APSIPA Transactions on Signal and Information Processing, 2024, 13, e

Original Paper

Image Classification via Subspace Learning Machine with Soft Partitioning (SLM/SP)

Hongyu Fu[1*], Xinyu Wang[1], Vinod K. Mishra[2] and C.-C. Jay Kuo[1]

[1] *University of Southern California, Los Angeles, CA, USA*
[2] *DEVCOM Army Research Laboratory, Adelphi, MD, USA*

ABSTRACT

Subspace partitioning plays a fundamental role in the design of effective classification methods. A novel subspace learning machine (SLM) was recently proposed. It projects feature vectors into a 1D feature subspace and partitions it into two disjoint sets. To effectively generalize the SLM method to high-dimensional feature space, SLM with soft partitioning, denoted by SLM/SP, is proposed in this work. By incorporating the Soft Decision Tree (SDT) data structure for decision learning, the SLM/SP begins with the adaptive learning of a tree structure using local greedy subspace partitioning. Once the tree structure is finalized, all parameters are globally updated. To apply SLM/SP to image classification tasks, we propose modulated designs for the topology of the SDT and a novel module for efficient local representation learning in the subspace learning diagram. The SLM/SP methodology offers efficient training, high classification accuracy, and small model size, underscored by experimental results on image classification benchmarks.

Keywords: subspace learning machine, image classification, machine learning, soft decision tree

*Corresponding author: Hongyu Fu, hongyufu@usc.edu

Received 10 January 2024; revised 05 June 2024; accepted 07 June 2024
ISSN 2048-7703; DOI 10.1561/116.20240003

1 Introduction

Image classification has seen significant advancements with the advent of machine learning (ML) and pattern recognition (PR). It has recently relied primarily on end-to-end optimization with a predefined model architecture in the deep learning (DL) paradigm, and traditional PR and ML approaches have been largely overlooked. However, these traditional methods have the potential to offer interpretable and efficient solutions for image classification tasks. We consider this learning paradigm and focus on a novel learning diagram design for image classification from the vantage point of high-dimensional feature space partitioning.

One of the critical limitations of traditional PR and ML methods is the lack of effective feature space partitioning. The classification work can be done in a single stage, such as the support vector machine (SVM), or multi-stages, such as the decision tree (DT) and multilayer perceptron (MLP). The model is much more efficient when utilizing multi-stage feature space partitioning, such as DT. At the same time, DT uses a single feature for decision-making at each node, which can lead to suboptimal performance. The recent subspace learning machine (SLM) [10, 11, 12] adopts the DT architecture while combining multiple features and outputs a new variable for decision-making at each node. SLM can be viewed as a generalized DT. The linear combination of multiple features can be written as the inner product of a projection vector and a feature vector. The effectiveness of SLM depends on selecting suitable projection vectors for effective feature space partitioning. Two projection vector selection methods were studied in Fu *et al.* [11], namely, probabilistic search and optimization-based search. SLM and DT apply a hard split to a feature using a threshold at a decision node. The effectiveness of SLM was demonstrated on datasets with feature dimensions up to 34D.

For tasks with higher dimensional feature spaces, SLM with hard split loses efficiency due to the exponentially increasing training complexity as a function of the feature dimension. We propose a novel SLM method that adopts soft partitioning, denoted as SLM/SP, to address the limitation. Our motivation is to develop a more efficient and effective feature space partitioning to handle high-dimensional feature spaces. We aim to leverage the strengths of SLM and soft decision trees (SDT) to create a more powerful and flexible framework for image classification.

This paper presents the SLM/SP method and demonstrates its effectiveness in image classification tasks. SLM/SP adopts the soft decision tree (SDT) data structure, and a novel topology is proposed with (i) inner nodes of SDT for data routing, (ii) leaf nodes of SDT for local decision-making, and (iii) edges between parent and child nodes in the SLM/SP tree for representation learning. Specific modules are designed for the nodes and edges, respectively. The training of an SLM/SP tree starts by learning an adaptive tree structure via local greedy

exploration between subspace partitioning and feature subspace learning. The tree structure is finalized once all leaf nodes' stopping criteria are met and all module parameters are updated globally. The proposed SLM/SP learning diagram design includes topology, data structure, and specific modules for classification-oriented feature learning and decision-making.

To apply SLM/SP to image classification tasks, we assign specific operations for the nodes and edges of the SDT topology and propose modulated designs for the adaptive learning of the whole SLM/SP model architecture. More specifically, we assign sample routing operation for the inner node, feature transformation operation for the edge, and local distribution estimation for the leaf nodes. For each operation in the SLM/SP model, we propose specific efficient modules for the SLM/SP model. For the SLM/SP image classification framework, the SLM/SP tree can serve as a joint feature-decision learning model or a pure decision learning model. For the joint feature-decision learning diagram of SLM/SP, we propose a novel subspace learning augmented block (SLAB) design as an efficient local representation learning module for the feature transformation operation on the edges of the SLM/SP tree.

