NCQS: Nonlinear Convex Quadrature Surrogate Hyperparameter Optimization

Sophia Abraham[1], Kehelwala Dewage Gayan Maduranga[2], Jeffery Kinnison[1], Jonathan Hauenstein[1], and Walter Scheirer[1]

[1]*University of Notre Dame, Notre Dame, IN 46556, USA*
[2]*Tennessee Technological University, Cookeville, TN 38505*

Abstract

Deep learning has revolutionized artificial intelligence and enabled breakthroughs across various domains. However, as deep learning models continue to grow in scale and complexity, optimizing their hyperparameters for efficient resource utilization becomes a critical challenge. Traditional optimization techniques often assume smooth and continuous loss functions, limiting their effectiveness in this context. In this work, we propose a novel data-driven approach to hyperparameter optimization using a convex quadrature surrogate. By leveraging a set of sampled hyperparameters and their corresponding performance, our method fits a multivariate quadratic surrogate model to identify the optimal hyperparameters. We demonstrate the practicality and effectiveness of our approach by improving the efficiency and performance of various hyperparameter strategies on both closed and open set benchmarks across diverse vision and tabular datasets. Additionally, we showcase its applicability in automatic target recognition tasks. This research contributes to the broader objective of resource-efficient deep learning for computer vision, fostering advancements in model efficiency, computational memory constraints, and latency considerations. Code available here.

1. Introduction

Deep learning has been pivotal in the recent resurgence of artificial intelligence, driving advancements in diverse applications such as human identification, self-driving vehicles, surveillance, and fire control systems. Despite these remarkable achievements, selecting the most suitable deep learning model for a specific task remains a challenging endeavor. Machine learning models comprise two types of parameters: hyperparameters and model parameters. Hyperparameters are user-defined parameters set before training, while model parameters are learned during the training process. In automated machine learning, the crucial task is to automatically determine these hyperparameters without manual intervention, aiming to enhance the performance of machine learning algorithms and improve the reproducibility and fairness of scientific studies.

In this work, we specifically focus on hyperparameter optimization. The performance of a machine learning model depends not only on the algorithm's architecture but also on the values assigned to its hyperparameters. Different hyperparameter configurations can lead to vastly different outcomes on the same dataset. Therefore, model selection is not solely an algorithmic decision but heavily influenced by the choice of hyperparameters. These parameters, which are free parameters associated with a particular machine learning model, optimize its learning capability. They need to be carefully set before training, distinct from the elementary parameters learned from the available data.

Various approaches have been proposed for automating the search for optimal hyperparameters, spanning from simple methods like grid search [9] and random search [4] to more sophisticated techniques such as Bayesian optimization [5, 3], gradient-based learning [2, 19], and surrogate model approaches [27].

Surrogate-based optimization methods [10, 26] are particularly valuable when the objective functions are computationally expensive to evaluate. The Surrogate Benchmarks for Hyperparameter Optimization [10] propose a strategy using inexpensive surrogates of real hyperparameter optimization benchmarks. These surrogates exhibit the same hyperparameter spaces and feature similar response surfaces. The approach involves training regression models on data representing the performance of a machine learning algorithm across a broad range of hyperparameter configurations. Instead of running the actual algorithm, hyperparameter optimization methods can be evaluated using the model's performance predictions, enabling more efficient

This work was funded by the DEVCOM Army Research Laboratory under the cooperative agreement W911NF-20-2-0218.

evaluations.

Our key contribution lies in a hyperparameter optimization strategy based on local nonlinear convex quadratic surrogates (NCQS). The fundamental idea behind NCQS is that at a local minimum, the Hessian matrix is positive semidefinite, allowing us to locally approximate the objective function with a convex quadratic function. Our focus primarily lies in the continuous domain, while also accommodating binary and categorical search domains using random strategies. Unlike other approaches, NCQS aims to approximate only regions of interest rather than the entire hyperparameter surface to approximate a global optimum. This approach results in a more efficient search methodology than the base strategies alone and does not rely on strong assumptions about the behavior of the underlying objective function. Additionally, we provide an implementation of NCQS within a search framework, enabling easy integration into general search problems.

2. Nonlinear Convex Quadrature Surrogate Optimization

To address the challenges of resource-efficient deep learning for computer vision, we propose a novel optimization method called Nonlinear Convex Quadrature Surrogate Optimization (NCQS). Traditional surrogate approximations [6, 8] try to approximate the objective function over large regions and thus are unable to effectively capture oscillations using relatively few samples. To mitigate this issue, we build a local convex multivariate approximation of the objective function using a quadratic that builds on the Morse lemma [20, Lemma 2.2] ensuring such a local convex approximation exists near the optimum. The approach is warmed up using any sampling method such as random sampling.

