2021 International Conference on Data Mining Workshops (ICDMW) | 978-1-6654-2427-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICDMW53433.2021.00077

Personalized Neural Architecture Search

Cedric Kulbach
FZI Research Center for Information Technology
Haid-und-Neu-Str. 10-14
76131 Karlsruhe, Germany
kulbach@fzi.de

Steffen Thoma
FZI Research Center for Information Technology
Haid-und-Neu-Str. 10-14
76131 Karlsruhe, Germany
thoma@fzi.de

***Abstract*—Existing approaches for Neural Architecture Search (NAS) aim at efficiently maximizing individual or sets of objectives (e.g. high accuracy or a low number of parameters) by exploiting Reinforcement Learning (RL), evolutionary algorithms, or Bayesian optimization. Most multi-objective NAS algorithms assume that all objectives are fully known and require them to be broadly explored to successfully approximate the Pareto front, which results in computational expensive search algorithms. To address this problem, we propose an interactive machine learning approach based on preference elicitation which enables end-users to explore and find a custom loss function and can be directly used for State-of-the-Art single-objective black-box optimization. We integrate our approach into State-of-the-Art single objective NAS algorithms and evaluate it against multi-objective approaches on the *NATS-Bench* benchmark dataset. Furthermore, we show that diverse end-user preferences can be successfully approximated in terms of loss functions, leading to suitable neural architectures.**

***Index Terms*—Neural Architecture Search, Personalization, User Centric AI, Ranking**

I. Introduction

Existing approaches for Neural Architecture Search (NAS) aim at efficiently maximizing individual or sets of objectives (e.g. high accuracy or low number of parameters) by exploiting Reinforcement Learning (RL) [1], [2], evolutionary algorithms [3]–[5] or Bayesian optimization [6], [7]. The resulting algorithms become increasingly efficient in terms of required computational resources for the search procedure and achieve exceptional predictive performance for a variety of architecture- and problem types. Especially for real-world applications, multi-objective NAS is a central tool for balancing available constraints, such as sufficiently high predictive accuracy, low energy consumption, or fast execution. The search goal in multi-objective NAS is therefore to approximate the Pareto frontier, which expresses the degree to which available objectives can be jointly met. The complexity of this approximation becomes increasingly difficult with more objectives, leading to the consideration of only a few objectives so far.

However, real end-user objectives are diverse, reaching beyond the global accuracy of a classifier or the execution time. For example, the number of floating point operations (FLOPS) or the number of trainable parameters could also be considered as a measurement for the complexity and therefore a measurement not only for the latency but also for the used memory.

To enable a user centric recommendation of suitable neural networks, we propose a human-in-the-loop approach for multi-objective NAS. Our interactive NAS setting incorporates techniques from preference elicitation, where end-users are supported to explore and find personalized objectives for NAS. Our system samples a diverse set of architectures and lets the end-user sequentially choose among pairs of candidates, which are then used to approximate a personalized ranking and thus a user driven metric. The learned ranking functions are suitable objective functions for single-objective NAS algorithms, leading to scalable and personalized multi-objective NAS approaches. The resulting neural architectures can maximize their satisfaction without approximating the full Pareto frontier, as the latter is gradually focused on user interaction. We present a resource-aware evaluation based on the *NATS-Bench* benchmark dataset for Cifar-10, Cifar-100, and ImageNet16-120 with the possibility to pursue different preferences such as (i) training accuracy, (ii) testing accuracy, (iii) FLOPS, (iv) trainable params, and (v) latency based features. Our results show that we can efficiently and successfully learn realistic preferences, leading to appropriate neural architecture recommendations for end-users.

In summary, we provide the following contributions:

1 An interactive NAS approach that enables learning user-specific metrics.
2 A formalization and implementation[1] of the proposed approach.
3 A resource-aware evaluation and comparison of recent multi-objective NAS approaches based on the *NATS-Bench* [8] dataset.
4 An OpenAI gym environment[2] to enable a broader application of NAS Algorithms to RL agents as well as the implementation of State-of-the-Art baselines for the *NATS-Bench* dataset.

II. Related Work

Our approach to a *Personalized Neural Architecture Search* approach compromises related work in Neural Architecture

This research was carried out with the support of the German Federal Ministry for Economic Affairs and Energy within project SeRoNet - Service Robotics Network

[1]https://github.com/kulbachcedric/PersonalizedNAS
[2]https://github.com/kulbachcedric/Natsbench-Gym

978-1-6654-2427-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDMW53433.2021.00077

Search and preference elicitation approaches. Neural Networks have shown great success in a broad range of applications, such as image classification tasks. Approaches for NAS can be classified on the one hand based on the underlying solution method but also based on the selection of the objective function. In Sec. II-A we provide a formalization for single objective neural architecture search and in Sec. II-C the background for ranking based preference learning approaches which enables interactive machine learning systems [9].

A. Single-Objective NAS

To automate the search for suitable structures, the concept of Neural Architecture Search (NAS) has been proposed by Pham et. al. [10] and shown great success on various datasets. Current research in NAS e.g. [11]–[14] have shown that NAS frameworks are able to outperform State-of-the-Art models and to set new benchmarks on established datasets. In reference to [15] the single objective NAS approach is formalized in Def. 1, in which neural networks process a dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Bauer et.al. [16] formalize neural networks as directed acyclic graphs (DAG) which transforms input variables X to output variables $\hat{y}$ with a set of nodes $z \in \mathcal{Z}$. Both ideas are used to formalize in Def. 1 NAS based on a single objective $\mathcal{L}$.

***Definition** 1:* NAS Problem, adopted from [15]:
Let D_{train} be a train and D_{valid} be a validation dataset, which is split into K cross validation folds $\{D_{train}^{(1)}, ..., D_{train}^{(K)}\}$ and $\{D_{valid}^{(1)}, ..., D_{valid}^{(K)}\}$. Furthermore, let $f_{g,\overrightarrow{z}} \in \mathcal{F}$ be a neural network, configured by a graph structure $g \in G$ and a set of nodes $z \in \mathcal{Z}$. Let $\mathcal{L}(f_{g,\overrightarrow{z}}(D_{\text{train}}^{(i)}), D_{\text{valid}}^{(i)})$ denote the loss function a neural network $f_{g,\overrightarrow{z}}$ achieves on $D_{valid}^{(i)}$ when trained on $D_{train}^{(i)}$. Then the neural architecture search problem is to find the neural network $f_{g^*,\overrightarrow{z}^*}$ that minimizes the loss:

$$g^*, \overrightarrow{z}^* \in \underset{g \in G, \overrightarrow{z} \in \mathcal{Z}}{\arg\min} \frac{1}{K} \sum_{i=1}^{K} \mathcal{L}(f_{g,\overrightarrow{z}}(D_{\text{train}}^{(i)}), D_{\text{valid}}^{(i)}) \quad (1)$$

Excessive research work has been applied to reinforcement learning (RL) and evolution based approaches [17] with heterogeneous search spaces. In RL MetaQNN [14], a q-learning approach, an agent interacts with the environment by choosing CNN architectures. Other RL approaches such as [1], [10], [18] use Policy Gradient algorithms. Evolution based approaches initialize a population of neural models and propagate through mutation [19], [20] or graph morphism [15]. However, the majority of early NAS approaches are single objective NAS approaches that optimize neural architectures regarding a single metric. With the increasing popularity of NAS and its application in a wide range of settings, a multi-criteria approach becomes necessary.

