ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230423

Neural Architecture Search for Explainable Networks

Yaoman Li[a;*] **and Irwin King**[a;**]

[a]The Chinese University of Hong Kong
ORCiD ID: Yaoman Li https://orcid.org/0000-0003-1191-2227, Irwin King https://orcid.org/0000-0001-8106-6447

Abstract. One of the main challenges in machine learning is providing understandable explanations for complex models. Despite outperforming humans in many tasks, machine learning models are often treated as black boxes that are difficult to interpret. Post-hoc explanation methods have been developed to create interpretable surrogate models that explain the behavior of black-box models. However, these methods have been shown to perpetuate bad practices and lack stability. Recently, inherent explainable approaches have been proposed to provide built-in explainability to models. However, most of these methods sacrifice performance. This paper proposes the Neural Architecture Search for Explainable Networks (NASXNet) approach to address the trade-off between performance and interpretability. Our method applies architecture search to generate high-performance and explainable neural networks for image classification tasks. We conduct experiments on four datasets: CUB-200-2011, Stanford Cars, CIFAR 10, and CIFAR 100. The results demonstrate that our models provide a high-level interpretation of prediction results, achieving state-of-the-art performance that is on par with non-explainable models. This paper contributes by solving the trade-off problem between performance and interpretability. It is the first to apply neural architecture search to develop explainable deep learning models, generating state-of-the-art explainable models that outperform existing approaches. Additionally, a new training process is proposed that enables faster convergence during model training.

1 Introduction

Deep learning has been successful in many applications, significantly impacting human lives. However, most deep learning models are black boxes that lack transparency and do not explain their results in a human-understandable way. This lack of accountability and transparency has the potential to cause reliability issues and severe consequences [31]. Therefore, explaining the deep learning model has become increasingly important.

Prior work in image processing problems has focused primarily on improving accuracy. Since the groundbreaking success of AlexNet [16] in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [32], increasingly complex deep learning models have been proposed. For instance, the VGG-16 [36], a powerful convolutional neural network (CNN), comprises more than 130 million parameters, while ResNet-152 [9] consists of 60 million parameters.

Designing a good deep learning model requires substantial time and resources during the trial-and-error process due to the complexity involved. To automate the architecture engineering process, Neural Architecture Search (NAS) methods [10] have been proposed. Recent NAS methods have achieved state-of-the-art results and outperformed handcrafted deep learning models in many tasks, such as image classification [44, 24, 40], image segmentation [21], and language modeling [11, 13]. However, deep learning models still lack transparency and interpretability, limiting their application in many critical decision-making tasks. The European Union's General Data Protection Regulation requires the "right to an explanation" for algorithms in individual decision-making, underscoring the urgency of transparency and interpretability in machine learning [8].

To better understand how deep learning models make predictions, post-hoc methods have been proposed. These methods aim to fit an explanation to the black-box models. Examples of deep learning post-hoc methods include deconvolution [42], activation maximization [35], and saliency visualization [34, 38]. However, these methods cannot explain the reasoning process and why the model makes such predictions. Since the explanations are not based on real reasoning processes, they can sometimes be unreliable and misleading.

To address these problems in post-hoc methods, built-in prototypes methods have been proposed [4, 27, 5, 33]. This method mimics how humans identify objects. For example, when we describe why an image looks like a clay-colored sparrow, we compare the bird's wing and head with some prototypical clay-colored sparrow images. Adding the prototype layers to models can provide built-in interpretability about the reasoning process. However, this method can sacrifice performance in many image classification tasks, making it difficult to achieve state-of-the-art results.

This work proposes the Neural Architecture Search for Explainable Networks (NASXNet) architecture, which generates high-performance and explainable models for image classification tasks. The motivation is to obtain high-performance and interpretable models, addressing two research problems.

Problem 1: *How to improve the performance of explainable models?* Some methods create built-in explainable models by adding prototype layers to existing neural networks. They do not redesign the prototype's backbone or feature extraction neural networks. In most cases, the decrease in performance is because the backbone network cannot effectively extract proper features for the prototype layer. Thus, this work proposes applying a NAS method to search for new convolutional neural networks (CNNs) to perform the feature extraction for the prototype layers.

Problem 2: *How to maintain interpretability and reduce training complexity?* By adding the NAS method, the model needs to learn both the neural architecture and prototypes at the same time, making it difficult to train. This work proposes dividing the training process into three stages and a self-adjustment for the number of prototypes

* Email: ymli@cse.cuhk.edu.hk.
** Email: king@cse.cuhk.edu.hk.

