is preferred within each data pair. Aiming to reduce the number of incorrectly ranked pairs as much as possible, many pairwise methods are proposed based on support vector machines (Herbrich et al., 2000; Joachims, 2002), boosting Freund et al. (2003), neural networks (Burges et al., 2005), and other machine learning models (Tsai et al., 2007; Zheng et al., 2007). Also, MP loss (Cortes et al., 2007) is proposed to keep the magnitude of the labeled documents when ranking the document pairs. Listwise methods consider the entire set of documents and ranked output a list that is most likely to be the ground truth. They can be roughly divided into two sub-categories: measure-specific (Taylor et al., 2008; Chakrabarti et al., 2008) and measure-irrelevant (Xia et al., 2008; Cao et al., 2007) loss functions. Additionally, some weighted ranking losses (Burges et al., 2006; Weston et al., 2011; Wang et al., 2018) are proposed to better recognize the top-rank documents. Despite the success of the ranking loss, it was originally designed for information retrieval or other fields. The large difference between document label and architecture label makes it nontrivial to adapt the ranking loss to the architecture ranking prediction task. To address this, we implement several well-known ranking losses tailored for performance predictors.

Intersection Between Ranking Loss and Predictors. Most existing performance predictors primarily utilize MSE loss, which belongs to pointwise ranking methods (Luo et al., 2018; Liu et al., 2021; Lu et al., 2023). However, predicting the relative ranking between architectures is more critical and effective than estimating the absolute accuracy of architectures (Xu et al., 2021). Recent predictors have introduced pairwise and listwise ranking loss functions to enhance their ranking ability. ReNAS (Xu et al., 2021) and FlowerFormer (Hwang et al., 2024) separately employ the hinge ranking loss and the margin ranking loss to preserve the rankings between different architectures. DCLP (Zheng et al., 2024) adopts ListMLE (Xia et al., 2008), a listwise method, to achieve better overall ranking performance. Although ranking loss enables the predictors to rank architectures better, no prior works conducted a comprehensive and fair comparison between them under consistent settings to the best of our knowledge.

3 Background

In this section, we first introduce the basics of different categories of ranking loss functions. Then we discuss the evaluation metrics used to measure the ranking ability of performance predictors. The detailed descriptions for 11 ranking loss functions are included in Appendix A.1.

3.1 Ranking loss Functions

In this paper, we compare four different categories of 11 ranking loss: pointwise ranking loss, pairwise ranking loss, listwise ranking loss and weighted ranking loss. Note that weighted ranking loss is mostly developed from pairwise and listwise ones. Formally, given N architecture-performance pairs $\{(x_i, y_i)\}_{i=1}^{N}$, the predictor takes architectures as input and output predicted values $\{\hat{y}_i\}_{i=1}^{N}$.

Pointwise Ranking Loss. MSE loss, the traditional option for performance predictors, is chosen to represent the pointwise ranking loss in our experiments. For each input architecture x_i, the predictor aims to minimize the square loss $(y_i - \hat{y}_i)^2$ between predicted scores and ground truth. After that, the architecture ranking is calculated based on the predicted scores.

Pairwise Ranking Loss. Four pairwise ranking loss functions are integrated into performance predictors. They are hinge ranking (HR) loss, margin ranking (MR) loss, bayesian personalized ranking (BPR) loss (Rendle et al., 2009) and magnitude preserving (MP) loss (Cortes et al., 2007). For each input architecture pair (x_i, x_j), the predictor is encouraged to predict relative ranking order correctly. The overall ranking is based on the massive relative ranking pairs. Among them, MP loss is special because it penalizes not only the mis-ranked pairs but also the correct ones if the magnitude of the prediction value is large.

Listwise Ranking Loss. ListNet (Cao et al., 2007) and ListMLE (Xia et al., 2008) losses are tested in our experiments. They optimize the entire architecture list $\{x_i\}_{i=1}^{N}$ to approach the correct ranking order and directly output a predicted ranking list for architectures.

Weighted Ranking Loss. Four weighted ranking loss functions are employed in our experiments which assign more weights to the top-ranked architectures when computed. We first develop the HR

loss for two weighted variants (short for WHR 1 and WHR 2) using different assignment manners. WHR 2 assigns more weights to the top-ranked architectures than WHR 1. Additionally, we introduce the well-known weighted approximate-rank pairwise (WARP) (Weston et al., 2011) loss and LambdaLoss (Wang et al., 2018) that assign weights in different manners.

3.2 Rank-based Metrics

In this paper, we employ the following rank-based metrics to evaluate the predictors' overall ranking performance and performance in identifying top-performing architectures.

Kendall Tau (Kendall, 1938) (τ). τ can reflect the ordinal association between a predicted score and the absolute accuracy of architectures for each task. We have $\tau \in [-1, 1]$ and the ranking correlation is stronger if τ is closer to 1. We opt for Kendall Tau as the representative ranking correlation metric for the sake of its popularity in the field of performance predictors (Ning et al., 2020; Liu et al., 2021; Lu et al., 2023; Hwang et al., 2024).

Weighted Kendall Tau (Vigna, 2015) (τ_w). Compared to vanilla τ, τ_w assigns weights to architectures with top performance. There is a hyperbolic drop-off in architecture importance according to the descending order of accuracy. Predictors will earn a higher τ_w if the top-performing architectures are better ranked.

N@K. N@K denotes the true rank of the architecture that has the highest accuracy among the predicted top K ones. It is used to evaluate predictors because the relative rankings between architectures with poor performance is of little importance (Ning et al., 2020). If a promising architecture is predicted to top K, the performance in identifying top-performing architectures is good.

Rel@K. Distinguished from N@K, Rel@K is sensitive to the actual value of architecture. Rel@K computes the ratio of the accuracy of architecture A_K to that of the best one A_{max}. We have Rel@K$\in (0, 1]$. In view of its success in other ranking problems (Li et al., 2021), this metric is also suitable for evaluating the effectiveness of the predictor in NAS because the majority of previous works only report the accuracy of the best architectures instead of the rank (Wen et al., 2020; Liu et al., 2021; Jing et al., 2022; Yi et al., 2023; Zheng et al., 2024).

4 Empirical Evaluation of Ranking Loss Functions

We divide our experiments into two parts: evaluating the performance of each ranking loss in a variety of rank-based metrics (Section 4.1) and evaluating the effectiveness of predictor-based NAS methods (Section 4.2). We start by describing the experimental settings.

Experimental Settings. The experiments are conducted in five popular search spaces: NAS-Bench-101 (Ying et al., 2019), NAS-Bench-201 (Dong & Yang, 2019), TransNAS-Bench-101 (Duan et al., 2021), NAS-Bench-NLP (Klyuchnikov et al., 2022) and DARTS (Liu et al., 2018b). Note that TransNAS-Bench-101 includes the macro and the micro search space. Most predictors are only evaluated on the latter while our experiments include the results on both. Detailed information about these search spaces will be elaborated in Section B.1. Although performance predictors can benefit from ranking loss functions, their effectiveness is heavily dependent on model selection, data scale and hyperparameter tuning. To achieve a fair comparison, we opt for a uniform performance predictor which is composed of a four-layer GCN encoder and a three-layer MLP in our experiments. The detailed hyperparameter configuration is included in Appendix B.2.

4.1 Evaluating the Performance Predictors

The predictor is trained with a certain number of architecture-performance pairs and predicts the architectures in the test set. We evaluate the ranking loss functions in 13 different tasks with respect to four rank-based metrics: Kendall Tau, weighted Kendall Tau, N@K and Rel@K. The ranking loss is compared based on three dimensions: training portion, test portion and rank-based metrics. We also apply them to representative predictors for performance improvements in rank-based metrics.

Figure 1: Kendall Tau of different ranking loss functions in 13 different tasks across four search spaces (scaled up by a factor of 100). The results are averaged over 100 trials.

4.1.1 Performance in Ranking Correlation

We first use Kendall Tau to evaluate the overall ranking performance of 11 ranking losses for 13 tasks across four search spaces. In this experiment, the training portion varies from 0.02% to 1.25% for different tasks according to the scale of the search space and the test portion is fixed to 100%. The results are reported in Figure 1. Each column represents the performance on a given task in a specific search space.