B. Multi-Objective NAS

Our approach to a *Personalized Neural Architecture Search* framework compromises research in multi-objective NAS and preference-based Machine Learning (ML). Whereby preference-based machine learning approaches are mainly based on preference learning [9] or ranking approaches introduced in Sec. II-C. Research in single-objective NAS approaches have shown, that optimizing only one metric can lead to high predictive performances, but also to inefficient network architectures in terms of trainable parameters, execution time, or latency and require a multi-objective perspective. The field of multi-criteria NAS comprises heterogeneous goals, such as enabling end-users to request novel quantities to be computed by the system. Goals of multi-criteria NAS are e.g. to allow the user to directly interact with neural structures and the key indicators for neural architectures. Recent work in multi-criteria NAS mainly defines a new objective from a variety of performance indicators [2] or calculate a Pareto frontier to meet the task of several objectives. According to [2], a multi-objective NAS problem is (i) multi-objective, (ii) Adaptive, (iii) Generalizable, (iv) Effective and (v) Scaleable. Pareto-Nash [5], NSGA-Net [21] and Nemo [4] use generic algorithms and MONAS [2] use a deep reinforcement learner to search for neural architectures with a pre-defined multi-objective metric (reward function). An approach that searches the Pareto frontier within a Lamarckian evolutionary algorithm is LEMONADE [22]. However, the question for the effectiveness and thus the generalisability to other search problems remains open, since in previous works mostly the performance on multi-objective metrics is measured, but not e.g. the number of trained neural networks it took to get the best performing neural networks. Our work compares (i) an effective approach to the number of trained neural networks. In terms of effectiveness and generalizability, we refer to the search space of neural architectures within the *NATS-Bench* [8] dataset and are thus able to evaluate the effectiveness of the search algorithm against existing work. Furthermore, we show the adaptivity as well as the scalability of *Personalized Neural Architecture Search* by evaluating it on different State-of-the-Art datasets such as CIFAR-10, CIFAR-100, ImageNet16-120 from the *NATS-Bench* dataset and on different metrics a user could want to pursue.

C. Preference learning

Learning from (human-) preferences has already been approached by reinforcement learning [23], in which reward functions are learning to rank models. Learning to rank approaches is an already well-researched topic which can be categorized into pointwise [24], [25], pairwise [26]–[29] and listwise [30], [31] approaches. The goal of these approaches is to learn based on features (documents or observations) a ranking model that is able to order a list of features. Pointwise (PRank [24] and McRank [25] approaches assume that one can rate documents or observations with absolute ratings, which can be reduced to a regression problem. Listwise approaches (Combined Regression and Ranking [29], PolyRank [28] and ES Rank [31]) are approaches where a user provides a relative, listwise feedback to learn the ranking model. Pairwise ranking approaches (Ranking SVM [26], RankNet [27], LambdaRank [32] and LambdaMART [33]) take ranked document pairs as input to classify each document pair into correctly or

incorrectly ranked classes. To provide an interactive approach to NAS, we base our work on a pairwise ranking approach, more precisely on the well-researched RankNet approach. The RankNet approach consists of two siamese base networks which are connected over a meta-network to learn an underlying ranking function (see Sec. II-C) Details for our approach are depicted in Sec. III-D in more detail.

III. Personalized Neural Architecture Search

In this section, we present our approach to a *Personalized Neural Architecture Search* framework. In Def. 2 we first formalize our approach based on Def. 1. Each component of our approach to the personalized NAS problem is then presented in more detail. Besides the personalization of single objective NAS frameworks (see Fig. 1), we integrate the personalization process (Alg. 1) into an evolutionary approach, which has the major advantage, that neural architectures from an initial population can already be used to learn a suitable loss function. We now formally define the Personalized NAS problem:

***Definition** 2:* Personalized Neural Architecture Search Problem

We extend the Neural Architecture Search (NAS) problem (see Def 1) where a loss function $\mathcal{L}$ is assumed to be given with a set of available loss functions $\mathbf{L} = \{\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(l)}\}$. The goal becomes to learn a novel loss function $\mathcal{L}^*$ which combines the individual loss functions:

$$\mathcal{L}^* : \mathcal{L}^{(1)} \times \cdots \times \mathcal{L}^{(l)} \rightarrow \mathbb{R}$$

By assuming that one can model a function $\phi : \mathbf{L} \rightarrow \mathbb{R}^l$ which generates a feature vector for the available set of metrics, we can reduce learning $\mathcal{L}$ to a regression problem, where we attempt to learn

$$h_{\mathcal{L}^*} : \phi(\mathbf{L}) \rightarrow \mathbb{R}$$

The resulting Personalized Neural Architecture Search Problem can then be defined as follows.

$$g^*, \overrightarrow{z}^*, \mathcal{L}^*, \in \underset{g \in G, \overrightarrow{z} \in \mathcal{Z}}{\arg\min} \frac{1}{K} \sum_{i=1}^{K} \hat{h}_{\mathcal{L}^*}(f_{g,\overrightarrow{z}}(D^{(i)}_{\text{train}}), D^{(i)}_{\text{valid}}) \quad (2)$$

where $\hat{h}_{\mathcal{L}^*}$ is the approximated regressor for the novel metric function $\mathcal{L}^*$ for which we we have to learn weights $\theta \in \mathbb{R}^l$, e.g. $\hat{h}_{\mathcal{L}^*} = \phi(\mathbf{L})^T\theta$ for the linear case.

