Learning Debuggable Models Through Multi-Objective NAS

Anonymous[1]

[1]Anonymous Institution

Abstract Monumental advances in deep learning have led to unprecedented achievements across various domains. While the performance of deep neural networks is indubitable, the architectural design and interpretability of such models are nontrivial. Research has been introduced to automate the design of neural network architectures through *neural architecture search* (NAS). Recent progress has made these methods more pragmatic by exploiting distributed computation and novel optimization algorithms. However, there is little work in optimizing architectures for interpretability. To this end, we propose a multi-objective distributed NAS framework that optimizes for both task performance and "introspectability," a surrogate metric for the debuggability of a model. We leverage the non-dominated sorting genetic algorithm (NSGA-II) and explainable AI (XAI) techniques to reward architectures that can be better comprehended by domain experts. The framework is evaluated on several image classification datasets. We demonstrate that jointly optimizing for task error and introspectability leads to more disentangled and debuggable architectures that perform within tolerable error.

1 Introduction

The success of deep learning is seemingly ubiquitous in a multitude of domains. A core component of its effectiveness is its ability to automate the feature engineering process. Under this perspective, a natural next step is the automation of the architecture design. To this end, *neural architecture search* (NAS) (Elsken et al., 2019b) has been proposed. Progress in NAS has led to results that supersede the state-of-the-art in several applications, such as image classification (Real et al., 2019) and object detection (Zoph et al., 2018).

While NAS has been effective in automating architecture discovery and topping leaderboards, little attention has been paid to the discovery of interpretable architectures. The automation of interpretability would further minimize the need for "human-in-the-loop" pipelines. Not only does this reduce the manual design needed to meet the constraints of an application, but it also enhances the comprehensibility of the discovered models. Increased comprehensibility aids in model debugging, decreases time to deployment, and instills greater trust.

To this end, we introduce a framework for the joint optimization of task performance and a surrogate for the debuggability of a model. In this work, we put forth the following contributions:

- We develop a new metric to quantify debuggability as the disentanglement between latent representations for different data classes: *introspectability*. We further extend this metric by exploiting hierarchical semantic information from the WordNet database.
- We formalize the discovery of interpretable architectures as a multi-objective optimization problem and propose an evolutionary approach to NAS, eXplainable NAS (`XNAS`), that maximizes both accuracy and introspectability by directly optimizing the Pareto frontier.
- We conduct analyses of the accuracy-introspectability trade-off, explore phylogenetic trees to understand the inheritability of objectives, visualize disentangled representations, analyze architectural motifs along the Pareto front, and demonstrate introspectability as a surrogate for trustworthiness and debuggability.

 © 2023 the authors, released under CC BY 4.0

2 Background & Related Work

Our work is at the intersection of neural architecture search, explainable AI (XAI), evolutionary algorithms, and multi-objective optimization. We present a brief overview of these topics and discuss related work to put our contributions in context.

Neural Architecture Search (NAS). Akin to how deep learning is used to automate feature engineering, NAS algorithms automate *architectural engineering* (Elsken et al., 2019b). NAS algorithms can generally be understood as the composition of three elements: (i) a *search space* that defines the possible neural architectures, (ii) a *search strategy* that explores a search space for candidate solutions, and (iii) a *performance estimation strategy* that determines the fitness of a solution. Of the many approaches to NAS, Bayesian optimization (BO), reinforcement learning (RL), and evolutionary algorithms are the most common. While BO is typically applied to low-dimensional problems, several works have applied it to NAS (Bergstra et al., 2013; Domhan et al., 2015) and it has even surpassed human experts on competition datasets (Mendoza et al., 2016). However, BO has mostly been overshadowed by RL ever since Zoph and Le achieved unprecedented results on NAS benchmarks (Zoph and Le, 2017).

While RL-based NAS has achieved state-of-the-art across many benchmarks, it tends to be compute-inefficient and can take thousands of GPU hours to converge (Real et al., 2017; Zoph et al., 2018). *Neuro-evolutionary* approaches are generally lightweight in comparison, and they notably perform the same as RL approaches on NAS benchmarks (Real et al., 2019). The use of evolutionary algorithms for NAS can be traced back decades, e.g. (Miller et al., 1989) uses genetic algorithms to propose architectures that are then trained using backpropagation. While evolutionary algorithms have been used to search for both weights and network architectures (Angeline et al., 1994; Stanley and Miikkulainen, 2002), it is more common to only apply evolution to the architecture and to train the weights with gradient descent (Real et al., 2019, 2017; Elsken et al., 2019a). Evolutionary algorithms evolve a population of candidate solutions to an optimization problem and each generation is derived from the last by applying mating operations to a set of selected parents. In NAS, an offspring may differ from its parents by an added layer, a changed connection, etc. The quality of solutions is judged by a fitness function and evolution is terminated when a resource or time budget is exceeded.

