Probabilistic Sequential Multi-Objective Optimization of Convolutional Neural Networks

Zixuan Yin
McGill University
Montreal, Canada
zixuan.yin@mail.mcgill.ca

Warren Gross
McGill University
Montreal, Canada
warren.gross@mcgill.ca

Brett H. Meyer
McGill University
Montreal, Canada
brett.meyer@mcgill.ca

***Abstract*—With the advent of deeper, larger and more complex convolutional neural networks (CNN), manual design has become a daunting task, especially when hardware performance must be optimized. Sequential model-based optimization (SMBO) is an efficient method for hyperparameter optimization on highly parameterized machine learning (ML) algorithms, able to find good configurations with a limited number of evaluations by predicting the performance of candidates before evaluation. A case study on MNIST shows that SMBO regression model prediction error significantly impedes search performance in multi-objective optimization. To address this issue, we propose *probabilistic SMBO*, which selects candidates based on probabilistic estimation of their Pareto efficiency. With a formulation that incorporates error in accuracy prediction and uncertainty in latency measurement, probabilistic Pareto efficiency quantifies a candidate's quality in two ways: its likelihood of being Pareto optimal, and the expected number of current Pareto optimal solutions that it will dominate. We evaluate our proposed method on four image classification problems. Compared to a deterministic approach, probabilistic SMBO consistently generates Pareto optimal solutions that perform better, and that are competitive with state-of-the-art efficient CNN models, offering tremendous speedup in inference latency while maintaining comparable accuracy.**

I. Introduction

As modern CNNs become deeper, larger and more complex [1]–[3], hardware efficiency has emerged as a challenging problem in both real-world and online deployment of CNNs [4]. Energy, inference delay, and memory footprint are all key metrics for resource-constrained hardware and applications, but are highly sensitive to the configuration of a large number of CNN hyperparameters.

Design automation (i.e., meta-learning, or AutoML) is one promising technology for optimizing CNN. Design automation has recently achieved major success on various computer vision problems, outperforming CNN hand-crafted by domain experts on image classification and object detection [5]. Previous automation approaches, however, tend to focus on improving accuracy; hardware performance is less often addressed [6], [7]. Furthermore, modern CNN hyperparameter optimization techniques require a tremendous amount of computing power and time to explore and evaluate CNN configurations. For instance, NASNet [5] trained and evaluated 20,000 networks across 500 GPUs over four days. Search inefficiency remains a key limitation preventing the adoption of automated CNN design frameworks by practitioners.

This research was supported by funding from the Natural Science and Engineering Research Council of Canada.

Sequential model-based optimization (SMBO) is an efficient method for hyperparameter optimization on highly parameterized algorithms. SMBO finds good configurations with a limited number of evaluations by using a regression model or a surrogate function to predict the performance of a candidate solution before evaluation [8]. SMBO has been employed in CNN hyperparameter optimization [9] and neural architecture search (NAS) [10], [11]. One challenge is that, given the sparsity of learning examples and the large search space, there is often error in candidate performance estimation. Inaccurate predictions lead to the evaluation of unfit configurations, wasting time and money; furthermore, promising configurations may be passed over. Model prediction error is a key search efficiency bottleneck in SMBO.

In this paper, we propose a heuristic strategy to improve search efficiency in the context of multi-objective model-based optimization. Our goal is to find efficient and accurate CNN hyperparameter configurations, *e.g.*, the number of convolutional (Conv) layers, number of fully-connected (FC) layers, number of filters, kernel size and stride. We formulate the problem as device-aware multi-objective optimization, aiming at finding a set of Pareto optimal solutions with the best accuracy-inference latency trade-offs by utilizing a meta-network that learns to predict the testing accuracy of candidate solutions. By incorporating error in accuracy prediction and uncertainty in latency measurement, we make probabilistic inference on candidates' Pareto efficiency in two regards: 1. their likelihood to become a Pareto optimal solution and 2. the expected number of current Pareto optimal solutions that will be dominated by them. In each iteration, our novel meta-heuristic algorithm compares a list of candidates and selects the candidate that will best improve the current collection of Pareto optimal solutions (Pareto front) based on the two estimations. In this way, we alleviate the effect of inaccurate meta-network prediction and utilize the meta-network more efficiently by comparing candidates' probabilistic Pareto efficiency under uncertainties.

