AIRCHITECT V2: Learning the Hardware Accelerator Design Space through Unified Representations

Jamin Seo*[1], Akshat Ramachandran*[1], Yu-Chuan Chuang[2], Anirudh Itagi[1], Tushar Krishna[1]
[1]Georgia Institute of Technology, Atlanta, USA
[2]National Taiwan University, Taipei, Taiwan
[1]{akshat.r, jseo89, anirudh.itagi}@gatech.edu, tushar@ece.gatech.edu
[2]frankchuang@access.ee.ntu.edu.tw

Abstract*—Design space exploration (DSE) plays a crucial role in enabling custom hardware architectures, particularly for emerging applications like AI, where optimized and specialized designs are essential. With the growing complexity of deep neural networks (DNNs) and the introduction of advanced large language models (LLMs), the design space for DNN accelerators is expanding at an exponential rate. Additionally, this space is highly non-uniform and non-convex, making it increasingly difficult to navigate and optimize. Traditional DSE techniques rely on search-based methods, which involve iterative sampling of the design space to find the optimal solution. However, this process is both time-consuming and often fails to converge to the global optima for such design spaces. Recently, AIRCHITECT V1, the first attempt to address the limitations of search-based techniques, transformed DSE into a constant-time classification problem using recommendation networks. However, AIRCHITECT V1 lacked generalizability and had poor performance in complex design spaces. In this work, we propose AIRCHITECT V2, a more accurate and generalizable learning-based DSE technique applicable to large-scale design spaces that overcomes the shortcomings of earlier approaches. Specifically, we devise an encoder-decoder transformer model that (a) encodes the complex design space into a uniform intermediate representation using contrastive learning and (b) leverages a novel unified representation blending the advantages of classification and regression to effectively explore the large DSE space without sacrificing accuracy. Experimental results evaluated on 10^5 real DNN workloads demonstrate that, on average, AIRCHITECT V2 outperforms existing techniques by 15% in identifying optimal design points. Furthermore, to demonstrate the generalizability of our method, we evaluate performance on unseen model workloads and attain a $1.7\times$ improvement in inference latency on the identified hardware architecture. Code and dataset are available at: https://github.com/maestro-project/AIrchitect-v2.

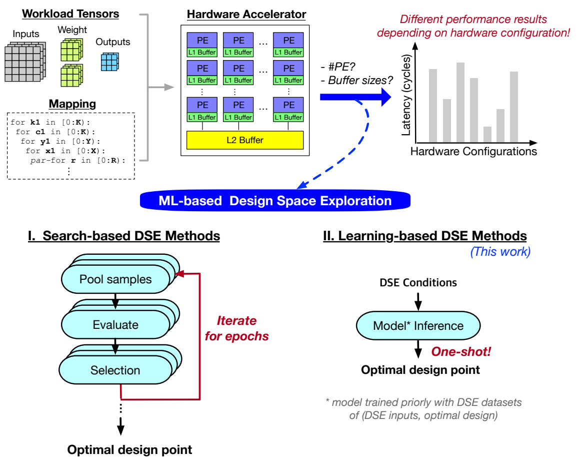
***Index Terms*—Design Space Exploration, DNN accelerator**

I. INTRODUCTION

In the rapidly evolving domain of deep neural networks (DNNs), hardware acceleration [1]–[3] is essential for the efficient deployment of models across a wide range of platforms, from cloud infrastructures to mobile and edge devices. However, the performance of DNN inference on hardware is dictated by the complex interaction between mapping strategies [4] and allocated hardware resources [5]. This complexity leads to a vast and intricate design landscape, making it difficult to explore and optimize for peak performance.

In the past, human experts have manually crafted the design choices based on their insights [6]–[8]. Such endeavors not only

*Equal contribution.



```
for k1 in [0:K):
 for c1 in [0:K):
  for y1 in [0:Y):
   for x1 in [0:X):
    par-for r in [0:R):
        :
```

Fig. 1: Different design choices yield a wide range of performance, necessitating automated design space exploration. (i) Search-based methods involve iterative exploration, (ii) learning-based methods enable one-shot inference.

require substantial time and resources but also may achieve sub-optimal solutions due to heuristics. Recently, motivated by the success of machine learning (ML) algorithms to perform complex tasks [9], [10], efforts are being made to automate DSE using ML techniques [4], [5], [11].

Mainstream methodologies using ML in DSE automation commonly involve iterative searches of samples and often rely on black-box optimization techniques. For instance, ConfuciuX [12] utilizes reinforcement learning (RL), Gamma [13], DiGamma [14] apply genetic algorithm (GA), and Hasco [15] employs Bayesian optimization (BO) for searching optimal hardware and/or mapping configurations. However, these techniques are often prohibitively time-consuming for large design spaces and the quality of identified designs is largely dependent on sampling efficiency [11], [13].

