Master Thesis

Uncertainty Calibration with Online Conformal Prediction in Neural Architecture Search: An Evaluation under the BANANAS Framework

Cheng Chen
(matriculation number 1662473)

July 31, 2025

Submitted to
Data and Web Science Group
Prof. Dr. Margret Keuper
University of Mannheim

Abstract

Some contents

Contents

List of Algorithms

List of Figures

List of Tables

1. Introduction

BANANAS with Conformal Prediction (BANANAS-CP)
Bayesian Optimization with Neural Architectures for Neural Architecture Search (BANANAS)
Neural Architecture Search (NAS)
Conformal Prediction (CP) Split Conformal Prediction (SCP) Conformal Prediction with Cross-validation (CrossVal-CP) Conformal Prediction with Bootstrapping (Bootstrap-CP)

1.1. Motivation

1.2. Related Work

1.3. Contributions and Limitations

1.4. Outline

Having gained an overview of the research question and the background, the remainder of this thesis is organized as follows. First, Chapter 2 reviews the related works on neural architecture search, uncertainty quantification, and in particular, conformal prediction. In Chapter 3, after proposing a novel framework to incorporate uncertainty calibration into the architecture search process in Section **??**, we describe its methodological steps in more detail. In Section **??**, we identify different types of conformal prediction algorithms that are applicable for NAS, and consider the use of the underlying surrogate models. In Section **??** and Section **??**, we further examine how the calibrated predictions can be applied in a Bayesian optimization process. In Chapter 4, we describe the benchmark dataset used for conducting experiments and comparing algorithm performance against the state-of-the-art techniques. In Chapter 5, we present the experiment setups and provide interpretations of the results. Finally, Chapter 6 concludes this work and discusses potential future work.

2. Literature Review

This chapter offers the technical background related to the research question of this work. We start by providing a comprehensive overview of NAS and introduce the three dimensions that characterize a NAS algorithm, followed by an anatomy of the high-performing search algorithm BANANAS. Then, we review the existing uncertainty quantification techniques, with a focus on CP algorithms, particularly those related to the novel framework we propose in Chapter **??**.

2.1. Neural Architecture Search

2.1.1. Background

In the recent decades, deep learning has achieved remarkable success in a variety of areas, including computer vision, natural language understanding, and machine translation. This success is partly attributed to the meticulously hand-crafted neural network architectures. With the rising demand for efficient architecture engineering in complex domains, NAS has emerged as a technique for automating the design of neural architectures for specific tasks.

NAS has been a rapidly progressing research domain in the past years. Since the seminal work that achieves competitive performance on CIFAR-10 [28], numerous NAS algorithms built on different techniques have been proposed. In general, NAS algorithms can be characterized by three key dimensions: search space, search strategy, and performance evaluation strategy [7, 24, 25]. Figure 2.1 illustrates a typical architetcture search process.

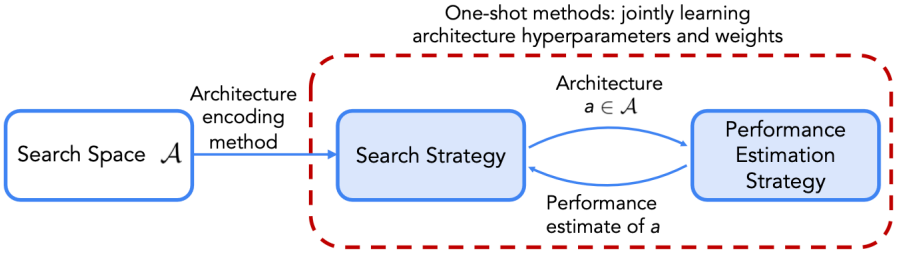

Figure 2.1: Overview of an architecture search process. The search strategy iteratively selects architectures from a predefined search space $\mathcal{A}$. The performance estimation strategy evaluates the model performance on the target dataset and returns the performance to the search strategy.

Next, we provide definitions of the terms and review the research progress of each domain.

Search Space A search space defines a set of architectures that the search algorithm is allowed to select. The search space is often the first step when setting up NAS and perhaps is also the most essential step, because the design of the search space represents an important trade-off between human bias and efficiency of search: a smaller search space incorporating more prior human knowledge and involving more manual decisions will enable NAS algorithms to find high-performing architectures more easily, in contrast a larger space with more primitive building blocks provides higher odds of discovering truly novel architectures [24]. Common search spaces range in size from a few thousand to over 10^{20}.