We evaluate our probabilistic approach on four image classification problems and find consistent improvement over deterministic approaches. On average, the Pareto optimal solutions found by probabilistic SMBO dominates 80% of the

Pareto optimal solutions found by deterministic SMBO, while achieving 57% improvement in terms of the hypervolume [12] of Pareto optimal fronts (POF). Furthermore, our searched models are also competitive with manually designed state-of-the-art efficient CNN models DenseNet [13] and MobileNetV2 [14], able to achieve up to $27\times$ speedup in inference latency while maintaining comparable accuracy.

II. Related Work

One common approach for accelerating automated CNN design is weight sharing. Cai et al. [15] explore the architecture space using network transformation/morphism [16], which modifies a trained network using function-preserving transformations. Brock et al. [17] propose a one-shot model architecture search method to bypass training child networks by training an auxiliary model to generate the weights of a candidate based on its architecture. Han et al. [18] construct an over-parameterized network, then select a candidate network through path-level pruning. Another common approach is to use cheaper proxy metrics to estimate a candidate's performance. Instead of searching for an entire network using a large dataset, Zoph et al. [5] search for small building blocks using a smaller dataset and construct full networks by connecting building blocks in series. Alternatively, candidates' performance can also be estimated without training by using surrogate functions or regression models to predict performance based on network configuration [9]–[11]. In this work, we adopt the SMBO method to estimate a candidate's performance without training. Specifically, we focus on improving candidate selection efficiency in the presence of accuracy estimation error, in order to find better results using the same amount of evaluations.

Most existing automation methods focus on optimizing a single objective: prediction accuracy on a given dataset [19]. However, device-related requirements such as inference latency are just as important for CNNs deployed to constrained devices. Stamoulis et al. [20] propose HyperPower, which uses Bayesian optimization in the context of power- and memory-constrained optimization and improves search efficiency by avoiding configurations that violate the memory or power budget. However, the best trade-offs are not sought for when constraints are satisfied. With multi-objective optimization, hardware performance and prediction accuracy can be optimized simultaneously, revealing new trade-offs.

Smithson et al. [9] use a weighted sum of FLOPs and weight count as a proxy for computational cost and used a SMBO algorithm to search for Pareto-optimal solutions. Kim et al. [21] apply an evolutionary algorithm to search for models with two objectives, inference speed and classification accuracy. Tan et al. [22] propose a platform-aware NAS with reinforcement learning that optimizes for inference latency on a mobile phone along with accuracy. Dong et al. [11] consider both device-agnostic metrics (*e.g.*, accuracy and model size) and device-aware metrics (*e.g.*, inference latency and memory usage) when searching for Pareto optimal architectures. While [22] and [21] use a preference criteria (fitness or reward) to rank or select candidates, [9] and [21] select a candidate if it is estimated to be Pareto optimal. In this work, we select candidates based on probabilistic Pareto efficiency to maximize the improvement with each evaluation.

III. Approach

A. Meta-network Prediction Error

When optimizing CNN for multiple metrics, often no single design simultaneously optimizes all objectives. Instead, a number of Pareto optimal solutions achieve the best trade-offs between objectives and form the Pareto front. Single-objective optimization can solve the multi-objective problem by combining the objectives in a single preference criteria [22]. However, such an approach can not guarantee diverse trade-offs in the solutions found. On the other hand, finding the Pareto optimal solutions exposes all interesting trade-offs for consideration.

In this work we adopt an SMBO algorithm to perform Pareto optimization on CNN [9], [11]. In each iteration, we select candidates for evaluation that achieve Pareto optimality compared to previously explored candidates (or skip a sampled candidate if it is not Pareto optimal). A meta-network is used to predict candidate accuracy.