To formalize personalized NAS (see Def. 2), we extend Def. 1 by multiple loss functions $\mathbf{L}$ an end-user might want to pursue. The goal is to learn a novel loss function $h_{\mathcal{L}^*}$ which combines individual loss functions based on an end-user preference. Our proposed framework (see Fig. 1) consists of an (i) *Architecture Sampler*, an (ii) *Evaluation Generator*, an (iii) *Human Preference* and a (iv) *Preference Executor* component, which can be paired with any NAS algorithm. An *Architecture Sampler* generates a set P of valid architectures $f_{g,\overrightarrow{z}} \in \mathcal{F}$ which are trained on a train dataset D_{train}. Note, that this component can also be part of a NAS algorithm e.g. in evolutionary algorithms (see Alg. 1), as it generates the initial population. Architectures can be generated by stacking multiple cells and modules (see [19]) or by graph morphism (see [15]). In Fig. 1 the *NAS* component (i) evaluates and (ii) generates features for the *Evaluation Generator* component and (iii) searches with a metric $\mathcal{L}^*$ efficiently for an optimal architecture $f^*_{g,\overrightarrow{z}}$. An *Evaluation Generator* component generates segment pairs U^{pair} from different architectures $f_{g,\overrightarrow{z}} \in P$ sampled by the *Architecture Sampler* component and evaluated by the *NAS* component. These segment pairs can be evaluated from humans within a *Human Preference* component, which is the interface that allows to indicate the users preferences. A *Preference Executor* component takes the ranked segment pairs U^{judged} (also referred to as judgment) as input and learns a new loss function $\mathcal{L}^*$ with given preferences. The new loss function $\mathcal{L}^*$ is used to rank samples from the *Architecture Sampler* component and to evaluate architectures within the NAS algorithm. The samples that perform best according to the *Preference Executor* component are trained in more extensive runs on D_{train} and used to predict D_{test}.

Evolutionary NAS algorithms have the advantage, that the ranking of each individual is not needed during the initial population generation step, so that the initial population can already be used to learn a new metric $\mathcal{L}^*$. The generated set P can be used to (i) train the ranking algorithm (*Preference Executor*) component and as (ii) initial population for the evolutionary NAS algorithm. Note, that approaches such as MONAS [2] can be used as NAS algorithm within our Personalized NAS approach. Besides the overall architecture shown in Fig. 1, we depict the overall workflow of our system exemplary based on a regularized evolutionary approach [3], [20] in Alg. 1 which is evaluated in Sec. IV.

In the following, we describe the main components depicted in Fig. 1 in more detail.

A. NAS Layer

The *NAS Layer* consists of a NAS algorithm and an *Architecture Sampler* component. The *Architecture Sampler* is part of the NAS Layer and generates from a search space $\mathcal{F}$ new architectures for the NAS algorithm. The goal of the *NAS* algorithm is to find the best architecture $f^*_{g,\overrightarrow{z}} \in \mathcal{F}$ and to enable the generation of features such as the accuracy, trainable parameters, training time. To illustrate the interface of the personalization approach more precisely, the *Architecture Sampler* is visualized in Fig. 1 as a separate component within the NAS Layer.

B. Evaluation Generator

The *Evaluation Generator* component takes the neural architecture's evaluation metrics from the *NAS Layer* and a maximum number of segment pairs (γ) to be generated as input. It generates a set of segments U^{pair}, which allows (i) the user to visualize the performance of neural architectures and also (ii) provide the features ($\mathbb{L}$) for the ranking algorithm. We denote a segment s_i as a tuple $(X_{valid}, y_{valid}, f^{(i)})$. Furthermore, we assume, that all metrics $\mathcal{L} \in \mathbb{L}$ for the user's visualization and the ranking algorithm can be generated by a tuple s_i. Based on the generated segments, we randomly sample γ segment pairs

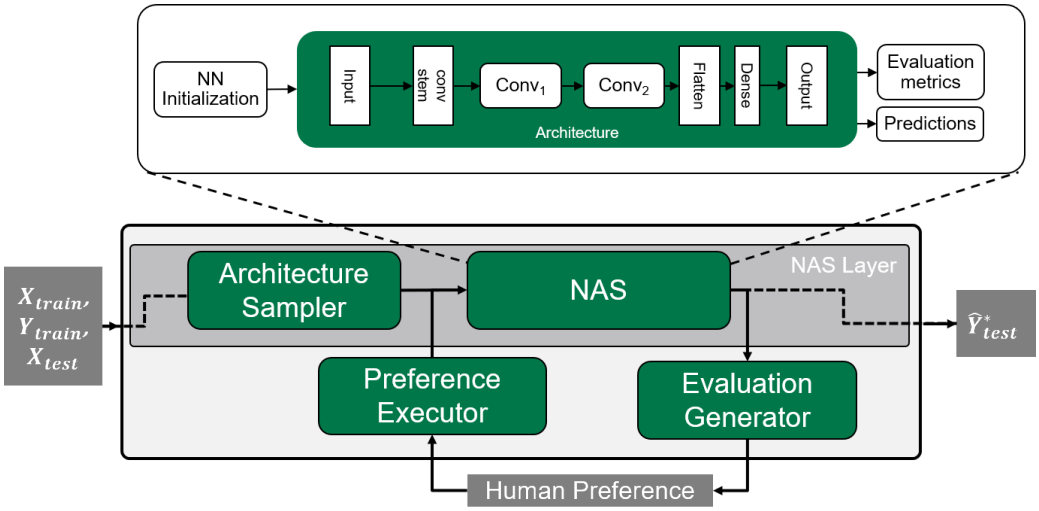

Fig. 1. Personalized AutoML framework

Algorithm 1 Evolutionary based Personalized NAS based on [3], [20]

Input:
Dataset $D_{\text{train}} = \{X_{\text{train}}, y_{\text{train}}\}$,
Dataset $D_{\text{valid}} = \{X_{\text{valid}}, y_{\text{valid}}\}$,
Features X_{test},
Number of pairwise comparisons ω,
PopulationSize p,
SampleSize s,
Cycles c
Output: Prediction $\hat{y}$
Ensure: $p \geq s \;\;\&\;\; c \geq p$
$P, H \leftarrow \emptyset$
for $i = 0$ to p **do** ▷ Generate Population P
  $f_{g,\overrightarrow{z}} \leftarrow RandomArch(G, \mathcal{Z})$
  $f_{g,\overrightarrow{z}}.fit(X_{\text{train}}, y_{\text{train}})$ ▷ train arch
  $P \leftarrow P \cup f_{g,\overrightarrow{z}}$
  $H \leftarrow H \cup f_{g,\overrightarrow{z}}$
end for
$U^{\text{pair}} \leftarrow \text{SegmentGenerator}(P, D_{\text{valid}}, \omega)$ ▷ see Alg. 2
$U^{\text{judged}} \leftarrow \text{HumanPreference}(U^{\text{pair}})$ ▷ see Sec. III-C
$X^{(1)}, X^{(-1)} \leftarrow SampleComparisons(U^{\text{judged}})$
$\mathcal{L}^* \leftarrow \text{RankNet}.fit(X^{(1)}, X^{(-1)})$ ▷ see Alg. 3
while iter $\leq c$ **do** ▷ Start NAS iterations
  $S \leftarrow$ sample s architectures $f^{(i)} \subset P$
  $f^{(\text{parent})} \leftarrow \arg\max_{f_{g,\overrightarrow{z}} \in S} \mathcal{L}^*(f_{g,\overrightarrow{z}}(X_{valid}), y_{valid})$
  $f^{(\text{child})}_{g,\overrightarrow{z}} \leftarrow f^{\text{parent}}_{g,\overrightarrow{z}}.\text{mutate()}$
  Train $f_{\text{child}}(X_{\text{train}}, y_{\text{train}})$
  $P \leftarrow P \cup f^{(\text{child})}_{g,\overrightarrow{z}}$
  $H \leftarrow H \cup f^{\text{child}}_{g,\overrightarrow{z}}$
  $P \leftarrow P \setminus P.\text{oldest}$
end while
$f^*_{g,\overrightarrow{z}} \leftarrow \arg\min_{f_{g,\overrightarrow{z}} \in H} \mathcal{L}^*(f_{g,\overrightarrow{z}}(X_{valid}), y_{valid})$
Return $f^*_{g,\overrightarrow{z}}$

$(s_i, s_j) \in U^{pair}$. Within the *Evaluation Generator* component the question which segments s to compare for ranking also takes place. In an initial implementation, the segment pairs are randomly sampled (see Alg. 2).