The application of SLM/SP to image classification tasks enables efficient training, high classification accuracy, and trim model size. It is shown by experimental results that an SLM/SP tree for image classification offers a lightweight and high-performance classification solution.

The contributions of this work are summarized below.

- We propose a novel, efficient, and effective image classification framework based on SLM.
- We generalize SLM from hard-partitioning to soft-partitioning and propose a novel SLM/SP learning diagram for classification.
- We propose a modularized design and adaptive topology for SLM/SP and a novel module SLAB for efficient local representation learning in the SLM/SP learning diagram.
- We demonstrate the efficiency and effectiveness of SLM/SP in image classification with model size, FLOPs, and accuracy with several benchmarking datasets for both pure decision learning and joint feature learning and decision learning paradigms.

The rest of this paper is organized as follows. Background information is reviewed in Section 2. The SLM/SP method is proposed in Section 3. The application of SLM/SP to image classification is discussed in Section 4. Experimental results show the excellent tradeoff between effectiveness and efficiency of SLM/SP in Section 5. Finally, concluding remarks and future extensions are given in Section 6.

2 Background Review

The proposed SLM/SP method follows the work of SLM, which targets an interpretable, computationally efficient, and small-size learning model following the traditional ML paradigm. SLM is an early work in green learning [22], in which we aim to develop green solutions to machine learning problems in contrast to deep learning. SLM is a supervised decision-learning module with a lightweight model and achieves effective classification and regression performance on low-dimensional data. To generalize the SLM to high-dimensional data and push the performance of green learning on image classification, we learn from soft decision tree and the state-of-the-art efficient image classification methods to propose SLM/SP image classification method as a green learning solution with lightweight model size and FLOPs, and performance comparable to deep learning. In this section, we introduce the DL and traditional ML-based image classification, green learning, subspace learning machine, and soft decision tree in the following sections, respectively. We discuss more details on the efficient image classification methods in Section 4.2 when introducing efficient representation learning and SLAB.

2.1 Deep Learning and traditional ML for Image Classification

Image classification methods can be generally categorized into DL and traditional ML methods. Traditional ML techniques comprise two sequential modules: 1) extraction of features from images and 2) classification based on these extracted features. In contrast, DL techniques perform feature extraction and classification concurrently within a single module. Convolutional neural network (CNN) models marked the inception of DL like AlexNet [21], VGG [38], GoogLeNet [40] and ResNet [13] have demonstrated remarkable accuracy levels. A significant advancement in this field is the Vision Transformer (ViT) model [9], which currently provides unparalleled performance on several image classification benchmark datasets. The success of DL techniques can be attributed to factors such as the availability of extensive training datasets, ample computational resources, end-to-end optimization, neural architecture search, and the implementation of large models such as transformers. However, DL techniques are hindered by their lack of interpretability, high computational expenses, and complex model structures. This research proposes an interpretable, computationally efficient, and compact learning model by adhering to the conventional ML paradigm.

2.2 Green Learning

Learning from the DL methods and traditional methods, green learning (GL) was proposed to address several concerns associated with DL, such as the

substantial carbon footprints produced by large DL networks in recent years. GL-based models are characterized by their low-carbon footprints, small model sizes, low computational complexity, and logical transparency. GL has been successfully applied to a wide range of applications. Examples include point cloud classification, segmentation and registration [46, 47], fake image detection [2, 3, 52, 53, 54], image generation [1, 28, 29, 51], blind image quality assessment [34], disease classification [32], face gender classification [36], and object tracking [48, 49, 50].

GL for image classification comprises three sequential modules: 1) unsupervised representation learning, 2) supervised feature learning, and 3) supervised decision learning. Modules 1 and 2 collectively correspond to the feature extraction module in the classical PR paradigm. However, there is a significant distinction: Modules 1 and 2 in GL are automated, whereas feature extraction in PR is conducted ad hoc. Automatic feature extraction is achieved by unsupervised representation learning, such as the Subspace Approximation with Adjusted Bias (Saab) transform [23]. The Saab transform is a joint spatial-spectral transform that decomposes a local patch into a DC (direct current) component and several AC (alternating current) components. The AC filters are derived from principal component analysis (PCA). Typically, multi-stage Saab transforms are applied. They are used to build several GL-based image classification methods, including Pixelhop [4], Pixelhop++ [5], and IPhop [44]. The Saab coefficients in various stages offer new representations for patches of different receptive fields. They are used as feature candidates. Note that Saab coefficients are not handcrafted since they are obtained automatically by exploiting the correlation of pixels in a local neighborhood. Besides, no supervision labels are used to derive the Saab transform. The dimension of the representation space is usually huge. It is essential to reduce its dimension using a supervised learning method. A supervised feature learning method, the discriminant feature test (DFT), was proposed in Y. Yang *et al.* [45]. The multi-stage Saab transforms and DFT serve as modules 1 and 2 of a GL system, respectively. The features extracted with modules 1 and 2 are usually called the successive subspace learning (SSL) features [4, 5, 44].