Our experiment on the MNIST dataset [23] shows that the prediction error of the meta-network has a significant impact on the search efficiency of SMBO. In a small search space of 4,352 CNNs, the SMBO algorithm was able to find all of the ten **true** Pareto optimal solutions in 46 iterations using a meta-network that predicts candidates' accuracy perfectly (Oracle network). In contrast, we yielded only two solutions in the true Pareto front in 100 iterations when training a meta-network iteratively to predict the accuracy of candidates, despite a low mean absolute prediction error of 0.46%. Repeating the experiment ten times results in similar discrepancies. Meta-network prediction error clearly has a big effect on the search efficiency of SMBO, leading to incorrect inference about candidates' Pareto optimality.

Inspired by Bayesian methodology, we propose to account for uncertainties in meta-network predictions to make probabilistic inference on Pareto optimality when selecting candidates. In the same search space, our probabilistic approach achieves better search performance and on average finds three more solutions in the true Pareto front in 100 iterations.

B. Probabilistic Pareto Efficiency

Given a model m, let $E(m)$ denote its error on the target task, and $L(m)$ denote its average inference latency measured on the processing platform. The goal of multi-objective optimization is to find all Pareto optimal solutions in the search space M. With two objectives, the solution set is partially ordered by a *Pareto dominance* relation. Given two solutions $m, n \in M$, m Pareto dominates n ($m \prec n$) when:

$$\begin{aligned} E(m) < E(n) \land L(m) \leq L(n) \quad \lor \\ E(m) \leq E(n) \land L(m) < L(n) \end{aligned} \tag{1}$$

Given a set of models explored $\Omega \subseteq M$, the current Pareto optimal set $\Lambda = \Psi(\Omega)$ is defined as:

$$\Psi(\Omega) = \left\{ m \in \Omega \,\middle|\, \nexists n \in \Omega \text{ s.t. } n \prec m \right\} \quad (2)$$

The projection of $\Lambda = \Psi(\Omega)$ in the objective space is the *Pareto front* associated with Ω, whereas $\Pi = \Psi(M)$ forms the *true Pareto front* in the objective space.

For a feasible candidate $x \in M \wedge x \notin \Omega$, its performance vector $v = f(x) = [\mathcal{E}_x, \mathcal{L}_x]$ is a 2-dimensional random vector uniformly distributed in a rectangular area $R_{\mathcal{V}}$ centered at point $(\,\hat{E}(x), L(x)\,)$ (see candidate selection in Figure 1):

$$R_{\mathcal{V}} = [\hat{E}(x) - e, \hat{E}(x) + e] \times [L(x) - l_x, L(x) + l_x] \quad (3)$$

where $\hat{E}(x)$ is the estimated performance of x, e is the mean absolute prediction error of the meta-network, and l_x is half the width of the 95% confidence interval calculated from measured inference latency. We then assess x by comparing it to each Pareto optimal solution $\lambda \in \Lambda$ in two ways: 1. the probability of x not being dominated by λ ($\Pr(\lambda \not\prec x)$) and 2. the probability of x dominating λ ($\Pr(x \prec \lambda)$). For the sake of this comparison, the latency of an evaluated Pareto solution λ is uniformly distributed in $[L(\lambda) - l_\lambda, L(\lambda) + l_\lambda]$. This accounts for variation in inference latency measurements caused by memory and communication overheads. The error of λ is fixed at $E(\lambda)$. While there is some variance in $E(\lambda)$, it is negligible compared with other sources of error, and measuring it requires multiple, costly, training runs.

We can derive the probability that x is not dominated by Λ:

$$\Pr(\Lambda \not\prec x) = \prod_{\lambda \in \Lambda} \Pr(\lambda \not\prec x). \quad (4)$$

The expected number of models in Λ to be dominated by x is:

$$\mathrm{E}[|\Lambda'_x|] = \sum_{\lambda \in \Lambda} \Pr(x \prec \lambda) \text{ where } \Lambda'_x = \{\lambda' \in \Lambda | x \prec \lambda'\} \quad (5)$$

Finally, the probabilistic Pareto efficiency of a given candidate x evaluated against the current Pareto optimal front Λ is:

$$\Gamma(x|\Lambda) = \Pr(\Lambda \not\prec x) + \mathrm{E}[|\Lambda'_x|] \quad (6)$$

Assuming that the current Pareto set Λ contains n models, the probabilistic Pareto efficiency $\Gamma(x|\Lambda)$ has a range $[0, n+1]$. Notice that the value of probabilistic Pareto efficiency allows candidates to be completely ordered as opposed to partially ordered based on deterministic Pareto optimality. Furthermore, previous SMBO approaches simply search for candidates that are not dominated by current POF solutions [9], [11]. With probabilistic Pareto efficiency, we can compare a group of candidates and select the one that has the best chance of improving the current POF.