Multi-Objective Optimization & NAS. In a multi-objective optimization problem, there are m objectives $\{f_1, \ldots, f_m\}$, which in the context of NAS may be accuracy, floating point operations (FLOPs), energy, etc. When $m > 1$, it becomes nontrivial to select the optimal solution among the set of all objective vectors $Y = \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y} = \{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\}\}$ where $\mathbf{x}$ is a candidate solution. There exists a variety of strategies to select solutions, such as optimizing for a weighted sum of the (normalized) objectives, lexicographic sorting, or maintaining Pareto-optimal solutions (Marler and Arora, 2004). We are most interested in the latter approach since it captures the trade-offs between objectives and allows the practitioner to choose the optimal compromise for their use case. The set of Pareto-optimal solutions, also called the Pareto frontier or Pareto front, is the set of non-dominated solutions $\{\mathbf{y}' \in Y \mid \{\mathbf{y} \in Y \mid \mathbf{y} \succ \mathbf{y}'\} = \emptyset\}$, where $\mathbf{a} \succ \mathbf{b}$ indicates that $\mathbf{a}$ strictly dominates $\mathbf{b}$, i.e. $|\{f_i(\mathbf{a}) \mid 1 \leq i \leq m, f_i(\mathbf{a}) > f_i(\mathbf{b})\}| = m$.

The non-dominated sorting genetic algorithm-II (NSGA-II) (Deb et al., 2002) is an elitist evolutionary approach to multi-objective optimization. Notably, the authors improve the non-dominated sorting algorithm from cubic to quadratic time complexity. The surviving members of a generation are selected in a binary tournament with preference given to members of the Pareto front. Additional offspring are generated from members in the ranked fronts, i.e. the Pareto fronts computed iteratively after removing the members of the previous front. When a ranked front needs to be subsampled, the crowding distance within the front is used to ensure the full front is represented.

Related to our work, NSGA-Net (Lu et al., 2019) is an evolutionary framework for NAS that employs NSGA-II for multi-objective optimization. Like most evolutionary NAS algorithms, NSGA-

Net explores and exploits the search space with a fixed-size population of candidate architectures. The authors demonstrate the effectiveness of population-based NAS and the superiority of NSGA-II over a weighted sum of objectives on the CIFAR-10 and CIFAR-100 datasets. While similar to our framework, our focus is on the design of objectives conducive to interpretability. Furthermore, we scale our method to a distributed cluster and evaluate on more datasets.

Multi-objective optimization of a weighted sum of objectives has been employed by many NAS works. The approach is attractive when the objectives are differentiable since it is amenable to gradient descent by backpropagation. For instance, Multi-Objective NAS (MONAS) (Hsu et al., 2018) uses RL with a weighted combination of accuracy, power, and multiply-accumulate operations (MACs) as the reward. However, there are limitations to optimizing for an aggregate of multiple objectives: it relies on manually tuned coefficients, struggles to accommodate objectives that range over multiple orders of magnitude, and tends to cluster in a small region of the Pareto front.

Explainable AI (XAI) & NAS. Some of the intersection between interpretability and NAS has been covered in prior work. In Ru et al. (2021), a NAS framework using the Bayesian optimization search strategy is proposed. For efficiency and interpretability, a Weisfeiler-Lehman graph kernel is used to define a Gaussian process surrogate on the search space, and the gradients are used to identify key motifs that lead to well-performing architectures. Similarly, Adam and Lorraine (2019); Zheng et al. (2020) use alternative techniques to identify key motifs used in the search process. However, their notions of interpretability and disentanglement focus on the search process rather than on the learned models themselves. In this work, we extend NAS to disentangle the latent space of learned models.

3 Proposed Framework: `XNAS`

Following the taxonomy in Elsken et al. (2019b), we break up our method into a search space, search strategy, and performance estimation strategy. We further discuss how we scale the search up to an arbitrary number of compute nodes in Appendix O.

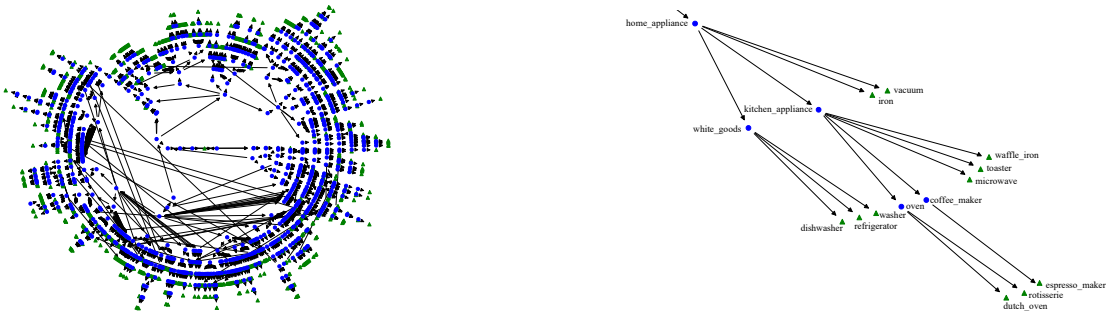
3.1 Search Space and Search Strategy

We are interested in exploring complex search spaces beyond simple chains, i.e. multi-branch networks such as ResNet (He et al., 2016) or DenseNet (Huang et al., 2017). To this end, we elect to use the popular NAS-Bench-201 search space (Dong and Yang, 2020), which is comprised of a macro skeleton and a searched cell. A full overview is given in Appendix A.