First, consider parameterizing the family of convex multivariate quadratic functions $f(x)$ where $x = (x_1, \ldots, x_n)$ using a matrix $M \in \mathbb{R}^{n\times n}$, a vector $y \in \mathbb{R}^n$, and a scalar $b \in \mathbb{R}$ such that

$$f(x) = \frac{1}{2}(x-y)^T M^T M(x-y) + b.$$

Clearly, the Hessian matrix of $f(x)$ is $M^T M$ which is always positive-semidefinite showing that $f(x)$ is convex with global minimum b which is attained at y, i.e., $f(x) \geq b$ for all $x \in \mathbb{R}^n$ and $f(y) = b$.

With this family parameterized by M, y, and b, the next step is to find optimal values (M^*, y^*, b^*) so that the corresponding quadratic best fits the objective function at the given data points. Namely, given sample points $x_1^*, \ldots, x_N^* \in \mathbb{R}^n$ with corresponding function values $o_1^*, \ldots, o_N^* \in \mathbb{R}$ and weights $d_1^*, \ldots, d_N^* \in \mathbb{R}_{\geq 0}$, one can compute the weighted least squares fit of the data by solving the unconstrained optimization problem

$$(M^*, y^*, b^*) = \underset{M,y,b}{\arg\min} \sum_{i=1}^{N} d_i^* \cdot (f(x_i^*) - o_i^*)^2. \quad (1)$$

This problem is easily solved using local methods. The corresponding value y^* is then suggested as the next set of hyperparameters to consider for solving the hyperparameter optimization problem.

Algorithm 1 provides a high-level description of the nonlinear convex quadrature surrogate optimization (NCQS) method. Initially, the algorithm collects data points by sampling hyperparameters using the base strategy of interest, ensuring exploration across the search space. A surrogate approximation function is then constructed in the neighborhood of the optimal sample point x_o^*. The weights d_i^* are used to localize the approximation, with one possible approach being to consider only the points within a ball centered at the current optimal sample point, with a radius D. In this approach, $d_i^* = 1$ if $x_i^* \in B(x_o^* D)$, and $d_i^* = 0$ otherwise. The choice of D can be adjusted by decaying its size over iterations to provide more accurate local approximations. To avoid excessively small regions, Algorithm 1 also incorporates a minimum distance threshold m, which was set to 0.1 in our experiments. An alternative approach is to select the weights based on a continuous function that is inversely proportional to the distance to x_o^*.

Once the weights are determined, the algorithm proceeds to solve the weighted least squares problem (1), yielding the optimal hyperparameters y^* which are then added to the list of data points along with the corresponding objective function value. The process iterates by collecting additional random sample points, which helps avoid being trapped in the region of the initial local minimum, and computing the minimum of each surrogate. The termination criterion of Algorithm 1 is based on either reaching a time bound or exceeding a trial bound.

The following two examples illustrate the ability of NCQS to locate the minimum of an objective function (provided in closed form) with Section 3 focusing on experiments involving optimizing hyperparameters.

2.1. One dimensional example

To illustrate the NCQS method, consider optimizing the following function from [14] on the domain $[0.5, 2.5]$:

$$g(x) = \frac{\sin(10\pi x)}{2x} + (x-1)^4 \quad (2)$$

which is plotted in Figure 1(a). The main idea behind NCQS is to treat the oscillatory behavior as noise and compute a best fit convex quadratic surrogate. This is demonstrated in Figure 1(b) where the surrogate function models the general trend of the data without oscillations. Using

Algorithm 1 Nonlinear Convex Quadrature Optimization

Input: T Time budget for optimization, N_T Maximum number of trials for optimization, W Number of samples for surrogate approximation, D Distance threshold for controlling surrogate localization, m Minimum distance threshold for surrogate localization control

Output: Optimal hyperparameter set that maximizes the objective function.

while Elapsed Time $\leq T$ **or** Completed Trials $\leq N_T$ **do**
  if Completed Trials $\leq W$ **then**
    Sample hyperparameters using base strategy and compute corresponding function evaluations.
  else
    Update the nonlinear local surrogate approximation $f(x)$.
    Find the optimal hyperparameter set from the surrogate by solving (1).
    Evaluate the model at the surrogate's optimal hyperparameter set.
    $D \leftarrow \max\left(\frac{D}{1 + 0.01 \cdot \text{Completed Trials}}, m\right)$.
    Collect additional sampled hyperparameters and compute their corresponding function evaluations.
  end if
end while
return Best hyperparameter set.

more randomly sampled data points, the surrogate becomes increasingly local so that the minimum of the surrogate converges to the global minimum as shown in Figure 1(c).

2.2. Two dimensional example

A standard planar example is the Griewank function [25]. To avoid having the global minimum attained at the origin, we consider the following modified function over the domain $[-20, 20]^2$:

$$g(x, y) = \frac{(x-5)^2 + (y+3)^2}{40} - \cos(x-5) \cdot \cos\left(\frac{y+3}{\sqrt{2}}\right) + 1 \quad (3)$$

which is plotted in Figure 2(a) with a global surrogate approximation built from 200 random sample points that models the general trend ignoring the oscillatory behavior in Figure 2(b). Increasingly more local approximations are shown in the plots in Figure 3 with the surrogate minimum converging to the global minimum.