C. Probabilistic SMBO Algorithm

During the multi-objective optimization, the SMBO algorithm iterates between fitting a response surface model (meta-network) and using it to make choices about which candidate to evaluate next. To initialize the response surface model and the Pareto optimal set, we evaluate a small number of candidates randomly sampled from the CNN search space at the beginning of a search process. After the initialization step, the general procedure of our proposed probabilistic SMBO can be broken down into the following steps:

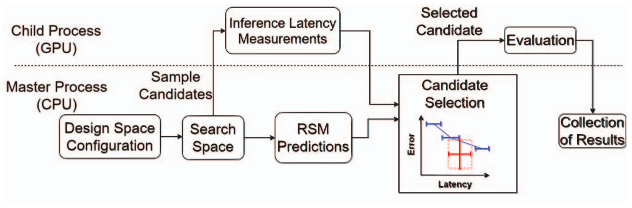


Fig. 1: Probabilistic SMBO flowchart. For candidate selection using probabilistic Pareto efficiency, the red rectangle represents the performance vector of a candidate whereas the blue bars represent Pareto solutions.

- **Sample:** Sample a group of K feasible candidates from the CNN design space. For each sample x, measure the inference latency $L(x)$ and estimate the prediction error $\hat{E}(x)$ using the meta-network. Compute the probabilistic Pareto efficiency of each candidate and select the candidate with the highest probabilistic Pareto efficiency to evaluate.
- **Evaluate:** Train the selected candidate on the target task and evaluate its actual prediction error $E(x)$.
- **Update:** Add the newly obtained results to the list of evaluated solutions and retrain the meta-network.

The sample-evaluate-update loop is repeated until the maximum number of model evaluations is reached. A flowchart describing the probabilistic SMBO implementation is shown in Figure 1. The software constructs a search space based user-specified design space configurations, and uses two processes (master and child) during the optimization. The master process runs the main search algorithm and delegates CNN-related tasks (*i.e.*, measuring inference latency, training and testing) to the child process, which has exclusive access to a GPU. After each finished task, the child process clears the GPU memory to ensure accurate latency measurements. Upon completion, all collected results (e.g., model configurations and weights) are saved for future deployment and re-use.

IV. Experimental Setup

We conduct experiments on four image classification datasets to demonstrate the effectiveness of probabilistic Pareto efficiency in multi-objective SMBO. The performance of our proposed SMBO algorithm is compared to the results obtained from multi-objective SMBO with a deterministic candidate selection policy on same search spaces. We also compare against manually designed, state-of-the-art, efficient CNN models, DenseNet [13] and MobileNetV2 [14].

The deterministic SMBO implementation follows the candidate selection policy proposed in [9]. During the sampling stage, a new candidate is sampled from a Gaussian distribution centered around the previously explored solution; it

TABLE I: Hyperparameter settings for each dataset.

Parameter	CIFAR-10	SVHN & GTSRB	Dogs vs. Cats
Conv depth	$6, 7, \dots, 10$	$6, 7, \dots, 12$	$6, 7, \dots, 12$
Conv features	$32, 48, \dots, 128$	$32, 48, \dots, 128$	$32, 40, \dots, 128$
Conv kernel size	$3, 5, 7$	$3, 5, 7$	$3, 5, 7$
Conv strides	$1, 2$	$1, 2$	$1, 2$
FC depth	$0, 1, 2$	$0, 1, 2$	$0, 1, 2$
FC units	$2^6, 2^7, 2^8$	$2^4, 2^5, ..., 2^8$	$2^4, 2^5, ..., 2^8$
FC dropout rate	$0, 0.1, 0.2$	$0, 0.1, 0.2$	$0, 0.1, 0.2$
Batch size	64	64	$40, 60, 80, 100$
Epochs	40	20	25
Learning rate	0.005	0.1	0.008

has $1 - 10^{-4}$ probability of being accepted if predicted to be Pareto optimal or 10^{-4} probability of being accepted if predicted to be dominated by the current Pareto front. The sampling process continues until a new candidate is accepted for evaluation.