Results on Different Tasks. From Figure 1, we can observe that no ranking loss functions consistently surpass other competitors across all 13 tasks. For instance, ListNet achieves the top-1 τ in NAS-Bench-101 while having the lowest τ in the TransNAS-Bench101-Micro Autoencoder task. Additionally, most pairwise ranking loss functions such as BPR, HR and MR obtain promising performance in general. Even in the difficult TransNAS-Bench101-Macro Autoencoder task, they still attain a high τ of over 0.74, outperforming most of the other loss functions by a large margin. Conversely, the majority of weighted ranking loss functions like WHR 2, WARP and LambdaLoss perform poorly in Kendall Tau because they focus more on the ranking between top-performing architectures. WHR 1 is an exception because its weight assignment is not significant and preserves the ability of HR in overall architecture ranking.

To explore the effect of ranking loss in various predictor scenarios, we further compare them with respect to Kendall Tau using 16 combinations of training portion and test portion in different tasks in Figure 2. For each (training portion, test portion) pair, the optimal ranking loss and the corresponding τ are marked. Note that the ranking losses of the same category are plotted with similar colors for clarity. For example, pairwise ranking losses are colored with different types of blue.

Results on Different Data Splits. For page limit, the complete results of other tasks are reported in Appendix C.1. Figure 2 clearly illustrates that the optimal ranking loss under different data splits is not consistent even in the same task. In most tasks, only three or four out of the 11 ranking loss own the best performance over 16 data splits. Besides, the best ranking loss changes frequently and gets significant performance improvements in each row. But from each column, the optimal loss changes a little and the performance is stable. This indicates that the training portion is more significant to the optimal option of ranking loss in terms of τ for each task.

When the training portion is small, the completion for the optimal loss is fierce. At the budget of 0.5% training data on NAS-Bench-201, ListMLE beats others in CIFAR10 and CIFAR100 dataset (Krizhevsky et al., 2009) while MSE performs best in ImageNet16-120 dataset (Chrabaszcz et al., 2017). For the largest NAS-Bench-101, MP achieves the highest τ. When it comes to TransNAS-Bench-101-Macro, WHR 1 is the most competitive both on the class-scene and class-

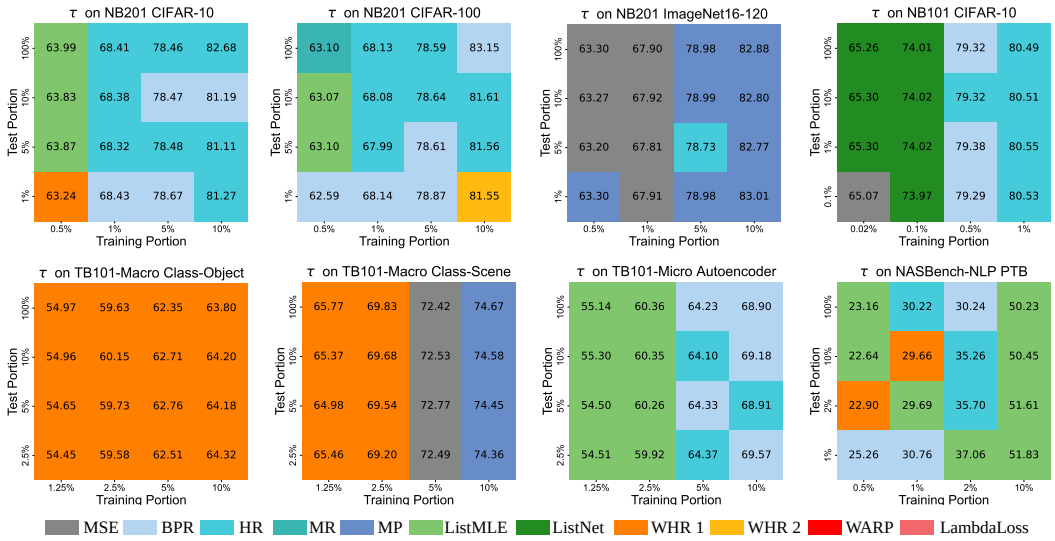


Figure 2: Kendall Tau of different ranking loss functions under various settings (scaled up by a factor of 100). The MSE loss is colored in gray. The pairwise ranking losses are colored in blue. The listwise ranking losses are colored in green. The weighted ranking losses are colored in red and orange. The results are averaged over 100 trials.

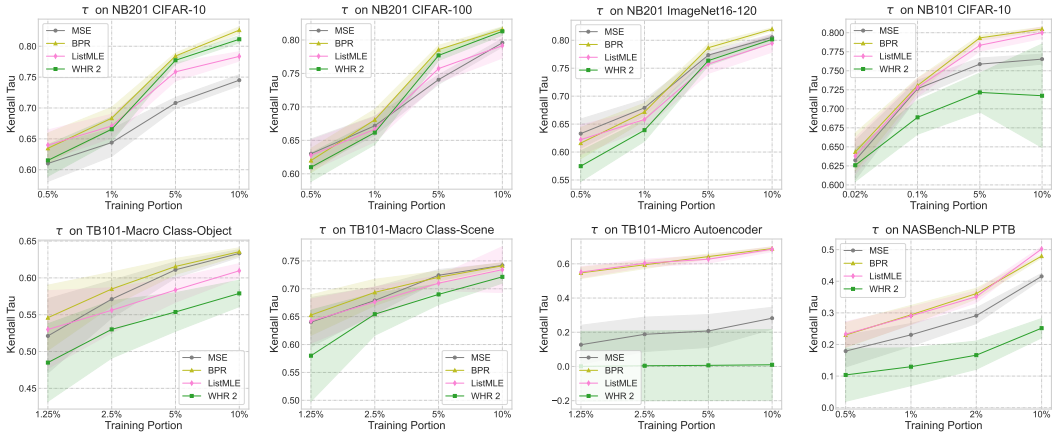


Figure 3: Kendall Tau of different ranking loss functions as the training portion grows. The results are averaged over 100 trials.

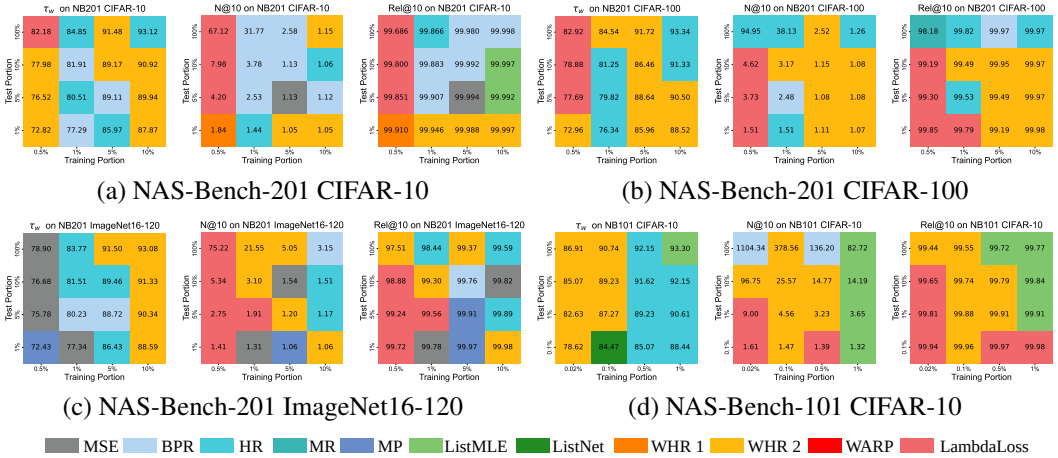
object tasks. For the challenging TransNAS-Bench-101-Micro Autoencoder and NAS-Bench-NLP PTB (Kombrink et al., 2011) tasks, ListMLE outperforms other ranking losses.

When the training portion becomes large, pairwise ranking losses such as BPR exhibit a dominant advantage in four tasks across three search spaces. This is consistent with the results in previous works (Xu et al., 2021; Hwang et al., 2024) that the performance predictor can achieve a much higher τ with pairwise ranking loss than MSE loss.

To better observe the performance of ranking losses as the training portion grows, we plot the τ of four ranking losses as representatives for different types of losses in Figure 3. We find that the pairwise and listwise ranking losses generalize better than the weighted and pointwise ranking ones because they optimize the overall ranking of architecture.

4.1.2 PERFORMANCE IN TOP-CENTERED RANK METRICS

To evaluate the performance in recognizing top-performing architectures, we compare τ_w, N@K and Rel@K of different ranking losses under 16 data splits in Figure 4. The K is set to 10 because we focus more on the top-performing architectures.



(a) NAS-Bench-201 CIFAR-10 (b) NAS-Bench-201 CIFAR-100

(c) NAS-Bench-201 ImageNet16-120 (d) NAS-Bench-101 CIFAR-10

Figure 4: τ_w, N@10 and Rel@10 of different ranking loss functions on NAS-Bench-101 and NAS-Bench-201. Higher is better for τ_w and Rel@10 while lower is better for N@10. The results are averaged over 100 trials.