3. Experiments

3.1. Benchmarks

In this study, we conducted evaluations of NCQS on both closed and open set recognition tasks to assess its effectiveness and performance in improving the performance and efficiency of popular hyperparameter optimization methods.

For the closed set experiments, we focused on classification tasks on tabular data using popular models such as SVM, Random Forest, and Logistic Regression. These benchmarks were accessed through HPOBench[11], a comprehensive platform for evaluating and comparing hyperparameter optimization algorithms. We compared NCQS's ability to augment the performance of various base methodologies, including random, TPE, Bayes, and SMAC. SMAC [18] is a well-known Bayesian method that utilizes a racing mechanism to efficiently decide which configurations perform better.

Furthermore, we employed NCQS to tune 30 hyperparameters for Tiny-YOLO[12], a widely used object detection model, and applied it in the context of automatic target recognition (ATR). ATR represents a complex optimization challenge due to the significant number of hyperparameters involved. By comparing the results against random sampling alone, we showcased NCQS's capability to efficiently navigate high-dimensional hyperparameter spaces and improve the overall optimization performance.

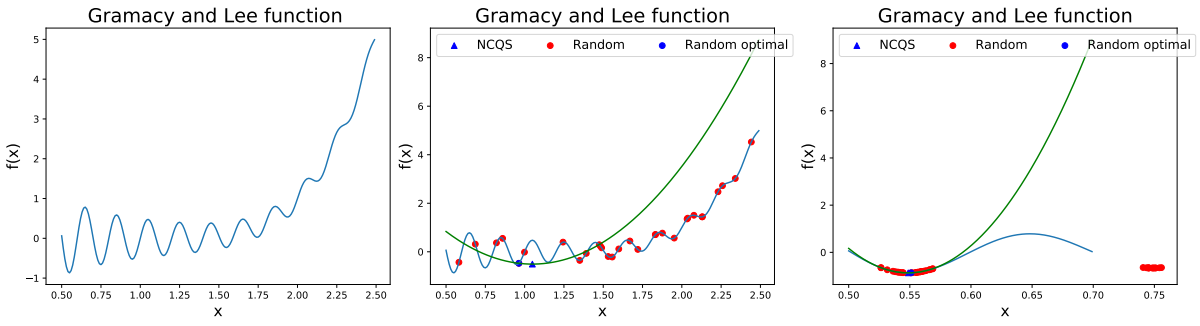
In addition to closed set recognition, we extended our exploration of NCQS to tackle the challenges posed by *open set recognition* tasks within the computer vision domain. Open set recognition involves scenarios where models encounter unknown classes during testing [22]. These situations can be particularly sensitive to hyperparameters, leading to complex and noisy loss landscapes with steep gradients.

To assess the efficacy of different hyperparameter optimization methods in the context of open set recognition, we conducted experiments comparing the performance and efficiency boost of NCQS against tree-structured parzen estimators (TPE) and random search. For open set recognition, we employed the Extreme Value Machine (EVM) [21], a scalable nonlinear classifier capable of handling incomplete knowledge during testing and rejecting inputs beyond the training set support.

The experimental results shed light on NCQS's capabilities and robustness in tuning hyperparameters for the EVM model, taking into account the unique challenges and complexities of open set recognition tasks in computer vision. By showcasing NCQS's effectiveness in real-world computer vision scenarios, we highlight its potential to facilitate more efficient and effective hyperparameter tuning for vision-based applications. The five hyperparameters and domains involved in training the EVM are outlined in the supplemental.

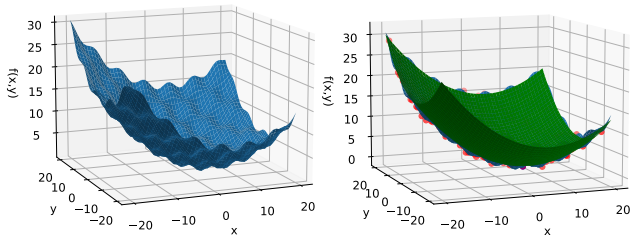
3.2. Optimization Framework

The baseline search framework utilized the massively Scalable Hardware-Aware Distributed Hyperparameter Op-



(a) (b) (c)

Figure 1: (a) Plot of Gramacy and Lee function (2) on the domain $[0.5, 2.5]$. (b) Surrogate approximation (green curve) avoids modeling oscillatory behavior but provides general trend of data. (c) Local surrogate approximation (green curve) whose minimum is converging to the global minimum.



(a) (b)

Figure 2: a) Plot of modified Griewank function (3) on $[-20, 20]^2$. (b) Surrogate approximation (green surface) using 200 randomly sampled points that avoids modeling oscillatory behavior but provides general trend of data.

timization (SHADHO) [17]. In addition to serving as a hyperparameter optimization tool, SHADHO has the added utility of considering hardware specifications when conducting a search. This framework calculates the relative complexity of each search space and monitors performance on the learning task over all trials. These metrics are then used as heuristics to assign hyperparameters to distributed workers based on their hardware. We extended SHADHO's framework by integrating `NCQS` which is publicly available for general search problems.