There are four major categories of search spaces in the NAS literature [24]. We start with two types of search spaces that have relatively simple architecture topologies. The macro search spaces [1, 10, 28] encode the entire neural architecture at a high level. Typically, an entire architecture is often represented by a Directed Acyclic Graph (DAG), with nodes defining the operation types and edges representing data flows. Each node is allowed to have distinct structures, such as convolution, pooling. As a result, macro search spaces are highly flexible and possess high representation power. Another type is the chain-structured search spaces. As suggested by the name, chain-structured search spaces consist of neural networks that can be written as a sequence of operation layers. These search spaces often take state-of-the-art manual designs as the backbone. For example, there are several chain-structured search spaces based on the convolutional networks [3] or the transformer architectures [26].

The third group is the cell-based search spaces, which perhaps are the most popular type of search spaces in NAS research. The cell-based search spaces are inspired by the fact that state-of-the-art human-designed architectures often consist of repeated blocks. For instance, the high-performing Transformer [21] contains 6 identical stacked encoder and decoder layers. Thus, instead of searching for the entire network architecture from scratch, [29] propose to only search over relatively small cells, and stack the cells according to a predefined skeleton to form the overall architecture. Building on this idea, [29] proposes the first modern cell-based search space, NASNet, which comprises of two types of cells: the normal cell that preserves the dimensionality and the reduction cell that reduces the spatial dimension, as illustrated in Figure 2.2. Since its emergence, many other cell-based search spaces have been developed. In general, these cell search spaces share a high-level similarity, but differ in the design of the fixed macro structure, the layout and constraints in the cells, and the choices of operations within the cells [6, 13, 27]. The cell-based approach significantly reduces the size and the complexity of the search space. However, it has been criticized for limiting the expressiveness of NAS, potentially hindering the discovery of highly novel architectures [24].

The last main category is the hierarchical search spaces. Different from the aforementioned types of search spaces that mostly have a flat representation, hierarchical search spaces involve designing patterns at different levels, where each higher-level pattern is often represented as a DAG of lower-level patterns [5, 12].

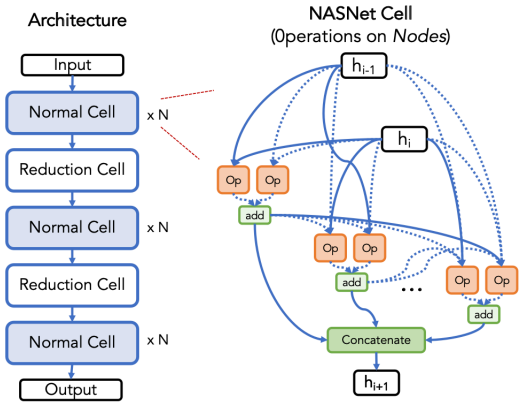

Figure 2.2: Overview of a cell-based search space NasNet. The outer skeleton across cells (left) is fixed, and the operations, represented by nodes, within the cells are searchable (right) [24].

In addition to the architecture topology, another important design accompanying a search space is the architecture encodings, because many NAS algorithms require representations of the architectures to, for example, mutate an architecture or train a predictive model to extrapolate its performance. For search spaces that can be represented by a DAG, adjacency matrix is a commonly used encoding method. In addition, other encoding techniques, including graph-based encoding [17], path-based encoding [23] and conditionally-structured encoding methods tailored for hierarchical search spaces have been proposed. [22] has shown that the effect of the encoding methods varies across different NAS subroutines.

Search Strategy According to [24], there are generally two main categories of search strategies: black-box optimization based techniques and one-shot techniques.

The black-box optimization based techniques largely overlap with another sub-area of AutoML: the hyperparameter tuning. Common techniques for hyperparameter tuning have been proven to be efficient for NAS as well, including reinforcement learning [28, 29], evolutionary algorithms [15, 19], gradient descent[13], and etc. In particular, we take a close look at the search strategies based on Bayesian optimization, since they are closely related to the research question of this work. Specifically, initial Bayesian optimization based approaches typically use the Gaussian Process (GP) as the surrogate model [10]. However, these algorithms often demonstrate under-performance compared to their competitors due to several limitations: 1) search spaces are usually high-dimensional, non-continuous, and graph-like; 2) GPs requires custom distance metrics among architectures, which involves a time-consuming matrix inversion step. Besides, GPs are difficult to scale since the computation complexity grows cubically with the number of observations. To address these challenges, a new framework that applies Bayesian optimization with a neural predictor has been proposed and demonstrated strong performance [14, 20, 23]. We review this framework in details in Section 2.1.2.