Weighted Kendall Tau Results. Figure 4 illustrates that WHR 2 achieves the highest τ_w under half of 16 data splits on NAS-Bench-201 CIFAR-10. Besides, BPR and HR maintain their competitive performance in τ and attain promising τ_w values. In addition, WHR 2 and LamadaLoss perform very well in τ_w but neither of them becomes the optimal loss for τ on the same task in Figure 2.

N@K and Rel@K Results. From Figure 4, we can see that weighted ranking loss shows excellent ability in N@10 and Rel@10. On NAS-Bench-201 CIFAR-10, WHR 2 achieves a very low N@10 of 1.15 with a training portion of 10% and a test portion of 100%. As the training portion grows, two pairwise ranking loss functions: BPR and HR yield impressive results in both metrics.

Figure 4 shows a very different trend from that of the overall ranking performance in Figure 2. Most Weighted ranking losses perform poorly in τ but exhibit high top-centered metrics especially when the training portion is low. Our results suggest that ranking losses which assign more weights to well-performing architectures can compensate for the limited training sample to identify top-performing architectures. Additionally, pairwise ranking losses such as BPR and HR can still achieve excellent results on top-centered metrics when the training portion gets larger. This indicates that pairwise ranking can leverage the ranking relationship between massive training samples to precisely recognize the top-performing architectures.

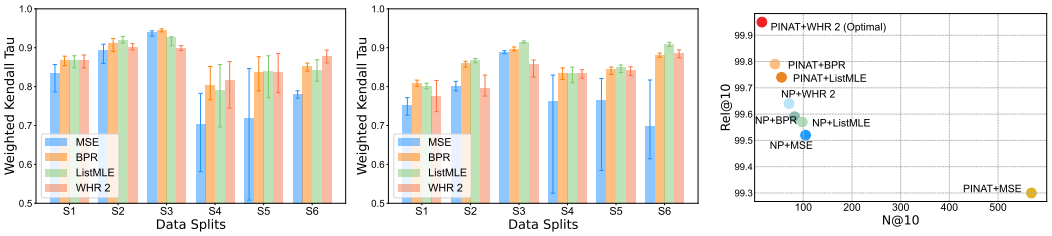
4.1.3 ENHANCING EXISTING PREDICTORS

To further investigate the power of ranking losses, we combine them with two representative performance predictors: NP (Wen et al., 2020) and PINAT (Lu et al., 2023). Both predictors have elaborately designed models but adopt the simple MSE loss function. In this part, BPR, ListMLE and WHR 2 loss are chosen as the representatives for their excellent performance in the rank-based metrics. The predictors are evaluated with four rank-based metrics for six data splits on NAS-Bench-101 and NAS-Bench-201. Note that the test portion is set to 100% in the following experiments. The results are averaged over 10 runs.

Results on Kendall Tau. Table 1 shows that changing the ranking loss in predictors can bring significant performance improvements in τ. For example, PINAT and NP obtain a 6.62% and 12.04% gain with ListMLE loss on NAS-Bench-201 CIFAR-10 using a training portion of 0.5%, respectively. We also find that ListMLE loss helps NP realize a high τ of 0.68 with only 1% training samples, which even outperforms the result of MSE loss at a training portion of 10%.

Datasets	NAS-Bench-101 CIFAR-10			NAS-Bench-201 CIFAR-10		
Training portion	0.02%	0.1%	1%	0.5%	1%	10%
PINAT + MSE	63.61±1.63	75.67±1.42	84.40±0.33	55.51±2.20	62.34±2.45	77.96±0.64
PINAT + BPR	66.21±0.71	76.75±1.04	84.66±0.02	61.32±2.09	67.11±1.81	85.18±0.07
PINAT + ListMLE	**66.35±0.91**	**77.56±0.48**	**84.69±0.12**	**62.13 ±2.10**	**67.65±1.70**	**85.60±0.30**
PINAT + WHR 2	65.59±0.98	70.65±0.61	68.14±0.06	61.79 ±1.90	65.10±2.05	75.29±1.99
NP + MSE	52.99±1.63	63.14±0.05	72.71±0.02	52.14±10.69	52.20±5.83	66.26±1.30
NP + BPR	61.21±0.04	69.59±0.03	75.91±0.01	63.87±0.18	66.03±1.40	76.33±0.05
NP + ListMLE	**62.06±0.33**	**70.41±0.22**	**78.92±0.07**	**64.18±2.32**	**68.22±1.64**	**80.26±0.35**
NP + WHR 2	48.40±0.91	53.49±0.97	60.53±0.56	62.45±1.24	63.91±1.05	73.67±0.72

Table 1: Kendall Tau (scaled up by a factor of 100, mean and standard deviation) on NAS-Bench-101 and NAS-Bench-201.



(a) τ_w for PINAT (b) τ_w for NP (c) N@10 and Rel@10

Figure 5: τ_w, N@10 and Rel@10 for different ranking loss with PINAT and NP. S1 to S6 correspond to the six data splits in Table 1. The N@10 and Rel@10 are computed with a 1% training portion on NAS-Bench-201 CIFAR-10.

Results on Top-centered Metrics. From Figure 5(a) and Figure 5(b), we can see that both PINAT and NP roughly attain a higher τ_w with BPR and ListMLE loss. Meanwhile, NP encounters a severe performance drop with MSE loss on NAS-Bench-201 as the training portion grows from 1% to 10% (S5 to S6) because MSE loss is weak at leveraging massive training data in exploration of well-performing architectures. As for N@10 and Rel@10, the results in Figure 5(c) demonstrate the superior performance of WHR 2 loss that PINAT can almost discover the optimal architecture within top10 prediction at a budget of only 1% training data on NAS-Bench-201.

These results suggest that a suitable ranking loss can significantly enhance predictors' capability in correct overall architecture ranking and identifying top-performing architectures. The importance of the loss function can match the model design and the number of training samples for performance predictors.

4.2 Evaluating Predictor-based NAS

Now we investigate to what extent ranking loss functions facilitate predictor-based NAS methods. In this section, we select MSE, BPR, ListMLE and WHR 2 to represent the pointwise, pairwise, listwise and weighted ranking losses for their good performance in rank-based metrics, respectively. From previous experiments, we can see that in the low training portion region, pairwise and listwise ranking loss achieve higher τ while weighted ranking loss performs better in top-centered metrics. We opt for two popular predictor-based NAS frameworks: predictor-guided random search (Bergstra & Bengio, 2012) and predictor-guided evolutionary search (Real et al., 2019). All the results are averaged over 20 trials. Additional experimental results for predictor-based NAS frameworks can be found in Appendix C.2.

4.2.1 Predictor-Based NAS Frameworks

We give a brief introduction to both frameworks and the detailed procedures are in Appendix A.2.

Predictor-Guided Random Search. This framework randomly samples candidate architectures from the search space several times and employs the predictor to select the top-ranked ones from

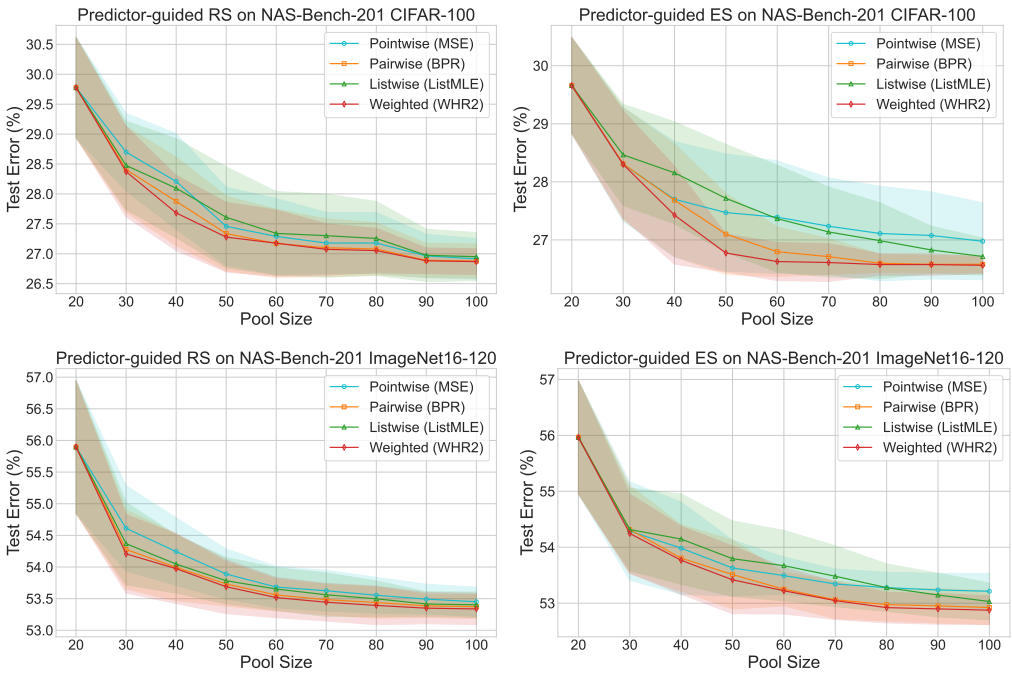


Figure 6: Test error vs. Pool size for the predictor-guided random search (RS) and the predictor-guided evolutionary search (ES) framework with different ranking losses on NAS-Bench-201.