2.3 Subspace Learning

Subspace learning is a powerful framework for data representation, widely used in various fields such as computer vision, machine learning, and signal processing. The core idea of subspace learning is to represent high-dimensional data in a lower-dimensional subspace, where the underlying structure of the data can be more easily identified and analyzed. In this section, we provide an overview of prominent techniques in the subspace learning paradigm, namely sparse representation and low-rank representation, and then compare and highlight the unique contributions of our SLM method.

Sparse representation is a technique that approximates a signal as a linear combination of a few basis vectors from a dictionary [35]. This approach has been employed to model the underlying structure of data, where each data point is represented as a sparse linear combination of atoms from a dictionary. The sparse representation of the data can be utilized as a feature vector for further analysis. The efficacy of sparse representation has been demonstrated, particularly when combined with other techniques such as dictionary learning [33].

Low-rank representation is another technique that has been applied to image classification [31]. This approach represents a set of data points as a low-rank matrix, where the rank of the matrix is smaller than the number of data points. Low-rank representation can be employed to model the underlying structure of data and is effective in tasks such as data denoising and feature extraction. Specifically, low-rank representation has been utilized to model the correlation between different data points and to identify the underlying patterns and structures in the data.

To offer an efficient and effective supervised decision learning module for GL, SLM is presented in Fu *et al.* [10, 11, 12]. Given a set of discriminant features as the input, SLM is used to output a probability vector of all possible image classes. SLM takes a tree structure, utilizes a subspace partitioning process for each node splitting and generating purer leaf nodes for final predictions. For subspace partitioning, SLM identifies a discriminant subspace, denoted by S^0, learns optimal projection vectors from a set of candidates in S^0 to yield the most discriminant subspace and then find the optimal partitions that split the subspace of a parent node into those of child nodes. The partitioning process is recursively applied at each child node to build an SLM tree.

While sparse and low-rank representations are powerful techniques, they differ from our SLM method in several aspects. SLM employs a feature space partitioning idea that clusters data into subspaces. In contrast, sparse and low-rank representations do not explicitly model the clustering structure of the data. Furthermore, SLM utilizes a machine learning framework to learn the subspace structure of the data, whereas sparse representation and low-rank representation are typically employed as feature extraction techniques.

The training complexity of getting effective projection vectors in SLM is high. Lowering the complexity of training is the motivation of this research. The training complexity of SLM can be significantly reduced by replacing hard partitioning with soft partitioning, leading to SLM/SP.

2.4 Soft Decision Tree

The generalization of SLM from hard feature space partitioning to soft partitioning with the SLM/SP method is inspired by research on the soft decision tree (SDT). The very first soft decision tree (SDT) was introduced by Suárez

and Lutsko [39]. Research on SDT in the 90s included Jordan and Jacobs [18] and Suárez and Lutsko [39]. The work by Suárez and Lutsko [39] examined a specific scenario where axis-aligned features are used as the input and parent nodes are either static distributions over classes or linear functions. A similar idea called the hierarchical mixture of experts (HMEs) [18] was introduced earlier. Parent nodes are linear classifiers, and the tree structure is fixed [18]. Several follow-up studies have been conducted in the last decade. A computationally efficient training method that directly optimizes hard partitioning through differentiation with stochastic gradient estimators was investigated by Léon and Denoyer [30]. More contemporary SDTs, such as Ioannou *et al.* [16] and Laptev and Buhmann [24], have incorporated MLPs or convolutional layers in parent nodes to enable more complex input space partitioning. Another direction is to combine nonlinear data transformations with DTs to enhance model performance. For example, the neural decision forest (NDF) [19] achieved state-of-the-art performance on the ImageNet in 2015. A similar idea was developed by Xiao and Xu [43], where an MLP was used as the root transformer. One can optimize it to minimize the loss of differentiable information gain. However, it is essential to note that the model architectures are predetermined and fixed in all these methods. The choice of effective architectures is still an open question.

Architecture growth is a key facet of DTs [7] and is typically performed greedily with a stopping criterion based on the validation set error [17, 39]. Prior research in DTs has endeavored to enhance this greedy growth strategy. Decision jungles [37] utilize a training mechanism to merge partitioned input spaces between different subtrees, thereby rectifying suboptimal splits caused by the locality of optimization. Irsoy *et al.* [17] introduced budding trees, which are incrementally grown and pruned based on the global optimization of existing nodes. Recently, a novel approach called Adaptive Neural Trees (ANTs) [41] was proposed that unites the paradigms of deep neural networks and decision trees. ANTs incorporate representation learning into a decision tree's edges, routing functions, and leaf nodes. This is achieved through a backpropagation-based training algorithm that adaptively grows the architecture from primitive modules, such as convolutional layers. The advantages of these neural tree models include lightweight inference via conditional computation, hierarchical separation of features useful to the task, and a mechanism to adapt the architecture to the size and complexity of the training dataset.