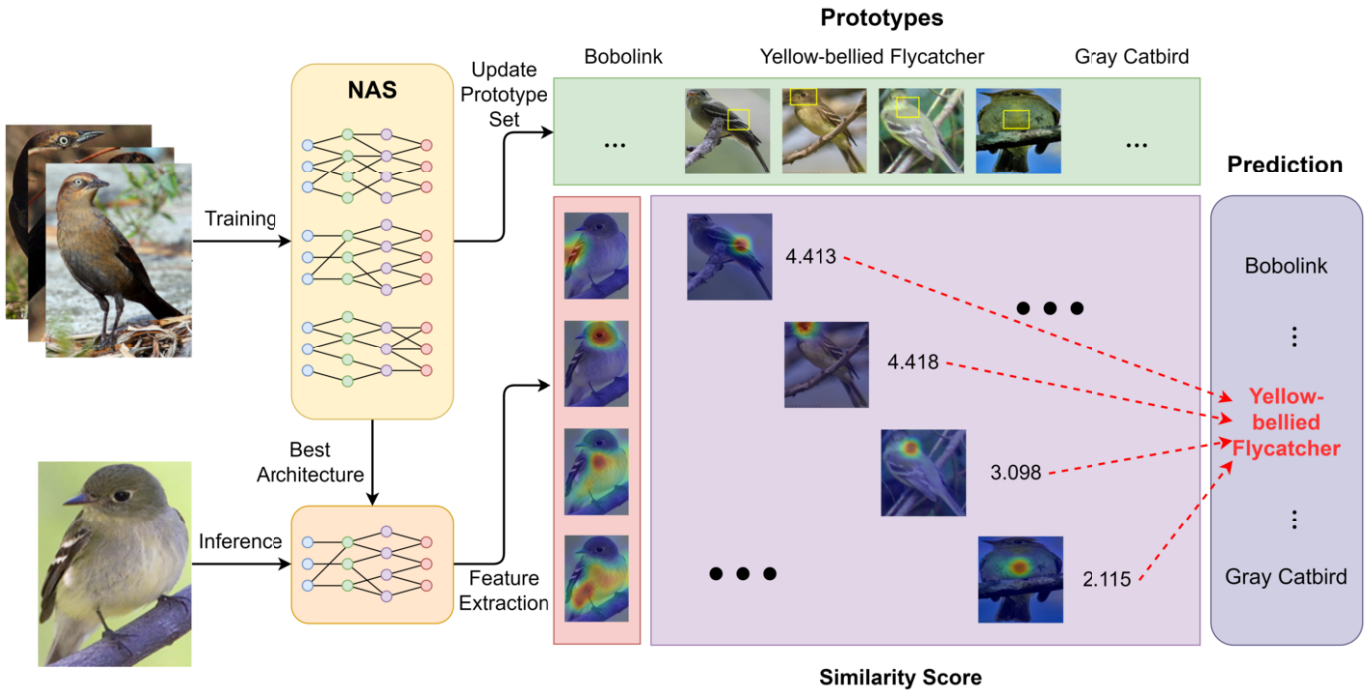


Figure 1: NASXNet prediction example for CUB-200-2011. The model would do self-adjust during training. When making predictions, each patch of the image would calculate the similarity scores with the prototypes which are auto-generated during the training. The similarity scores would pass to the linear layer to calculate the final prediction result. The sample shows the parches and prototypes of the top 4 similarity scores.

for each class, allowing the model to automatically learn the prototype distribution.

By addressing these problems, the proposed NASXNet can obtain state-of-the-art performance and a high level of interpretability. The contributions of this paper are threefold:

- To the best of the authors' knowledge, this is the first application of neural architecture to search for explainable deep learning models.
- The method can generate a state-of-the-art explainable model that outperforms existing approaches.
- A novel training process for the explainable model is proposed to enable faster convergence during training.

In the following sections, the related work of neural architecture search and prototypical network is introduced. Next, the architecture and training method are presented. The experiments and results are analyzed. Finally, the limitations and future improvement of NASXNet are discussed.

2 Related Work

We first do a brief review on current NAS works in image classification and then introduce current explanation methods, which can be divided into two categories, one is the post-hoc explanation [17, 25], another is the inherent explanation [20, 12, 5].

Neural Architecture Search. NAS achieves promising results in various deep learning tasks [7, 26]. It automates the architecture engineering process of neural networks. The NAS problem can be formally defined as follows. Given a search space $\mathcal{A}$, the target of our algorithm is to search the optimal neural network architecture $\alpha \in \mathcal{A}$ which minimizes the validation loss $\mathcal{L}_{val}$. It can be written as:

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}(w^*(\alpha), \alpha), \\ \text{s.t.} \quad & w^*(\alpha) = \operatorname{argmin}_w \mathcal{L}_{train}(w, \alpha), \end{aligned} \tag{1}$$

where w^* denotes the optimal parameters learned for the architecture in the training set. This is a bilevel optimization problem.

Direct computation of the gradient of $\mathcal{L}_{val}$ is almost impossible due to the high dimensionality of the model parameters w. Thus we calculate the validation loss by following approximation scheme [23]:

$$\begin{aligned} &\nabla_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha) \approx \\ &\nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha). \end{aligned} \tag{2}$$

Prior works mostly belongs to reinforcement learning category [2, 44, 45] and evolutionary algorithm category [37, 22, 29]. Due to the expensive cost of these methods, various works have been proposed to reduce the search costs [6, 28, 41]. DARTS [23, 43] is a differentiable architecture search method which relax the discrete neural architecture search problem to a continuous one and solve it by gradient descent, i.e., for each operation $o(\cdot)$ in the operation set $\mathcal{O}$, we can relax the choice of a particular operation to a softmax:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x), \tag{3}$$

where $\alpha^{(i,j)}$ is the operation of mixing weights for a pair of nodes (i, j). DARTS allows us to search for high-performance neural networks much faster than discrete search methods. Though NAS can generate state-of-the-art deep learning models which outperform handcrafted models, these generated models are still black boxes.