	BANANAS	CATE	DCLP	WeakNAS	Ours (ES)			
					MSE	BPR	ListMLE	WHR2
# Queries	150	150	300	200	150	150	150	150
Test Error (%)	5.92	5.88	5.83	5.82	5.95	5.86	5.84	**5.81**

Table 2: Comparison with previous works on NAS-Bench-101.

the candidates. Note that the predictor is trained on an initial architecture pool P in advance. The newly chosen architectures are then added to P and the predictor is updated upon the expanded architecture pool. This update process is iterated until the size of P reaches the query budget. The best architecture in P is chosen as the final search result.

Predictor-Guided Evolutionary Search. The predictor-guided evolutionary search framework also searches for promising architectures with an updated performance predictor. To expand the architecture pool during the search, we evolve the architectures in the pool to generate new candidate individuals and utilize the predictor to pick out ones that are predicted to perform well.

4.2.2 Results

Results on NAS-Bench-201. The results on NAS-Bench-201 are presented in Figure 6. We find that weighted ranking loss achieves the lowest test error under both frameworks. For instance, WHR 2 outperforms other ranking losses consistently in CIFAR-100 using the predictor-guided ES framework. Although BPR loss also yields favorable results with the full query budget, WHR 2 loss has a significant advantage in the region of low pool size.

Results on NAS-Bench-101. We adopt a predictor-guided evolutionary search framework and compare our method with previous SotA predictors: BANANAS (White et al., 2021), CATE (Yan et al., 2021), WeakNAS (Wu et al., 2021) and DCLP (Zheng et al., 2024). From Table 2, We find a major decrease in test error when changing the loss function from MSE to WHR 2. Meanwhile, WHR 2 loss helps our method outperform the previous SotA methods with the fewest queries. Note that the predictor used in our method is only composed of a basic GCN encoder and a simple MLP regressor.

Tasks	Cls.O.	Cls.S.	Auto.	Jigsaw	Avg. Rank
Metric	Acc.↑	Acc.↑	SSIM↑	Acc.↑	
RS (Bergstra & Bengio, 2012)	45.16	54.41	55.94	94.47	61.00
REA (Real et al., 2019)	45.39	54.62	**56.96**	94.62	27.50
MSE	45.36	54.66	55.59	94.75	41.25
BPR	45.47	54.76	56.20	94.77	24.25
ListMLE	45.53	54.66	55.95	94.63	35.5
WHR 2	**45.66**	**54.83**	56.57	**94.84**	**15.75**
Global Best	46.32	54.94	57.72	95.37	1

Table 3: Searching results on TransNAS-Bench-101-Micro.

Predictor	Test Err.(%)	Params(M)	GPU Days	Search Method	Loss Function
TNASP (Lu et al., 2021)	2.57±0.04	3.6	0.3	Evolution	MSE
CDP (Liu et al., 2022)	2.63±0.08	3.3	0.1	Random	MSE
NPENAS (Wei et al., 2022)	2.54±0.10	3.5	1.8	Evolution	MSE
PINAT (Lu et al., 2023)	2.54±0.08	3.6	0.3	Evolution	MSE
Ours (GCN + MLP)	2.74±0.04	4.4	0.14	Random	MSE
	2.57±0.04	3.6	0.14	Random	BPR
	2.62±0.07	3.8	0.14	Random	ListMLE
	2.48±0.04	4.3	0.14	Random	WHR 2

Table 4: Performance comparison between the architectures searched by different ranking loss functions and other predictor-based methods on CIFAR-10.

Results on TransNAS-Bench-101-Micro. We also search for architectures on TransNAS-Bench-101-Micro using a predictor-guided evolutionary framework and compare with baseline methods in Table 3. The query budget is set to 50. Among all, WHR 2 loss attains the best average rank on four tasks. Meanwhile, we find the average rank of MSE and ListMLE losses are inferior to the baselines. This performance gap indicates that weighted ranking loss can significantly boost the accuracy of the predictor and lead to a promising solution in the search space.

Results on DARTS. Table 4 demonstrates the results of the architectures discovered by four representative ranking losses on DARTS search space for CIFAR-10. We find that WHR 2 loss achieves the lowest test error rate of 2.48% in only 0.14 GPU Days, which outperforms previous SotA predictor-based methods. Besides, pairwise and listwise ranking loss perform better than MSE loss within the same predictor, which reflects the significance of the ranking loss function for predictor-based NAS methods. This suggests that when the training samples for performance predictors are limited, the weighted ranking loss is the optimal choice in pursuit of a promising architecture for new tasks because of their excellent performance on top-centered rank metrics.

5 Conclusion

In this paper, we provide the first comprehensive study for 11 ranking loss functions in performance predictors, including pointwise, pairwise, listwise and weighted ranking loss. We compare their performance in ranking correlation metrics and top-centered metrics under various settings. Meanwhile, we combine these ranking loss functions into predictor-based NAS methods to search for promising architectures. We find ranking loss with excellent performance on top-centered rank metrics can help find architectures with high quality. In addition, our experiments show that a well-designed ranking loss can greatly improve the performance of existing predictors and predictor-based NAS methods. In the future, we will study some complex ranking loss functions which are composed of several single losses to further enhance performance predictors.

REFERENCES

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pp. 89–96, 2005.

Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems*, 19, 2006.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136, 2007.

Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. Structured learning for non-smooth ranking losses. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 88–96, 2008.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. Magnitude-preserving ranking algorithms. In *Proceedings of the 24th international conference on Machine learning*, pp. 169–176, 2007.

David Cossock and Tong Zhang. Subset ranking using regression. In *Learning Theory: 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006. Proceedings 19*, pp. 605–619. Springer, 2006.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *Proc. of ICLR*, 2019.

Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5251–5260, 2021.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7036–7045, 2019.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. 2000.

Minbin Huang, Zhijian Huang, Changlin Li, Xin Chen, Hang Xu, Zhenguo Li, and Xiaodan Liang. Arch-graph: Acyclic architecture relation predictor for task-transferable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11881–11891, 2022.

Dongyeong Hwang, Hyunju Kim, Sunwoo Kim, and Kijung Shin. Flowerformer: Empowering neural architecture encoding using a flow-aware graph transformer. In *Proc. of CVPR*, pp. 6128–6137, 2024.

Kun Jing, Jungang Xu, and Pengfei Li. Graph masked autoencoder enhanced predictor for neural architecture search. In *Proc. of IJCAI*, 2022.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142, 2002.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, Alexander Filippov, and Evgeny Burnaev. Nas-bench-nlp: neural architecture search benchmark for natural language processing. *IEEE Access*, 10:45736–45747, 2022.

Stefan Kombrink, Tomas Mikolov, Martin Karafiát, and Lukás Burget. Recurrent neural network based language modeling in meeting recognition. In *Interspeech*, volume 11, pp. 2877–2880, 2011.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yandong Li, Xuhui Jia, Ruoxin Sang, Yukun Zhu, Bradley Green, Liqiang Wang, and Boqing Gong. Ranking neural checkpoints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2663–2673, 2021.

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. In *Proc. of ECCV*, 2018a.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *Proc. of ICLR*, 2018b.

Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

Yuqiao Liu, Yehui Tang, and Yanan Sun. Homogeneous architecture augmentation for neural predictor. In *Proc. of ICCV*, 2021.

Yuqiao Liu, Yehui Tang, Zeqiong Lv, Yunhe Wang, and Yanan Sun. Bridge the gap between architecture spaces via a cross-domain predictor. *Advances in Neural Information Processing Systems*, 35:13355–13366, 2022.

Shun Lu, Jixiang Li, Jianchao Tan, Sen Yang, and Ji Liu. Tnasp: A transformer-based nas predictor with a self-evolution framework. *Advances in Neural Information Processing Systems*, 34:15125–15137, 2021.