3.3. Datasets

We evaluated `NCQS` on both closed set and open set recognition tasks using a diverse range of datasets to ensure comprehensive testing and validation of our method's effectiveness.

Closed Set Recognition For the closed set experiments, we utilized a collection of classification tasks on tabular data, which were accessed through HPOBench [11]. To assess the performance of `NCQS` across different problem domains, we conducted these experiments on four separate dataset tasks obtained from the OpenML library [13].

Automatic Target Recognition (ATR) To demonstrate the capabilities of `NCQS` on more challenging scenarios involving a higher number of hyperparameters, we used the ATR Algorithm Development Image Database from the Defense Systems Information Analysis Center (DSIAC) [7]. This dataset contains visible and mid-wave infrared (MWIR) imagery collected by the US Army Night Vision and Electronic Sensors Directorate (NVESD). The images consist of frames from video sequences capturing different ground-based vehicle targets traveling around a 100-meter diameter circular path at distances ranging from 1000 to 5000 meters. For our experiments, we used the data from the closest set of recordings (1000–2000 meters).

Open Set Recognition in Computer Vision To validate and compare the performance of `NCQS` against existing hyperparameter optimization methods for open set recognition tasks in computer vision, we utilized datasets of varying complexity. We carefully selected these datasets to reflect real-world scenarios where models must deal with unknown classes during testing. Specific details about the datasets can be found in the supplemental material.

The number of trials for each experiment was determined based on the time required to train a single model for the given dataset. In cases where datasets contained numerous images per class, training the Extreme Value Machine (EVM) model required longer compute time, leading us to adjust the number of search trials accordingly.

Here are the datasets used for open set recognition tasks:

MNIST: In this dataset, we designated handwritten digits 0 to 5 as known classes and digits 6 to 9 as unknown classes. Each image was represented as a flattened vector of the image pixels (784 features).

Labeled Faces in the Wild (LFW): For LFW, we assigned classes with 30 or more face image samples as known classes (34 classes) and assigned the remaining 5715 classes as the unknown set. We used an ArcFace-based feature extractor [1] trained on the MS-Celeb-1M dataset [15] to ob-

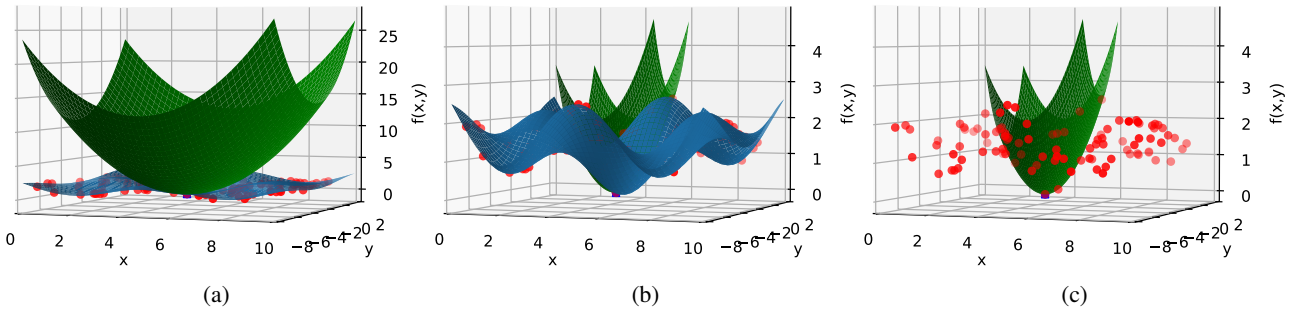


Figure 3: Three surrogate approximations (green surface) of the modified Griewank function using increasingly more local surrogates with the surrogate minimum converging to the global minimum as the number of random samples increase (red).

tain a 512-dimensional feature vector for each image.

ImageNet: For ImageNet, we used an open set benchmark subset provided by Abhinav Shrivastava [24] from the DARPA SAIL-ON research program [23]. To overcome memory and training time constraints, we randomly selected 40 out of the 413 known classes provided and retained the same unknown class partition. We used a 512-dimensional deep feature space representation of the images derived from the penultimate layer of ResNet-34 [16] and applied principal component analysis to reduce the feature size to 250 components.

3.4. Method Specifications

The HPOBench set of models were run for 500 iterations, repeated across 5 separate seeds, and the results were averaged. Each iteration refers to a single cycle of the optimization process where the hyperparameter search algorithm evaluates a set of hyperparameters and their corresponding model performance on the given task or dataset. In the case of the automatic target recognition (ATR) benchmark, due to its complex and computationally intensive nature, we conducted a single run for 100 iterations. We made this decision based on the considerable time and resources required for each iteration in this particular task. We compared the performance of random sampling of the hyperparameters against `NCQS` fit to random samples to assess the impact of our proposed approach in boosting the performance and efficiency of random sampling alone.