As we are interested in discovering neural architectures that are both accurate and debuggable, we propose to use multi-objective optimization. We explore and exploit the search space using the Non-Dominated Sorting Genetic Algorithm II (NSGA-II), as introduced in Section 2, with two objectives: accuracy and *introspectability* (introduced in Section 3.2). Because we search for architectures that are both accurate and interpretable, we refer to our approach as eXplainable NAS (`XNAS`). We generate the initial set of solutions by uniformly sampling the layers of the searched cell (see Appendix A for details). These candidates comprise the first generation of the *population*. Thereafter, the offspring of the proceeding generation are produced by mating the parents comprising the prior generation. Parents are selected based on the ranked Pareto fronts of the population as described in Section 3.2 and Deb et al. (2002). Because of this selection, there is no notion of a single best solution, but rather a set of non-dominated solutions that characterize the optimal trade-off between all objectives. The details of crossover and mutation are provided in Appendix J.

3.2 Performance Evaluation Strategy

We evaluate the performance of an architecture using two objectives: task performance and interpretability (debuggability). The former is simple to define quantitatively as the classification



(a) (b)

Figure 1: (a) The WordNet hyponym-hypernym graph where leaf nodes (green) are the labels of the ImageNet dataset. (b) An inset of the full graph containing the synset "home appliance" and its hyponyms.

accuracy on the held-out validation split of a dataset. However, interpretability is often treated far more qualitatively and an objective definition eludes community consensus. Furthermore, explaining a model is dependent on the audience, data modality, modeling task, and questions being asked. To disambiguate interpretability in the context of the framework, we state our assumptions: that the user has some technical understanding (e.g. a data scientist or domain expert), that we are interested in understanding the model in classification tasks (e.g. as opposed to the data or the NAS evolution process), and that models that maximize the metric lead to qualitatively discernible trends. To this end, we propose to quantify the interpretability of models as the introspectability of disentangled elements, which we describe in the subsequent subsections. We measure this for supervised classification tasks using the pairwise distances between latent representations of individual classes.

Introspectability. Here we formalize the score that we denote as *introspectability*: the degree to which the representations of disparate classes within a neural network $\mathcal{M}$ are disentangled. Let us denote the subset of validation data belonging to class c as $\mathfrak{X}^{(c)} \in \mathbb{R}^{N^{(c)} \times H \times W \times C}$. Given $\mathfrak{X}^{(c)}$ as input to $\mathcal{M}$, denote the activations of layer l as $\mathbf{\Phi}^{(c,l)} \in \mathbb{R}^{N^{(c)} \times d_1^{(l)} \times \cdots \times d_n^{(l)}}$. We reshape the activations to have a single dimension of size $d^{(l)} = \prod_{i=1}^{n} d_i^{(l)}$ such that $\mathbf{\Phi}^{(c,l)} \in \mathbb{R}^{N^{(c)} \times d^{(l)}}$. We denote all activations for class c within $\mathcal{M}$ as $\mathbf{\Phi}^{(c)} = \|_{l=1}^{L} \mathbf{\Phi}^{(c,l)}$ where $\|$ is the matrix concatenation operator along the columns and L is the number of layers in $\mathcal{M}$. The mean activations for class c are then $\bar{\mathbf{\Phi}}^{(c)} = \frac{1}{N^{(c)}} \sum_{i=1}^{N^{(c)}} \mathbf{\Phi}_i^{(c)}$ where $|\bar{\mathbf{\Phi}}^{(c)}| = \sum_{l=1}^{L} d^{(l)}$. With these definitions, we then formulate introspectability as (1)

$$\text{Introspectability}(\mathcal{M}, \mathfrak{X}) = \frac{1}{\binom{N_C}{2}} \sum_{c=1}^{N_C} \sum_{k=c+1}^{N_C} D(\bar{\mathbf{\Phi}}^{(c)}, \bar{\mathbf{\Phi}}^{(k)}) \tag{1}$$

where $D(\cdot, \cdot)$ gives the cosine distance between its two vector arguments and N_C is the number of classes in the classification task.

Introspectability and WordNet. We derive a second definition of introspectability based on WordNet (Miller, 1995), a lexical database of the English language. WordNet comprises sets of synonyms (*synsets*) and arises into a hierarchical representation by embedding the transitive relations *hyponyms* (more specific sub-names) and *hypernyms* (more abstract super-names). In computer vision, the labels of the ImageNet database (Deng et al., 2009) are notably derived from WordNet synsets. We visualize all of the labels covered by ImageNet in the hyponym-hypernym