Shun Lu, Yu Hu, Peihao Wang, Yan Han, Jianchao Tan, Jixiang Li, Sen Yang, and Ji Liu. Pinat: A permutation invariance augmented transformer for nas predictor. In *Proc. of AAAI*, 2023.

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. *Proc. of NeurIPS*, 2018.

Xuefei Ning, Yin Zheng, Tianchen Zhao, Yu Wang, and Huazhong Yang. A generic graph-based neural architecture encoding scheme for predictor-based nas. In *Proc. of ECCV*, 2020.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pp. 4095–4104. PMLR, 2018.

Przemyslaw Pobrotyn, Tomasz Bartczak, Mikolaj Synowiec, Radoslaw Bialobrzeski, and Jaroslaw Bojar. Context-aware learning to rank with self-attention. *ArXiv*, abs/2005.10084, 2020.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International conference on machine learning*, pp. 2902–2911. PMLR, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proc. of UAI*, 2009.

Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pp. 77–86, 2008.

Ming-Feng Tsai, Tie-Yan Liu, Tao Qin, Hsin-Hsi Chen, and Wei-Ying Ma. Frank: a ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 383–390, 2007.

Sebastiano Vigna. A weighted correlation index for rankings with ties. In *Proc. of WWW*, pp. 1166–1176, 2015.

Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pp. 1313–1322, 2018.

Yujing Wang, Yaming Yang, Yiren Chen, Jing Bai, Ce Zhang, Guinan Su, Xiaoyu Kou, Yunhai Tong, Mao Yang, and Lidong Zhou. Textnas: A neural architecture search space tailored for text representation. In *Proc. of AAAI*, 2020.

Chen Wei, Chuang Niu, Yiping Tang, Yue Wang, Haihong Hu, and Jimin Liang. Npenas: Neural predictor guided evolution for neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8441–8455, 2022.

Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *Proc. of ECCV*, 2020.

Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*. Citeseer, 2011.

Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10293–10301, 2021.

Junru Wu, Xiyang Dai, Dongdong Chen, Yinpeng Chen, Mengchen Liu, Ye Yu, Zhangyang Wang, Zicheng Liu, Mei Chen, and Lu Yuan. Stronger nas with weaker predictors. *Advances in Neural Information Processing Systems*, 34:28904–28918, 2021.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proc. of ICML*, 2008.

Yixing Xu, Yunhe Wang, Kai Han, Yehui Tang, Shangling Jui, Chunjing Xu, and Chang Xu. Renas: Relativistic evaluation of neural architecture search. In *Proc. of CVPR*, 2021.

Shen Yan, Kaiqiang Song, Fei Liu, and Mi Zhang. Cate: Computation-aware neural architecture encoding with transformers. In *International Conference on Machine Learning*, pp. 11670–11681. PMLR, 2021.

Yun Yi, Haokui Zhang, Wenze Hu, Nannan Wang, and Xiaoyu Wang. Nar-former: Neural architecture representation learning towards holistic attributes prediction. In *Proc. of CVPR*, 2023.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *Proc. of ICML*, 2019.

Shenghe Zheng, Hongzhi Wang, and Tianyu Mu. Dclp: Neural architecture predictor with curriculum contrastive learning. In *Proc. of AAAI*, 2024.

Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 287–294, 2007.

Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *Proc. of ICLR*, 2016.

APPENDIX

The appendix includes detailed information on the materials that are not covered in the main paper for the page limit and additional experimental results. The organization of this file is presented as follows:

- Appendix A - We elaborate on the details of different ranking loss functions and predictor-based NAS frameworks used in our experiments.
- Appendix B - We provide detailed experiment settings including search space and hyper-parameter configuration.
- Appendix C - We provide additional experimental results for the evaluation of the performance predictors, predictor-based NAS methods, computational costs and visualization results.

A DETAILED DESCRIPTIONS OF BASELINE METHODS

A.1 DETAILED DESCRIPTIONS OF RANKING LOSS FUNCTIONS

Now we give detailed descriptions of 11 different ranking loss functions that we used.

Mean Square Error (MSE). MSE loss is the most widely used loss function in existing performance predictors (Wen et al., 2020; Lu et al., 2021; Liu et al., 2022; Lu et al., 2023). They first predict absolute architecture performance precisely and then rank architectures according to the predicted values. We directly use the official implementation from the Pytorch package.

Bayesian Personalized Ranking (BPR). BPR loss (Rendle et al., 2009) is very popular in the field of recommendation system and was first combined with the performance predictor by GMAE-NAS (Jing et al., 2022). For each sample in a batch, BPR compares its predicted score with other samples which have higher true accuracy. BPR optimizes the relative ranking between architecture pairs, encouraging the predictor to capture the difference between high-rank and low-rank architectures for better ranking ability. We use the implementation from the open source of GMAE-NAS.

ListMLE. ListMLE loss (Xia et al., 2008) is a classic listwise ranking loss in information retrieval and was introduced into performance predictor by DCLP (Zheng et al., 2024). ListMLE loss aims to maximize the likelihood of the correct architecture ranking, making the predictor better estimate the ranking order more accurately. We use the implementation from the open source of DCLP.

Margin Rank (MR). MR loss was used in FlowerFormer (Hwang et al., 2024) to optimize the ranking of architecture pairs. It promotes correct ranking by comparing pairs of architecture scores and maintaining a minimum margin between the high-rank architecture and the low-rank one. We directly use the official implementation from the Pytorch package.

Hinge Rank (HR). HR loss can be categorized into pairwise ranking and was applied to performance predictor by ReNAS (Xu et al., 2021) to predict relativistic architecture performance. A hinge function is used to calculate the loss. The loss will be 0 only when the architecture pair is correctly ranked and the score difference is within the margin. We use the implementation from the open source of ReNAS.

Weighted Hinge Rank (WHR1 & WHR2). We design two versions of weighted hinge ranking loss functions based on Hinge Rank by assigning more weights to architectures with better performance. Given a pair of architecture scores s_i and s_j, the loss function is calculated by $\mathcal{L}_{\mathcal{HR}} \times w_i \times w_j$, where w_i and w_j are the weights for two architectures. For WHR1, the weight of each architecture X is set to $(N - rank(X) + 1)/N$, where $rank(X)$ denotes the true ranking of X. As for WHR2, the weight of each architecture X is formulated as $exp(s_X) - 1$. WHR2 assigns a higher weight to top-performing architectures compared to WHR1. We implement two WHR loss functions on the foundation of Hinge Rank.

ListNet. ListNet loss (Xia et al., 2008) is another classic listwise ranking loss that converts both the predicted scores and true scores into probability distributions with the Softmax function and calculates the cross-entropy between them to enhance the model's ranking ability. We develop the original code of ListNet to fit the task of architecture performance.

Weighted Approximate-Rank Pairwise (WARP). WARP loss (Weston et al., 2011) was originally designed for image annotation. To calculate the loss, pairs of items are randomly sampled until a wrongly ranked pair is found and the predicted scores of the two incorrect items. A weight will be assigned to the loss according to the times it takes to find a wrongly ranked pair. Based on this idea, we tailor the WARP loss for the performance predictor. For each architecture in the batch, we randomly choose others to construct pairs until a pair is ranked incorrectly. If the incorrect pair appears soon, we think the predictor is poor at ranking and will assign a large weight to this sample when calculating the loss. Instead, if we need more samples to find a wrong pair, the performance of the predictor is already good and the loss should be small. We borrow the core idea of the original WARP and implement a new version for architecture performance ranking.

Magnitude Preserving (MP). MP loss (Cortes et al., 2007) combines the idea of MSE loss and pairwise ranking loss and preserves the magnitude of ground truth. MP loss aims to make the difference between the predicted scores approach that between the true accuracy within each data pair for better ranking ability. So the correctly ranked pairs will still be penalized. We develop the original code of MP and apply it to predict architecture performance.

LambdaLoss. LambdaLoss (Wang et al., 2018) is a probabilistic framework for ranking metric optimization advanced from the classic LambdaRank (Burges et al., 2006), which contains different metric-driven loss functions for the model. To enhance the model's ability to identify top-performing architectures, we design the loss function based on the one that was originally used to optimize the model performance in Normalized Discounted Cumulative Gain (NDCG), a top-rank centered metric. We use the implementation from the AllRank framework (Pobrotyn et al., 2020).

A.2 Algorithms of Predictor-based NAS frameworks

Now we provide detailed algorithms of the two predictor-based NAS frameworks that we employed in Algorithm 1 and Algorithm 2.