To mitigate the inefficiencies of search-based DSE, recent techniques [5], [16] propose a constant time optimization method by employing different DNN models trained/fine-tuned for DSE to predict the optimal hardware configuration. AIRCHITECT V1 [5] proposed and demonstrated this idea on several hardware and mapping DSE tasks on systolic arrays [17], by training a multi-layer perceptron (MLP)-based classification model. Despite achieving good results on the systolic array design space, the accuracy and generality of this scheme are still less than expected (§IV). First, it did not address the non-uniform and long-tailed distribution of DSE data (§II), which significantly impacts the learnability of DNN models. Second, modeling DSE as a classification-only problem significantly prunes the design space with a fixed number of labels, i.e. configuration, and restricts its scalability for larger DSE due to a commensurate increase in model size. And lastly, the design space explored by this technique is small and relatively simple.

Contributions. To address the above issues, we propose AIRCHITECT V2, an enhanced version to enable more accurate, generalizable, and scalable learning-based DSE. We leverage contrastive learning to learn and encode the input feature representations into a uniform and smooth feature embedding space[1]. We also propose a novel unified representation namely, *Unified Ordinal Vectors* [18], that enables joint classification and regression to leverage the unified benefits of both these techniques for DSE to overcome the limitation of the classification-only approach. Finally, we study the applicability of the proposed technique to a sufficiently complicated design space, the hardware resource allocation (for a given workload and mapping) on an accelerator modeled by MAESTRO [19].

Our key contributions can be summarized as follows:

- We propose AIRCHITECT V2, featuring (i) a novel encoder-decoder, multi-head self-attention-based model, (ii) contrastive learning approach for uniform and smooth embedding space, and (iii) a unified ordinal vector representation of output, combining classification and regression. (§III)
- Through extensive ablations and experiments, we demonstrate that AIrchitect v2 surpasses existing techniques by an average of 15% in discovering optimal design points and achieves a $\sim 1.7\times$ improvement in inference performance on the predicted hardware architecture. (§IV)
- We release our MAESTRO [19]-based DSE training dataset comprising of 10^5 real DNN workloads along with our trained models to advance learning-based DSE research.

II. Background and Motivation

A. Learning-based DSE

Training a DNN model that predicts the optimal design choice enables one-shot inference and mitigates sensitivity to sampling efficiency unlike search-based techniques (Figure 1). AIRCHITECT V1 [5] formulated several DSE tasks for systolic arrays as a classification problem, where its MLP model outputs probability distributions over labels, i.e. encoded design choices. The model is trained on a dataset of DSE input parameters and their corresponding optimal design choice, generated by the Scale-Sim [20] simulator. Given its effectiveness and advantages in the DSE domain, we expand this direction.

B. Contrastive Learning

Contrastive learning, widely adopted in self-supervised settings [21], [22] has proven to be effective in mitigating overfitting and improving generalization by regularization against negative samples. Recent work [23], [24] has also demonstrated

[1]In this work, we use intermediate representation and embedding space interchangeably.

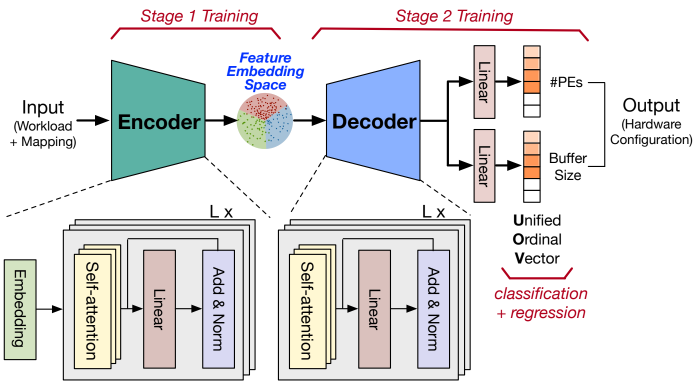


Fig. 2: Overview of AIRCHITECT V2, highlighting (1) multi-head self-attention-based encoder and decoder structure, (2) latent embedding space improved by contrastive learning (3) **UOV** output representation combining classification and regression.

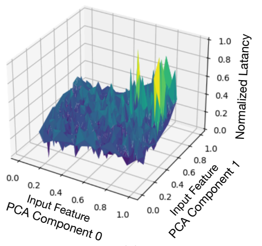
its benefits in smoothening the loss landscape. The core idea of contrastive learning, which is based on the infoNCE loss [25] is, to balance the learning of a data sample (anchor) by aligning the corresponding positive sample pairs and repulsing the negative sample pairs. As we shall demonstrate in §III, by selecting positive samples from DSE data points belonging to the same class and aligning them in the embedding space, while simultaneously pushing away samples from different classes, contrastive learning promotes the creation of a more uniform embedding space and simultaneously combats the effects of long-tailed distributions [26].

