

Improving Neural Architecture Search with Class Visualization and Bilevel Optimization for Imbalanced Data

Alejandro Rosales-Pérez*

alejandro.rosales@cimat.mx

Centro de Investigación en
Matemáticas

Monterrey, Nuevo León, Mexico

Saúl Zapotecas-Martínez

szapotecas@inaoep.mx

Instituto Nacional de Astrofísica,
Óptica y Electrónica

Puebla, Mexico

Carlos A. Coello Coello

ccoello@cs.cinvestav.mx

Centro de Investigación y Estudios
Avanzados del IPN

Mexico City, Mexico

Abstract

Neural Architecture Search (NAS) has proven to be a highly effective technique for automating neural network design, overcoming challenges posed by complex datasets, class imbalance, and the high time consumption associated with manual architecture selection. Although NAS shows promise, it frequently suffers from high computational costs and the risk of data loss due to insufficient training. This paper presents a novel NAS method that incorporates class visualization to mitigate computational costs and a bilevel optimization framework to address class imbalance. The hierarchical structure comprises an upper level formulated as a multi-objective optimization problem to maximize performance, measured by the Matthews correlation coefficient, while minimizing model complexity. The lower level focuses on neural network training using a reduced dataset obtained by class visualization. The proposed approach combines NSGA-II (for the upper-level problem) with stochastic gradient descent (for the lower-level problem). This method significantly reduces NAS runtime on MNIST and CIFAR-10 subsets while maintaining accuracy and robustness.

CCS Concepts

• **Applied computing** → **Operations research**; *Decision analysis*; Multi-criterion optimization and decision-making.

Keywords

Neural architecture search, class visualization, NSGA-II

ACM Reference Format:

Alejandro Rosales-Pérez, Saúl Zapotecas-Martínez, and Carlos A. Coello Coello. 2025. Improving Neural Architecture Search with Class Visualization and Bilevel Optimization for Imbalanced Data. In *Genetic and Evolutionary Computation Conference (GECCO '25 Companion)*, July 14–18, 2025, Malaga, Spain. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3712255.3726630>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

GECCO '25 Companion, Malaga, Spain

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1464-1/2025/07

<https://doi.org/10.1145/3712255.3726630>

1 Introduction

Deep learning is a branch of machine learning that leverages multi-layered neural networks to capture intricate patterns and representations within large-scale, high-dimensional datasets. The capacity of deep learning to process unstructured data and generalize from large datasets is paramount to modern artificial intelligence.

The efficacy of deep learning hinges on both the underlying architecture and the data it uses. Building effective neural networks is a complex process requiring significant specialized knowledge. The challenge of exploring every configuration is a significant undertaking. Furthermore, suboptimal design leads to poor performance and generalization issues. To address these challenges, Neural Architecture Search (NAS) has proven to be an effective automated design solution.

Class visualization [8] has recently emerged as a viable alternative to model compression [1]. Based on this idea, we explored the idea of class visualization as an alternative to construct a surrogate representation of the original dataset for its application in NAS. This paper proposes a novel approach to address these challenges by integrating bilevel optimization and class visualization. In Bilevel NAS, the optimal architecture is found at the upper level, while the lower level focuses on the model's parameter training and fine-tuning. The upper level addresses a multi-objective optimization problem, maximizing both the Matthews Correlation Coefficient (MCC)—a metric robust to class imbalance—and minimizing FLOPs to ensure efficient and robust architectures. Empirical analysis across diverse MNIST and CIFAR-10 dataset variations reveals that the novel bilevel NAS architecture, integrated with class visualization, attains superior accuracy with substantially reduced computational cost.

The remainder of this paper is organized as follows. Section 2 discusses the key concepts related to bilevel optimization and neural architecture search. Section 3 elaborates on the proposed framework, while Section 4 details our experimental study. Finally, Section 5 summarizes the findings and outlines potential directions for future research.

2 Preliminaries

2.1 Neural Architecture Search

NAS automates the design of optimal neural network architectures tailored to specific machine learning tasks like image recognition and natural language processing. Traditionally, this was a labor-intensive process, demanding significant expertise in network

configuration, including layer types, interconnections, and hyperparameter tuning. NAS streamlines this process by examining a pre-set range of potential architectures and automatically selecting the best one. This not only enhances the network's performance but also considers other factors like computational efficiency and resource utilization, which are critical in real-world applications.

The NAS process generally uses three key parts: a search space, a search strategy, and an evaluation method [5]. The search space defines the range of neural network architectures that NAS can explore. This encompasses differences in the quantity and kinds of layers (like convolutional, fully connected, or recurrent layers), their interconnectivity, and other design decisions. The search strategy dictates how NAS explores this massive space to find the best architectures. This is often achieved using techniques such as reinforcement learning, evolutionary algorithms, or gradient-based optimization, which guide the search process by iteratively refining network designs. The evaluation method is used to assess the quality of each candidate architecture. This usually means training the network on a dataset and then measuring its accuracy, precision, or other relevant metrics. However, the computational demands of this evaluation are significant, given the need to fully train multiple architectures, a process consuming considerable time and resources.