Algorithm 1: Predictor-guided Random Search Framework

Initialize: Sample a few architectures randomly from search space S to construct an initialization space S_0, train them for ground-truth performance and add them to $history$.
while $|history| + |S_0| < N$ **do**
  Train predictor using architectures in $history$;
  Sample M candidate architectures at random from $S - history$;
  Utilize predictor to predict the performance of each candidate architecture ;
  Update $history$ by adding architectures with top-K predicted values ;
return Architecture A' with best ground-truth performance in $history$

Algorithm 2: Predictor-guided Evolutionary Search Framework

Initialize : Sample a few architectures randomly from search space S to construct an initialization space S_0, train them for ground-truth performance and add them to $history$ and population P.
while $|history| + |S_0| < N$ **do**
  Train the predictor using architectures in $history$;
  Select T well-performing architectures in P through Binary Tournament Selection ;
  Mutate the selected architectures and generate M candidate architectures ;
  Utilize predictor to predict the performance of each candidate architecture ;
  Update $history$ and P by adding architectures with top-K predicted values ;
  Remove the oldest architecture from P;
return Architecture A' with best ground-truth performance in $history$

B Detailed Experimental Settings

B.1 Search Spaces

We discuss the search spaces we used in the following part.

NAS-Bench-101. NAS-Bench-101 (Ying et al., 2019) is a cell-based search space composed of over 423k architectures. An operation is represented by a node in the cell. Each architecture cell includes at most seven nodes and nine edges. The search space provides the accuracy of architectures on the CIFAR-10 dataset.

NAS-Bench-201. NAS-Bench-201 (Dong & Yang, 2019) is also a cell-based search space which contains 15625 architectures. The operation type is represented by the edge on NAS-Bench-201. There exist four nodes and six edges in the architecture cell. NAS-Bench-201 provides the accuracy of architectures on the CIFAR-10, CIFAR-100 and ImageNet16-120 datasets.

TransNAS-Bench-101-Micro & **Macro.** TransNAS-Bench-101 (Duan et al., 2021) is composed of two parts: a micro (cell-based) search space containing 4096 architectures and a macro (skeleton-based) search space containing 3256 architectures. The constitution of the architecture cell in the micro search space is the same as NAS-Bench-201. As for the macro space, we use the same encoding in Arch-Graph (Huang et al., 2022). The search space provides architecture performance across diverse tasks including segmentation, regression, pixel-level prediction and self-supervised tasks. In our experiments, we test our methods on four tasks: Class-Object, Class-Scene, Jigsaw and Autoencoder.

NAS-Bench-NLP. NAS-Bench-NLP contains 14332 architectures and a cell-based search space. There are at most 24 nodes and 26 edges in each cell. Different from the other search spaces, this one aims to design well-preforming architectures for NLP tasks. NAS-Bench-NLP provides test results of the architectures which are trained on Penn Tree Bank dataset (Kombrink et al., 2011) for 50 epochs.

DARTS. DARTS (Liu et al., 2018b) is an open-domain cell-based search space without available architecture performance. It is the largest search space in our experiments which contains around 10^{18} architectures. Each architecture consists of a normal cell and a reduction cell, which contains seven nodes and eight edges in a single cell.

Search Space	Max Node / Edge Num.	Size	Is Cell-based?	URL
NAS-Bench-101	7 / 9	423k	✓	https://github.com/google-research/nasbench
NAS-Bench-201	4 / 6	15625	✓	https://github.com/D-X-Y/NAS-Bench-201
TransBench-101-Micro	4 / 6	4096	✓	https://github.com/yawen-d/TransNASBench
TransBench-101-Macro	8 / 7	3256	×	
NAS-Bench-NLP	24 / 26	14322	✓	https://github.com/fmsnew/nas-bench-nlp-release
DARTS	7 / 14	10^{18}	✓	https://github.com/quark0/darts

Table 5: Basic information of different search spaces used in our experiments.

The URLs of search spaces are reported in Table 5.

B.2 Hyperparameter Settings

Search Space	Query Budget (N)	Candidate Number (M)	Update Number (K)	Initial Size ($\|S_0\|$)
NAS-Bench-101	150	200	10	20
NAS-Bench-201	100	200	10	20
TransBench-101-Micro	50	100	5	20

Table 6: Hyperparameter settings for predictor-based NAS methods in our experiments.

For all the ranking and searching experiments, we combine different ranking loss functions with a simple GCN encoder to construct our predictor for a fair comparison. The predictor is composed of a four-layer GCN and a three-layer MLP. The hyperparameter setting of the predictor is the same across different search spaces and tasks. The batch size is set to one-tenth of the training number.

Figure 7: Weighted Kendall Tau of different ranking loss functions in 13 different tasks across four search spaces (scaled up by a factor of 100).

Figure 8: N@10 of different ranking loss functions in 13 different tasks across four search spaces.

For example, if the training number is 424 on NAS-Bench-101, the batch size should be 43. The predictor is trained with a learning rate of 1e-3, a weight decay of 1e-3 and a dropout of 0.15. The number of epochs is set to 300 for most ranking losses except for WARP and LambdaLoss. They use 30 epochs by default because a longer training epoch leads to very poor performance on all the rank-based metrics. The configuration for the search experiments is included in the Table 6. All the experiments are conducted on a single RTX 3090.

C Additional Experiments Results

C.1 Full results for the evaluation of performance predictors

In Figure 7, Figure 8 and Figure 9, we plot the weighted Kendall Tau, N@10 and Rel@10 of different ranking loss functions in 13 tasks. The training portion is set to the smallest in the respective range of data splits and the test portion is set to 100%.

In Figure 10 and Figure 11, we provide the complete comparison between 11 ranking loss functions in 12 tasks. In Figure 12, we plot the performance of ranking loss functions in Kendall Tau and Weighted Kendall Tau as the training portion grows. The test portion is fixed to 100%. For the page limit, the results on NAS-Bench-NLP are included in Figure 13.

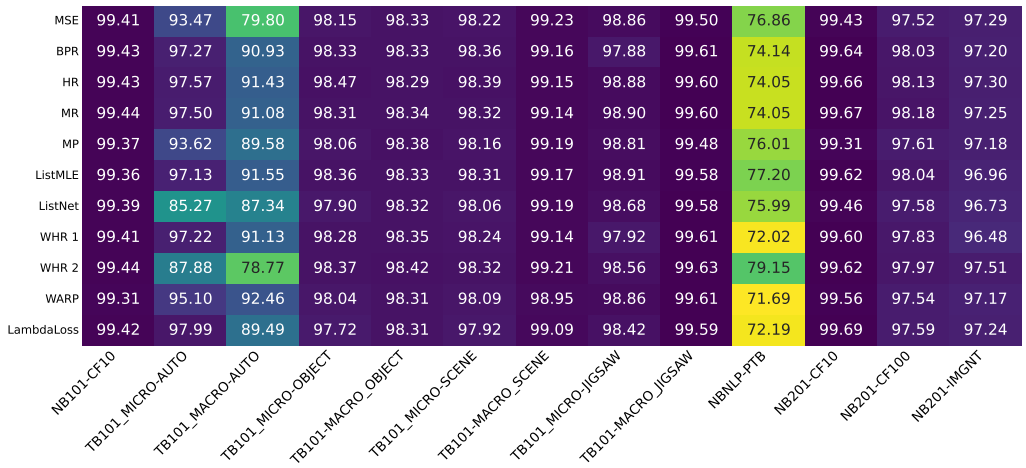


Figure 9: Rel@10 of different ranking loss functions in 13 different tasks across four search spaces (scaled up by a factor of 100).

C.2 Full results for the evaluation of predictor-based NAS methods

We plot the search results of predictor-based NAS frameworks using different ranking losses on NAS-Bench-201 CIFAR-10 and NAS-Bench-101 in Figure 14 and Figure 15.

C.3 Computational Cost

Loss Function	MSE	BPR	HR	MR	MP	ListMLE	ListNet	WHR 1	WHR 2	WARP	LambdaLoss
Training Time	55.21	110.17	54.64	55.65	56.67	58.02	56.11	55.86	55.71	24.98	6.27
Inference Time	39.24	38.94	37.58	37.91	37.97	38.70	38.91	37.79	38.90	38.91	38.54

Table 7: Training and inference time on NAS-Bench-101 with a training portion of 0.1% and a test portion of 100%.

Loss Function	MSE	BPR	ListMLE	WHR 2
Training Time	750.6	991.5	745.8	751.3
Inference Time	253.2	254.7	250.5	252.8

Table 8: Training and inference time on NAS-Bench-101 with a training portion of 0.1% and a test portion of 100% for PINAT.