3 SLM Tree with Soft Partition (SLM/SP)

3.1 Overview of SLM/SP

For SLM with Soft Partition (SLM/SP), we consider a K-class classification problem. The input feature space X contains N samples, where each sample has a D-dimensional feature vector. A sample is denoted by

$$\boldsymbol{x}_n = (x_{n,1} \cdots x_{n,d} \cdots, x_{n,D})^T \in \mathbb{R}^D, \quad n = 1, \cdots, N. \tag{1}$$

The partitioning in the feature space in SLM can be expressed mathematically in the form of

$$\boldsymbol{a}^T \boldsymbol{x} + b = 0, \tag{2}$$

where b is the bias and

$$\boldsymbol{a} = (a_1, \cdots, a_d, \cdots, a_D)^T, \qquad ||\boldsymbol{a}|| = 1, \tag{3}$$

is the unit vector that points to the surface in a normal direction. It is also called the projection vector. Then, the full space, S, is split into two half subspaces:

$$S_+ : \boldsymbol{a}^T \boldsymbol{x} \geq -b, \text{ and } S_- : \boldsymbol{a}^T \boldsymbol{x} < -b. \tag{4}$$

The above process can be conducted recursively to lead to a binary decision tree that offers a hierarchical partition of the feature space. One challenge in Eq. (4) lies in finding a good projection vector $\boldsymbol{a}$ at each intermediate (or called inner) node so that samples of different classes are better separated. This is related to the distribution of samples of different classes at the node. The ultimate objective is to lower the weighted entropy of all leaf nodes.

The output of the root node contains two child nodes, denoted by S_+ and S_-. Hard partitioning assigns an input sample to one of the two. With soft partitioning, its assignment is a probabilistic one. For linear soft partitioning, the probabilities of going to S_+ and S_- are

$$p_+(x) = \sigma(\boldsymbol{a}^T \boldsymbol{x} + b), \text{ and } p_-(x) = 1 - \sigma(\boldsymbol{a}^T \boldsymbol{x} + b), \tag{5}$$

where σ is the sigmoid logistic function, respectively. The dimension of $\boldsymbol{x}$ determines the complexity of each soft partitioning. The soft partitioning Eq. (5) can be generalized to any differentiable linear or nonlinear function. For example, to achieve higher modeling capability, the linear function $\boldsymbol{a}^T \boldsymbol{x} + b$ in Eq. (5) can be replaced by a simple MLP with one hidden layer. With nonlinear activation functions such as ReLU or Leaky ReLU, an MLP can be trained via back-propagation. This work uses a single hidden layer MLP as a pre-processing unit for soft partitioning. Then, the probability of going to S_+ can be written as

$$p_+(x) = \sigma(\boldsymbol{a'}^T ReLU(\boldsymbol{a}^T \boldsymbol{x} + b) + b'). \tag{6}$$

where $\boldsymbol{a}$ and b are for the first layer and $\boldsymbol{a'}$ and b' are for the second layer of the MLP. The same soft partitioning process can be repeated at child nodes recursively. For example, we conduct soft partitioning on S_+ so that the probability for an input to arrive at the $(+, +)$-grand-child node $p_{+,+}(x)$ is the cascaded multiplication of $p_+(x)$ at inner nodes. It is straightforward to get the mathematical expressions for $p_{+,+}(x)$, $p_{+,-}(x)$, $p_{-,+}(x)$, and $p_{-,-}(x)$. Combining SLM and soft partitioning leads to an SLM/SP tree. SLM/SP learns the parameters in the training stage. After training, SLM/SP employs them to assign an input sample to one of a set of partitioned subspaces with a path probability. The generalization of the SLM/SP process to the tree topology, the probabilistic inference of the SLM/SP trees, and the application of the SLM/SP tree to image classification are discussed in Section 3.2, Section 3.3, and Section 4 respectively.

3.2 Design of SLM/SP Tree

In this section, we formalize the definition of SLM/SP trees, including the topology of the SLM/SP trees, the determination of the tree structure, and the general formulation of the parameter learning. In general, the topology of the SLM/SP tree is the form of DT enhanced with SLM/SP, which aims to learn the conditional distribution $p(y|x)$ from a set of N labeled samples denoted in Eq. (1) as training data.

The design of an SLM/SP tree consists of two main choices: 1) determining the hierarchical tree structure, 2) determining optimal parameters (i.e., $\boldsymbol{a_i}$ and b_i) at inner nodes. This section introduces the SLM/SP tree and the respective modules and operations corresponding to the topology. Then, the two design choices above are elaborated.