The one-shot techniques are introduced to avoid training each architecture from scratch, The key idea is to train a *supernetwork* that comprises all possible architectures in the search space as subnetworks. Once a supernet is trained, each architecture from the search space can be evaluated by inheriting the weights from the corresponding subnet within the supernet [2, 13].

Performance Evaluation The performance evaluation refers to the process of estimating the performance of architectures. The estimated performance is communicated back to the search algorithm to guide the next search. The simplest performance estimation strategy is to fully train an architecture on the training data and then evaluate its performance on the validation data. However, training each architecture demands substantial computation resources and typically takes serval hours or days on a GPU. Consequently, many methods for speeding up the performance evaluation process for architectures have been proposed. One popular line of work is to predict the performance of neural networks before they are fully trained using the zero-cost proxies [16].

In this work, we mainly run experiments on the benchmark dataset NAS-Bench-201 [6], which offers queryable validation and test accuracies for all architectures in the search space and eliminates the need to train neural networks when simulating NAS experiments. Therefore, we provide only a brief overview of this aspect and refer the readers to [24] for a comprehensive introduction to performance evaluation techniques.

2.1.2. BANANAS

includling.... BANANAs, a BO + poricter based strong performance..

BANANAS is an Bayesian optimization based search strategy.

We begin by an introduction of bayesian optimization... Give an introduction how Bayesian optimization works. We list the five engineering decisions and review each field's related works. Maybe briefly cite Gaussian Process.

Bayesian optimization is a sequential decision-making process that seeks to find a global minimum x argminxX f(x) of an unknown black-box objective function f : X R over an input space X RD.

Architecture Encoding

Neural Predictor Feedforward Neural Networks (FNNs)

Uncertainty Estimation

Acquisition Function The acquisition function trades off exploration and exploitation during the search. In the experiments, five commonly used acquisition functions are examined: Thompson Sampling (TS), Independent Thompson Sampling (ITS), Upper Confident Bound (UCB), Probability of Improvements (PI), and Expected Improvements (EI). Each function is adapted to the Gaussian assumption, thereby requiring only the mean and standard deviation estimates to compute the acquisition scores. In the experiments, overall ITS yields the best performance among all the options, although the

marginal outperformance is subtle. The results indicate that the acquisition function does not have as significant impact on the search performance as the other examined components in the framework.

Acquisition Optimization In each iteration of the Bayesian optimization, the goal is to select a candidate from the search space that maximizes the acquisition score. Evaluating the acquisition function for every architecture available in the search space is computationally infeasible, therefore [23] proposes to create a set of 100 to 1000 candidates and then choose the architecture with the maximal acquisition score in this set. Specifically, [23] explores various approaches for creating this candidate set. The simplest and most natural way is to draw architectures at random. Consider that architectures close in edit distance to those used for training the surrogate model are likely to have more accurate estimates, an alternative is a mutation-based sampling approach, where the candidate set is created via local search by randomly modifying an operation or an edge of the best-performing architectures that have been evaluated so far. In additional, [23] also examines a hybrid approach that combines random search with mutation-based search. Their experiments show that the mutation-based approach outperforms its competitors and suggest it is better to search locally rather than globally.

2.2. Uncertainty Quantification Methods

Understanding uncertainty is important for real-world application of artificial intelligence, e.g., in autonomous driving, medical diagnosis.

Aleatory Uncertainty (data uncertainty): uncertainty that arises due to inherent variations and randomness, and cannot be reduced by collecting more information

Epistemic Uncertainty (model uncertainty): uncertainty that arises due to lack of knowledge, and can be reduced by collecting more information.

- Bayesian-based: e.g., Bayesian Neural Network
- Ensemble-based: e.g., Monte-Carlo dropout
- Bootstrapping

But these techniques are limited in several perspectives. First, quantifying uncertainty requires training models for several times, which means that the models cannot be applied for real-time prediction or in an online-learning setup. Second, some models are pre-trained and are only accessible via API. Besides, models (pre-)trained on certain datasets may struggle to generalize across different domains or contexts.