We perform hyperparameter optimization on CNN topologies derived from cost-efficient MobileNets [24], which replace standard convolutions with depthwise separable convolutions in the hidden layers. Depthwise separable convolutions factorize a standard convolution into a depthwise convolution and a pointwise convolution. Each convolution operation is followed by batch normalization [25] and ReLU activation. A depthwise separable convolution operation is counted as one layer in terms of Conv depth, with tunable Conv kernel size and Conv stride in depthwise convolution and tunable number of output features (channels) of pointwise convolution. Down sampling of feature maps is handled with strided convolution in the depthwise convolutions or the input layer. To predict candidates' error rate, we adopt the meta-network implementation and its input encoding scheme from [9].

We picked four datasets with diverse characteristics and application purposes. Each dataset presents a different challenge to the multi-objective CNN design.

CIFAR-10. The CIFAR-10 [26] dataset consists of 32×32 RGB images from 10 categories. The training and testing sets contain 50,000 and 10,000 images respectively.

SVHN. The Street View House Numbers (SVHN) dataset [27] contains 32×32 RGB images of cropped house number digits from street view images. There are 73,257 images for training and 26,032 images for testing.

GTSRB. The German Traffic Sign Recognition Benchmark [28] contains 43 classes of colored traffic sign images. The training and testing sets contain 39,209 and 12,630 images respectively. We resize all images to 32×32 pixels.

Dogs vs. Cats. This dataset contains 25,000 colored images of dogs and cats [29]. We resize the images to 224×224 and randomly split the full dataset into 20,000 images for training and 5,000 images for testing.

We train and evaluate CNNs on three different GPUs: the CIFAR-10 experiments are conducted on an NVIDIA K20 GPU; the SVHN and GTSRB experiments are run on one GK210 GPU out of the pair available in an NVIDIA K80; the Dogs vs. Cats experiments are run on an NVIDIA GTX 1080 Ti. Table I details all hyperparameters along with their possible values for each dataset.

TABLE II: Evaluation of hypervolume and coverage rate.

Dataset	$hv(\Lambda_P)$	$hv(\Lambda_D)$	$\mathcal{C}(\Lambda_P, \Lambda_D)$	$\mathcal{C}(\Lambda_D, \Lambda_P)$
CIFAR-10	**0.35**	0.63	**100%**	0%
SVHN	**0.23**	0.50	**82%**	0%
GTSRB	**0.16**	0.45	**80%**	0%
Dogs vs. Cats	**0.25**	0.75	**60%**	0%

Searches using small datasets (CIFAR-10, SVHN, and GTSRB) evaluate 100 models, including five random initial evaluations. For the Dogs vs. Cats dataset, training takes significantly longer than smaller datasets. Therefore, we evaluate 30 models per experiment with three initial evaluations. We measure inference latency with a batch size of 32 and report the average latency per inference from 25 measurements. All CNN models are trained using the Adam optimizer [30] with categorical cross entropy loss without data augmentation. To balance the runtime between sampling candidates and evaluating selected candidates, we set the number of feasible network samples K to 200 for probabilistic SMBO. In Section V-C we assess the effect of K on solution quality and optimization efficiency.