3.2.1 Topology and Operations

The SLM/SP Tree overview is shown in Figure 1. In short, the SLM/SP Tree is characterized by a set of hierarchical partitions of input space X, a series of transformations, and separate predictive models in the respective component regions. For the model's topology, we restrict it to be a binary tree, defined as a graph with its node as either an inner node or a leaf node and is the child of exactly one parent node, except for the root node at the top. We define the topology $\mathcal{T}$ of the tree as $\mathcal{T} := \{\mathcal{N}, \mathcal{E}\}$, which $\mathcal{N}$ is the set of all nodes and $\mathcal{E}$ is the edges between them. Nodes with no children are leaf nodes N_{leaf}, and all the other nodes are inner nodes N_{inner}. Every inner node $i \subset N_{inner}$ has two child nodes, represented as $left(i)$ and $right(i)$. It also $\mathcal{E}$ contains an edge that connects input space X to the root node as shown in Figure 1.

For each node and edge, operations are assigned, which act upon the allocated data samples as illustrated in Figure 1. The process begins at the

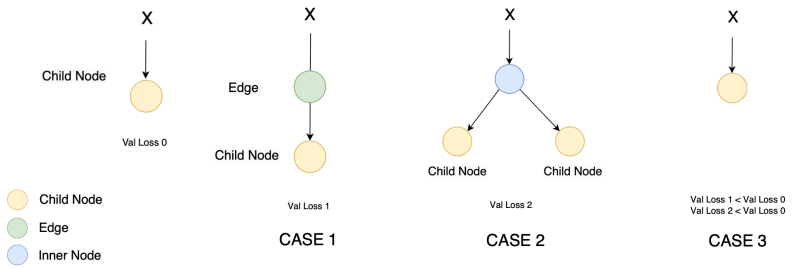

Figure 1: The hierarchical determination of SLM/SP tree.

root, where each sample undergoes transformation and navigates through the tree according to the assigned operation. An SLM/SP tree is constructed based on three fundamental modules of differentiable operations.

Inner Node $\mathbb{I}$: each inner node $i \subset N_{inner}$ is assigned with an inner module, $I_i^\theta : X_i \to [0,1] \subset \mathbb{I}$, parameterized with θ, in which X_i denotes the representation at node i. Each inner node routes the samples from the incoming edge to either the left or right child, we use the soft partitioning described in Section 3.1 where the decision output from the node denotes the probability of routing to the left child.

Edge $\mathbb{E}$: each edge $e \subset \mathcal{E}$ is assigned to one or multiple edge modules, and each edge module $E_e^\psi :\subset \mathbb{E}$, parameterized with ψ, transforms samples from the previous module and passes them to the next one.

Leaf Node $\mathbb{L}$: each leaf node $l \subset \mathcal{L}$ is assigned to a leaf module, each leaf module $L_l^\phi :\subset \mathbb{L}$, parameterized with ϕ, estimates the conditional distribution $p(y|x)$. For classification tasks with K classes, for example, we can define l^ϕ as a linear classifier on the transformed feature space $\mathcal{X}_l$, that outputs the distribution over classes q_k^l. Each leaf node, l has a K-D output vector, $\boldsymbol{q}^l$, whose kth element is equal to:

$$q_k^l = \frac{\exp(\phi_k^l)}{\sum_{k=1}^K \exp(\phi_k^l)}, \quad k = 1, \cdots, K. \tag{7}$$

where q_k^l is the probability of class k at the l-th leaf.

From input space X, data is passed through edges, inner nodes, and leaf nodes. For example, in Figure 1, to reach the distribution $\boldsymbol{q}^l$ at the leaf node $l = 1$, input X undergoes a series of transformations $X \to X_0^\psi := E_0^\psi(X) \to X_1^\psi := E_1^\psi(X_0^\psi) \to X_2^\psi := E_2^\psi(X_1^\psi)$ and the leaf module L_1 yields the predicted distribution $\boldsymbol{q}^l = p^{\phi,\psi}(y) := L_1^\phi(X_2^\psi)$. The probability of selecting this path is given by $(1 - I_0^\theta(X_0^\psi))I_1^\theta(X_1^\psi)$, which is the cascaded multiplication of the probability of the inner node module I_0^θ routing right and inner node module I_1^θ routing left.

The definition of specific operations assigned to the SLM/SP tree topology, $\mathcal{T}$, can be generalized to any differentiable functions. It enables the generalizability of the SLM/SP tree. For example, when the identity transform is assigned to edge modules $\mathbb{E}$, the topology of the SLM/SP tree is reduced to standard binary SDT. Commonly used operations in CNNs and ViTs could be applied to the SLM/SP tree operations to improve the model's effectiveness. For high-dimensional data sources (say, images), further details about operation assignments are discussed in Section 4.

3.2.2 Hierarchical Tree Determination

The tree architecture growth process is illustrated in Figure 2. We use a greedy search to find parameters at each node l to decide whether there is a gain in reducing the loss function by splitting the node or extending the node l with an edge. We employ a loss function that minimizes the cross-entropy at each leaf weighted by its path probability and the target distribution. Mathematically, the loss function at the th leaf node can be written as

$$L_\ell(x) = -\sum_{n=1}^{N}\sum_{l=1}^{L} P^\ell(x_n) \sum_{k=1}^{K} T_k^l \log q_k^l, \tag{8}$$

where $P^\ell(x_n)$ is the probability for input x_n to arrive at leaf node ℓ, and q_k^l is the probability distribution at leaf node ℓ for class k, and T_k^ℓ is the target distribution of class k at node l. The target distribution T_k^l is obtained by putting all training samples through the tree and finding the statistics of all classes at the node ℓ. The loss function evaluates the discrepancy between the predicted distribution q_k^l and the target distribution T_k^l, taking into account its path probability.