Algorithm 2 Segment Generation

Input:
Set of neural architectures P,
Number of segment pairs to be generated ω,
Validation dataset $D_{\text{valid}} = \{X_{\text{valid}}, y_{\text{valid}}\}$,
Output:
Set of segment pairs U
while $|U^{\text{pair}}| \leq \omega$ **do**
  $i, j \leftarrow$ Select random $i, j \in P$
  $s_i \leftarrow (X_{\text{valid}}, y_{\text{valid}}, f^{(i)})$
  $s_j \leftarrow (X_{\text{valid}}, y_{\text{valid}}, f^{(j)})$
  if $(s_i, s_j) \notin U^{\text{pair}}$ **then**
    $U^{\text{pair}} = U^{\text{pair}} \cup \{(s_i, s_j)\}$
  end if
end while
Return U^{pair}

C. Human Preference Interface

The *Human Preference* component is the interface between the *Evaluation Generator* and the *Preference Executor* component. It takes the sampled segment pairs U^{pair} from the *Evaluation Generator* component and the preferences c_k for each segment pair $(s_i, s_j)_k \in U^{pair}$ as input. Based on the visualization of each segment, the end-user can specify his or her preference. While each segment $s^i \in U^{\text{judged}}$ is related to a neural architecture f_i, we can visualize the architecture itself, its features, but also predictions on D_{valid}. The output from the *Human Preference* component are judgments $(s_i, s_j, c) \in U^{judged}$. With $s_i = \{X_{valid}, y_{valid}, f^{(i)}\}$, we are

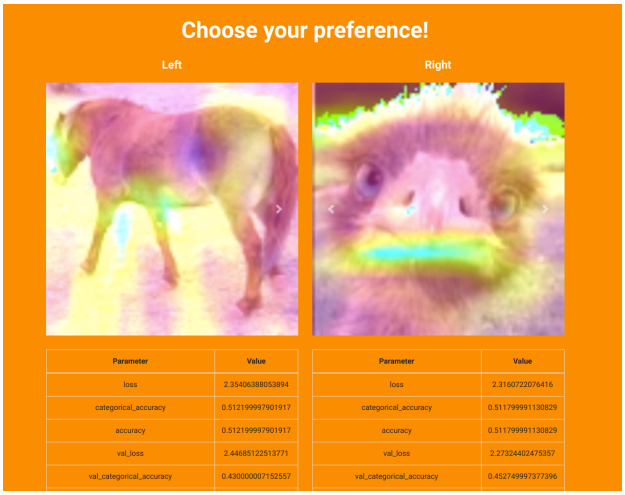

Fig. 2. Human Preference Interface visualizing different preformance metrics, as well as activations

able to compute $\hat{y}_{valid} = f^{(i)}(X_{valid})$ and thus performance metrics such as the accuracy. Besides the computation of performance metrics, such as the accuracy, we are able to compute a latency metrics or architecture specific futures, that enables a user centric comparison of neural architectures. In Fig. 2 we present a prototypical implementation for the *Human Preference* component and how a user could state his or her preference. Since the architecture is part of the segment s_i, one can also visualize different activations (see Fig. 2) of $f^{(i)}(X_{valid})$ so that a user could prefer architectures that pay attention to the areas that a user could want to pursue to classify an image. In Fig. 2, we used grad cam [34] to visualize the important regions for a neural network. Based on the visualization of the performance metrics, as well as the activation functions, the user can state his or her preference within pairwise comparisons.

D. Preference Executor

The *Preference Executor* component is the core component to personalize NAS. From the *Human Preference* component, it takes a set of judgments U^{judged} as input to generate a new, personalized loss function $\mathcal{L}^*$. While each segment is related to a neural architecture or a *NATS-Bench* [8] entry, we can compute all metrics/losses $\mathcal{L} \in \mathbf{L}$. Within the *Preference Executor* component a pairwise ranking model is trained with the segments' metrics and the judgments from the user. We implemented the RankNet [27] algorithm to learn the underlying pairwise ranking function (see Alg. 3). The RankNet approach takes two training datasets $X^{(1)}_{\text{train}}$ for the selected segments and $X^{(-1)}_{\text{train}}$ for the non-selected segments from all judgments U^{judged}. Both datasets have the same structure. We replaced the base network from the RankNet (see [27]) approach with a Multilayer Perceptron Network, since our experiments initially have a small number of features and learning convolutional features did not lead to any improvement in the results. The output from the base network is used as a scoring function to rank the features X. To train the pairwise ranking approach, the meta network has a siamese architecture and predicts within two base networks the scores on $X^{(1)}_{\text{train}}$ and $X^{(-1)}_{\text{train}}$. The difference between both scores is parsed to a sigmoid function which is used to distinguish if the predicted scores provide a correct ranking within $X^{(1)}_{\text{train}}$ and $X^{(-1)}_{\text{train}}$. In Alg. 3, we present the algorithm for a pairwise *Preference Executor*. The obtained loss function $\mathcal{L}^*$ from Alg. 3 is used as new loss

Algorithm 3 Rank based Scoring function $\mathcal{L}^*$

Input:
Ground truth y, and prediction $\hat{y}$
Trained RankNet model $\text{NN}_{\text{RankNet}}$
Set of metric functions $L = \{\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(l)}\}$

Output:
Metric $\mathcal{L}^*(x)$

Initialize $X \leftarrow \emptyset$
Initialize $\mathcal{L}^* \leftarrow \text{NN}_{\text{RankNet}}$
for $\mathcal{L}^{(i)} \in \mathbf{L}$ **do**
    $x \leftarrow x \cup \mathcal{L}^{(i)}(\hat{y}, y)$
end for
Return $\mathcal{L}^*(x)$ \\prediction

function within the NAS Layer and to rank new samples from the *Architecture Sampler*.

IV. Empirical Evaluation

In this section we evaluate our proposed approach based on the *NATS-Bench* [8] dataset. Regarding the evaluation of interactive machine learning systems proposed in [9], we provide an algorithm-centered evaluation. First, we present in line with MONAS [2] the metrics that the search algorithm should pursue. We show in Sec. IV-B that the pairwise ranking model (*Preference Executor* component) is able to pursuit different user preferences. Furthermore, in Sec. IV-C we evaluate our proposed framework from Sec. III and show that the *Preference Executor* component is able to steer NAS algorithms.