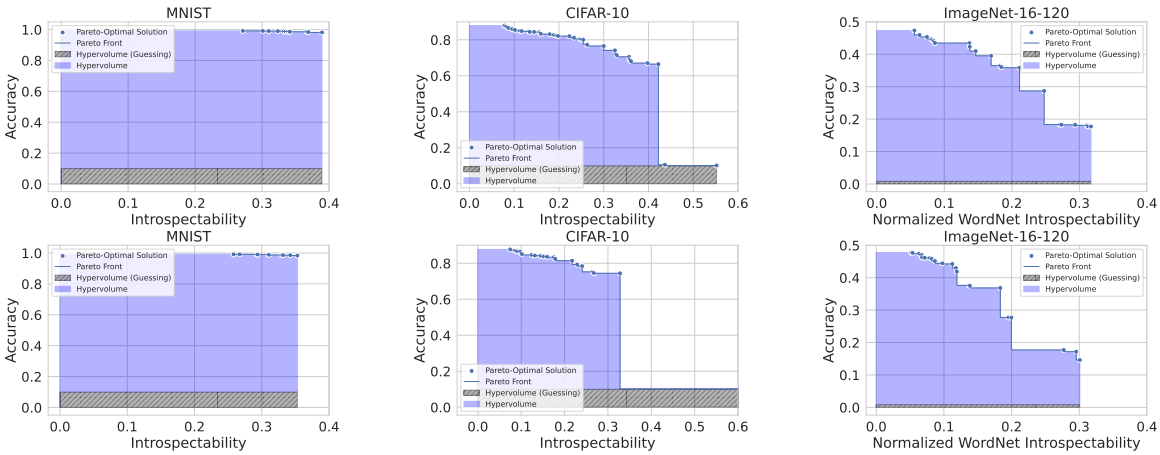


Figure 2: The Pareto fronts and Pareto-optimal solutions are shown for each task. The shaded region visualizes the hypervolume achieved by XNAS. In the first row, the front is visualized for multi-objective XNAS and for single-objective (accuracy only) in the second row. Normalized $\text{Introspectability}_{\text{WordNet}}$ is shown for ImageNet-16-120.

graph shown in Figure 1a and 1b. The shortest path distances between two labels in the hyponym-hypernym graph can be used to compute semantic similarity as shown in (2)

$$\texttt{path_sim}(w_a, w_b) = \frac{1}{\texttt{shortest_path}(w_a, w_b) + 1} \tag{2}$$

where w_a and w_b are label names. We then weigh the pairwise distances between classes by this similarity as (3)

$$\text{Introspectability}_{\text{WordNet}}(\mathcal{M}, \mathfrak{X}) = \frac{1}{\binom{N_C}{2}} \sum_{c=1}^{N_C} \sum_{k=c+1}^{N_C} D(\bar{\Phi}^{(c)}, \bar{\Phi}^{(k)}) \times S(c, k). \tag{3}$$

The similarity between labels S is given by $S(c, k) = \texttt{path_sim}(\texttt{name}(c), \texttt{name}(k))$ where $\texttt{name}(\cdot)$ maps the label index to the corresponding label name. Intuitively, the score penalizes models with relatively small distances between dissimilar labels and compensates for small distances between similar labels. To ensure the range of WordNet introspectability is comparable to that of the baseline definition, we normalize the score by dividing by the mean `path_sim` value among all pairs of labels in the dataset.

4 Experiments & Results

We evaluate XNAS on three image classification datasets: MNIST LeCun et al. (2010), CIFAR-10 Krizhevsky (2009), and ImageNet-16-120 (Dong and Yang, 2020). Thereafter, we conduct analyses to understand the evolution process of XNAS, characterize the Pareto front of each task, and demonstrate the debuggability of higher-introspectability architectures.

4.1 Setup and Implementation

The experimental setup and all hyperparameters are detailed in Appendix B. We implement XNAS in Python with code built on the `pymoo` (Blank and Deb, 2020), `DeepHyper` (Balaprakash et al., 2018), and `Ray` (Moritz et al., 2018) libraries. The source code is publicly available at <redacted>.

Dataset	Multi-Objective	Gen.	Population-Level Statistics				Hypervolume
			Max Acc.	Median Acc.	Maximum Intros.	Median Intros.	
MNIST	reg	–	98.9%	98.6%	0.384	0.291	0.341
	✗	18	99.1%	98.8%	0.353	0.255	0.314
	✓	18	99.1%	98.6%	0.390	0.258	0.347
	✓+reg	18	99.0%	98.6%	0.503	0.303	**0.424**
CIFAR-10	reg	–	85.9%	72.8%	0.331	0.178	0.229
	✗	34	87.7%	84.3%	0.328	0.077	0.237
	✓	34	87.9%	74.6%	0.552	0.196	0.293
	✓+reg	34	87.7%	74.2%	0.654	0.249	**0.361**
ImageNet-16-120	reg	–	44.8%	36.2%	0.318	0.087	0.099
	✗	11	47.8%	44.9%	0.301	0.053	0.099
	✓	11	47.3%	39.1%	0.317	0.104	0.111
	✓+reg	11	47.4%	39.0%	0.380	0.109	**0.117**

Table 1: XNAS experimental results on image classification datasets listing the population-level accuracy and introspectability (Intros.) scores, and hypervolume. Introspectability can be used as a regularizer ("reg"), as described in Section 4.3. We compare to this as a baseline, as well as to using the multi-objective approach followed by fine-tuning with the regularizer ("✓+reg"). Normalized $\text{Introspectability}_{\text{WordNet}}$ is shown for ImageNet-16-120. Multi-objective XNAS with regularization yields the greatest hypervolume across all tasks.