Starting from the root node, one of its child nodes grows in the breadth-first order and incrementally modifies the hierarchy of the SLM/SP tree. More specifically, as illustrated in Figure 2, we evaluate three cases during the process of each leaf node: 1) extend the node, 2) split the node into two child nodes, 3) keep the child node. We use the validation dataset to evaluate the effectiveness of the three cases with the loss in Eq. (8). That is, we fix the previously optimized parameters and optimize the parameters of the newly added modules, compare the validation loss improvement of the 1)extend and 2)split, and greedily select the case with the lowest validation loss. The intuition of evaluating the three cases is to utilize the local distribution and greedily explore the most effective option between soft feature space partitioning and richer local representation learning. We stop the split at a particular leaf node if there is no further validation loss improvement on the validation dataset, which is greedily adopted during the tree growth. We also use the maximum tree depth and the minimum sample number per node as stopping criteria.

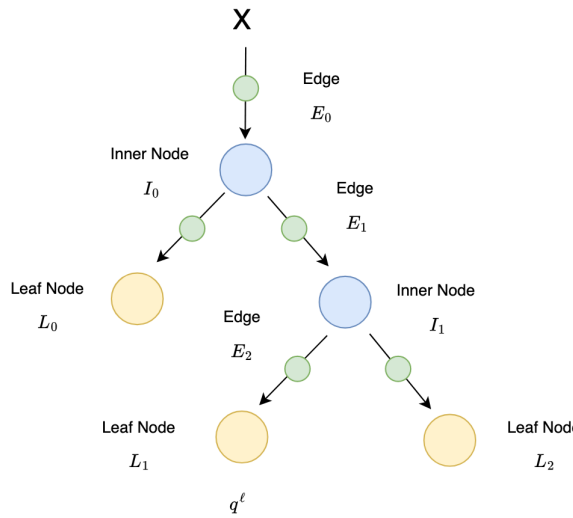

Figure 2: The overview of an example SLM/SP tree.

3.2.3 Module Parameters Determination

After the tree structure is finalized, we apply global optimization to update the projection vector and the bias at inner nodes for furthermore performance improvement. The total loss function from all leaf nodes can be written as

$$\begin{aligned}\mathcal{L} &= \sum_{n=1}^{N} L(\boldsymbol{x}_n) = \sum_{n=1}^{N} \sum_{\ell \in \text{LeafNodes}} L_\ell(\boldsymbol{x}_n) \\ &= -\sum_{n=1}^{N} \sum_{\ell \in \text{LeafNodes}} P^\ell(\boldsymbol{x}_n) \sum_k T_k^\ell \log q_k^\ell. \end{aligned} \tag{9}$$

Since $\mathcal{L}$ as well as all modules and operations assigned to the nodes and edges of the SLM/SP tree are differentiable; we use the mini-batch stochastic gradient descent (SGD) method to determine parameters and minimize $L(x)$. The global optimization of the module parameters can correct suboptimal decisions made during the local optimization during the growth and determination of the hierarchical tree structure, and empirically improve the generalization error.

3.3 Probabilistic Inference with SLM/SP Tree

The SLM/SP tree models the conditional distribution $p(y|x)$, which is a root-to-leaf path in the tree, with the final distribution determined by the leaf node. Each input x_n to the SLM/SP tree stochastically traverses the tree based on the decisions of inner node modules and undergoes a sequence of transformations with each edge module until it reaches a leaf node where the corresponding module predicts the label $\hat{y}$.

The inference with SLM/SP can be implemented in two schemes based on a the tradeoff between accuracy and computation: 1) full-path inference and 2) single-path inference.

3.3.1 Full-Path Inference

The full-path inference calculates the probabilistic distributions over all leaves. The predicted class of a single test sample, $\boldsymbol{x}_n$, is given by

$$\hat{y}(\boldsymbol{x}_n) = \arg\max_k \sum_{l=1}^{L} P^{\ell}(\boldsymbol{x}_n) q_k^{\ell}(\boldsymbol{x}_n). \tag{10}$$

To get the global estimation of $\hat{k}(\boldsymbol{x}_n)$, we need to traverse the full SLM/SP tree to estimate the probability $P^{\ell}(\boldsymbol{x}_n)$ for input $(\boldsymbol{x}_n)$ to arrive at a leaf node ℓ, and the local probabilistic distribution $q^{\ell}(\boldsymbol{x}_n)$ among k classes. When the depth of the SLM/SP tree increases, the time and memory complexity increases exponentially and may become too high to be practical if the total number of test samples is large.