Tree-structured parzen estimators (TPE) had additional configuration settings known as "meta-hyperparameters" which control how the optimization algorithm operates. In TPE, the algorithm maintains two separate probability density functions (PDFs) for the hyperparameters: one for the hyperparameters leading to good performance (top-k) and another for those leading to poor performance. The "top-k" refers to the top-performing configurations. The TPE algorithm then generates new candidate samples from the PDF representing the hyperparameters with good performance, enabling it to focus on promising regions of the hyperparameter space. For the experiments using the TPE algorithm, we set aside 20% of the top-performing hyperparameter configurations (referred to as the "top-k mixture model") to influence TPE's decision-making process. Additionally, we started the TPE algorithm by randomly sampling 20 hyperparameter configurations to explore the search space initially. Afterward, we generated 10 new candidate hyperparameter samples based on the information collected so far from the top-performing configurations. These settings were carefully chosen to strike a balance between exploring the hyperparameter space effectively and exploiting promising configurations that show good performance while taking computational resources into account.

`NCQS` also has meta-hyperparameters listed in Algorithm 1. For the open set recognition experiments we use 50 samples for the surrogate approximation ($W = 50$) motivated by the need to explore a larger portion of the hyperparameter space, given the challenges posed by unknown classes in the open set recognition tasks. These samples are obtained using the sampling methodology that is compared against to assess how effectively `NCQS` boosts the performance and efficiency of the base strategy. For the closed set experiments, we used 20 samples to ensure a comprehensive search while maintaining computational efficiency. In certain experiments involving datasets with reduced numbers of trials, such as MNIST and ImageNet, approximating the surrogate model became more complex. To address this, we set the initial distance threshold to $D = 0.5$ to effectively localize the region of interest over the objective function. We further employed the distance decay introduced in Algorithm 1 to adaptively adjust the region of interest localization. In contrast, for all other experiments, we set $D = 1$ and allowed the distance decay to control the surrogate model's localization.

3.5. Evaluation Measures

The performance of the hyperparameter optimization methods is assessed using different evaluation measures depending on the type of experiments conducted.

For the closed set machine learning benchmarks, the optimizer's performance is evaluated based on the validation performance, which is the objective value seen by the optimizer during the search process. This objective value was minimized using 1-accuracy, and a summary of the results across all benchmark experiments is reported.

To illustrate the performance of each method in the closed set experiments, we use the concept of *simple regret*. The *simple regret* at iteration t is computed as:

$$R_t = f(x_{\text{best-so-far}}) - \min_{i=1}^{t} f(x_i), \tag{4}$$

where R_t is the regret at iteration t, $f(x_{\text{best-so-far}})$ is the minimum value found so far in the optimization process, and $\min_{i=1}^{t} f(x_i)$ is the global minimum for that dataset, obtained by taking the minimum value across all benchmarks and methods for that dataset. This measure provides insights into how close the algorithm is to finding the best hyperparameter configuration as the optimization progresses.

In the automatic target recognition (ATR) experiment, we illustrate the performance with the raw observed validation performance scores over 100 iterations. The objective in this benchmark is the validation box loss, which assesses the accuracy of the predicted bounding box in comparison to the ground-truth bounding box.

For the open set experiments, the Extreme Value Machine (EVM) is trained to minimize the *loss* on a validation set containing samples from both known and unknown classes within the specified number of search trials. This *loss* is defined as the negative f-measure with *weighted* averaging. The best set of hyperparameters found through the search is evaluated on a separate test set, and the resulting *loss* and accuracy of the EVM on this test set are reported. To ensure clarity, the performance *loss* is reported as the positive $F1$ score for both validation and testing.

All data used for experimentation, including data splits and scripts for training, validation, and testing, are included in the supplemental material, providing transparency and reproducibility for the evaluation process.

4. Results

Figures 4–6 plot the average regret for each of the methods across five separate runs for each dataset experiment. The dataset IDs on the plots correspond to the OpenML dataset ID. For the SVM benchmark, NCQS not only improved the best observed loss for almost all the experiments (Table 2), but also converged quicker to a better optimum on all four of the datasets. By converging faster to better optima, NCQS reduces the number of iterations needed to achieve good performance, leading to more efficient hyperparameter optimization. This efficiency is crucial in practical applications where computational resources and time are limited, allowing practitioners to make the most of their available resources while achieving competitive performance. A similar trend can be seen in the Random Forest benchmarks results. In the Logistic Regression benchmark, NCQS demonstrates improved performance in some of the experiments; however, it does not consistently boost the convergence of the methods (Figure 6).