A. Metrics

To evaluate the pairwise ranking approach, as well as the performance of multi-objective NAS approaches, we evaluate our approach in accordance to the metrics defined in [2]. These can be divided into selective (see Sec. IV-A1) and combined metrics (see Sec. IV-A2). Note, that each metric $\mathcal{L}^{(i)} \in \mathbb{L}$ is normalized so that lower return values are better than higher return values.

1) Selective Metrics: In a selective scenario a user pursues a specific metric $\mathcal{L}^{(i)} \in \mathbb{L}$. The learning problem for the *Preference Executor* is to select from the set of metrics $\mathbb{L}$ the one that is pursued. The metric $h_{accuracy}$ (Eq. 3) pursues

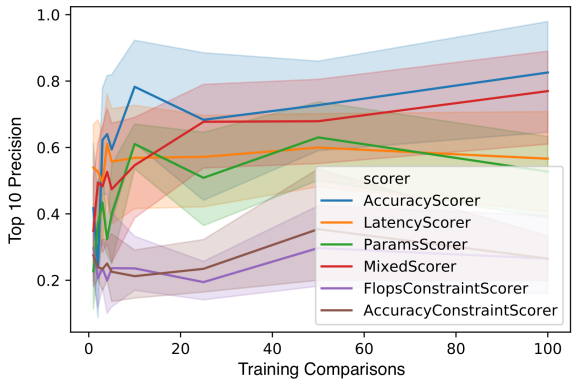

Fig. 3. RankNet - Top - 10 precision on ranking architectures (Cifar-100) with different metrics to pursue number of pairwise comparisons.

the test accuracy, $h_{latency}$ (Eq. 4) pursues the networks latency and h_{params} (Eq. 5) follows the number of trained parameters.

$$h_{accuracy}(\mathbf{L}) = \mathcal{L}^{(accuracy\ test)} \tag{3}$$

$$h_{latency}(\mathbf{L}) = \mathcal{L}^{(latency)} \tag{4}$$

$$h_{params}(\mathbf{L}) = \mathcal{L}^{(params)} \tag{5}$$

2) Combined Metrics: In a combined scenario a user pursues a combination of metrics. In Eq. 6, $h_{weighted}$ represents a weighted sum of all $\mathcal{L}^{(i)} \in \mathbb{L}$. The equation Eq. 7 and Eq. 8 represent constraint metrics, in which the return is zero if the number of FLOPS or the accuracy exceeds a threshold. For the evaluation, we set the threshold in Eq. 7 and Eq. 8 to the median value of $\mathcal{L}^{(flops)}$ and $\mathcal{L}^{(accuracy)}$ for all neural networks within the *NATS-Bench* dataset. Furthermore, we normalized all metrics $\mathcal{L} \in \mathbb{L}$ to $[0, 1]$. The weights in Eq. 6 are equally distributed within the evaluation.

$$h_{weighted}(\mathbf{L}) = \sum_{\mathcal{L}^{(i)} \in \mathbf{L}} \alpha^{(i)} * \mathcal{L}^{(i)} \tag{6}$$

$$h_{\textit{flop constraint}}(\mathbf{L}) = \begin{cases} \mathcal{L}^{(accuracy)}, & \textit{if } \mathcal{L}^{(flop)} > threshold \\ 0, & \textit{else} \end{cases} \tag{7}$$

$$h_{\textit{accuracy constraint}}(\mathbf{L}) = \begin{cases} \mathcal{L}^{(flop)}, & \textit{if } \mathcal{L}^{(accuracy)} > threshold \\ 0, & \textit{else} \end{cases} \tag{8}$$

B. Preference Executor Evaluation

In order to evaluate the *Preference Executor* component we randomly generated 100 architectures $f_{g,\vec{z}} \in P$ for training and 100 architectures for testing. Based on the architectures in P and the synthetic preferences $\mathcal{L} \in \mathbb{L}$, we generated 1, 2, 3, 4, 5, 10, 25, 50 and 100 segment pairs U^{pair}_{train} for training the RankNet approach. To show that the *Preference Executor* component can use the user preferences in the context of NAS, we evaluated the *Preference Executor* on the Cifar-10, Cifar-100, and Imagenet16-120 datasets. The generated test architectures are synthetically evaluated and ranked by the pre-defined metrics Eq. 3 - 8 in Sec. IV-A. To evaluate the performance of the RankNet approach, we applied the top-10 precision metric on the test architectures and measured whether the RankNet approach manages to rank the top 10 out of the 100 generated test architectures. Fig. 3 shows the top 10 precision for different training sizes (comparisons) for the Cifar-100 dataset. For the Cifar-10 and the Imagenet16-120 dataset, the results are presented in Tab. I. Fig. 3 and Tab. I show that about 20 pairwise comparisons are needed to learn a suitable preference, whereby nonlinear constraint metrics $h_{\textit{accuracy constraint}}$ and $h_{\textit{flop constraint}}$ are more difficult to approximate and perform worse than other metrics. In Tab. I, we trained the *Preference Executor* component on various subsets of U^{judged}_{train} and evaluated it on U^{judged}_{test}. In the experimental setup, a top-10 precision of 0.2 means that the ranking algorithm can recommend from the 100 generated test architectures 2 correctly within the top 10 architectures (20%). With the conducted experiments we qualitatively show that the *Preference Executor* needs ~ 10 pairwise comparisons (see Fig. 3) to learn a suitable metric. Whereby the constraint metrics (Eq. 7 and Eq. 8) are more difficult to learn than selective or weighted metrics. While the selective metrics and the weighted metric are already well approximated by the ranker after a few pairwise comparisons (see Tab. I) on all datasets the constraint metrics achieve their maximum top-10 precision after 50 pairwise comparisons. Considering the integration of the *Preference Executor* component into the evolutionary algorithm, a rough learning direction already enables a performance improvement regarding the user's preference.