C. Motivation 1: Non-Uniform Performance Landscape

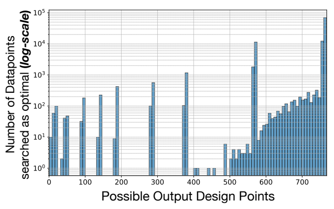
As we can observe from Figure 3 (a), the normalized performance (latency) landscape, drawn from the DSE dataset introduced in §III-A, the distribution is highly non-uniform and non-convex. This makes it particularly challenging for search-based techniques to reach the global optimum due to a high probability of getting trapped within the multiple local minima [23], [24]. Additionally, learning-based techniques also struggle to achieve good performance in the presence of such a non-uniform design space (§IV). For instance, even insignificant variations in the input features may cause the predictions to have large discrepancies since the model might not have converged to the global optimum. Furthermore, they are also prone to overfitting the training data because such a landscape can cause the model to fit too closely to the specific training data points, reducing its ability to generalize. In this work, we leverage contrastive learning to smoothen the landscape and encode it into a simpler and uniform embedding space.

D. Motivation 2: Long-tailed Data Distribution

Figure 3 (b), plots the number of data samples for each DSE output design point, based on the same dataset (§III-A) and using a strategy akin to that described in [5]. We observe that the DSE dataset is imbalanced and exhibits a long-tailed distribution [26], where a small subset of output design points are favored by the majority of data samples, while many other design points are sparsely chosen. Such a data distribution poses a significant challenge for learning-based DSE techniques and



(a)



(b)

Fig. 3: Prominent challenges on DSE dataset: (a) non-uniform and non-convex landscape (b) long-tailed distribution of data samples over labels. *Drawn from the problem space in III-A*

impacts performance and generalizability (§IV). In this work, we extend the advantages of contrastive learning to address the long-tailed data distribution. By employing contrastive learning, the learnt embedding space converges to one where output design points are more uniformly distributed, reducing the imbalance in representation (see Figure 5).

E. Motivation 3: Classifications v/s Regression

Depending on how the optimal DSE output is determined, we categorize existing DSE techniques into two broad categories: *classification* and *regression*. Classification-based techniques, such as AIRCHITECT V1 [5], partition the design space into a set of predefined classes or labels, each representing a distinct design configuration. It predicts the optimal design point by selecting the most appropriate class from this fixed set, simplifying the search process and better constraining the search space. However, this approach can limit flexibility and scalability when dealing with large or complex design spaces.

We take the liberty of categorizing techniques as regression-based when they are capable of predicting DSE output hardware configurations as continuous values, rather than selecting from a predefined set of discrete options as described above. Consequently, under this definition, all search-based techniques [4], [11]–[15] can be interpreted as regression-based. Regression-based techniques [4], [11], [16] are highly scalable since increasing design space size or complexity does not necessitate an increase in model size [11], [16]. However, with large and complex design spaces, these methods result in an unconstrained learning problem [27], which can greatly impact accuracy and increase the risk of overfitting. In this work, we propose a novel representation **UOV**, or *Unified Ordinal Vectors* that can leverage the unified benefits of both these techniques while mitigating their specific drawbacks.

III. AIRCHITECT V2

A. DSE Problem Formulation

To delve into the aforementioned challenges in DSE, we select the following task as our target scenario for exploration: hardware resource assignment on MAESTRO [19]-modeled accelerator. This problem, previously also explored by ConfuciuX [12] using RL-based search, has been shown to be a sufficiently complex design space.

We translate the DSE task into a learnable formulation, by encoding the design parameters following Table I, modified from [12]. As the DSE inputs, the tenser dimensions for GEMM operation (M, N, K) are numerical integer values, while dataflow is categorical data chosen among: weight stationary [6], output stationary [8], and row stationary [7].

The output is the optimal hardware resources for the given per-layer inputs, configured as the number of PEs and L2 buffer size, while L1 buffer size is fixed following the search assumptions in [12]. The dataset is generated by executing ConfuciuX [12] on the randomized input parameters, with the optimization metric (i.e. reward) set as latency.

This DSE task forms a large design space of complexity, $O(10^9)$ derived as the product of input feature dimensions from Table I. Figure 4 visualizes its significant complexity, exhibiting irregular and non-trivial characteristics that hardly suffice with simple techniques such as decision trees or support vector machines [5]. Moreover, the dataset shows a highly non-uniform performance landscape as well as imbalanced data sample distribution as highlighted in Figure 3.