2.3. Conformal Prediction

2.3.1. Theoretical Background

Starting from i.i.d data, and provide an intuitive demonstration how the prediction interval is constructed (can add a figure illustrating why conformal prediction works, i.e., symmetry). From the most intuitive expression to the finite-sample adjusted expression.

Notation Then, relax the i.i.d assumption to exchangeability, and lay a formal definition of the conformal prediction. And list the most importance three ingredients of the conformal predictions.
- A trained predictor f
- A conformity score function s. The conformity score is an important engineering decision and has an impact on the size on the prediction set, i.e., the efficiency. The conformity score function can be either a negatively- or positively oriented, in which ... And it can be a random variable as well.
- A target coverage alpha

Marginal coverage is guaranteed regardless of the choices in dataset and black box model. Only the model predictions are required to apply the technique.

A Link to Statistical Testing (clarify the relationship between conformal prediction and hypothesis testing) In this video (22:21), it is explained the intuition why conformal prediction guarantees the coverage, which is quite similar to the spirit of hypothesis testing.

CP	**Hypothesis Testing**
(desired) Coverage level	Confidence level
Nominal error level (1 - Coverage level)	Significance level
The conformity score of the new instance	p-value (is an empirical term)

The coverage parameters which should be pre-set plays a similar role as the confidence interval in hypothesis testing. Conformal prediction is like hypothesis testing with hypotheses:
H0: test instance i conforms to the training instances.
H1: test instance i does not conform to the training instances.

2.3.2. Full Conformal Prediction

Full Conformal Prediction (FCP)

2.3.3. Extensions of Conformal Prediction

Since the transductive version of CP was first proposed in [8], several variants have been developed with different computational complexities, formal guarantees, and practical applications.

To address the aforementioned inefficient computation problem of FCP, Split Conformal Prediction (SCP), also known as Inductive Conformal Prediction (ICP), was first

introduced in [18] by replacing the transductive inference with inductive inference. aims to learn a general prediction rule about the data using the observed records. Then, this rule can be applied directly to obtain predictions when new data arrives in sequence, without re-using the training data and retraining the model repeatedly. The main concept involves splitting the data into two non-overlapping subsets, designated for training and calibration, respectively. A predictive model is fit exclusively on the training set, then non-conformity measures are computed on the calibration set to determine the prediction interval's width. Due to its simplicity and computational efficiency, SCP is one of the most commonly used techniques in the CP family. We delve into methodological steps of SCP with pseudo-code in Section **??**.

Limitations of split conformal predictions: - Distribution shift. The conformal prediction is built on the core assumption of exchangeability, which means the data points are identically distributed. However, this assumption is hard to meet in real-world application. For example, with time-series data this assumption is generally violated due to the temporal relationships. - Adaptivity. Once the conformity scores are computed on the calibration set, the decision threshold is settled and is applied to all test datapoints, regardless of the intrinsic complexity of the exact example. It is desirable that the threshold can adapts to the difficulty of the problem and produce a larger prediction interval/set on hard-to-solve example and smaller prediction interval/set on easy-to-solve example. This limitation echoes with the characteristic of Conformal Prediction that the guaranteed coverage is only marginal over all datapoints but not conditional on a specific data points..

Variations of Conformal predictions have been proposed to overcome the limitations. There are three main streams: - find an empirical coverage rate which leads to the desired coverage level. For example, if the desired coverage rate is 90- find an efficient conformity score: Alternatively, [. . .] apply the conformal prediction in an online setting to dynamically incorporate the conformity score of new data points. - find suitable predictor: The trained predictor can be just a poor approximation of the real data generation process.

Besides, [. . .] proposes a CP algorithm that samples datapoints using Monte-Carlo sampling to approach the real distribution of labels in case the ground-truth is ambiguous and consequently cause a biased distribution in manually-annotated labels.

3. Dataset

To compare the performance of BANANAS–CP with the original BANANAS algorithms and assess the role of uncertainty calibration, we run experiments on the widely used tabular benchmark dataset NAS-Bench-201 [6] [1].