4.2 Metrics

To quantitatively assess the quality of the solutions from a multi-objective search algorithm, we look to *hypervolume* as introduced in Zitzler and Thiele (1998) and improved in Fonseca et al. (2006). Hypervolume is the area of the union of rectangles where each rectangle is formed by a point on the Pareto front and a reference point (such as $(0, 0)$). This notion can easily be extended to higher dimensions, i.e. rectangular cuboids are formed with three objectives and hyperrectangles are formed with four or more objectives. In this work, we set the reference point to $(0, 1/N_C)$ where $1/N_C$ is the classification rate of random guessing with balanced data. Note that this reduces the hypervolume range from $[0, 1]$ to $[0, (1-1/N_C)]$. We are interested in setting the reference point here to avoid rewarding models that have not learned the task in any significant capacity.

4.3 Results

Table 1 contains the aggregate results for each task and demonstrates the efficacy of using the proposed multi-objective approach. XNAS is compared with the single-objective NAS baseline. While the median population-level accuracy falls slightly compared to single-objective NAS, the maximum accuracy is still comparatively high and there is a substantial increase in hypervolume across experiments. Note that the best accuracy of XNAS on ImageNet-16-120 is on par with the best-performing methods as evaluated in Dong and Yang (2020). We plot the Pareto front of every NAS search result and shade in the hypervolume in Figure 2. The visualizations make clear where the multi-objective approach makes up hypervolume over single-objective (accuracy). Across all tasks, multi-objective covers a larger range of introspectability values. As one would expect, focusing on accuracy tends to cluster the majority of non-dominated solutions in the upper left of the front. The hypervolume of random guessing illustrates why we set the hypervolume reference point to $(0, 1/N_C)$: some solutions manage to achieve high introspectability but are effectively useless since their predictions are no better than random.

As another baseline, introspectability is employed as a *regularization* term. Introspectability is differentiable and in turn, can be used to train models as an auxiliary loss. However, this drastically

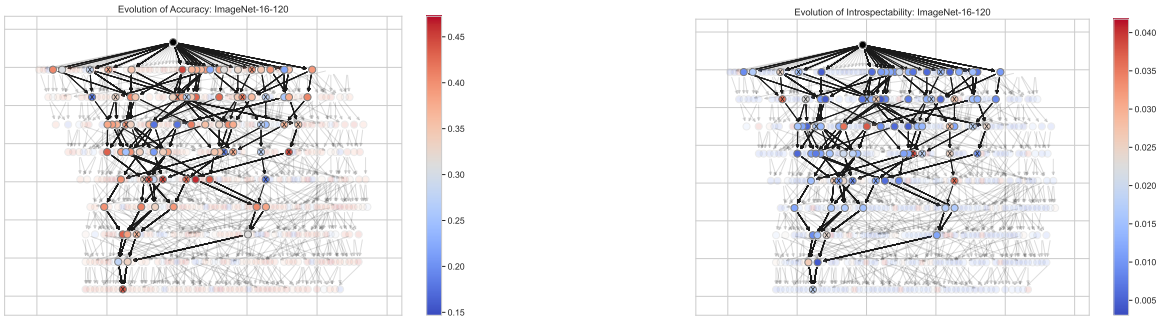


Figure 3: Phylogenetic trees showing the evolution of the population in XNAS on the ImageNet-16-120 task up to the eighth generation. The mating path from the initial population to a solution in the final Pareto front of the search is emphasized. Along the path, parents that were Pareto-optimal within their own generation are marked with an ×. The evolution of accuracy (top) and introspectability (bottom) within the population are shown.

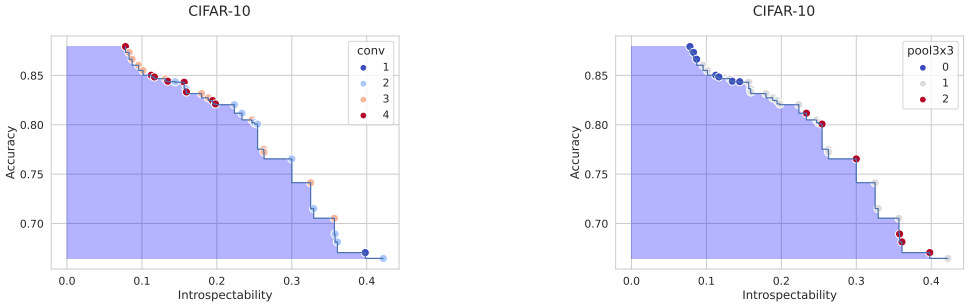


Figure 4: The Pareto front for the CIFAR-10 task with solutions colored by the number of convolutional (left) and pooling (right) layers per cell.

increases the training computational complexity and memory utilization due to the calculation of pairwise distances and the accumulation of activations. In experiments, this slows down training by several orders of magnitude. Accordingly, the number of search space evaluations is limited. Regularization achieves introspectability scores competitive with the multi-objective approach on MNIST and ImageNet-16-120, but not CIFAR-10. In addition, the achieved accuracy and hypervolume are hindered due to the reduced evaluations. Synergistically, we also evaluate regularization applied to the Pareto front of the multi-objective approach – XNAS is capable of discovering high-accuracy solutions that are predisposed to higher introspectability via the regularization approach. This combination performs best but demands the additional computation.