Loss Function	MSE	BPR	ListMLE	WHR 2
Training Time	110.5	190.6	117.3	111.5
Inference Time	42.1	41.7	40.8	42.2

Table 9: Training and inference time on NAS-Bench-101 with a training portion of 0.1% and a test portion of 100% for NP.

We also calculate the computational cost of all the ranking loss functions on NAS-Bench-101 in Table 7. Among them, BPR is the most time-consuming option and takes two times longer than MSE when training. In contrast, WARP and LambdaLoss have very low training time because they require fewer epochs to train. In general, it is still very fast to finish training and inference because of the simple predictor.

We also calculate the computation cost for two representative predictors (Lu et al., 2023; Wen et al., 2020) in Table 8 and Table 9. The trend is similar to Table 7 but the time increases a lot for all losses. This indicates that the computational cost is more related to the composition of the predictor while the choice of loss function makes little difference.

C.4 Visualization of Search Results on DARTS

We visualize the architecture cells searched by different ranking losses in Figure 16.

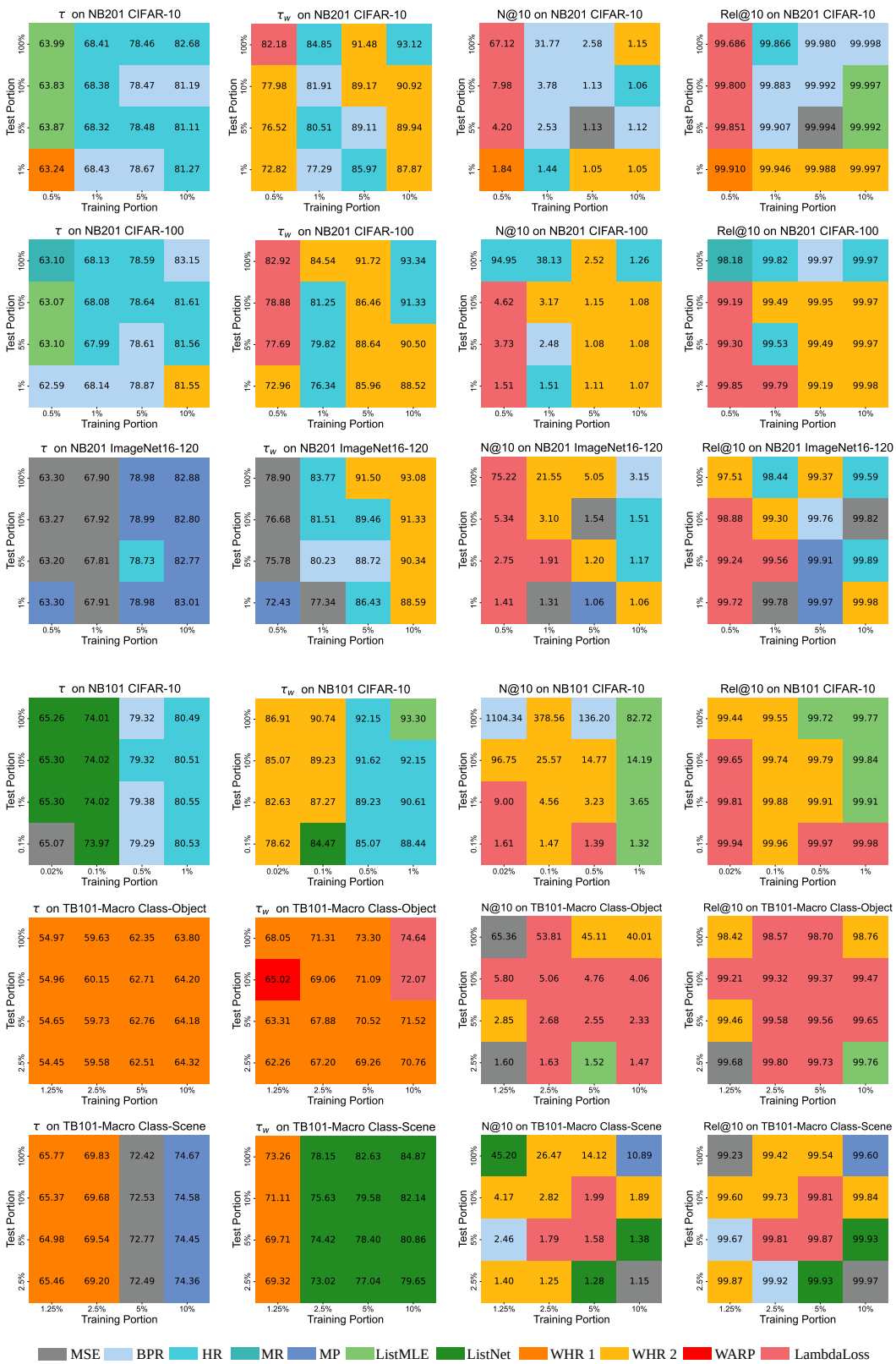


Figure 10: Kendall Tau of different ranking loss functions under various settings (τ, τ_w and Rel@10 are scaled up by a factor of 100). The results are averaged over 100 trials.

Figure 11: Results of different ranking loss functions under various settings (τ, τ_w and Rel@10 are scaled up by a factor of 100). The results are averaged over 100 trials.

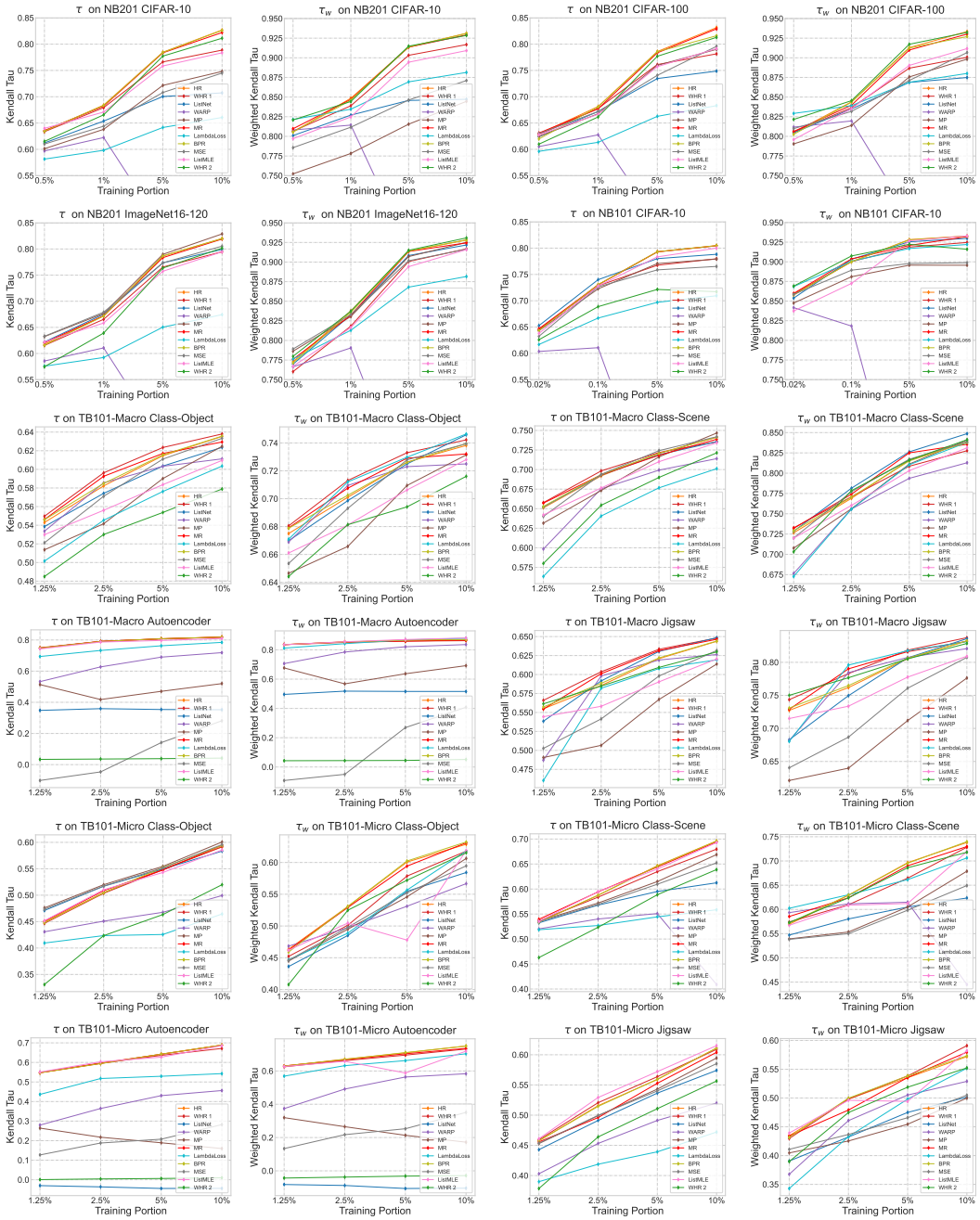


Figure 12: τ and τ_w of 11 ranking loss functions as the training portion grows. The test portion is fixed to 100%. The results are averaged over 100 trials.