Post-hoc Explanation. Post-hoc explanations of black-box models are widely used for complex models such as deep neural networks. Many methods have been proposed [25, 30] recently. The

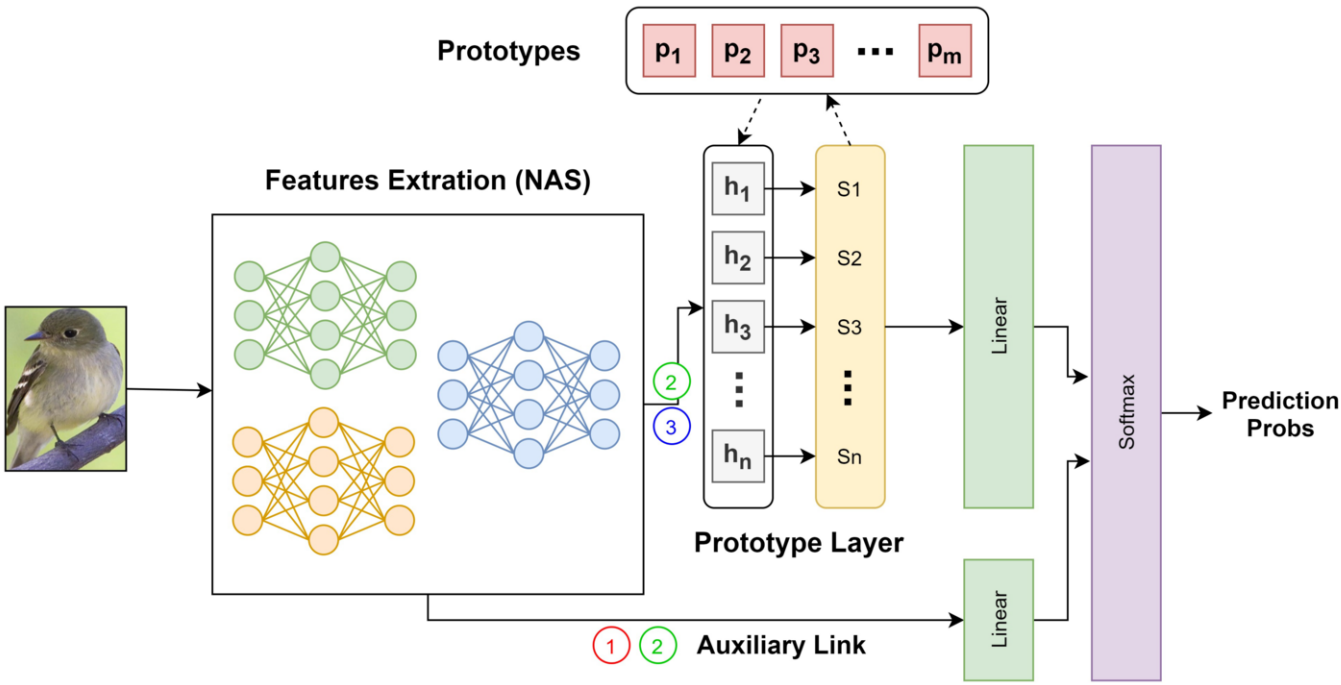


Figure 2: NASXNet Architecture. It consists of a Feature Extraction Network and Classification Layers. Classification layers contain two paths, one is the auxiliary (traditional) classification path, another is the prototype classification path. In the first stage of training, we only use the auxiliary classification path. In the second stage of training, the both auxiliary classification path and prototype classification path would be activated. In the third stage of training, we only use the prototype classification path to do the final model fine-tuning.

basic idea of post-hoc approaches is to build a surrogate model to explain the behavior of the original one. Based on the scope of approximation, they can be divided into local and global explanations methods. For instance, some local explanation methods compute saliency maps to capture feature importance for an individual prediction by computing gradient with respect to the input [35, 38], some global methods provide an explainable model to summarize the whole black box [19, 3]. However, recent works on post-hoc explanations show that black-box explanations can be unstable and unreliable [1, 18, 31].

Inherent Explanation. Since post-hoc approaches cannot explain the original model precisely [31], build-in interpretability is important. Recent works of [4, 27] define a new form of interpretability for image processing and sequence learning. The basic idea is inspired by the way humans describe their thinking in classification tasks.

ProtoPNet [4] compares image parts to learned prototypes and based on the similarity to classify the class of the new example. The model first extracts several parts of an image and then calculates each part's similarity scores to the prototypes. Thus, the model can interpret how they classify the image based on the classes of the prototypes.

The ProtoPNet consists of three components, the convolutional network f, the prototype layer g_P, and the fully connected layer h. The convolutional network uses convolutional layers from pre-trained models such as ResNet-34, VGG-16, VGG-19, DenseNet-121. Given an input image $\mathbf{x}$, the convolutional network extracts useful features $\mathbf{z} = f(\mathbf{x})$, whose shape is $H \times W \times D$. The prototype layer learns m prototypes $\mathbf{P} = \{\mathbf{p}_j\}_{j=1}^{m}$. Each prototype represents a prototypical part of an image in one class. Its shape is $H_1 \times W_1 \times D$, where $H_1 < H$, $W_1 < W$. Given an extracted feature $\mathbf{z}$, the prototype layer would calculate the square L^2 distance between each prototype $\mathbf{p}_j$ with all patches in $\mathbf{z}$. Then the distance would convert to the similarity between $\mathbf{z}$ and prototype $\mathbf{p}_j$. The similarity score represents how similar the image is to the prototype. The number of prototypes m_k for each class $k \in 1, ..., K$ is pre-determined. Lastly, the fully connected layer would calculate the output logits based on the similarity scores.

Though ProtoPNet can provide a new level of build-in interpretability for image classification, it cannot achieve state-of-the-art results in many tasks. Our NASXNet method would solve this problem by searching for a new convolutional network for prototype feature extraction and allowing self-adjustment of the prototypes' distribution. It can achieve better performance without losing interpretability.

3 Method

In this section, we first introduce our architecture and key components. Then we show our training algorithm and loss function. Finally, we describe the method and implementation details of each component.