2.2 Class Visualization

Class visualization aims to generate images that best represent each class, leading to better classification. Gradient descent helps identify images that reveal the most active parts of deep neural networks layers. Class visualizations are defined as the images that maximally activate neurons in the final layer.

A class visualization technique is aimed at identifying the image producing the highest softmax layer score for a particular class [8]. An algorithm begins by generating a random input image. Subsequently, backpropagation is utilized to calculate the gradient with respect to the image; iterative optimization via gradient descent is then performed until a predetermined termination condition is satisfied. The optimization problem is formulated as follows:

$$\arg \max_I Sc(I) - \lambda \|I\|_2^2 \quad (1)$$

where I is the image, $Sc(\cdot)$ is the score of class c , and λ is the regularization parameter.

Recent research has investigated the potential of class visualization as both an interpretative and as a training technique. [1] introduces a novel methodology employing visualizations to enhance neural network training efficiency.

2.3 Bilevel Optimization

Bilevel optimization is a specialized field in optimization where one problem, known as the upper-level (or leader) problem, is embedded within another, referred to as the lower-level (or follower) problem. The defining characteristic of bilevel optimization is its hierarchical structure, where the solution to the upper-level problem depends on the optimal solution to the lower-level problem [3]. In this framework, the upper-level decision-maker controls a set of variables, while the lower-level decision-maker manages others, resulting in a complex interdependence between the two levels. Formally, a

bilevel optimization problem can be formulated as follows:

$$\min_{x \in X} F(x, y) \quad (2)$$

$$\text{subject to: } G(x, y) \leq 0 \quad (3)$$

$$\min_{y \in Y} f(x, y) \quad (4)$$

$$\text{subject to: } g(x, y) \leq 0 \quad (5)$$

The primary optimization task in bilevel optimization is the upper-level problem, which incorporates a secondary, nested optimization problem as part of its constraints. A solution to the upper-level problem is only deemed valid if the lower-level problem has been solved optimally. This layered structure adds considerable complexity, as each attempt to solve the upper-level problem requires repeated resolution of the lower-level problem. The lower-level decision-maker typically faces a separate objective function and set of constraints, which are influenced by decisions made at the upper level. This interdependence between the two levels can lead to situations where traditional optimization methods struggle, as they are not designed to handle the feedback loop created by this hierarchical relationship.

3 EBiNAS: Efficient Bilevel Neural Architecture Search based on Class Visualizations

Algorithm 1 EBiNAS

Require: \mathcal{X} , the set of images,

y , the set of class labels,

u_s , upper population size,

m_e , maximum number of evaluations at the upper-level,

p , number of surrogate images by class,

m , maximum number of iterations for the lower-level.

Ensure: A set of neural architectures.

- 1: Generate randomly an initial population for the upper-level, \mathcal{P}_u , with u_s individuals
 - 2: Generate p surrogate images for each class from \mathcal{X} using class visualization.
 - 3: **for** each individual p_u in \mathcal{P}_u **do**
 - 4: Learn the parameters for the architecture encoded in p_u with stochastic gradient descent and the surrogate dataset as training samples.
 - 5: Compute the MCC and FLOPS for the p_u architecture.
 - 6: **end for**
 - 7: **while** a stopping criterion is not met **do**
 - 8: Apply evolutionary operators to produce an upper-level offspring population, O_u .
 - 9: **for** each individual o_u in O_u **do**
 - 10: Learn the parameters for the architecture encoded in o_u with stochastic gradient descent and the surrogate dataset as training samples.
 - 11: Compute the MCC and FLOPS for o_u architecture.
 - 12: **end for**
 - 13: Set \mathcal{P}_u based non-dominated sorting on $\mathcal{P}_u \cup O_u$.
 - 14: **end while**
 - 15: Retrain the set of non-dominated architectures using \mathcal{X} , y .
-

3.1 Bilevel Formulation of NAS

EBiNAS balances model complexity and performance by minimizing training error and generalizing to new data. This is accomplished through a bilevel optimization strategy. The upper level maximizes the MCC and minimizes model complexity (measured in FLOPs), while the lower level learns model parameters by minimizing Binary Cross-Entropy (BCE) loss.

3.1.1 The upper-level. By focusing on maximizing the MCC score at the upper level, the proposed formulation seeks to identify architectures that excel in handling imbalanced classification problems. The MCC is defined as follows:

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (6)$$

where TP is the number of instances correctly classified as positive, TN is the number of instances correctly classified as negative, FP is the number of instances incorrectly classified as positive, and FN is the number of instances incorrectly classified as negative. As the outputs are the probability values of belonging to a class, an instance is associated with the class with the highest probability.