TABLE I: Input and output formulations for MAESTRO [19]-based hardware resource allocation, modified from [12]

	Features (size)
Input	M (256), N (1677), K (1185), dataflow (3)
Output	PE (64), buffer size (12)

- Assuming GEMM operation $(M, K) \times (K, N) = (M, N)$
- dataflow=choice among [[6], [8], [7]]

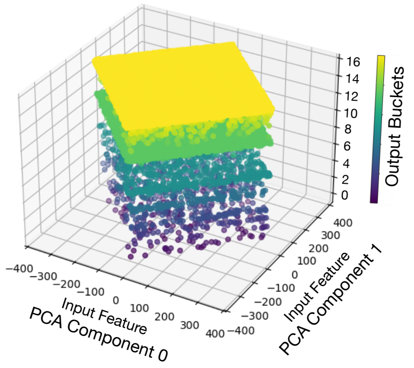


Fig. 4: Complexity of the problem space from Table I, visualizing the input features (xy-plane, processed with PCA) and output configuration (z-axis, plotted into **UOV** buckets). This justifies the need for sophisticated model architecture.

B. AIRCHITECT V2 *Overview*

We present an overview of AIRCHITECT V2 framework in Figure 2. To effectively learn the complex DSE space, we design an encoder-decoder transformer-based model architecture following the structure in [28] (see §IV for reasoning). AIRCHITECT V2 takes as input workload specifications as outlined in Table I and outputs optimized hardware design configurations that are geared towards improving overall latency and/or energy.

Both the encoder and decoder have identical and complementary structures, consisting of L layers of stacked self-attention blocks, a feed-forward network, and a downsampling (encoder) / upsampling (decoder) units [28]. The decoder also has two

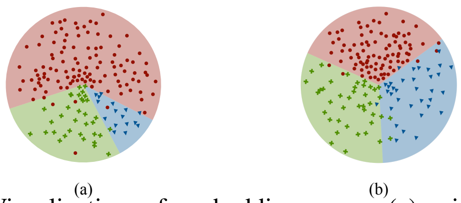


Fig. 5: Visualization of embedding space (a) without contrastive learning and b) with contrastive learning. Employing contrastive learning results in a uniform embedding space. Different colors represents different classes of data samples.

UOV heads (explained later) corresponding to the two hardware design configurations predicted by the framework.

The encoder and decoder in the AIRCHITECT V2 framework decompose the hardware DSE learning and prediction process into two distinct stages. In stage 1 (§III-C), the encoder is responsible for learning to construct a uniform and smooth intermediate representation of the input design space, and during prediction, identifies a point in this learned embedding space that closely approximates the input specifications. In stage 2 (§III-D), the decoder learns to process the identified point in the intermediate representation and finally predict the optimal design configuration via **UOV**.

C. AIRCHITECT V2 *Stage 1*

The goal of stage 1, i.e. encoder, is to generate a uniform and smooth intermediate representation of the input design space. To guide the encoder in efficiently learning this intermediate space, we leverage a combined objective consisting of the contrastive term and performance prediction term.

Contrastive Learning. As pointed out in §II, contrastive learning enables the encoder to learn to create a uniform and smooth embedding space by aligning positive samples together while simultaneously pushing away negative samples. In the context of stage 1 training, for each workload configuration within an input batch (anchor), positive and negative samples correspond to configurations that belong to the same and different **UOV** buckets, respectively. Inspired by [21], [24], we adopt the infoNCE loss variant [21] of contrastive loss, and augment it to balance the positive and negative samples. The contrastive objective can be defined as,

$$\mathcal{L}^C = -\log \frac{\sum_{p+} \exp(\lambda^p \cdot \lambda^{p+}/\tau)}{\sum_{p+} \exp(\lambda^p \cdot \lambda^{p+}/\tau) + \sum_{p-} \exp(\lambda^p \cdot \lambda^{p-}/\tau)} \tag{1}$$

where, λ is the output embedding representation from the encoder, and p, p+, and p- are the anchor, positive, and negative samples, respectively. τ is empirically determined to be 0.4.

Performance Predictor. Training the encoder with a vanilla contrastive objective will create an embedding space with no semantic meaning [11]. Therefore, we augment the training objective with an L1-based performance prediction loss, $\mathcal{L}^{perf}$ to add semantic information to the learnt embedding. The performance here is the optimization goal of DSE, e.g. latency. This design is motivated by earlier work [11], [29], that emphasizes the influence of performance predictors in organizing the embedding space.

The final stage-1 objective is, $\mathcal{L}^{stage1} = \mathcal{L}^C + \mathcal{L}^{perf}$. During stage 1 training, the encoder is trained with L^{stage1} as the objective, enabling the encoder to learn the embedding space that keeps similar DSE samples close while distancing dissimilar ones (Figure 5) and incorporate enriched semantic information that aligns with the resulting performance. This contributes to the formation of a uniform and smooth intermediate representation space that is also robust to the imbalanced and long-tailed data distribution.

D. AIRCHITECT V2 *Stage 2*

Once stage 1 training is complete, we train the decoder in stage 2 while keeping the encoder's weights fixed to prevent the backpropagation of gradients. In this stage, the decoder learns to predict the optimal DSE hardware configuration given a point in the embedding space identified by the encoder. Unlike previous works [5], [11], [16] the decoder is augmented to predict our proposed **UOV** through **UOV** heads (Figure 2) which are simple feed-forward layers that learn to predict **UOV** s guided by the stage 2 objective. Since the DSE space explored in this study has two configurations, i.e. the number of PEs and buffer size, the decoder has two **UOV** heads.