Figure 7 displays a raw plot of the objective validation box loss observed over 100 iterations, comparing a ran-

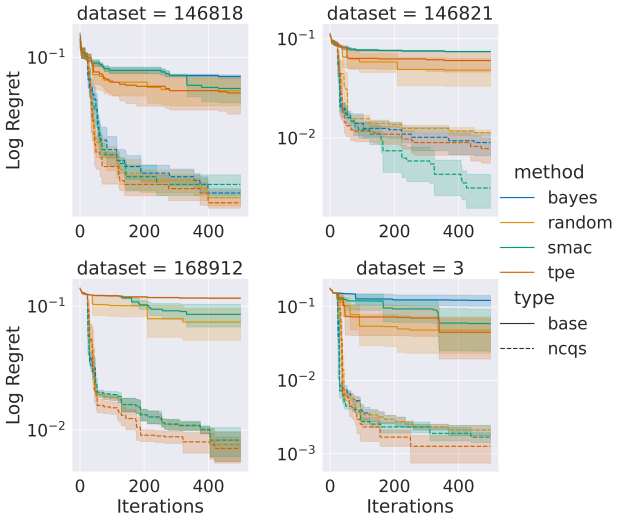


Figure 4: Comparison of all methods on 4 different tasks for the **SVM Benchmark** from the HPOBench suite. The mean and standard error of the regret at each iteration are displayed across 5 repetitions.

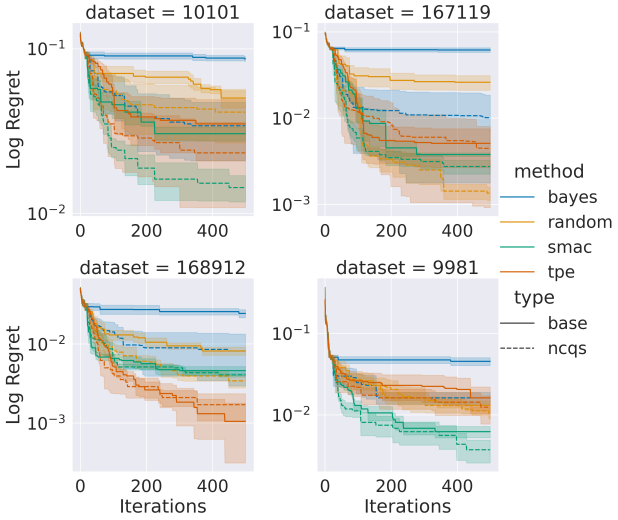


Figure 5: Comparison of all methods on 4 different tasks for the **Random Forest Benchmark** from the HPOBench suite. The mean and standard error of the regret at each iteration are displayed across 5 repetitions.

Table 1: Average percent improvement for the best observed loss and standard error over five trials of NCQS over the base methodology for each dataset and method in the SVM benchmark. Bolded values indicate where NCQS outperforms the base methodology.

Dataset	Method			
	bayes	**random**	**smac**	**tpe**
146818	**60.95** ± 0.95	**61.90** ± 1.51	**59.05** ± 2.43	-56.00 ± 4.00
146821	**87.57** ± 3.27	**83.43** ± 1.07	**95.68** ± 1.62	**89.19** ± 3.08
168912	**65.89** ± 1.26	**60.65** ± 1.45	**62.12** ± 1.46	**66.21** ± 1.04
3	**97.71** ± 0.24	**90.32** ± 1.02	**98.02** ± 0.19	**72.50** ± 6.12

Table 2: Average percent improvement for the best observed loss and standard error over five trials of NCQS over the base methodology for each dataset and method in the Random Forest benchmark. Bolded values indicate where NCQS outperforms the base methodology.

Dataset	Method			
	bayes	**random**	**smac**	**tpe**
10101	**30.24** ± 7.18	**11.18** ± 9.46	**13.57** ± 2.08	-9.17 ± 11.59
167119	**38.51** ± 6.57	**33.64** ± 0.22	**1.61** ± 0.69	-6.21 ± 5.19
168912	**54.06** ± 12.37	**30.62** ± 3.03	**7.20** ± 1.50	-15.29 ± 5.76
9981	**48.33** ± 7.17	-10.00 ± 12.75	-10.00 ± 18.71	**0.00** ± 30.33

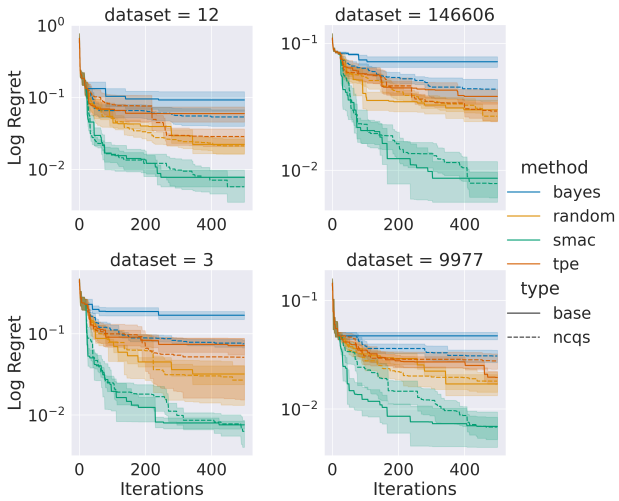


Figure 6: Comparison of all methods on 4 different tasks for the **Logistic Regression Benchmark** from the HPOBench suite. The mean and standard error of the regret at each iteration are displayed across 5 repetitions.

dom search with NCQS fit to random samples. Both methods converge to an optimal solution within 40 iterations; however, NCQS outperforms random search with a validation box loss of **0.001267**, while random search results in a higher loss score of **0.056445**.