3.1 Overall Architecture

Our NASXNet architecture can be divided into two parts, feature extraction and classification. Feature extraction is a neural network generated by our NAS method. The classification part contains two paths, one is the auxiliary (traditional) classification path, another is the prototype classification path. The whole architecture is shown in Figure 2.

The feature extraction network extracts image patch features to compare with prototypes. The auxiliary classification path is only used in our training process's first and second stages. It helps the

NAS algorithm to adapt to the classification task and generate high-quality architecture more efficiently. The prototype layer contains m prototypes, which are automatically learned from training data. For each instance, we calculate the similarity scores to the prototypes and based on the scores to classify the instance. This path would be used in the second and third stages of the training process.

After the architecture search, we would report the best-performing architecture. Thus we could obtain our final feature extraction network. Then we retrain and evaluate the final explainable network. Since we do not need to do NAS in the generated explainable network, this final model only needs to apply the second and third stage training.

3.2 Training Process

Our training process consists of three stages: (1) Architecture search adaptation. (2) Joint training of the auxiliary classification and prototype classification paths. (3) Fine-tuning the final model. We would first apply the three-stage training process to obtain the feature extraction neural network. Then we fix the feature extraction network with the best architecture generated by NAS and retrain the final model. When retraining the final model, it does not need the adaptation stage.

NAS Adaptation. At the beginning of the training process, we need to warm up the architecture search. In this stage, the prototype layers are not used. We only train the feature extraction network and auxiliary layer. This stage would help our architecture search algorithm converge faster and generate reasonable architectures.

Joint training. After the NAS adaptation stage, we would add the prototype layers to the training process. We joint train feature extraction network and two classification paths in this stage. The output of the final prediction is the weighted sum of the auxiliary layer and prototype layer output. This joint training stage would allow the prototype layer to make the classification based on the prototype similarity scores and let the architecture search algorithm start to learn architecture that is helpful to prototype feature extraction.

Fine-tuning. The final fine-tuning stage only trains the feature extraction network and prototype layers. The auxiliary classification path is removed. In this stage, our prediction only relies on the prototype similarity scores. This allows the algorithm to fine-tune the feature extraction network architecture and prototypes.

3.3 Loss Function

In the NAS adaptation stage, our goal is to learn an initial feature extraction network. The training loss is

$$\mathcal{L}_{train} = \frac{1}{n}\sum_{i=1}^{n} \text{CrsEnt}(h1 \circ f(\mathbf{x}_i), \mathbf{y}_i), \tag{4}$$

where CrsEnt represents cross entropy loss, $h1$ is the auxiliary classification layer, f is feature extraction network, $\mathbf{x}_i$ is the i-th input image, $\mathbf{y}_i$ is i-th label. The validation loss optimization is based on Equation (1).

In the joint training stage, we would combine the prototype classification path with auxiliary classification path. Thus the loss would be

$$\begin{aligned}\mathcal{L}_{train} =& \frac{1}{n}\sum_{i=1}^{n} \text{CrsEnt}(h \circ g_{\mathbf{p}} \circ f(\mathbf{x}_i), \mathbf{y}_i) \\ &+ \lambda_1 \frac{1}{n}\sum_{i=1}^{n} \text{CrsEnt}(h1 \circ f(\mathbf{x}_i), \mathbf{y}_i) \\ &+ \lambda_2 \text{Sep},\end{aligned} \tag{5}$$

where $g_{\mathbf{p}}$ is the prototype layer, h is the linear layer, Sep is defined by

$$\text{Sep} = -\frac{1}{n}\sum_{i=1}^{n} \min_{j:j\neq i} \|\mathbf{p}_i - \mathbf{p}_j\|_2^2. \tag{6}$$

The separation cost (Sep) encourages the model to obtain more various prototypes.

In the fine-tuning stage, the auxiliary path is removed, we only need to consider the prototype classification output, so the loss is:

$$\mathcal{L}_{train} = \frac{1}{n}\sum_{i=1}^{n} \text{CrsEnt}(h \circ g_{\mathbf{p}} \circ f(\mathbf{x}_i), \mathbf{y}_i) + \lambda_2 \text{Sep}. \tag{7}$$

These losses could help our model to focus on different parts at each stage. Such that the training could converge faster.

3.4 Architecture Search

To generate feature extraction networks efficiently, we apply a differentiable NAS method to search for the neural architecture. We optimize the cell structures which are stacked to define full network architecture. Following the same setting of DARTS, we search for two types of cells, i.e., normal cell and reduction cell. Each cell contains two input nodes, one output node and a set of intermediate nodes. Each node in the cell represents a feature map $x^{(j)}$. The operation for each node can be chose by computing the a mixture of candidate operations, i.e., $x^{(j)} = \sum_{i<j}\sum_{o\in\mathcal{O}} \frac{exp(\alpha_o^{i,j})}{\sum_{o'\in\mathcal{O}} exp(\alpha_{o'}^{i,j})} o(x^{(i)})$, where $x^{(i)}$ are the predecessor nodes, $\mathcal{O}$ denotes the set of candidate operations, e.g., 3×3 convolution, 3×3 max pooling, skip connection, etc. Then we can optimize both the weight of the neural network and architecture parameters alternatively by gradient descent. Finally the operations are selected based on the weighting factor $\frac{\exp(\alpha_o^{i,j})}{\sum_{o'\in\mathcal{O}} \exp(\alpha_{o'}^{i,j})}$.

However, DARTS can be unstable and fail in some cases. Thus, we apply the adaptive DARTS algorithm in work [43], i.e., whenever the dominant eigenvalue in DARTS starts increasing rapidly, we increase the regularization strength.