Unified Ordinal Vectors (**UOV**). The **UOV** representation [18] scheme enables embedding the large-scale classification labels into reduced-size and scalable representation via "bucketization". Based on the scheme, the model jointly and implicitly predicts the classification *bucket* while regressing to the actual DSE configuration within each bucket. The classification bucket consists of *ranges* of DSE points. We employ Space Increasing Discretization [30] for the given DSE space and obtain K discretized buckets, $\Lambda = \{r_0, r_1, ..., r_{K-1}\}$. The higher the value of K, the lower the range of DSE points covered by each bucket. Following algorithm 1, any DSE configuration D can be encoded as a K-length **UOV** such that,

- Assuming D lies in bucket r_n
- Bucket values preceding r_n are non-zero and monotonically increasing (algorithm 1 line 3)
- Bucket values following r_n are zero (algorithm 1 line 6)

As a result, the final **UOV** (O) of a given D is,

$$\{O_i\}_{i=0}^{K-1} = \begin{cases} 1 - f(|D - r_i|), & \text{if } D \geq r_i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where f can be any choice of monotonically increasing function (we select the exponential function in algorithm 1). Figure 6 intuitively visualize this process, and decoding the **UOV** back to the actual DSE configuration is the exact reverse of algorithm 1. Our proposed unified ordinal representation captures essential ordering information and is well-suited for the model to learn and predict thanks to its regular structure. For evaluation, we empirically set K as 16 (see §IV-D).

Decoder Training. The decoder and **UOV** heads are trained with the same data used for stage 1. By leveraging the transformer decoder as the backbone, each **UOV** head is trained to predict the corresponding hardware configuration by unifying (1) classification to identify range buckets and (2) regression for finer-granularity search within the bucket.

```
Algorithm 1: Ordinal Encoding
Input  : Ground-truth DSE-config $D_{gt} \in \mathbb{R}$, $K$ buckets
         produced $\Lambda = \{r_0, r_2, ..., r_{K-1}\}$, $r_i \in \mathbb{R}$
Output: Ordinal vectors $\{O_i \in \mathbb{R}\}_{i=1}^{K}$
1 for $i = 0$ to $K - 1$ do
2     if $D \geq r_i$ then
3         $O_i \leftarrow 1 - \exp^{-|D - r_i|}$;
4     end
5     else
6         $O_i \leftarrow 0$;
7     end
8 end
9 return $\{O_i\}_{i=1}^{K}$
```

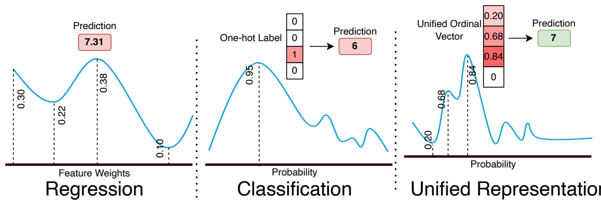


Fig. 6: Visualization of how the same configuration is represented for regression, classification, and the proposed **UOV**.

Unification Loss. We make use of the *Unification Loss* (L_o) as the primary training loss. Due to the nature of the **UOV**, we adopt a loss function similar to [31] to guide stage 2 training,

$$L_o = \sum_{i=0}^{K-1} \begin{cases} \alpha |q_i - u_i|^{\gamma} BCE(u_i, q_i), & \text{if } q_i > 0 \\ (1-\alpha) u_i^{\gamma} BCE(u_i, q_i), & \text{otherwise} \end{cases} \tag{3}$$

$$BCE(u_i, q_i) = -q_i log(u_i) - (1 - q_i) log(1 - u_i) \tag{4}$$

where, BCE corresponds to Binary Cross-Entropy, u, q are the predicted and ground-truth **UOV** s respectively, and $\alpha = 0.75$, $\gamma = 1$ are empirically determined. This formulation for the unification loss penalizes the predictions buckets farther from the ground-truth bucket more heavily than those closer to the ground-truth. In addition, it penalizes the actual likelihood/regression within a predicted bucket to simultaneously ensure accurate bucket prediction and correct estimation of the actual DSE point within the bucket.

E. AIRCHITECT V2 *Deployment Pipeline*

AIRCHITECT V2 is trained and inferred on a per-layer basis, recommending the optimal hardware resources for single-layer execution. For model-level deployment, we mention two methods (which apply to any general layer-granularity DSE).