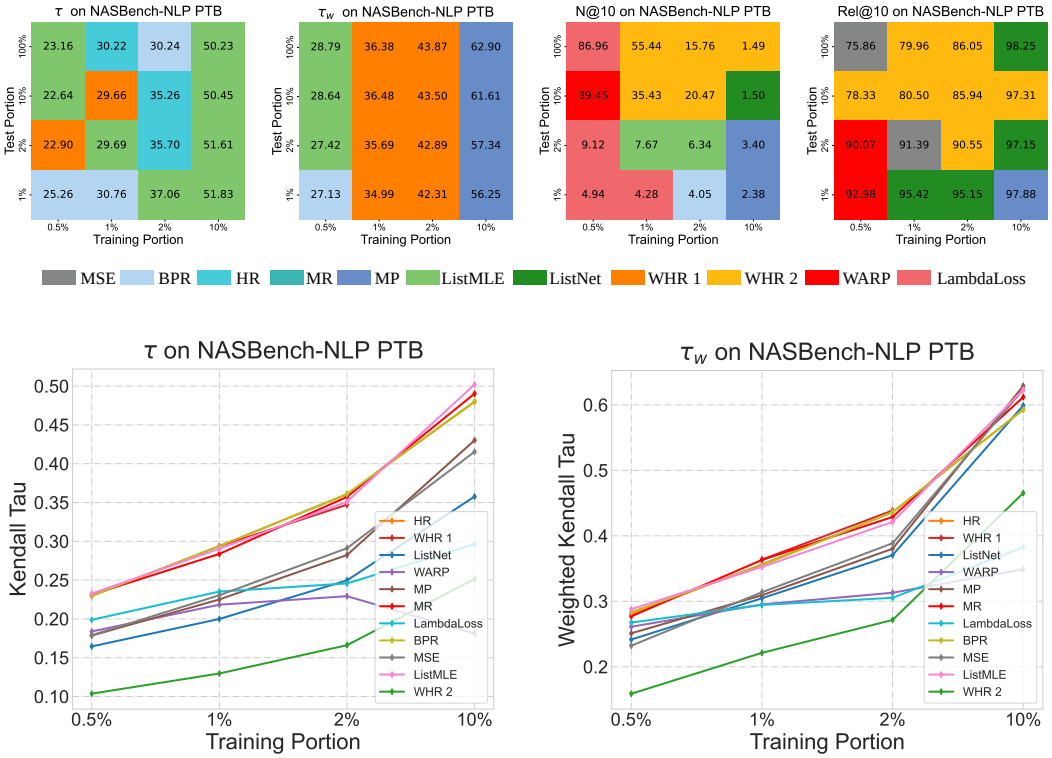


Figure 13: Results of different ranking loss functions on NAS-Bench-NLP. The results are averaged over 100 trials.

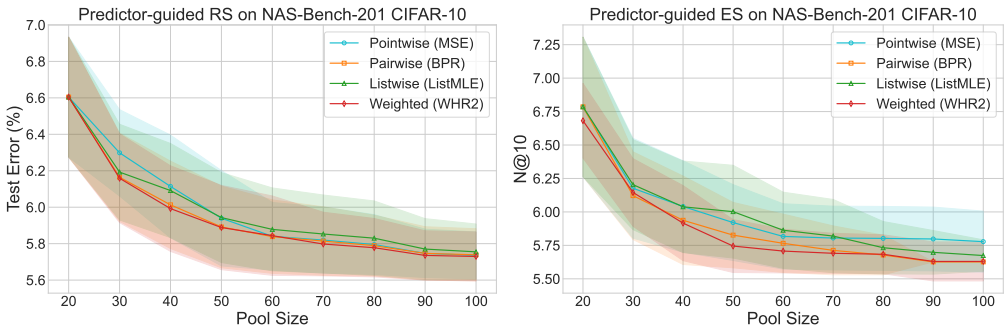


Figure 14: Test error vs. Pool size for the predictor-guided random search (RS) and the predictor-guided evolutionary search (ES) framework with different ranking losses on NAS-Bench-201 CIFAR-10.

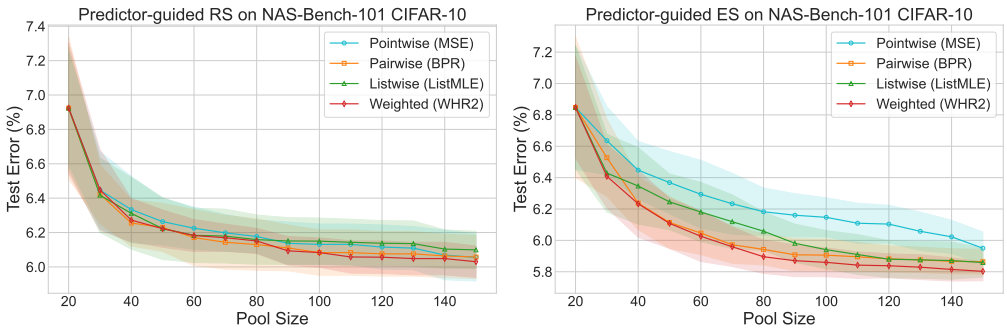
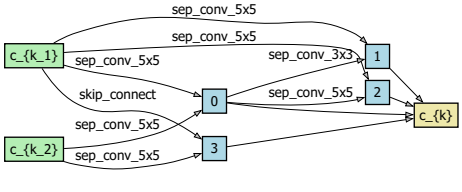
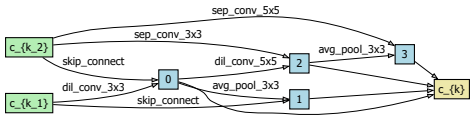


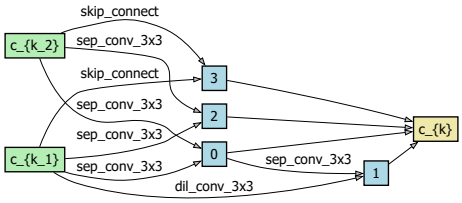
Figure 15: Test error vs. Pool size for the predictor-guided random search (RS) and the predictor-guided evolutionary search (ES) framework with different ranking losses on NAS-Bench-101.



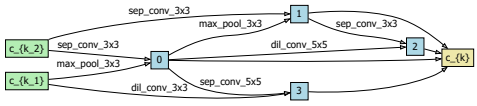
(a)Normal cell found by MSE loss.



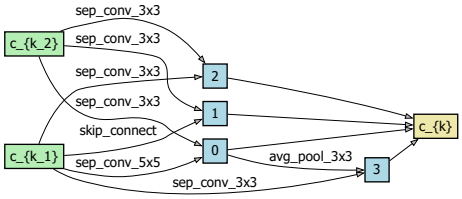
(b)Reduction cell found by MSE loss.



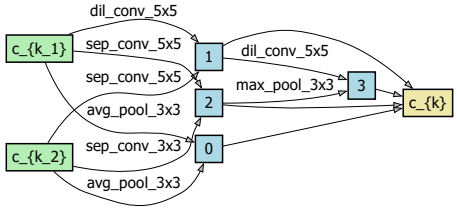
(c)Normal cell found by BPR loss.



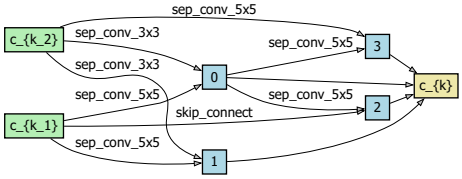
(d)Reduction cell found by BPR loss.



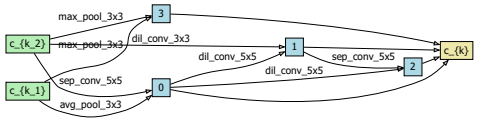
(e)Normal cell found by ListMLE loss.



(f)Reduction cell found by ListMLE loss.



(g)Normal cell found by WHR 2 loss.



(h)Reduction cell found by WHR 2 loss.

Figure 16: Cell architectures found by different ranking losses on CIFAR-10 dataset.