3.5 Learning Prototypes

At the beginning of the training, we first randomly initialize each prototype. Then we divide the prototypes $\mathbf{p}_i$ into K groups $\mathcal{G} = \{G_j\}_{j=1}^{K}$. Each group belongs to one class. We initialize the weight matrix W of last linear layer h as follows:

$$w_{i,j} = \begin{cases} 1 & \text{if } \mathbf{p}_i \in G_j \\ -0.5 & \text{if } \mathbf{p}_i \notin G_j \end{cases}. \tag{8}$$

For every t epochs of training, we project the nearest latent patch of images into the prototype $\mathbf{p}_j$, such that $\mathbf{p}_j$ can be viewed as a

Table 1: Classification results comparison. We apply the non-explainable methods as backbones in ProtoPNet. Though a better backbone can help increase the ProtoPNet performance, the backbones are not designed for prototype extraction. There is still a significant performance gap between non-explainable methods and explainable methods. Our NASXNet could re-design the backbone specifically for the prototype network. Thus it could achieve higher performance with a smaller model size.

Model	Type	#Params	Accuracy (%)			
			CIFAR-10	CIFAR-100	CUB-200-2011	Stanford Cars
VGG-19	Non-Explainable	144M	93.56 ± 0.25	70.48 ± 0.23	75.14 ± 0.37	85.90 ± 0.32
ResNet-101	Non-Explainable	44.5M	93.75 ± 0.22	74.41 ± 0.31	81.50 ± 0.29	90.21 ± 0.30
EfficientNet-B6	Non-Explainable	43M	98.03 ± 0.22	87.21 ± 0.31	83.32 ± 0.29	93.13 ± 0.30
ProtoPNet + VGG-19	Explainable	138M	92.99 ± 0.15	69.80 ± 0.11	78.03 ± 0.22	86.10 ± 0.25
ProtoPNet + ResNet-101	Explainable	45M	93.88 ± 0.24	68.87 ± 0.18	77.32 ± 0.33	86.85 ± 0.37
ProtoPNet + EfficientNet-B6	Explainable	44M	95.88 ± 0.31	74.93 ± 0.13	79.25 ± 0.43	88.98 ± 0.22
NASXNet	Explainable	30M	**97.13 ± 0.23**	**75.10 ± 0.23**	**82.42 ± 0.35**	**91.24 ± 0.26**

Figure 3: NASXNet prediction example for Stanford Cars. The left part is the input image and top four activation maps. The right part is the corresponding activation maps and images of the prototypes with the highest similarity score.

Figure 4: Training process comparison on CIFAR-10. In NASXNet, epoch 1 to epoch 5 is the NAS adaptation stage, epoch 6 to 45 is the joint training stage, epoch 46 to 50 is the fine-tuning stage. For direct training, there is no adaptation stage and auxiliary joint training stage.

prototype of the class that image belongs to. Mathematically, we perform following update for each prototype $\mathbf{p}_j$:

$$\begin{aligned} &\mathbf{p}_j \leftarrow \text{argmin}_{\mathbf{z}\in Z_i} \|\mathbf{z} - \mathbf{p}_j\|_2, \\ &\text{where } Z_i = \{\mathbf{z} : \mathbf{z} \in \text{patches}(f(\mathbf{x}_i))\}. \end{aligned} \tag{9}$$

Unlike ProtoPNet, we allow a global update of the prototype projection, thus the model can automatically adjust the number of prototypes for each class. We can easily prove that the projection operation of prototypes has a limited effect on the accuracy of the models.

Theorem 1. *Let $\mathbf{p}_l^k$ be the nearest latent patch of class k, $\mathbf{b}_l^k$ be the value of the l-th prototype for class k before projection, $\mathbf{a}_l^k$ be the value after the projection with the nearest patch in the same class, $\mathbf{a}_l$ be the value after the projection with the global nearest patch. If we project the global nearest patch to the prototypes, then the change of top-2 classes $\Delta_{top2} \leq 2\Delta_{\max}$*

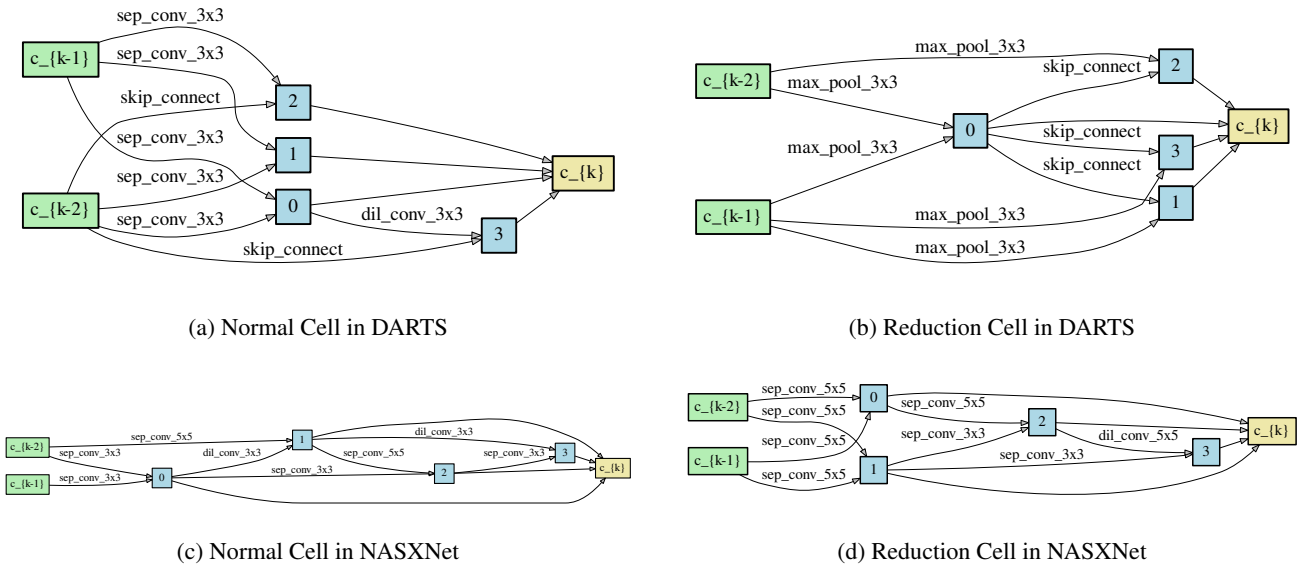
Proof. Suppose after project prototypes to their nearest patch of the same class $\mathbf{p}_l^k$, the output logit for every incorrect classes can increase at most by $\Delta_{\max} = m' \log((1+\delta)(2-\delta))$, and the output logit for the correct class can decreased at most by $\Delta_{\max}$. The proof can be referred to the Theorem 2.1 in [4].