3.1.2 The lower-level. The lower level minimizes the BCE loss function, which governs the accuracy of predictions during model training. The BCE loss measures the disparity between the predicted probabilities and the actual labels, assigning greater penalties to incorrect predictions. This ensures that the model focuses on learning to predict correctly, especially in cases where class distributions are skewed. The BCE loss is calculated for each individual prediction and then averaged across the entire dataset to provide an overall measure of error. This makes it a widely used metric in binary classification tasks, particularly when precision in probabilistic predictions is crucial. The BCE loss function is formally defined as:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^n (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \quad (7)$$

where y is the true label, which can be either 0 or 1; \hat{y} is the predicted probability that the output is 1; and $\log(\hat{y})$ is the natural logarithm of the predicted probability.

Due to the high computational cost of training neural architectures at lower levels, we propose employing class visualizations to create a surrogate dataset. The surrogate dataset is employed within the lower level, as evidenced by lines 4 and 10 of Algorithm 1.

3.2 Search Strategy

To solve the bilevel optimization problem, we adopted a hierarchical approach. At the lower level, the Stochastic Gradient Descent (SGD) algorithm was employed. At the upper level, we used the well-known NSGA-II [2] to search for optimal neural architectures. The architecture of each neural network is encoded based on the representation proposed in [9]. More precisely, an architecture is represented as $x = (x^{(1)}, x^{(2)}, \dots, x^{(n_b)})$, where n_b denotes the number of blocks in the network. Each element in this sequence corresponds to a block, which is modeled as a directed acyclic graph (DAG) with a fixed number of nodes. This graph describes the operations within each block using a binary string, where specific bits represent different operations or connections.

Table 1: Subsets from the MNIST dataset used in the experimental study.

Subset	Classes	Samples	Rate of samples per class
1	3/8	6501	10%/90%
2	0/3/8	12170	7%/45%/48%
3	0/3/4/8	16222	13%/17%/34%/36%
4	0/3/4/8/9	8755	5%/20%/5%/40%/30%

Table 2: Subsets from the CIFAR-10 dataset used in the experimental study.

Subset	Classes	Samples	Rate of samples per class
1	cat/dog	6000	17%/83%
2	dog/horse	4000	30%/70%
3	auto/truck	5000	80%/20%

Architectures evolve through crossover and mutation operations. The crossover process is conducted using the half-uniform crossover method. On the other hand, the bit-flip operator was adopted for mutation.

The population for the next generation is selected using the NSGA-II selection scheme, which combines the individuals from both the parents and offspring populations and performs a non-dominated sorting on the merged population to select the best ones based on the non-dominance level.

4 Experimental Setup and Results

4.1 Dataset

For assessing the performance of the proposed method, we used two well-known datasets in machine learning: the MNIST dataset [4] and the CIFAR-10 dataset [6]. The MNIST dataset consists of a set of handwritten digit images, widely used for training and testing pattern recognition algorithms. The selection of MNIST in numerous studies is due to its manageable size, easy accessibility, and its ability to efficiently evaluate classification algorithms in a supervised learning context.

The MNIST dataset is composed of 60,000 training samples and 10,000 validation samples, where each observation has a size of 28×28 pixels and is represented in grayscale. These samples are evenly distributed across 10 classes. From the training set, subsets have been created, which are detailed in Table 1.

The CIFAR-10 dataset consists of 60,000 32×32 color images in 10 classes. We adopted from the dataset 50,000 training samples and 1,000 from each class for the test set. Table 2 shows the distribution of the generated imbalanced subsets.

Configuring different subsets allows assessing the performance under imbalanced scenarios. Each of the subsets detailed in Tables 1 and 2 was divided into three sets: training, validation, and test sets. The distribution is of 64%, 16%, and 20% of the data, respectively. The test set was reserved exclusively for obtaining the final results presented in this work, while the validation set was used for the final evaluation of the architectures during the search process.

Table 3: Reported results for each problem for EBiNAS and NSGA-NET. The reported results are the MCC score, FLOPS and the average time.