C. Personalized NAS Evaluation

In this section, we evaluate the framework proposed in Sec. III as a whole and the impact of different metrics as well as the learned metric on the neural network recommendations. Furthermore, to compare different multi-objective NAS algorithms, we implemented MONAS [2] as well as LEMONADE [22] based on the search space defined within the *NATS-Bench* dataset. For MONAS and the Regularized Evolutionary algorithm, we performed the search based on a synthetic preference and a preference learned within the *Preference Executor* component. Since LEMONADE searches the Pareto frontier so that all metrics $\mathcal{L} \in \mathbb{L}$ are pursued without personalization, we evaluated LEMONADE without the *Evaluation Generator* and *Preference Executor* components. We ran each NAS algorithm for 100 epochs on Cifar-10, Cifar-100 and ImageNet16-120 based on the *NATS-Bench* dataset. The population generation within the Regularized Evolutionary algorithm is included in the number of epochs. To train the *Preference Executor* component, we sampled 5 random architectures to generate 10 segment pairs U^{pair} and ranked them based on the chosen synthetic preference. As synthetic preference, we evaluated in Fig. 4 the synthetic, selective preference defined in Eq. 3 and in Fig. 5 the combined preference defined in Eq. 8. For the Regularized Evolutionary as well as for the LEMONADE

TABLE I
RANKNET - TOP - 10 PRECISION ON TEST ARCHITECTURES

dataset	metric	train comparisons								
		1	2	3	4	5	10	25	50	100
Cifar-10	accuracy constraint	$0,216$	$0,265$	$0,284$	$0,269$	$0,206$	$0,382$	$0,285$	$0,394$	$0,369$
	accuracy	$0,105$	$0,617$	$0,570$	$0,471$	$0,705$	$0,746$	$0,692$	$0,622$	$0,833$
	flops constraint	$0,282$	$0,215$	$0,288$	$0,168$	$0,234$	$0,241$	$0,232$	$0,294$	$0,419$
	latency	$0,713$	$0,696$	$0,512$	$0,770$	$0,703$	$0,753$	$0,659$	$0,671$	$0,665$
	weighted	$0,642$	$0,703$	$0,663$	$0,740$	$0,810$	$0,735$	$0,766$	$0,898$	$0,858$
	params	$0,321$	$0,312$	$0,428$	$0,418$	$0,308$	$0,682$	$0,577$	$0,539$	$0,565$
Cifar-100	accuracy constraint	$0,275$	$0,239$	$0,234$	$0,250$	$0,226$	$0,212$	$0,235$	$0,354$	$0,265$
	accuracy	$0,417$	$0,244$	$0,623$	$0,640$	$0,594$	$0,783$	$0,683$	$0,727$	$0,826$
	flops constraint	$0,296$	$0,206$	$0,236$	$0,199$	$0,236$	$0,236$	$0,194$	$0,297$	$0,265$
	latency	$0,539$	$0,527$	$0,494$	$0,611$	$0,557$	$0,569$	$0,572$	$0,600$	$0,566$
	weighted	$0,349$	$0,494$	$0,483$	$0,527$	$0,474$	$0,546$	$0,677$	$0,679$	$0,770$
	params	$0,227$	$0,369$	$0,434$	$0,324$	$0,400$	$0,610$	$0,509$	$0,630$	$0,527$
ImageNet16-120	accuracy constraint	$0,257$	$0,257$	$0,159$	$0,162$	$0,184$	$0,144$	$0,240$	$0,265$	$0,317$
	accuracy	$0,242$	$0,696$	$0,637$	$0,679$	$0,698$	$0,527$	$0,803$	$0,790$	$0,832$
	flops constraint	$0,217$	$0,213$	$0,226$	$0,227$	$0,119$	$0,232$	$0,275$	$0,337$	$0,326$
	latency	$0,391$	$0,310$	$0,323$	$0,340$	$0,322$	$0,148$	$0,191$	$0,268$	$0,421$
	weighted	$0,421$	$0,572$	$0,573$	$0,576$	$0,594$	$0,616$	$0,628$	$0,742$	$0,685$
	params	$0,331$	$0,489$	$0,351$	$0,545$	$0,444$	$0,688$	$0,571$	$0,759$	$0,604$

approach we chose a population size p of 10 and a sample size s of 5. The pairwise ranking approach manages to steer the population over the cycles towards a stronger synthetic preference within MONAS and the Regularized Evolutionary approach (see. Sec. IV-B).

When evaluating the NAS algorithms based on a selective accuracy metric (Eq. 3), the Regularized Evolutionary approach achieves within 100 epochs 99.7% on Cifar-10, 99.6% on Cifar-100 and 99.8% on ImageNet16-120 (Fig. 4) the best performances. Note, that the metrics defined in Sec. IV-A are normalized, so that a score of 99.7% on $h_{accuracy}$ means that the NAS algorithm found an architecture that has 99.7% of the performance from the best architecture within the *NATS-Bench* dataset. LEMONADE achieves with $h_{accuracy}$ as metric 99.4% on Cifar-10, 96.6% on Cifar-100 and 99.8% on ImageNet16-120. Personalizing MONAS and the Regularized Evolutionary approach (Alg. 1) with the *Evaluation Generator* and the *Preference Executor* by training the RankNet ranker within the *Preference Executor* component with 10 synthetically generated preferences U^{judged} (see Sec. IV-B) MONAS achieves 91.5% and the Regularized Evolutionary approach $99,8\%$ on Cifar-10. The Cifar-100 and ImageNet16-120 datasets further show that MONAS performs inferior with both the predefined metric (58.7% on ImageNet16-120 and 64.9% on Cifar-100) and the metric learned within the *Preference Executor* (64.8% on Cifar-100 and 58.8% on ImageNet16-120). Furthermore, the Regularized Evolutionary approach achieves with the learned metric on Cifar-100 99.2% and 97.1% on ImageNet16-120. In Fig. 5, we evaluate the combined metric $h_{weighted}$ (Eq. 6). In this experiment, it is shown that MONAS (79.3%) can compete with LEMONADE (79.5%) and the Regularized Evolutionary (79.4%) approach on Cifar-10. However, on Cifar-100 and ImageNet16-120 MONAS performs marginally weaker than LEMONADE and the Regularized Evolutionary approach. Whereby the Regularized Evolutionary and the LEMONADE approach work similarly well. By comparing the performances of MONAS and the Regularized Evolutionary approach on $h_{weighted}$ and the learned metric of the *Preference Executor*, we show that the trained metric manages to steer neural architectures in Fig. 4 into the direction of accuracy based metric and in Fig. 5 into an equally distributed weighted metric. The differences between the defined metrics and the trained one are negligible, so that a fruitful direction to personalized NAS is shown. We presented a novel approach for multi-criteria NAS based on pairwise comparisons and evaluated it based on synthetic preferences. A central motivation for synthetic experiments was to empirically prove that the personalization approaches can be integrated into the NAS process. Since the evaluation of the *Preference Executor* component has shown that a suitable metric can be learned by pairwise comparisons and thus lead to a suitable recommendation of neural architectures. Thus, we show that the *Preference Executor* manages to steer within 10 pairwise comparisons NAS algorithms into the direction a user might want to pursue. Furthermore, we show within the evaluation of the proposed personalized NAS approach in Sec. IV-C the integration of the *Preference Executor* component.