Given a model with N layers, $\mathbf{M} = \{L_0, L_1, ..., L_N\}$, assume AIRCHITECT V2 has recommended $\mathbf{HW} = \{HW_{L_0}, HW_{L_1}, ..., HW_{L_N}\}$ for each layer. We can determine the final hardware configuration HW_M from either:

Method 1. For each HW_{L_i} in **HW**, estimate the model-wise latency (we use MAESRO [19] in this work) across all layers in M. Select the HW_{L_i} that yields the minimum as HW_M.

Method 2. Identify the bottleneck layer L_n among L_i that results in the largest latency when executed on its recommended HW_{L_i}. Choose the HW_n as HW_M.

TABLE II: AIRCHITECT V2 stage 1 ablations.

$\mathcal{L}^C$	$\mathcal{L}^{perf}$	Accuracy (%)
✗	✗	79.43
✗	✓	81.27
✓	✗	89.97
✓	✓	**91.17**

TABLE III: Comparison with other learning-based techniques.

Method	Accuracy (%)
GANDSE [16]	84.39
AIRCHITECT V1 [5]	77.60
AIRCHITECT V2 **(Ours)**	**91.17**

We demonstrate *Method 1* on representative models in §IV-C, with per-layer DSE from AIRCHITECT V2 and baselines, to highlight the practical effectiveness of our approach.

IV. RESULTS AND DISCUSSION

A. *Experimental Setup*

Implementation. The AIRCHITECT V2 framework is implemented in Pytorch and evaluated on the DSE task introduced in §III-A with a dataset consisting of 100K samples, split into 80K for training and 20K for testing. We train AIRCHITECT V2's stage 1 and stage 2 individually for 500 and 100 epochs, respectively. All experiments were conducted on a single NVIDIA H100 GPU. The access to the dataset, scripts for training, and the pre-trained encoder and decoder models are provided in https://github.com/maestro-project/AIrchitect-v2.

Baselines. We compare AIRCHITECT V2 framework against existing SoTA learning-based techniques, including the MLP-based AIRCHITECT V1 [5], generative adversarial network GANDSE [16] and variational autoencoder VAESA [11] combined with a search-based technique (BO), which are trained and evaluated on the same dataset (§III) for fair evaluation.

B. *Layer-level Prediction Accuracy*

As shown in Table III, AIRCHITECT V2 demonstrates considerable improvement in prediction accuracy over other baselines. In particular, the shallow MLP model and classification head used in AIRCHITECT V1 cause significant overfitting to the training data and inability to handle the complexities of the DSE landscape and data distribution, resulting in the lowest accuracy of 77.60%. Although GANDSE achieves higher accuracy than AIRCHITECT V1, it is still limited by the large unconstrained learning problem due to its generative approach, impacting the quality of DSE outputs. In contrast, AIRCHITECT V2 achieved a notably high accuracy of 91.17%, benefiting from our solutions outlined in §III-B, §III-C, and §III-D.

C. *Model-level Deployment Evaluation*

We further assess the performance in practical model deployment on representative DNN and LLM models [32]–[34], *which were never seen during training*. Figure 7 compares the model-level latency achieved by various DSE techniques mentioned in §IV-A. AIRCHITECT V2 consistently outperforms others across workloads in identifying the hardware configuration with the lowest latency. Particularly, AIRCHITECT V1 and GANDSE [16] achieve poor performance due to overfitting and solutions being trapped in local optima, as they lack addressing the non-uniform and non-convex DSE performance landscape. VAESA with BO [11] is the only method that achieves performance close to ours, as it is able to construct a continuous and low-dimensional latent space through a variational autoencoder

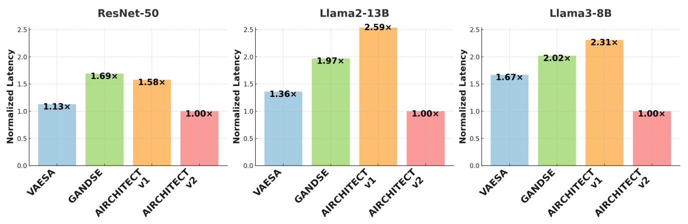


Fig. 7: DSE performance (latency) comparison of different techniques on popular DNNs and LLMs. Normalized to AIRCHITECT V2 (red). Latency is estimated using MAESTRO [19].

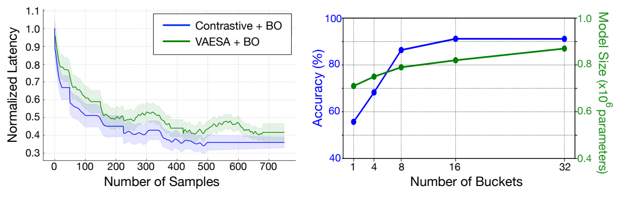


Fig. 8: (a) DSE performance comparison of searching using BO on the contrastive embedding v/s VAE-generated embedding [11] for a Llama2-7B model. (b) Impact of number of **UOV** intervals within each bucket, on accuracy and model size.

(VAE). However, as demonstrated in §IV-D, the embedding space generated through contrastive learning is superior to VAE.