Then since $\mathbf{a}_l$ is the projection with global nearest path, we have $\|\mathbf{a}_l - \mathbf{b}_l^k\|_2 \leq \|\mathbf{a}_l^k - \mathbf{b}_l^k\|_2$. Thus if we project the global nearest patch to the prototypes, the change of top-2 classes $\Delta_{\text{top2}} \leq 2\Delta_{\max}$. It means if the output logits between the top-2 classes are at least $2\Delta_{\max}$ apart, the projection would not change the prediction result. □

Intuitively speaking, the global update of the prototype projection would have less effect on the model's accuracy. Moreover, if the prototype projection does not change the prototypes by much, the prediction results would not change. We would periodically project the image patches to the prototypes during the training. The corresponding image patches and images would be saved to visualize and interpret the prediction results.

4 Experiments

In this section, we evaluate NASXNet for image classification tasks. First, we analyze performance on four popular image classification



(a) Normal Cell in DARTS

(b) Reduction Cell in DARTS

(c) Normal Cell in NASXNet

(d) Reduction Cell in NASXNet

Figure 5: Generated architecture comparison in CIFAR-10. The first line cell structures are the architectures generated with traditional classification layers. The structures in the second line are generated by our NASXNet which contains explainable prototype layers.

datasets, i.e., CIFAR-10, CIFAR-100, CUB-200-2011 and Stanford Cars datasets [15, 39, 14]. Then we visualize prediction results and conduct interpretability analysis on the CUB-200-2011. Last, we show the generated feature extraction network architecture of our approach.

4.1 Setup & Visualization

In our experiments, we used the NVIDIA TITAN RTX 24G GPU and followed the same settings as in [4], where we set the prototype shape as $H_1 = W_1 = 1$, the input image would be transformed to make sure the dimensions of the embedding are aligned. Additionally, we used the following values for our loss function: $\lambda_1 = 0.8$ and $\lambda_2 = 0.001$.

During the prototype projection phase, we obtain the proto-image for each prototype $\mathbf{p}_j$. To get the patch of the proto-image that corresponds to $\mathbf{p}_j$, we analyze the pixels of the image that were most activated when passing through NASXNet. This helps us visualize $\mathbf{p}_j$ as the minimal rectangular patch of the proto-image $\mathbf{x}$ that encloses 95% of the activation values in the map.

To visualize the activation maps of patches for the input image, we analyze the similarity scores between the input patches and prototypes. This allows us to see how the model classifies the image. By analyzing the similarity scores between the input patches and prototypes, we can gain insights into how the model is making its predictions.

4.2 Performance Analysis

In our study, we compared the performance of NASXNet with several black-box models, including VGG-10, ResNet-101, and EfficientNet-B6, as well as the explainable model ProtoPNet with different backbones. The feature extraction network is stacked with the best cell structures searched from CIFAR-10. We presented the accuracy results on four datasets in Table 1.

From the results, we observed that ProtoPNet's performance in some datasets could be improved by using a higher-performance backbone. However, we also found a significant performance gap compared to the corresponding non-explainable method. Therefore, we proposed NASXNet that generates smaller and higher-performing explainable models by designing a new architecture specifically for the prototype feature extraction. The accuracy has a significant improvement by using our NASXNet framework.

In addition, we analyzed how the three-stage training process works in NASXNet. We removed the NAS adaptation and joint training stages and directly trained the NAS and prototype layer. In Figure 4, we can see that without adaptation and joint training, the model cannot converge and cannot search for a proper neural architecture for the prototype layer. The three-stage training process, which we proposed, enables faster convergence during training and helps the model to automatically learn the prototype distribution.

4.3 Interpretability Analysis

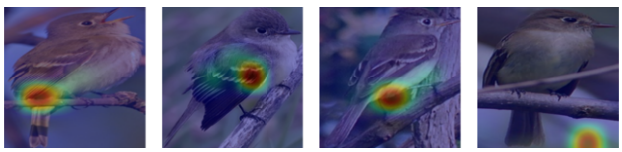
In our proposed method, we set the number of prototypes as $10 \times K$, where K is the number of classes in the dataset. This means that the average number of prototypes per class is 10. The prototype layer generates similarity scores for each prediction result with the prototypes, allowing us to obtain highly activated prototypes based on these scores. By visualizing the activation map in the prototypes and the original image, we can easily understand how the deep learning model classifies the instance.