MNIST	NSGA-NET			EBiNAS			CIFAR-10	NSGA-NET			EBiNAS		
	MCC	FLOPS	Time	MCC	FLOPS	Time		MCC	FLOPS	Time	MCC	FLOPS	Time
1	0.940	387,651	10:40:58	0.934	261,726	02:54:49	1	0.932	443,285	8:28:03	0.940	234,900	03:01:49
2	0.987	481,478	16:24:53	0.958	256,788	02:39:43	2	0.944	383,612	5:26:11	0.923	248,313	02:43:26
3	0.981	923,450	24:11:01	0.972	355,552	02:55:12	3	0.964	422,086	7:34:29	0.940	249,313	02:37:21
4	0.524	1,355,546	20:15:00	0.936	429,626	02:49:23							

4.2 Experimental Settings

Our proposed EBiNAS algorithm is compared to the widely recognized evolutionary NAS algorithm, NSGA-NET [7]. For both NSGA-NET and EBiNAS, the initial population is generated uniformly and randomly across the entire search space. The crossover and mutation probabilities are set at 0.9 and 0.02, respectively. The size of the initial population is 20 individuals, and 20 generations are performed for exploration. For the lower-level EBiNAS, the learning rate was dynamically adjusted, starting at 0.025 and gradually decreasing to 0.001 throughout 25 epochs. This dynamic learning rate allows for more aggressive updates at the early stages of training, while ensuring more refined adjustments as the model converges. For EBiNAS, the number of images per class, p , is set to one.

4.3 Experimental Results

Our proposed bilevel NAS was executed five times for each problem to ensure robust performance evaluation. The outcomes of these runs are summarized in Table 3, which reports the performance of the bilevel NAS across all problems outlined in Table 1.

The effectiveness of the EBiNAS formulation is clearly shown by the results, which indicate that EBiNAS consistently achieves superior performance, exceeding 0.9 across all problem instances. These findings are highly significant, demonstrating considerable capability within the field of neural architecture search. Compared to the state-of-the-art NAS algorithm NSGA-NET, this method obtains a marginally superior MCC score across several subsets of both MNIST and CIFAR-10 datasets. Wilcoxon signed-rank tests conducted on each subset revealed no statistically significant differences for subsets 1, 3, and 4 of the MNIST dataset.

The NSGA-NET algorithm achieved performance scores exceeding 0.9 in only four of the seven test problems. Remarkably, its performance dropped below 0.6 in a problem where the minority class accounted for less than 5% of the samples, indicating its limitations in managing extreme class imbalance.

Concerning FLOP counts, our proposed EBiNAS approach yields architectures of superior simplicity compared to those produced by NSGA-NET. Therefore, simple architectures can achieve a high degree of accuracy. The FLOP counts of models discovered using EBiNAS are up to 68% less than those identified by NSGA-NET.

Regarding the computational cost, the runtime of EBiNAS remained stable at two to three hours across all subsets. The experiments were conducted using an RTX GPU equipped with 24 GB of RAM. In contrast, the NSGA-NET method demanded a longer processing time, reaching up to 24 hours. This shows the efficacy of class visualization in mitigating computational costs.

5 Conclusions

EBiNAS address the problem of optimizing deep learning models. We focus on improving the performance of models trained on imbalanced datasets by combining bilevel optimization and class visualization to reduce the computational cost of NAS. This bilevel approach involves two optimization stages. The upper level optimizes neural network architectures to enhance robustness in the face of class imbalance and control model complexity.

Surrogate dataset created by class visualization reduced computational costs by up to 90%. The execution time of EBiNAS remained almost unaffected by increases in the number of samples and classes. Because of the surrogate dataset contains only one image per class, incorporating a new class introduces only a single data point, resulting in a negligible increase in computational demands.

Our results illustrate the exceptional efficiency of our proposed EBiNAS, consistently outperforming conventional approaches across diverse test cases. The use of MCC serves to highlight its efficiency in addressing real-world problems exhibiting considerable asymmetry. This technique proves to be invaluable when designing architectures capable of adapting to diverse data distributions, particularly those exhibiting substantial class imbalance.

Acknowledgments

This work was funded by *Secretaría de Ciencia, Humanidades, Tecnología en Innovación* through project CBF2023-2024-2797.

References

- [1] José R. Abreu-Pederzini, Guillermo A. Martínez-Mascorro, José C. Ortiz-Bayliss, and Hugo Terashima-Marin. 2021. Exploring the Knowledge Embedded in Class Visualizations and Their Application in Dataset and Extreme Model Compression. *Applied Sciences* 11, 20 (2021), 9374.
- [2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197. doi:10.1109/4235.996017
- [3] Stephan Dempe. 2002. *Foundations of Bilevel Programming*. Springer US, Boston, MA. 309 pages. doi:10.1007/b101970
- [4] Li Deng. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142. doi:10.1109/MSP.2012.2211477
- [5] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *Journal of Machine Learning Research* 20, 55 (2019), 1–21.
- [6] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
- [7] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. 2019. NSGA-Net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (Prague, Czech Republic) (GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 419–427. doi:10.1145/3321707.3321729
- [8] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).
- [9] Lingxi Xie and Alan Yuille. 2017. Genetic CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 1388–1397. doi:10.1109/ICCV.2017.154