V. CONCLUSION

We presented an approach for *Personalized NAS* [3], which enables human interaction in order to scale multi-objective NAS to larger numbers of objectives. Personalized NAS therefore learns a preference-based loss function based on end-user preferences for possible architectures, which can directly be used as an objective for single-objective NAS algorithms. To compare the search efficiency of different NAS algorithms, we

[3]https://github.com/kulbachcedric/PersonalizedNAS

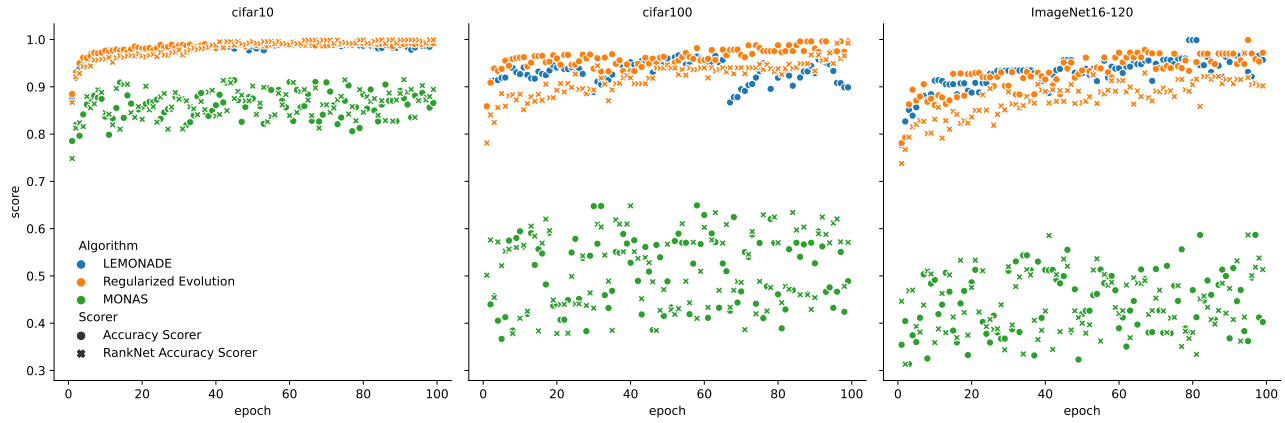

Fig. 4. $h_{accuracy}$ scores for MONAS, LEMONADE and Regularized Evolutionary Algorithm based on Cifar 10, Cifar 100 and ImageNet16-120

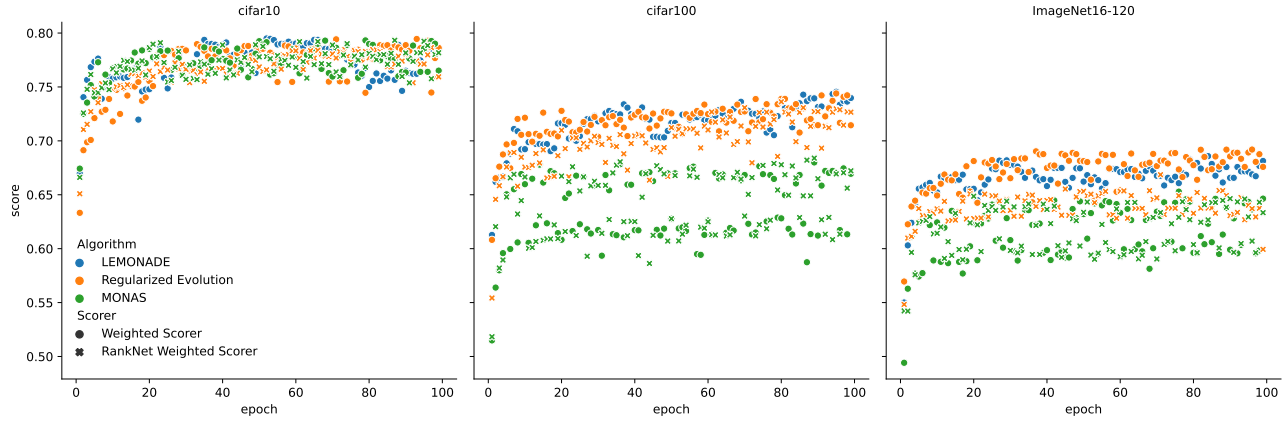

Fig. 5. $h_{weighted}$ scores for MONAS, LEMONADE and Regularized Evolutionary Algorithm based on Cifar-10, Cifar-100 and ImageNet16-120

implemented an OpenAI gym environment [4], which enables the integration and thus the evaluation of RL based NAS algorithms on the *NATS-Bench* dataset. Additionally, it enables the research of further NAS approaches. To show (i) the possibility of adapting the search direction of NAS algorithms to user centric needs, we evaluated in Sec. III-B the *Evaluation Generator* and to show (ii) the impact of a personalized loss function on NAS, we evaluated in Sec. IV-C the influence on NAS performances. Our results for Cifar-10, Cifar-100, and ImageNet16-120 from the *NATS-Bench* dataset are promising in that the available metric functions can be successfully weighted and combined to efficiently find the exemplary user preferences. Furthermore, we presented a resource-aware evaluation by evaluating which multi-objective NAS algorithm delivers the best performance in a small number of epochs and thus recommends suitable neural networks.

[4]https://github.com/kulbachcedric/Natsbench-Gym

VI. Outlook

A fruitful direction to extend our personalized NAS approach is the integration of further metrics such as the activation function proposed in Sec. III-C. Furthermore, we plan to evaluate *Personalized Neural Architecture Search* based on human preferences within the *Human Preference Interface* introduced in Sec. III-C, where end-users use the system to find architectures based on defined sets of objectives. Thus, our approach enables the incorporation of real-world objectives, such as per-class accuracies or measures from explainable machine learning. We proposed a first visualization draft within the *Human Preference Interface* which enables pairwise rankings and the human interaction with the proposed framework. Since a human evaluation is dependent on an adequate interface to allow the user to express his preference, it is essential to conduct further research based on which a user can make decisions. At this point, one can investigate the necessity to conduct multiple rounds of collecting user preferences and learning the ranking function. An interesting research direction is exploring few-shot learning for *Personalized NAS*, where

the goal is to warm-start the *Architecture Sampler* component based on learned preferences from previous users. Finally, the *Human Preference* component could be improved by active learning, where pairwise comparisons would be chosen based on the actual information gain and uncertainty of the preference-based loss function.

REFERENCES

[1] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=r1Ue8Hcxg

[2] C. Hsu, S. Chang, D. Juan, J. Pan, Y. Chen, W. Wei, and S. Chang, "MONAS: multi-objective neural architecture search using reinforcement learning," *CoRR*, vol. abs/1806.10332, 2018. [Online]. Available: http://arxiv.org/abs/1806.10332

[3] E. Real, S. Moore, and A. S. et. al., "Large-scale evolution of image classifiers," in *ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 2902–2911. [Online]. Available: http://proceedings.mlr.press/v70/real17a.html

[4] Y.-H. Kim, B. Reddy, S. Yun, and C. Seo, "Nemo: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy," in *JMLR: Workshop and Conference Proceedings*, vol. 1, 2017, pp. 1–8.