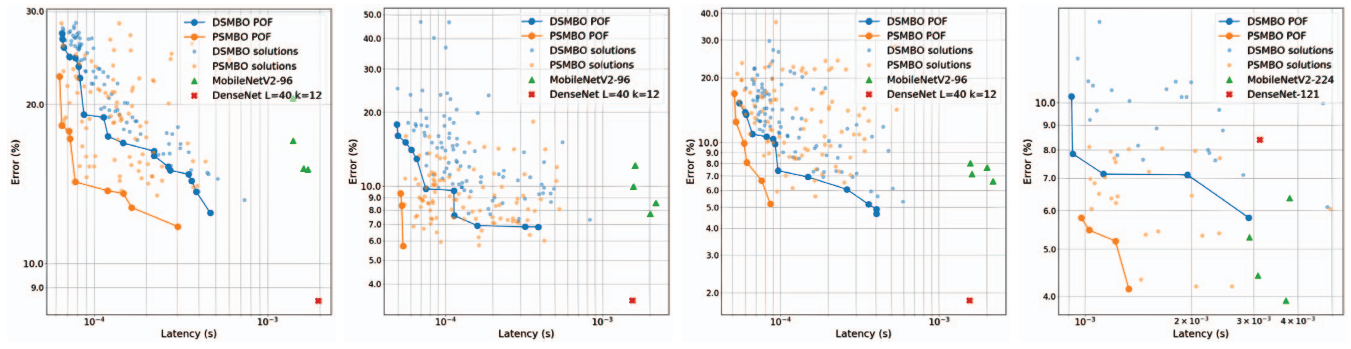
V. Results

Figure 2 shows the results of probabilistic and deterministic SMBO as well as their performance in comparison with MobileNetV2 and DenseNet on each dataset.

A. Comparison with Deterministic SMBO

We use *hypervolume hv* and *coverage* $\mathcal{C}$ metrics to measure the relative quality of one solution set with respect to another [12], [31]. The hypervolume *hv* is the normalized dominating area under a given POF in the objective space, where at least one Pareto solution is dominated. A better Pareto optimal front should have a lower hypervolume. The coverage function $\mathcal{C}$ measures the percentage of solutions in a set $X' \subseteq M$ which is dominated by solutions in another set $X'' \subseteq M$. $\mathcal{C}(A, B) = 1$ corresponds to the case when all the solutions in B are dominated by those in A, whereas $\mathcal{C}(A, B) = 0$ represents the case when none of the solutions in B are dominated by those in A.

Table II lists the hypervolumes and coverage rates of the Pareto optimal sets of probabilistic and deterministic SMBO (Λ_P, Λ_D respectively). Both metrics show that probabilistic SMBO consistently outperforms deterministic SMBO on all four datasets. In particular, Λ_P entirely dominates Λ_D on CIFAR-10 and achieves 67% improvement in terms of hypervolume on GTSRB and Dogs vs. Cats. On average, Λ_P reduces hypervolume by 57% over Λ_D. It is also noteworthy that Λ_P dominates the majority of solutions in Λ_D, while none of the solutions in Λ_P are dominated by those in Λ_D ($\mathcal{C}(\Lambda_D, \Lambda_P) = 0$), a clear display of the superior search efficiency of probabilistic SMBO across all datasets: given the same budget for evaluations, probabilistic SMBO finds ample solutions that surpass the Pareto front of deterministic SMBO, at times in dramatic fashion (e.g., SVHN and Dogs vs. Cats in Figure 2).



(a) CIFAR-10 (b) SVHN (c) GTSRB (d) Dogs vs. Cats

Fig. 2: Experimental results on image classification datasets.

TABLE III: CNN performance results. Inference latency are provided in unit of millisecond.

	CIFAR-10		SVHN		GTSRB		Dogs vs. Cats	
	E	*L*	*E*	*L*	*E*	*L*	*E*	*L*
Prob. SMBO	11.7	**0.3**	5.7	**0.054**	5.2	**0.086**	4.1	**1.3**
MNV2-1.0	15.1	1.7	8.5	2.2	6.6	2.2	**3.9**	3.8
MNV2-0.75	15.1	1.6	7.7	2.0	7.6	2.0	6.3	3.7
MNV2-0.5	17.1	1.4	12.2	1.6	7.1	1.6	4.4	3.1
MNV2-0.35	20.5	1.4	10.0	1.5	8.0	1.5	5.3	2.9
DenseNet	**8.5**	2.0	**3.4**	1.5	**1.8**	1.5	8.4	3.1

B. Comparison with Manual Designs

To further validate the effectiveness of probabilistic SMBO, we compare the performance of its solutions to hand-crafted efficient CNNs MobileNetV2 [14] and DenseNet [13] after the same number of epochs of training. Since both designs come in different-sized variants, we selected the variants whose inference speed are closer to the level of our searched models'. On small image datasets, we selected MobileNetV2 models with the lowest input resolution, 96 (images are resized accordingly), and width multipliers of 0.35, 0.5, 0.75, 1; for DenseNet we selected an efficient model with number of layers $L = 40$ and growth rate $k = 12$. On the Dogs vs. Cats dataset, we deployed variants of MobileNetV2 with input resolution of 224 and the same width multipliers; we also deployed DenseNet-121, which is designed for 224×224 images. We follow the training procedures in the literature, except for MobileNetV2 on Dogs vs. Cats, where we fine-tuned from weights pre-trained on ImageNet; training from scratch results in much lower accuracy.