3.3.2 Single-Path Inference

For a larger test sample size, we must simplify the multi-path inference for higher memory- and time efficiency at the expense of lower accuracy. The simplified scheme adopted in our experiments is the single-path inference. That is, it greedily traverses the tree in the direction of highest confidence of inner nodes to reach a leaf node, ℓ, and then predicts its class based on the maximum likelihood at that node, namely,

$$\hat{y}(\boldsymbol{x}_n) = \arg\max_k q_k^{\ell}, \tag{11}$$

where q_k^{ℓ} is the probability at the leaf node ℓ for class k learned from the training samples.

4 Image Classification with SLM/SP

The image classification framework with the SLM/SP tree is illustrated in Figure 3. We utilize the SSL feature as input feature space X for the proposed SLM/SP tree image classification framework. With the topology $\mathcal{T}$ of the SLM/SP tree, we adopt flexible designs of the tree module with different modules for the nodes $\mathcal{N}$ and edges $\mathcal{E}$. To evaluate the efficiency and effectiveness of the capabilities of the SLM/SP tree, we propose two learning roles of the SLM/SP tree in the image classification framework, i.e., 1) pure decision learning and 2) decision learning with supervised local representation learning. For the supervised local representation learning in 2), we propose a novel, efficient local representation learning module named SLAB for the edges $\mathcal{E}$ in the SLM/SP tree topology $\mathcal{T}$. To evaluate the learning capability of SLM/SP with the two learning roles, we adopt two module designs for the SLM/SP tree, respectively.

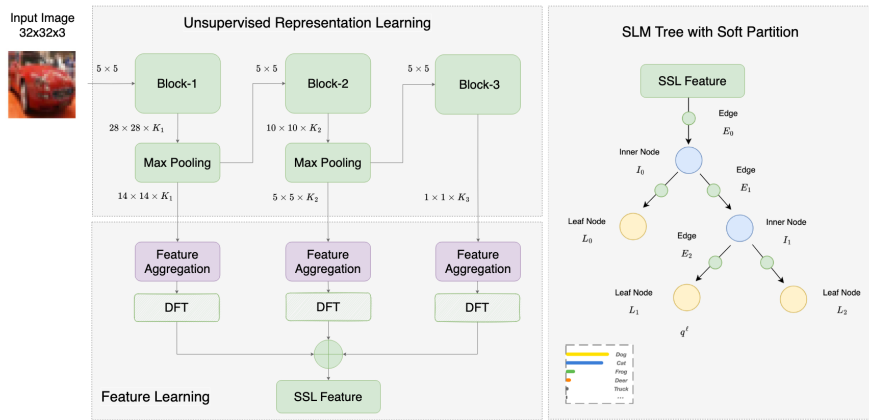

Figure 3: Image Classification Framework with SLM/SP Tree.

In this section, we first discuss the module designs with SLM/SP as two learning roles in Section 4.1, and then discuss the details of the proposed SLAB module in Section 4.2.

4.1 Design of SLM/SP Modules

4.1.1 SLM/SP for Pure Decision Learning

With extracted image representations, SLM/SP can be applied to the image classification as a pure decision-learning module. To evaluate the decision learning capability of the SLM/SP tree, we adopt the identity function for the edges $\mathcal{E}$ in the tree for the image classification framework. The topology

of the SLM/SP tree is then reduced to the standard binary SDT tree graph. With certain restriction, the SLM/SP tree nodes process the fixed global feature learned with SSL, and the goal of the tree is to model the conditional distribution $p(y|X)$ by soft subspace partitioning with the inner nodes I and local conditional distribution estimation with the leaf nodes L.

We adopt MLP with two hidden layers for the inner nodes I to achieve effective soft feature space partitioning. Then, we use linear classifiers for the leaf nodes L for the local conditional distribution estimation. We follow the same SLM/SP tree designs described in Section 3.1, including tree growth, inference strategies, and stopping criterion. The SSL feature for SLM/SP tree input X has 1D spatial dimension following [4, 5]. With SLM/SP as pure decision learning module, we can directly compare the SLM/SP decision learning capability with other classification methods like SVM and XGBoost as in Section 5, and demonstrate that SLM/SP serves as efficient and effective supervised decision learning module.

4.1.2 SLM/SP for Decision Learning with Supervised Local Representation Learning

To enhance the capability of the SLM/SP tree for image classification, SLAB is designed for the edges $\mathcal{E}$ in the tree for the image classification framework and serves as an effective local representation learning module. With the flexibility of the topological design of the SLM/SP tree, we adopt SLAB for edges $\mathcal{E}$ instead of identity function in pure decision learning. SLAB can be stacked on the edge of an SLM/SP tree with the extension strategy in tree growth. The local representation learned with SLAB discriminates the samples from more global to more local when going deeper into the SLM/SP tree. The detailed design of SLAB is presented in Section 4.2.