For instance, Figure 1 shows a prediction example in CUB-200-2011, while Figure 3 shows a visualization example in Stanford Cars. We select the four prototypes with the highest score for a prediction and visualize them as training image patches.

However, we have noticed that there are many duplicated prototypes in ProtoPNet, e.g., the first and second prototypes in the example shown in Figure 6. This is because the cluster cost in ProtoPNet tends to select similar prototypes in the same class. To avoid this problem, we have added a new separation cost between any two prototypes.

This new separation cost is a novel element we have introduced in our method. It allows for more flexible adjustment in the number of

(a) Top four prototypes in ProtoPNet



(b) Top four prototypes in NASXNet

Figure 6: Prototypes for the same class that generated by ProtoPNet and NASXNet. ProtoPNet would generate many duplicated prototypes. Our new loss function in NASXNet could reduce the duplicated prototypes, thus including more various prototypes in the prototype set.

prototypes between different classes, guiding the model to search for more diverse prototypes. By avoiding duplicated prototypes, we can expect a more accurate and robust performance from our models.

Our approach to prototype selection is a critical aspect of our method, as it allows us to gain insights into how the deep learning model makes its predictions. By visualizing the activation maps of patches for the input image, we can analyze the similarity scores between the input patches and prototypes, gaining valuable insights into how the model classifies the image.

4.4 Generated Architectures

The results presented in Table 1 demonstrate the effectiveness of our method, which generates smaller models with better performance. Our approach optimizes the feature extraction network architecture by jointly training the NAS with the prototype layers, resulting in a more suitable prototype feature extraction. In comparison to the best-performing architectures of DARTS, the architectures generated by our NASXNet are different from the original ones. This indicates that the architecture has been adjusted to fit the prototype classification.

To provide a more in-depth understanding of our approach, we show the normal cell and reduction cell structures of the best model in CIFAR-10 in Figure 5. These structures were carefully designed to enhance the performance of our approach, and we believe that the newly designed architecture would also contribute to the field of prototype classification.

5 Conclusion

Interpretability is a critical factor in many areas and applications that employ deep learning models. ProtoPNet has proposed adding prototype layers to interpret image classification predictions. However, there remains a significant performance gap between explainable models and state-of-the-art models, with built-in explainable models often suffering from low-performance issues. This is where our NASXNet comes in, providing a way to obtain an explainable and high-performing deep learning model.

To achieve this, we propose a new loss function and search for a feature extraction network that can properly utilize prototypes. Our method also proposes a new training architecture to obtain efficient and high-performance explainable models. These contributions are significant in that our proposed NASXNet can provide both inherent interpretability and state-of-the-art results. Thus, we show that there might not necessarily be a trade-off between performance and interpretability.

However, we do acknowledge that while prototype layers provide a level of interpretability, the inference process of the feature extraction network is still difficult to understand. As such, in future research, we may focus on interpreting the feature extraction network to gain a better understanding of how it works and obtain a fully explainable deep neural network. In summary, our work demonstrates the importance of interpretability in deep learning models and provides a promising path towards achieving both interpretability and high-performance in deep learning applications.

Acknowledge

The work described here was partially supported by grants from the National Key Research and Development Program of China (No. 2018AAA0100204) and from the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 14222922, RGC GRF, No. 2151185).

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim, 'Sanity checks for saliency maps', in *NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, eds., Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, pp. 9525–9536, (2018).

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar, 'Designing neural network architectures using reinforcement learning', in *ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, (2017).

[3] Osbert Bastani, Carolyn Kim, and Hamsa Bastani, 'Interpretability via model extraction', *CoRR*, **abs/1706.09773**, (2017).

[4] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan Su, 'This looks like that: Deep learning for interpretable image recognition', in *NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, eds., Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 8928–8939, (2019).

[5] Jon Donnelly, Alina Jade Barnett, and Chaofan Chen, 'Deformable protopnet: An interpretable image classifier using deformable prototypes', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10265–10275, (2022).

[6] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter, 'Efficient multi-objective neural architecture search via lamarckian evolution', in *ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, (2019).

[7] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter, 'Neural architecture search: A survey', *J. Mach. Learn. Res.*, **20**, 55:1–55:21, (2019).

[8] Bryce Goodman and Seth Flaxman, 'European union regulations on algorithmic decision-making and a "right to explanation"', *AI Magazine*, **38**(3), 50–57, (Oct. 2017).

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, (2016).

[10] Xin He, Kaiyong Zhao, and Xiaowen Chu, 'Automl: A survey of the state-of-the-art', *Knowl. Based Syst.*, **212**, 106622, (2021).

[11] Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu, 'Improved differentiable architecture search for language modeling and named entity recognition', in *EMNLP-IJCNLP*, eds., Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, pp. 3583–3588. Association for Computational Linguistics, (2019).

[12] Carolyn Kim and Osbert Bastani, 'Learning interpretable models with causal guarantees', *CoRR*, **abs/1901.08576**, (2019).

[13] Efi Kokiopoulou, Anja Hauth, Luciano Sbaiz, Andrea Gesmundo, Gábor Bartók, and Jesse Berent, 'Task-aware performance prediction for

efficient architecture search', in *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, eds., Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 1238–1245. IOS Press, (2020).

[14] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei, '3d object representations for fine-grained categorization', in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, (2013).

[15] Alex Krizhevsky, Geoffrey Hinton, et al., 'Learning multiple layers of features from tiny images', (2009).