Figure 2 illustrates the results. Compared to MobileNetV2, models generated by probabilistic SMBO offer tremendous speedup with similar or better accuracy. Our searched models strike a trade-off with DenseNet on small image datasets and dominate DenseNet on Dogs vs. Cats. For a more comprehensive comparison, we select the most accurate CNN models generated by probabilistic SMBO and compare their performance to manual designs in Table III. On the CIFAR-10 dataset, our model runs $\mathbf{6.6}\times$ faster than DenseNet, with 3.2% lower accuracy. On SVHN, our model only requires a fraction ($\mathbf{3.6}\%$) of the inference latency of DenseNet, with 2.3% reduction in accuracy. On GTSRB, our model achieves $\mathbf{17.4}\times$ speedup over DenseNet, with 3.4% lower accuracy. On Dogs vs. Cats dataset, our model is almost as accurate as MobileNetV2-1.0 (MNV2-1.0) but runs $\mathbf{2.8}\times$ faster. While probabilistic SMBO does not find the most accurate model, the low inference latency of the models can be very appealing to some applications such as real-time video applications.

Furthermore, results on different datasets expose some major issues with manually designed CNNs. First, manual designs' performance are inconsistent across datasets. MobileNetV2 models are more accurate on Dogs vs. Cats; DenseNet performs better on small image datasets. Probabilistic SMBO is able to adapt to different tasks and find competitive models on all datasets using relatively similar design space configurations.

Second, scaling a baseline network is not an effective way to explore trade-offs between accuracy and hardware performance. For instance, a MobileNetV2 model with a width multiplier of 0.35 (MobileNetV2-0.35) reduces the number of MACs and weights by 80% and 52% respectively over a full-size baseline model (MobileNetV2-1.0). However, the actual inference latency reduction over the baseline model is much worse than what's expected in theory, ranging from 17.6% to 31.8% on different GPUs. Recent research has shown that reducing the number of MACs or weights does not necessarily improve inference speed or energy, since device-related performance metrics are not only dependent on the CNN configuration but also the implementation of hardware and software [11], [32]. Multi-objective optimization finds better solutions than scaling manually designed baseline models.

C. Varying the Amount of Samples

TABLE IV: Time consumption (hours) with different K value.

	Total runtime	Funtional runtime	Overhead
$K = 100$	6.9	4.7 (67.6%)	2.2 (32.4%)
$K = 200$	6.3	3.2 (50.1%)	3.1 (49.9%)
$K = 400$	7.5	2.7 (36.5%)	4.8 (63.5%)

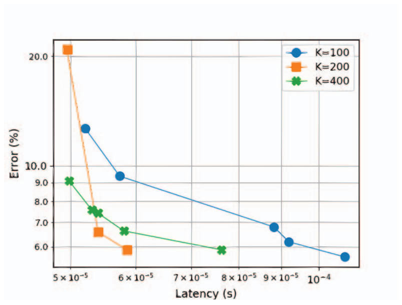


Fig. 3: Pareto fronts as a function of K on GTSRB.

Our final experiments analyze the effect on the performance and runtime of probabilistic SMBO when varying the amount of samples per iteration (K). By ranking candidates based on their probabilistic Pareto efficiency, probabilistic SMBO compares a group of candidates and picks the most promising candidate for evaluation; higher K means more designs are considered. Time spent on training and testing sampled CNN models (*evaluate* step) is functional runtime whereas time spent sampling candidates (*sample* step) or training the meta-network (*update* step) is overhead. We run three sets of experiments on GTSRB with $K = 100, 200, 400$, using the settings in Table I. A total of 50 evaluations are performed with five initial random evaluations. The resulting Pareto fronts are plotted in Figure 3. Table IV summarizes the time consumption of probabilistic SMBO with different K values.