To understand the evolution process of XNAS, consider the phylogenetic trees shown in Figure 3. It shows the ancestry for a Pareto-optimal solution from the eighth generation with the ImageNet-16-120 task. While most solutions on a given generation's Pareto front are not directly descended from the previous generation's Pareto front, they typically have many Pareto-optimal ancestors. This suggests that Pareto optimality is mildly heritable, although not enough to ensure direct transmission between generations. In addition, the final Pareto-optimal solution had the second-highest cumulative accuracy at its generation. It is not surprising that its ancestors tended to have above-average accuracy and below-average introspectability.

We conduct an analysis of emerging patterns in the architectures discovered across the Pareto front for each task. The methodology for selecting motifs of interest is described in the supplemental material and is accompanied by the corresponding visualizations. In Figure 4, we elect to visualize a pattern that holds for all tasks but is shown for CIFAR-10. Therein, we observe that more accurate models have fewer pooling layers and more convolutional layers, whereas models with greater

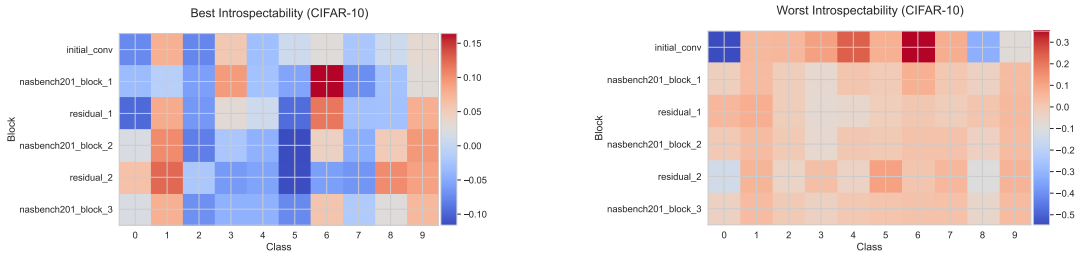


Figure 5: Mean activations heatmap of the model with highest (top) and lowest (bottom) introspectability on the CIFAR-10 task.

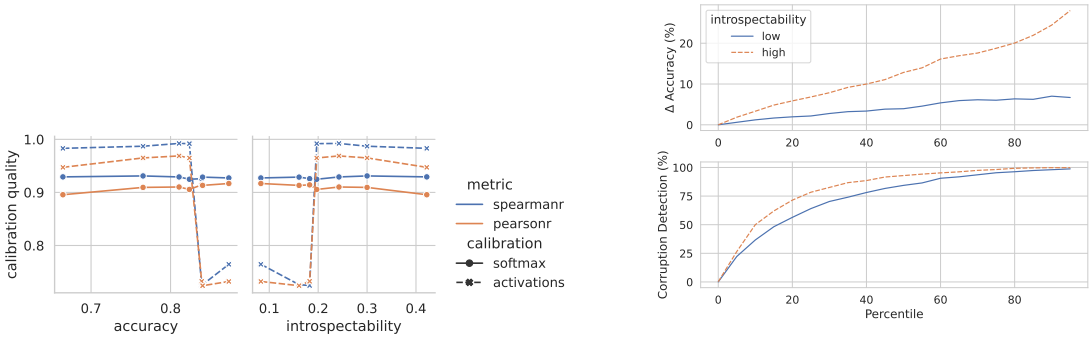
introspectability exhibit the opposite tendency. These layer types can be seen as one knob that controls the accuracy-introspectability trade-off. Furthermore, a study of the impact of accuracy and introspectability of the generalization error, convergence speed, and the number of parameters is presented in Appendix J. High-introspectability models have lower generalization error, fewer parameters, and faster training times, whereas high-accuracy models exhibit the inverse trend.

To gain a better qualitative understanding of the introspectability metric, we visualize the activations of the Pareto-optimal solutions of each task. In Figure 5, the solutions of the highest and lowest introspectability are shown for CIFAR-10. The MNIST and ImageNet-16-120 activations are shown in the supplemental material. Within each layer, the activations are normalized using z-score normalization. The activations within each block per class are then averaged for the purpose of visualization. The differences between the highest- and lowest-scoring models are quite apparent; the activation patterns for each class in higher-scoring models have notable variance, whereas they are quite constant in lower-scoring models.

Introspectability, trustworthiness, and debuggability. Figure 5 demonstrates that increased class-wise disentanglement is a result of optimizing for introspectability. Here, we exploit this to show that introspectability is a surrogate for improved trustworthiness and debuggability of discovered architectures. Typically, *softmax calibration* is employed to assess the confidence and trustworthiness of predictions. This approach interprets the softmax-activated readout layer outputs of DNN classifiers as probabilities. We propose to use an *activations-based calibration* that `XNAS` indirectly optimizes for. In activations-based calibration, we 1) collect the mean activations per class from training data, 2) collect the activations of the held-out data, 3) compute the cosine distances between the activations of held-out data and the mean activations per class (similar to Eq. (1)), and 4) interpret the distances of the predicted classes as probabilities. To assess calibration quality, we use the Pearson and Spearman rank correlation coefficients of the calibrated probabilities and the corresponding actual accuracy scores at each probability range on a held-out test set[1]. Fig. 6a demonstrates that introspectability estimates calibration quality and outperforms softmax calibration when introspectability> 0.18. Notably, the calibration quality tapers off rapidly at this point, indicating that class-wise latent representations are no longer as disentangled. This is an important observation of this approach – *DNN trustworthiness is a function of class-wise disentanglement*. This experiment serves as motivation that higher-introspectability models are more conducive to trustworthiness and debuggability, as we will demonstrate.