NAS-Bench-201 is a cell-based architecture search space. Each cell is expressed as a densely connected DAG with in total 4 nodes and 6 edges. The nodes within a cell represents the sum of all feature maps transformed through the associated operations of the edges pointing to this node, and the edges represent the architectures operation that are chosen from the predefined operation set. Specifically, the operation set comprises 5 representative operations: 1) zeroize, 2) skip connection, (3) 1-by-1 convolution, (4) 3-by-3 convolution, and (5) 3-by-3 average pooling layer. This search space contains all possible architectures generated by 4 nodes and 5 associated operation options, which results in 15,625 cell candidates in total. The macro structure of an architecture is defined as a chain of blocks, which is initiated with one 3-by-3 convolution with 16 output channels and a batch normalization layer, and consists of three stacks of cells that are connected by a residual layer. Figure 3.1 illustartes the structure of an architecture in this search space.

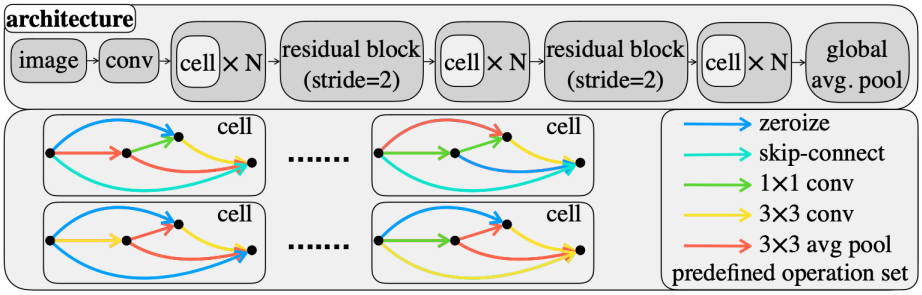

Figure 3.1: Illustration of the skeleton (top) and the design of individual cells (bottom) of architectures in NAS-Bench-201 [6].

Architectures in the search space are evaluated on three datasets that are widely used for image classification tasks: CIFAR-10, CIFAR-100 [11] and ImageNet-16-120 [4]. Each dataset is split into the training, validation, and test sets and NAS-Bench-201 provides the training, validation, and test losses as well as accuracies for all architectures in the search space. We give a brief introduction to these datasets:

[1] In this work, the dataset for running experiments are downloaded from the NASLib repository: https://github.com/automl/NASLib/tree/Develop

CIFAR10 : The dataset consists of 60K 32×32 color images in 10 classes. In NAS-Bench-201, 25K images with 10 classes are assigned into the training and the validation sets, respectively. The test set contains 10K images, with 1K images per class.

CIFAR100 : This dataset has the same images as CIFAR-10 but in 100 classes. The training set has 50K images, and each of the validation and the test sets has 5K images.

ImageNet-16-120 : This dataset contains 151.7K training images, 3K validation images, and 3K test images with 120 classes. Each image has 16×16 pixels.

NAS-Bench-201 has contributed to the NAS community in several aspects. First, it offers assess to precomputed performance metrics for all architectures, allowing subsequent research to focus on the core search algorithm. With a unified dataset splitting strategy, it also enables consistent comparisons across different NAS algorithms. Additionally, NAS-Bench-201 serves as a foundation for extending benchmark datasets. For example, [9] introduces a dataset for benchmarking the robustness of neural networks.

In this work, we leverage the API offered by NAS-Bench-201 and directly query the performance metrics of the pre-trained architectures. In addition, NAS-Bench-201 also provides several analytical metrics, e.g., model ranking and accuracy correlations across the three datasets, which further guides our post-hoc analysis of the observed results.

4. Experiments and Results

4.1. Setup and Implementation

..declare which codes are from the NASlib library. Validation accuracy is a commonly used supervision signal for NAS.

4.2. Baseline

5. Conclusion

This chapter presents the central findings of this work as well as their critical discussion. Finally, it highlights limitations and corresponding opportunities for further research.

Bibliography

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 550–559. PMLR, July 2018.

[3] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.

[4] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the CIFAR datasets. *arXiv preprint arXiv:1707.08819*, 2017. 36,000 3232 ImageNet images over 1,000 classes (also variants at 1616 and 6464).

[5] Aristeidis Christoforidis, George Kyriakides, and Konstantinos Margaritis. A novel evolutionary algorithm for hierarchical neural architecture search. arXiv preprint arXiv:2107.08484, 2021. Published July 18, 2021.