As K increases, we observe a decrease in functional evaluation time and an increase in overhead. While the increase in overhead can be attributed to extra work in the *sample* step, the decrease in functional evaluation time is most likely due to more Pareto-efficient models being selected. Although the inference latency and training time of a CNN model are not directly correlated, our hypothesis is that models with lower inference latency on GPUs tend to consume less time for the same amount of training. As a result, the total time required with $K = 200$ is actually lower than with $K = 100$.

In terms of performance, the POFs generated with $K = 200$ and $K = 400$ clearly dominates the POF generated with $K = 100$. There is no significant improvement in search efficiency when increasing K from 200 to 400. In this case, $K = 200$ results strike the best trade-off.

VI. Conclusions

This paper presents a probabilistic approach to improve the search efficiency of SMBO in multi-objective optimization of CNNs. The key idea is to probabilistically evaluate candidates' Pareto efficiency by incorporating uncertainty in meta-network prediction and variance in latency measurement. With probabilistic Pareto efficiency, our proposed method compares a group of sampled candidates and select the one with the best chance of extending the current POF. Results on four image classification datasets demonstrate that our approach achieves superior search efficiency and finds better Pareto optimal solutions than deterministic approaches. CNN models generated by probabilistic SMBO are also competitive with state-of-the-art hand-crafted mobile CNNs, providing much lower inference latency while maintaining similar accuracy.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
[2] C. Szegedy, et al., "Going deeper with convolutions," in *CVPR*, June 2015, pp. 1–9.
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CVPR*, pp. 770–778, 2016.
[4] D. Marculescu, D. Stamoulis, and E. Cai, "Hardware-aware machine learning: Modeling and optimization," in *ICCAD*, 2018.
[5] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.
[6] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2016.
[7] E. Real, et al., "Large-scale evolution of image classifiers," in *ICML*, 2017.
[8] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *LION*, 2011.
[9] S. C. Smithson, G. Yang, W. J. Gross, and B. H. Meyer, "Neural networks designing neural networks: Multi-objective hyper-parameter optimization," *ICCAD*, 2016.
[10] C. Liu, et al., "Progressive neural architecture search," in *ECCV*, 2018.
[11] J.-D. Dong and et al., "DPP-Net: Device-aware progressive search for pareto-optimal neural architectures," in *ECCV*, 2018.
[12] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms - a comparative case study," in *PPSN V*, 1998, pp. 292–301.
[13] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, July 2017.
[14] A. M. Sandler, et al. Zhmoginov and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, June 2018.
[15] H. Cai, T. Chen, W. Zhang, and Y. Yu, "Efficient architecture search by network transformation," in *AAAI*, 2018.
[16] T. Wei, C. Wang, Y. Rui, and C. W. Chen, "Network morphism," in *ICML*, 2016.
[17] A. Brock, T. Lim, J. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through hypernetworks," in *ICLR*, 2018.
[18] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations*, 2019.
[19] A.-C. Cheng, et al., "Searching toward pareto-optimal device-aware neural architectures," in *ICCAD*, 2018.
[20] D. Stamoulis, E. Cai, D. Juan, and D. Marculescu, "Hyperpower: Power- and memory-constrained hyper-parameter optimization for neural networks," in *DATE*, March 2018.
[21] Y.-H. Kim, B. Reddy, S. Yun, and C. Seo, "Nemo: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy," in *ICML 2017 AutoML Workshop*, 2017.
[22] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *CVPR*, 2019.
[23] Y. Lecun, et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, Nov 1998.
[24] A. G. Howard, et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.
[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.
[26] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.
[27] Y. Netzer, et al., "Reading digits in natural images with unsupervised feature learning," *NIPS*, Jan 2011.
[28] J. Stallkamp, et al., "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *IJCNN*, 2011.
[29] J. Elson, et al., "Asirra: A captcha that exploits interest-aligned manual image categorization," in *ACM CCS*, Oct 2007.
[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.
[31] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, Nov 1999.
[32] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," *CVPR*, 2017.