[5] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=ByME42AqK7

[6] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, M. Urban, M. Burkart, M. Dippel, M. Lindauer, and F. Hutter, "Towards automatically-tuned deep neural networks," in *Automated Machine Learning - Methods, Systems, Challenges*, ser. The Springer Series on Challenges in Machine Learning, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Springer, 2019, pp. 135–149. [Online]. Available: https://doi.org/10.1007/978-3-030-05318-5_7

[7] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, ser. JMLR Workshop and Conference Proceedings, vol. 28. JMLR.org, 2013, pp. 115–123. [Online]. Available: http://proceedings.mlr.press/v28/bergstra13.html

[8] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-Bench: Benchmarking nas algorithms for architecture topology and size," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021, doi:10.1109/TPAMI.2021.3054824.

[9] N. Boukhelifa, A. Bezerianos, and E. Lutton, "Evaluation of interactive machine learning systems," in *Human and Machine Learning - Visible, Explainable, Trustworthy and Transparent*, ser. Human-Computer Interaction Series, J. Zhou and F. Chen, Eds. Springer, 2018, pp. 341–360. [Online]. Available: https://doi.org/10.1007/978-3-319-90403-0_17

[10] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 4092–4101. [Online]. Available: http://proceedings.mlr.press/v80/pham18a.html

[11] J. Jiang, F. Han, Q. Ling, J. Wang, T. Li, and H. Han, "Efficient network architecture search via multiobjective particle swarm optimization based on decomposition," *Neural Networks*, vol. 123, pp. 305–316, 2020. [Online]. Available: https://doi.org/10.1016/j.neunet.2019.12.005

[12] A. Anderson, J. Su, R. Dahyot, and D. Gregg, "Performance-oriented neural architecture search," *CoRR*, vol. abs/2001.02976, 2020. [Online]. Available: http://arxiv.org/abs/2001.02976

[13] Y. Jaâfra, J. L. Laurent, A. Deruyver, and M. S. Naceur, "Reinforcement learning for neural architecture search: A review," *Image Vis. Comput.*, vol. 89, pp. 57–66, 2019. [Online]. Available: https://doi.org/10.1016/j.imavis.2019.06.005

[14] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=S1c2cvqee

[15] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, Eds. ACM, 2019, pp. 1946–1956. [Online]. Available: https://doi.org/10.1145/3292500.3330648

[16] F. L. Bauer, "Computational graphs and rounding error," *SIAM Journal on Numerical Analysis*, vol. 11, no. 1, pp. 87–96, 1974.

[17] X. Chu, B. Zhang, R. Xu, and H. Ma, "Multi-objective reinforced evolution in mobile neural architecture search," *CoRR*, vol. abs/1901.01074, 2019. [Online]. Available: http://arxiv.org/abs/1901.01074

[18] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 2820–2828.

[19] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-bench-101: Towards reproducible neural architecture search," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 7105–7114. [Online]. Available: http://proceedings.mlr.press/v97/ying19a.html

[20] C. Saltori, S. Roy, N. Sebe, and G. Iacca, "Regularized evolutionary algorithm for dynamic neural topology search," in *Image Analysis and Processing - ICIAP 2019 - 20th International Conference, Trento, Italy, September 9-13, 2019, Proceedings, Part I*, ser. Lecture Notes in Computer Science, E. Ricci, S. R. Bulò, C. Snoek, O. Lanz, S. Messelodi, and N. Sebe, Eds., vol. 11751. Springer, 2019, pp. 219–230. [Online]. Available: https://doi.org/10.1007/978-3-030-30642-7_20

[21] Z. Lu, I. Whalen, V. Boddeti, Y. D. Dhebar, K. Deb, E. D. Goodman, and W. Banzhaf, "Nsga-net: neural architecture search using multi-objective genetic algorithm," in *Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, A. Auger and T. Stützle, Eds. ACM, 2019, pp. 419–427. [Online]. Available: https://doi.org/10.1145/3321707.3321729

[22] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ByME42AqK7

[23] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 8022–8034.

[24] K. Crammer and Y. Singer, "Pranking with ranking," in *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2001, pp. 641–647. [Online]. Available: http://papers.nips.cc/paper/2023-pranking-with-ranking

[25] P. Li, C. J. C. Burges, and Q. Wu, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *Advances in Neural Information Processing Systems 20, Proceedings of the 21. Annual Conference on Neural Information Processing Systems, 2007*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2007, pp. 897–904. [Online]. Available: http://papers.nips.cc/book/advances-in-neural-information-processing-systems-20-2007

[26] T. Graepel, R. Herbrich, and K. Obermayer, "Bayesian transduction," in *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, S. A. Solla, T. K. Leen, and K. Müller, Eds. The MIT Press, 1999, pp. 456–462. [Online]. Available: http://papers.nips.cc/paper/1712-bayesian-transduction

[27] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender, "Learning to rank using gradient descent," in *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, ser. ACM International Conference Proceeding Series, L. D. Raedt and S. Wrobel, Eds., vol. 119. ACM, 2005, pp. 89–96. [Online]. Available: https://doi.org/10.1145/1102351.1102363

[28] I. F. D. Oliveira, N. Ailon, and O. Davidov, "A new and flexible approach to the analysis of paired comparison data," *J. Mach. Learn. Res.*, vol. 19, pp. 60:1–60:29, 2018. [Online]. Available: http://jmlr.org/papers/v19/17-179.html

[29] D. Sculley, "Combined regression and ranking," in *ACM SIGKDD 2010*, B. Rao, B. Krishnapuram, A. Tomkins, and Q. Yang, Eds. ACM, 2010, pp. 979–988.

[30] F. Çakir, K. He, X. Xia, B. Kulis, and S. Sclaroff, "Deep metric learning to rank," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 1861–1870. [Online]. Available: http://openaccess.thecvf.com/CVPR2019.py

[31] O. A. S. Ibrahim and D. Landa-Silva, "Es-rank: evolution strategy learning to rank approach," in *Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco, April 3-7, 2017*, A. Seffah, B. Penzenstadler, C. Alves, and X. Peng, Eds. ACM, 2017, pp. 944–950. [Online]. Available: https://doi.org/10.1145/3019612.3019696

[32] C. J. C. Burges, R. Ragno, and Q. V. Le, "Learning to rank with nonsmooth cost functions," in *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, B. Schölkopf, J. C. Platt, and T. Hofmann, Eds. MIT Press, 2006, pp. 193–200. [Online]. Available: http://papers.nips.cc/paper/2971-learning-to-rank-with-nonsmooth-cost-functions

[33] J. H. Friedman and J. J. Meulman, "Multiple additive regression trees with application in epidemiology," *Statistics in Medicine*, vol. 22, no. 9, pp. 1365–1381, 2003. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.1501

[34] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 618–626. [Online]. Available: https://doi.org/10.1109/ICCV.2017.74

[35] *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/group?id=ICLR.cc/2017/conference