D. Ablations

Encoder Training Loss. We investigate the impact of contrastive loss and performance prediction loss to the training, as shown in Table II. Without both objectives (and using only an L2-loss term), the model struggles to handle the non-uniform DSE landscapes and skewed data distributions, resulting in low accuracy similar to AIRCHITECT V1. Incorporating contrastive loss significantly alleviates these issues, leading to a substantial increase in prediction accuracy (+10.54%). The addition of performance prediction loss further enhances learning (+1.2%) by providing semantic information to the learnt feature embedding, achieving the highest final prediction accuracy.

Impact of Contrastive learning. To study the effectiveness of the proposed contrastive learning on DSE feature embeddings, we further evaluate the constructed embedding space. Following [11], we adopt BO to search from both the embedding space constructed by contrastive learning and the VAE-generated latent space [11]. We train a separate decoder for each technique that converts a point from the latent space into a hardware configuration, and estimate the corresponding latency using MAESTRO [19]. As observed in Figure 8 (a), searching within our contrastive embedding space leads to significantly faster convergence and lower latency compared to the VAE-generated embedding space, implying a more uniform and smoother performance landscape.

UOV v/s Classification. Figure 9 compares the effectiveness of the proposed **UOV** formulation against the conventional classification approach, for both AIRCHITECT V1 and AIRCHITECT V2. We observe that in both scenarios, **UOV** formulation improves prediction performance because classification overly discretizes and constrains the design space, while **UOV** combines the benefits of classification and regression, enabling

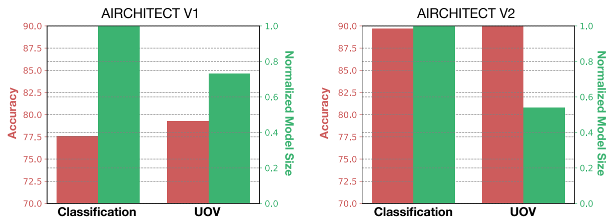


Fig. 9: Effect of **UOV** on prediction accuracy(red) and model size(green), for AIRCHITECT V1 and AIRCHITECT V2.

fine-grained prediction. Additionally, **UOV** significantly reduces model size, which highlights its applicability in larger design spaces. By demonstrating the advantages of **UOV** for two different techniques, we show that it is not specific to AIRCHITECT V2 and can be adapted for similar benefits in other methods.

UOV Hyperparameter Evaluation. Increasing the number of **UOV** buckets improves accuracy through finer-granularity prediction or reduced discretization, but also increases model size due to the larger output vector. As shown in Figure 8 (b), model size (green) grows along with the number of buckets, while accuracy (blue) begins to saturate beyond 16 buckets. We select 16 **UOV** buckets for our DSE learning to achieve the optimal trade-off between accuracy and model size. Notably, as the number of buckets increases, the problem shifts toward classification, while a single bucket reverts it to regression!

V. RELATED WORKS

Search-based Optimizations Approaches. [13], [14] utilize GA over genomes encoding design points. [12] uses RL for coarse-grained search, followed by GA for fine-tuning. [15] performs a two-step optimization combining multi-objective BO with Q-learning algorithms. [35] employs a differentiable surrogate model to guide sampling via input gradients.

Supervised Learning-based Approaches. [5] trains an MLP model to predict optimal design choices on systolic arrays, framing the DSEs as a classification problem. [16] trains a GAN that generates design points to meet user-specified objectives, in a higher-dimensional design space. [11] focuses on DSE feature embedding space by training a variational autoencoder.

VI. CONCLUSION

Current supervised learning-based DSE studies insufficiently addressed DSE-specific challenges in the dataset and learning formulation. In this paper, we propose AIRCHITECT V2, which employs an encoder-decoder transformer model to learn the complicated DSE space and leverages contrastive learning and **UOV** representation to tackle the non-uniform embedding space and long-tailed data distribution. AIRCHITECT V2 improved prediction accuracy on the test set by ∼15% over competing baselines, while also achieving 1.7× higher performance of identified designs on unseen workloads.

ACKNOWLEDGMENT

We thank Ananda Samajdar for their valuable assistance in helping set up AIRCHITECT V1 experiments for comparison and their insightful discussions on learning-based DSE. This work was funded in part by JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] A. Ramachandran, Z. Wan, G. Jeong, J. Gustafson, and T. Krishna, "Algorithm-hardware co-design of distribution-aware logarithmic-posit encodings for efficient dnn inference," *arXiv preprint arXiv:2403.05465*, 2024.

[2] H. Kwon, A. Samajdar, and T. Krishna, "Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 461–475, 2018.

[3] A. Ramachandran, S. Kundu, and T. Krishna, "Microscopiq: Accelerating foundational models through outlier-aware microscaling quantization," *arXiv preprint arXiv:2411.05282*, 2024.