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, 'Imagenet classification with deep convolutional neural networks', in *Advances in neural information processing systems*, eds., Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, pp. 1106–1114, (2012).

[17] Himabindu Lakkaraju, Nino Arsov, and Osbert Bastani, 'Robust and stable black box explanations', in *International conference on machine learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5628–5638. PMLR, (2020).

[18] Himabindu Lakkaraju and Osbert Bastani, '"how do I fool you?": Manipulating user trust via misleading black box explanations', in *AIES '20: AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, February 7-8, 2020*, eds., Annette N. Markham, Julia Powles, Toby Walsh, and Anne L. Washington, pp. 79–85. ACM, (2020).

[19] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec, 'Faithful and customizable explanations of black box models', in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, AIES 2019, Honolulu, HI, USA, January 27-28, 2019*, eds., Vincent Conitzer, Gillian K. Hadfield, and Shannon Vallor, pp. 131–138. ACM, (2019).

[20] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al., 'Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model', *Annals of Applied Statistics*, **9**(3), 1350–1371, (2015).

[21] Yaoman Li and Irwin King, 'Architecture search for image inpainting', in *ISNN 2019, Moscow, Russia, July 10-12, 2019, Proceedings, Part I*, eds., Huchuan Lu, Huajin Tang, and Zhanshan Wang, volume 11554 of *Lecture Notes in Computer Science*, pp. 106–115. Springer, (2019).

[22] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu, 'Hierarchical representations for efficient architecture search', in *ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, (2018).

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang, 'DARTS: differentiable architecture search', in *ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, (2019).

[24] Zhichao Lu, Gautam Sreekumar, Erik D. Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti, 'Neural architecture transfer', *IEEE Trans. Pattern Anal. Mach. Intell.*, **43**(9), 2971–2989, (2021).

[25] Scott M. Lundberg and Su-In Lee, 'A unified approach to interpreting model predictions', in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, eds., Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 4765–4774, (2017).

[26] Yao Ma, Shilin Zhao, Weixiao Wang, Yaoman Li, and Irwin King, 'Multimodality in meta-learning: A comprehensive survey', *Knowl. Based Syst.*, **250**, 108976, (2022).

[27] Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren, 'Interpretable and steerable sequence learning via prototypes', in *Proceedings of the 25th ACM SIGKDD*, eds., Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, pp. 903–913. ACM, (2019).

[28] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean, 'Efficient neural architecture search via parameter sharing', in *International conference on machine learning*, eds., Jennifer G. Dy and Andreas Krause, volume 80 of *Proceedings of Machine Learning Research*, pp. 4092–4101. PMLR, (2018).

[29] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le, 'Regularized evolution for image classifier architecture search', in *Proceedings of the aaai conference on artificial intelligence*, pp. 4780–4789. AAAI Press, (2019).

[30] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin, '"why should I trust you?": Explaining the predictions of any classifier', in *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, eds., Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, pp. 1135–1144. ACM, (2016).

[31] Cynthia Rudin, 'Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead', *Nature Machine Intelligence*, **1**(5), 206–215, (2019).

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li, 'Imagenet large scale visual recognition challenge', *Int. J. Comput. Vis.*, **115**(3), 211–252, (2015).

[33] Mikolaj Sacha, Dawid Rymarczyk, Lukasz Struski, Jacek Tabor, and Bartosz Zielinski, 'Protoseg: Interpretable semantic segmentation with prototypical parts', in *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2023, Waikoloa, HI, USA, January 2-7, 2023*, pp. 1481–1492. IEEE, (2023).

[34] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra, 'Grad-cam: Visual explanations from deep networks via gradient-based localization', in *ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 618–626. IEEE Computer Society, (2017).

[35] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 'Deep inside convolutional networks: Visualising image classification models and saliency maps', in *ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, eds., Yoshua Bengio and Yann LeCun, (2014).

[36] Karen Simonyan and Andrew Zisserman, 'Very deep convolutional networks for large-scale image recognition', in *ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, (2015).

[37] Kenneth O Stanley and Risto Miikkulainen, 'Evolving neural networks through augmenting topologies', *Evolutionary computation*, **10**(2), 99–127, (2002).

[38] Mukund Sundararajan, Ankur Taly, and Qiqi Yan, 'Axiomatic attribution for deep networks', in *International conference on machine learning*, eds., Doina Precup and Yee Whye Teh, volume 70 of *Proceedings of Machine Learning Research*, pp. 3319–3328. PMLR, (2017).

[39] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, 'The Caltech-UCSD Birds-200-2011 Dataset', Technical Report CNS-TR-2011-001, California Institute of Technology, (2011).

[40] Wenna Wang, Xiuwei Zhang, Hengfei Cui, Hanlin Yin, and Yannnig Zhang, 'FP-DARTS: fast parallel differentiable neural architecture search for image classification', *Pattern Recognit.*, **136**, 109193, (2023).

[41] Yihang Yin, Siyu Huang, and Xiang Zhang, 'BM-NAS: bilevel multimodal neural architecture search', in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 8901–8909. AAAI Press, (2022).

[42] Matthew D. Zeiler and Rob Fergus, 'Visualizing and understanding convolutional networks', in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, eds., David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, volume 8689 of *Lecture Notes in Computer Science*, pp. 818–833. Springer, (2014).

[43] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter, 'Understanding and robustifying differentiable architecture search', in *ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, (2020).

[44] Barret Zoph and Quoc V. Le, 'Neural architecture search with reinforcement learning', in *ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, (2017).

[45] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le, '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*, pp. 8697–8710. IEEE Computer Society, (2018).