In Table 4, we present the validation $F1$ score, testing $F1$ score, and accuracy for the open set experiments. The results for NCQS are generally comparable to both random search and TPE. In the MNIST experiment, NCQS identifies better hyperparameters for both validation and testing

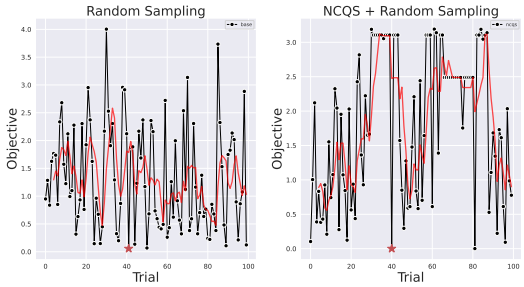


Figure 7: Validation box loss per trial for random sampling (left) and NCQS (right). Red line represents a smoothed line of the data with a rolling average over five points. Although both methods found an optimum in 40 iterations (red star), NCQS found a lower optimal value within the first 40 iterations.

measures (Table 4). Moreover, in the LFW experiment, NCQS achieves the highest validation score and performs on par with random search and TPE for testing. NCQS also attains an accuracy equivalent to TPE, which exceeds the performance of random search.

In the ImageNet experiment, NCQS successfully surpasses both TPE and random search and falls slightly below TPE for test score and accuracy.

5. Conclusion & Limitations

In conclusion, this paper introduces NCQS as a novel approach for hyperparameter optimization in computer vision. The key innovation lies in the use of nonlinear con-

Table 3: Average percent improvement for the best observed loss and standard error over five trials of NCQS over the base methodology for each dataset and method in the Logistic Regression benchmark. Bolded values indicate where NCQS outperforms the base methodology.

Dataset	Method			
	bayes	**random**	**smac**	**tpe**
12	-2.42 ± 24.83	-4.29 ± 21.26	-35.00 ± 33.17	**0.00** ± 24.63
146606	**8.26** ± 1.94	-2.19 ± 0.58	**2.74** ± 0.47	**1.12** ± 1.30
3	**44.48** ± 4.55	**33.42** ± 17.20	**3.12** ± 6.85	**44.93** ± 25.53
9977	**2.21** ± 4.72	**0.25** ± 5.41	**12.15** ± 2.38	-5.34 ± 5.93

Table 4: Comparison of NCQS against random sampling and TPE for the EVM experiments on MNIST, LFW, and ImageNet datasets. Metrics include VAL F1, TEST F1, and ACC. The values in bold indicate where NCQS outperforms the base methodology. Across all experiments, NCQS performs comparably or better to both TPE and random search.

Dataset	Metric	**NCQS** (Mean)	**Random** (Mean)	**TPE** (Mean)
MNIST	VAL F1	**0.919**	0.903	0.904
	TEST F1	**0.913**	0.902	0.902
	ACC	**0.914**	0.902	0.902
LFW	VAL F1	**0.968**	0.967	0.967
	TEST F1	0.959	0.959	0.959
	ACC	0.962	0.962	0.962
ImageNet	VAL F1	**0.532**	0.525	0.526
	TEST F1	0.868	0.868	**0.882**
	ACC	0.875	0.874	**0.904**

Table 5: Minimum objective validation box loss values and percent improvement for random base sampling and NCQS. NCQS demonstrates a significant improvement in the loss.

random	**NCQS**	**improvement**
0.05645	0.00127	97.75%

vex quadratic surrogate functions, constructed based on the Morse lemma, which allows for a local convex approximation that captures the general trend of the data without introducing oscillations to effectively tune hyperparameters with improved efficiency, enabling better utilization of resources and achieving higher performance in various computer vision and tabular tasks.

While NCQS demonstrates strong performance across various examples, it does have certain limitations that should be considered:

Dependency on Neighborhood of Global Minimum: As shown in Figure 1, NCQS requires sample points in the vicinity of the global minimum to converge effectively. Without a sufficient number of such samples, its behavior resembles that of random search. The selection methodology for sampling hyperparameters may impact this behavior. Developing more intelligent sampling strategies, such as weighted sampling based on performance scores, could lead to better approximations and improved results.

Single Minimum Surrogate Model: NCQS relies on a convex quadratic approximation, leading to surrogate models with only one minimum. This could limit its performance on loss functions with multiple optimal points, as it may converge to a local minimum instead of a global one.