Identifying mispredictions We explore the effect of introspectability on the ability to identify mispredictions. To do so, low-confidence predictions are isolated according to the distribution of activation calibration scores. The quality of identified mispredictions is then quantified as the increase in accuracy due to removing samples. Figure 6b compares misprediction identification between low- and high-introspectability models on CIFAR-10 – high-introspectability models demonstrate greater improvement owing to superior calibration and trustworthiness.

[1]We use 50 linear bins to estimate these quantities.



(a) (b)

Figure 6: (a) Calibration quality of Pareto-optimal models discovered by XNAS. Calibration using activations improves on softmax for introspectability>0.18. (b) (Top) The change in accuracy due to identified mispredictions as a function of the percentile of sample-wise activation calibration scores. (b) (Bottom) The detection rate of corrupted labels as a function of the percentile of sample-wise activation calibration scores. All results are on CIFAR-10.

Debugging data We further demonstrate the improved model comprehensibility to debug data. We mislabel samples at a corruption rate of 20% and assess the ability of models to identify the mislabeled data following the distribution of activation calibration scores. Figure 6b shows that high-introspectability models are better equipped to identify bugs in the data on CIFAR-10, achieving a ∼20% higher detection rate at the 20th percentile of scored samples.

Debugging models Further model debugging experiments are included in Appendix K – we identify and repair bugs in models based on pairwise activation distances.

5 Discussion

As can be observed in Figure 2, there exists a clear trade-off between accuracy and introspectability, which is more pronounced as the classification task grows more complex, i.e. in the order of MNIST, CIFAR-10, and ImageNet-16-120. We discussed how this trade-off is influenced by the selection of operators within a cell in the previous section. Interestingly, the observation of this phenomenon at first seems to contradict the argument by Rudin et al. that the accuracy-interpretability trade-off is a false dichotomy (Rudin and Radin, 2019). However, an important distinction is that our observation is based on models of the same class and derived from the same search space, rather than on a comparison between different model classes. Moreover, the trade-off is specific to our definition of interpretability and how it is intertwined with performance on the selected datasets.

Our proposed introspectability metric is best suited for technical users, as discussed in Section 3.2, and should not be confused for an all-telling measure of fairness, trust, or reliability. However, the introspectability of models discovered by `XNAS` has the potential to serve as a requisite criterion before being deployed to users or trusted as a valid model. Furthermore, the metric is designed with simplicity and generality in mind. It can be gamed by an adversary with model-level access, e.g. by inserting futile blocks that are zeroed out and bypassed with a skip connection. This should be addressed in later work, ideally on a per-application basis.

There are several routes for improvement of the `XNAS` framework. Foremost, the efficiency can be increased by using weight-sharing techniques (Elsken et al., 2019b; Lu et al., 2019), which reduce the evaluation time of offspring. Furthermore, there are uninteresting regions of the Pareto front, depending on the application or end user – NSGA-II can be modified to use reference points of interest to guide the multi-objective search towards more desirable solutions (Deb and Sundar, 2006).

6 Broader Impact Statement

Our work aims to bridge the gap between neural architecture search (NAS) and explainable AI (XAI) – by optimizing for both task performance and a surrogate objective for model debuggability, we discover performant architectures that enable some aspects of interpretability. Our framework is a proof of concept of how these two subfields can work together. Several new applications stem naturally. Additional objectives can be utilized from the XAI literature in multi-objective search, even in combination. Existing NAS techniques can be applied to discovering novel architectures conducive to the goals in XAI research. While our approach has potential for broad impact, the limitations of the surrogate metric should be noted. As the text details, the surrogate is not useful to all types of stakeholders nor all aspects of interpretability. Introspectability is particularly useful for data scientists and researchers who understand neural networks at a technical level. The nature of interpretability concerns model trustworthiness, learned generalizations, and understanding the latent space. The surrogate has utility for these stakeholders seeking this type of interpretability. However, potential harm can arise from inappropriate use by users or use cases. As we call for in the text, additional interpretability objectives should be explored in future work for these other settings.

7 Submission Checklist

1. For all authors…

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The contributions stated in the abstract, as itemized in the introduction, are all demonstrated in the experiments and discussion sections.

   (b) Did you describe the limitations of your work? [Yes] Limitations are discussed in the discussion section.

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] This is addressed in both the broader impact statement as well as the discussion.

   (d) Have you read the ethics author's and review guidelines and ensured that your paper conforms to them? `https://automl.cc/ethics-accessibility/` [Yes] We have read and verified that our work complies with the guidelines.

2. If you are including theoretical results…

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments…

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] Code is provided with adequate instructions to reproduce the results.

   (b) Did you include the raw results of running the given instructions on the given code and data? [Yes] See the supplemental material and the code used to process such data.