[4] C. Hong, Q. Huang, G. Dinh, M. Subedar, and Y. S. Shao, "Dosa: Differentiable model-based one-loop search for dnn accelerators," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 209–224. [Online]. Available: https://doi.org/10.1145/3613424.3623797

[5] A. Samajdar, J. M. Joseph, M. Denton, and T. Krishna, "Airchitect: Learning custom architecture design and mapping space," 2021. [Online]. Available: https://arxiv.org/abs/2108.08295

[6] NVIDIA, "Nvdla deep learning accelerator," http://nvdla.org, 2017.

[7] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[8] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," in *International Symposium on Computer Architecture (ISCA)*, 2015.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[10] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 418–434.

[11] Q. Huang, C. Hong, J. Wawrzynek, M. Subedar, and Y. S. Shao, "Learning a continuous and reconstructible latent space for hardware accelerator design," in *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2022, pp. 277–287.

[12] S.-C. Kao, G. Jeong, and T. Krishna, "Confuciux: Autonomous hardware resource assignment for dnn accelerators using reinforcement learning," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 622–636.

[13] S.-C. Kao and T. Krishna, "Gamma: Automating the hw mapping of dnn models on accelerators via genetic algorithm," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–9.

[14] S.-C. Kao, M. Pellauer, A. Parashar, and T. Krishna, "Digamma: Domain-aware genetic algorithm for hw-mapping co-optimization for dnn accelerators," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 232–237.

[15] Q. Xiao, S. Zheng, B. Wu, P. Xu, X. Qian, and Y. Liang, "Hasco: Towards agile hardware and software co-design for tensor computation," *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1055–1068, 2021.

[16] L. Feng, W. Liu, C. Guo, K. Tang, C. Zhuo, and Z. Wang, "Gandse: Generative adversarial network-based design space exploration for neural network accelerator design," *ACM Transactions on Design Automation of Electronic Systems*, vol. 28, no. 3, pp. 1–20, 2023.

[17] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A systematic methodology for characterizing scalability of dnn accelerators using scale-sim," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2020, pp. 58–68.

[18] A. Ramachandran, A. Dhiman, B. S. Vandrotti, and J. Kim, "Ntransnet: A multi-scale neutrosophic-uncertainty guided transformer network for indoor depth completion," in *2023 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2023, pp. 905–909.

[19] H. Kwon, P. Chatarasi, V. Sarkar, T. Krishna, M. Pellauer, and A. Parashar, "MAESTRO: A data-centric approach to understand reuse, performance, and hardware cost of DNN mappings," *IEEE Micro*, vol. 40, no. 3, pp. 20–29, 2020.

[20] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A systematic methodology for characterizing scalability of dnn accelerators using scale-sim," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2020, pp. 58–68.

[21] Y. Fu, Q. Yu, M. Li, X. Ouyang, V. Chandra, and Y. Lin, "Contrastive quant: quantization makes stronger contrastive learning," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 205–210.

[22] Y.-H. Cao, P. Sun, Y. Huang, J. Wu, and S. Zhou, "Synergistic self-supervised and quantization learning," in *European Conference on Computer Vision*. Springer, 2022, pp. 587–604.

[23] N. Frumkin, D. Gope, and D. Marculescu, "Jumping through local minima: Quantization in the loss landscape of vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 16 978–16 988.

[24] A. Ramachandran, S. Kundu, and T. Krishna, "Clamp-vit: Contrastive data-free learning for adaptive post-training quantization of vits," *arXiv preprint arXiv:2407.05266*, 2024.

[25] J. Wang, J. Li, W. Li, L. Xuan, T. Zhang, and W. Wang, "Positive–negative equal contrastive loss for semantic segmentation," *Neurocomputing*, vol. 535, pp. 13–24, 2023.

[26] T. Li, P. Cao, Y. Yuan, L. Fan, Y. Yang, R. S. Feris, P. Indyk, and D. Katabi, "Targeted supervised contrastive learning for long-tailed recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 6918–6928.

[27] R. Roelofs, V. Shankar, B. Recht, S. Fridovich-Keil, M. Hardt, J. Miller, and L. Schmidt, "A meta-analysis of overfitting in machine learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[28] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[29] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic chemical design using a data-driven continuous representation of molecules," *ACS central science*, vol. 4, no. 2, pp. 268–276, 2018.

[30] R. Miao, W. Liu, M. Chen, Z. Gong, W. Xu, C. Hu, and S. Zhou, "Occdepth: A depth-aware method for 3d semantic scene completion," *arXiv preprint arXiv:2302.13540*, 2023.

[31] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 002–21 012, 2020.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[34] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[35] K. Hegde, P.-A. Tsai, S. Huang, V. Chandra, A. Parashar, and C. W. Fletcher, "Mind mappings: Enabling efficient algorithm-accelerator mapping space search," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 943–958. [Online]. Available: https://doi.org/10.1145/3445814.3446762