Meta Hyperparameter Tuning for NCQS: The method introduces additional hyperparameters within its own framework, which may require fine-tuning for different models and datasets. Properly configuring these hyperparameters is essential to ensure optimal performance.

Convergence to Global Minimum: While NCQS aims to approximate the objective function effectively, its quadratic approximation cannot guarantee convergence to a global minimum. Ensuring convergence often requires a sufficient number of warm-up samples to navigate the hyperparameter space effectively.

Selection of Local Region for Model Building: The current methodology for selecting the local region to construct the surrogate model may lead to failure if hyperparameters change significantly on different length scales. Addressing this issue and developing more robust region selection techniques could improve NCQS' performance.

Trade-off with Increasing Dimensions: As the number of dimensions grows, there may be a trade-off between the number of local points used for quadratic approximation and the number of points needed to estimate the model parameters. This balance becomes critical as the dimensionality of the hyperparameter space increases.

Addressing these limitations and exploring ways to enhance the methodology would contribute to the continued improvement and applicability of NCQS across various optimization problems. As with any optimization approach, understanding the potential challenges and constraints is essential for leveraging the method effectively in different scenarios.

References

[1] Vítor Albiero, Kai Zhang, and Kevin W Bowyer. How does gender balance in training data affect face recognition accuracy?, 2020. 4

[2] Yoshua Bengio. Gradient-Based Optimization of Hyperparameters. *Neural Computation*, 12(8):1889–1900, 08 2000. 1

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. 1

[4] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. 1

[5] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, jul 2015. 1

[6] Mohamed Amine Bouhlel, John T. Hwang, Nathalie Bartoli, Rémi Lafage, Joseph Morlier, and Joaquim R.R.A. Martins. A python surrogate modeling framework with derivatives. *Advances in Engineering Software*, 135:102662, 2019. 2

[7] Defense Systems Information Analysis Center. Atr algorithm development image database. https://dsiac.org/databases/atr-algorithm-development-image-database/. 4

[8] Souvik Chakraborty and Rajib Chowdhury. Multivariate function approximations using the d-morph algorithm. *Applied Mathematical Modelling*, 39(23):7155–7180, 2015. 2

[9] Kai-Bo Duan and S. Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In *Proceedings of the 6th International Conference on Multiple Classifier Systems*, MCS'05, page 278–285, Berlin, Heidelberg, 2005. Springer-Verlag. 1

[10] Katharina Eggensperger, F. Hutter, H. Hoos, and Kevin Leyton-Brown. Surrogate benchmarks for hyperparameter optimization: Semantic scholar, Jan 1970. 1

[11] Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, René Sass, Aaron Klein, Noor Awad, Marius Lindauer, and Frank Hutter. Hpobench: A collection of reproducible multi-fidelity benchmark problems for hpo. *arXiv preprint arXiv:2109.06716*, 2021. 3, 4

[12] Wei Fang, Lin Wang, and Peiming Ren. Tinier-yolo: A real-time object detection method for constrained environments. *IEEE Access*, 8:1935–1944, 2019. 3

[13] Matthias Feurer, Jan N Van Rijn, Arlind Kadra, Pieter Gijsbers, Neeratyoy Mallik, Sahithya Ravi, Andreas Müller, Joaquin Vanschoren, and Frank Hutter. Openml-python: an extensible python api for openml. *The Journal of Machine Learning Research*, 22(1):4573–4577, 2021. 4

[14] Robert B. Gramacy and Herbert K. H. Lee. Cases for the nugget in modeling computer experiments. *Stat. Comput.*, 22(3):713–722, 2012. 2

[15] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European conference on computer vision*, pages 87–102. Springer, 2016. 4

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[17] J. Kinnison, N. Kremer-Herman, D. Thain, and W. Scheirer. Shadho: Massively scalable hardware-aware distributed hyperparameter optimization. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 738–747, 2018. 4

[18] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, André Biedenkapp, and Frank Hutter. Smac v3: Algorithm configuration in python. https://github.com/automl/SMAC3, 2017. 3

[19] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning, 2015. 1

[20] J. Milnor. *Morse theory*. Annals of Mathematics Studies, No. 51. Princeton University Press, Princeton, N.J., 1963. Based on lecture notes by M. Spivak and R. Wells. 2

[21] Ethan M Rudd, Lalit P Jain, Walter J Scheirer, and Terrance E Boult. The extreme value machine. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):762–768, 2017. 3

[22] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2012. 3

[23] Ted Senator. Science of artificial intelligence and learning for open-world novelty (sail-on). 5

[24] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016. 5

[25] Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Griewank function, 2013. 3

[26] Weicheng Xie, Wenting Chen, Linlin Shen, Jinming Duan, and Meng Yang. Surrogate network-based sparseness hyperparameter optimization for deep expression recognition. *Pattern Recognition*, 111:107701, 2021. 1

[27] Yudong Zhang, Shuihua Wang, and Genlin Ji. A comprehensive survey on particle swarm optimization algorithm and its applications, Oct 2015. 1