   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [Yes] See the code in the supplemental material.

(d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [Yes] The scripts are well-documented.

(e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes] See the supplemental material.

(f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes] See experiments section and supplemental material for all details confirming that this is the case.

(g) Did you run ablation studies to assess the impact of different components of your approach? [Yes] See the experiments section and the supplemental material.

(h) Did you use the same evaluation protocol for the methods being compared? [Yes] The same NAS approach was applied for both single- and multi-objective runs.

(i) Did you compare performance over time? [Yes] See the supplemental material for the target metrics over time.

(j) Did you perform multiple runs of your experiments and report random seeds? [No] While the random seeds are reported, multiple runs were omitted as 1) this does not give our method an advantage over other approaches in the experimental setting, 2) the Pareto front is more robust to random noise than aggregate scalar metrics, 3) the variance of the solutions of the Pareto front is not clear to present, and 4) multiple runs for all settings is expensive given our current resources.

(k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] See previous response.

(l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [No] This is outside the scope of this paper.

(m) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the supplemental material for resources and compute times.

(n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [N/A] No hyperparameter tuning was performed.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets…

(a) If your work uses existing assets, did you cite the creators? [Yes] See the experiments section for citations of each dataset.

(b) Did you mention the license of the assets? [Yes] See the supplemental material.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code URL is in the main text (redacted for anonymity but public after publication).

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See the supplemental material.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See the supplemental material.

5. If you used crowdsourcing or conducted research with human subjects…

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

References

Adam, G. and Lorraine, J. (2019). Understanding neural architecture search techniques. *CoRR*, abs/1904.00438.

Angeline, P. J., Saunders, G. M., and Pollack, J. B. (1994). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans. Neural Networks*, 5(1):54–65.

Balaprakash, P., Salim, M. A., Uram, T. D., Vishwanath, V., and Wild, S. M. (2018). Deephyper: Asynchronous hyperparameter search for deep neural networks. In *25th IEEE International Conference on High Performance Computing, HiPC 2018, Bengaluru, India, December 17-20, 2018*, pages 42–51. IEEE.

Bergstra, J., Yamins, D., and Cox, D. D. (2013). 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, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 115–123. JMLR.org.

Blank, J. and Deb, K. (2020). pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deb, K. and Sundar, J. (2006). Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, page 635–642, New York, NY, USA. Association for Computing Machinery.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society.

Domhan, T., Springenberg, J. T., and Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In Yang, Q. and Wooldridge, M. J., editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 3460–3468. AAAI Press.

Dong, X. and Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Elsken, T., Metzen, J. H., and Hutter, F. (2019a). 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.

Elsken, T., Metzen, J. H., and Hutter, F. (2019b). Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017.

Fonseca, C. M., Paquete, L., and López-Ibáñez, M. (2006). An improved dimension-sweep algorithm for the hypervolume indicator. In *IEEE International Conference on Evolutionary Computation, CEC 2006, part of WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, pages 1157–1163. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision - ECCV 2016 - 14th European Conference*, volume 9908 of *Lecture Notes in Computer Science*, pages 630–645. Springer.

Hsu, C., Chang, S., Juan, D., Pan, J., Chen, Y., Wei, W., and Chang, S. (2018). MONAS: multi-objective neural architecture search using reinforcement learning. *CoRR*, abs/1806.10332.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y. D., Deb, K., Goodman, E. D., and Banzhaf, W. (2019). NSGA-Net: neural architecture search using multi-objective genetic algorithm. In Auger, A. and Stützle, T., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 419–427. ACM.

Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395.

Mendoza, H., Klein, A., Feurer, M., Springenberg, J. T., and Hutter, F. (2016). Towards automatically-tuned neural networks. In Hutter, F., Kotthoff, L., and Vanschoren, J., editors, *Proceedings of the 2016 Workshop on Automatic Machine Learning (AutoML)*, volume 64 of *JMLR Workshop and Conference Proceedings*, pages 58–65. JMLR.org.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Miller, G. F., Todd, P. M., and Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In Schaffer, J. D., editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 379–384. Morgan Kaufmann.

Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., and Stoica, I. (2018). Ray: A distributed framework for emerging AI applications. In Arpaci-Dusseau, A. C. and Voelker, G., editors, *13th USENIX Symposium on Operating Systems Design and Implementation*, pages 561–577. USENIX Association.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 4780–4789. AAAI Press.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., and Kurakin, A. (2017). Large-scale evolution of image classifiers. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2902–2911. PMLR.

Ru, B. X., Wan, X., Dong, X., and Osborne, M. A. (2021). Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Rudin, C. and Radin, J. (2019). Why are we using black box models in ai when we don't need to? a lesson from an explainable ai competition. *Harvard Data Science Review*, 1(2).

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

Zheng, X., Wang, P., Wang, Q., and Shi, Z. (2020). Disentangled neural architecture search. *CoRR*, abs/2009.13266.

Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms - A comparative case study. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H., editors, *Parallel Problem Solving from Nature - PPSN V, 5th International Conference*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–304. Springer.

Zoph, B. and Le, Q. V. (2017). 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.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710. Computer Vision Foundation / IEEE Computer Society.