[6] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

[7] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. Published online August 16, 2018.

[8] Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 148–155, Madison, WI, USA, 1998. Morgan Kaufmann.

[9] Steffen Jung, Jovita Lukasik, and Margret Keuper. Neural architecture design and robustness: A dataset. In *International Conference on Learning Representations (Poster Track)*, 2023.

[10] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabás Póczos, and Eric Xing. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems*, pages 2020–2029, 2018.

[11] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 2009-TR-XXX, University of Toronto.

[12] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations (Poster)*, April 2018. Originally published on arXiv Nov 1, 2017 as arXiv:1711.00436.

[13] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018. Published 2019 at ICLR.

[14] Lizheng Ma, Jiaxu Cui, and Bo Yang. Deep neural architecture search with deep graph bayesian optimization. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 500–507, IEEE, 2019. arXiv preprint arXiv:1905.06159.

[15] Krzysztof Maziarz, Mingxing Tan, Andrey Khorlin, Marin Georgiev, and Andrea Gesmundo. Evolutionary-neural hybrid agents for architecture search. *CoRR*, abs/1811.09828, 2018.

[16] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. Neural architecture search without training. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7588–7598. PMLR, July 18–24 2021.

[17] Xuefei Ning, Yin Zheng, Tianchen Zhao, Yu Wang, and Hua-Zhong Yang. A generic graph-based neural architecture encoding scheme for predictor-based nas. In *Computer Vision – ECCV 2020*, volume 12358, pages 189–204. Springer, 2020.

[18] Harris Papadopoulos, Kostas Proedrou, Vladimir Vovk, and Alexander Gammerman. Inductive confidence machines for regression. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Machine Learning: ECML 2002*, volume 2430 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2002.

[19] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, pages 4780–4789, Honolulu, Hawaii, USA, 2019. AAAI Press.

[20] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *Advances in Neural Information Processing Systems*, volume 29 of *NeurIPS*, pages 4134–4142, Barcelona, Spain, 2016.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[22] Colin White, Willie Neiswanger, Sam Nolen, and Yash Savani. A study on encodings for neural architecture search. In *Advances in Neural Information Processing Systems*. NeurIPS, 2020. Originally released as arXiv:2007.04965 in July 2020.

[23] Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[24] Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers. *CoRR*, abs/2301.08727, 2023.

[25] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. A survey on neural architecture search. *CoRR*, abs/1905.01392, 2019.

[26] Jin Xu, Xu Tan, Renqian Luo, Kaitao Song, Jian Li, Tao Qin, and Tie-Yan Liu. Nas-bert: Task-agnostic and adaptive-size bert compression with neural architecture search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1933–1943, 2021.

[27] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-bench-101: Towards reproducible neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7105–7114. PMLR, June 2019.

[28] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[29] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710, 2018.

Acronyms

BANANAS	Bayesian Optimization with Neural Architectures for Neural Architecture Search
BANANAS-CP	BANANAS with Conformal Prediction
Bootstrap-CP	Conformal Prediction with Bootstrapping
CP	Conformal Prediction
CrossVal-CP	Conformal Prediction with Cross-validation
DAG	Directed Acyclic Graph
EI	Expected Improvements
FCP	Full Conformal Prediction
FNNs	Feedforward Neural Networks
ITS	Independent Thompson Sampling
NAS	Neural Architecture Search
PI	Probability of Improvements
SCP	Split Conformal Prediction
TS	Thompson Sampling
UCB	Upper Confident Bound

A. Program Code and Data Resources

The source code and a documentation are available at the GitHub repository: https://github.com/chengc823/Thesis. The datasets used for experiments and algorithm evaluations are sourced from the NASLib repository.

In case of access or permission issues to the private repository, please reach out at: chechen@mail.uni-mannheim.de.

B. Additional Experimental Results

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Bachelor-, Master-, Seminar-, oder Projektarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und in der untenstehenden Tabelle angegebenen Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Declaration of Used AI Tools

Tool	Purpose	Where?	Useful?
ChatGPT	Rephrasing	Throughout	+
ChatGPT	Debugging LaTeX syntax errors	Equation	+
ChatGPT	Rendering LaTeX tables from Python frame	Tables	+

Unterschrift
Mannheim, den 31.07.2025