Like the SLM/SP for decision learning, the inner nodes I in the SLM/SP tree serve as a feature space soft partitioning module. The SLAB-based local representation enables more efficient design for the inner node module, i.e., a global average pooling layer for the input feature space and one hidden layer MLP for sample routing. Furthermore, we adopt a global average pooling layer before the linear classifier for the leaf node modules to fit with SLAB edge modules.

4.2 Supervised Local Representation Learning

4.2.1 Preliminaries

For the supervised efficient local feature learning module design, preliminaries include depth-wise separable convolutions, squeeze, and excitement blocks, and linear feature subspace learning.

Depth-wise Separable Convolutions. To enhance computational efficiency, depth-wise separable convolutions are utilized as a replacement for standard convolutions. As outlined by Howard *et al.* [14], a convolution with a weight tensor of dimensions k x k x M x N (where k x k represents the kernel size, and M and N denote the number of input and output channels, respectively) can be decomposed into two separate convolutions. The first is a depth-wise, or channel-wise, convolution with an M-channel k x k kernel, independently learning spatial correlations within each channel. The second is a pointwise convolution that combines channels to generate new features. Since combining a pointwise convolution and a k x k depth-wise convolution significantly reduces the number of parameters and computations, incorporating depth-wise separable convolutions into basic building blocks can drastically reduce parameters and computational costs. Our proposed SLAB employs these separable convolutions.

Squeeze and Excitement Block. As introduced by Hu *et al.* [15], the squeeze and excitement(SE) block is designed to enhance the representational power of the feature map by enabling it to perform dynamic channel-wise feature recalibration. The SE block takes a convolutional feature map as input, investigates the relationship between channels, and models the interdependencies between the channels of the convolutional features. Each channel in the input feature map is squeezed into a single numeric value using global average pooling, yielding a 1D tensor feature map. Then, two pointwise convolution layers are applied to the feature map; the first layer performs a dimension reduction of the input, followed by a ReLU activation function. The second layer performs a dimension increase, followed by a sigmoid activation function. The output of the second layer is used to perform a channel-wise multiplication with the original input feature map. This process allows the network to adaptively adjust each feature map's importance, enhancing its representational power. The SE block has been shown to improve performance significantly with negligible additional computational cost.

Linear Feature Subspace Learning. For a given set of real images as input, the features in the convolutional layers form a manifold of interest. It has been widely assumed that these manifolds of interest in neural networks could be embedded in low-dimensional subspaces. This implies that when we examine all individual pixels of a deep convolutional layer, the information encoded in those values resides in some manifold, which can be embedded into a low-dimensional subspace. This fact can be leveraged by simply reducing a layer's dimension, thus reducing the dimension of the operating space. This approach has been effectively utilized by Howard *et al.* [14] and has been incorporated into efficient model designs of other networks. With the standard design of the linear convolution and nonlinear activation setting of the dimensional reduction embedding, an activation function such as ReLU is applied to a particular channel that inevitably loses information. However, if we have

numerous channels and structures in the activation manifold, there could be redundant information that might still be preserved in other channels.

4.2.2 Subspace Learning Augmented Block (SLAB)

Inspired by the designs of depth-wise separable convolutions, squeeze and excitement blocks, and linear feature subspace learning, we propose a novel feature learning block named SLAB. SLAB utilizes the depth-wise separable convolution for efficient convolution design, squeeze-and-excitement block for enhancing the local representational power of the feature map, and linear feature subspace learning to preserve the information on the manifold of interest.

An overview of SLAB is illustrated in Section 4. We use the relative size of the cubic blocks to represent the spectral dimension difference between different feature maps. For the convolution layer design in the SLAB, we utilize pointwise convolution for feature channel adaptation and depth-wise convolution for channel-wise spatial information learning. In particular, with the input of a low-dimensional manifold consisting of features of interest, we first apply pointwise convolution to expand the spectral dimension, as discussed in the preliminary section. More redundant information is introduced to the high-dimensional feature map with channel expansion. We learn effective nonlinear local representation without losing much information from the manifold of interest. Then, with the high-dimensional feature map, a depth-wise convolution follows the pointwise convolution for expressive spatial dense feature learning capability. Afterward, the high-dimensional feature is embedded into a low-dimensional subspace as the transformed manifold of interest, and linear transformation is applied during the feature dimension reduction to preserve the information in the subspace manifold of interest. For the residual learning setting, the SLAB utilizes the low-dimensional subspace embedding learned for residual connection, i.e., the skip residual connection is between the input and output subspace manifold of interest.

SE block is adopted in the proposed SLAB design to enhance the local representative power. As discussed in the preliminaries, the SE block can improve the performance with negligible additional computational cost. To maximize the performance improvement capability, we attach the SE block to the high-dimensional feature map in the SLAB, after the dense spatial feature learning step with depth-wise convolution as shown in Figure 4.

SLAB is a novel design inspired by efficient convolution designs from deep learning. With the novel design of the pointwise and depthwise convolution sequence, linear and nonlinear transformation combination, SE block, and residual connection, we utilize the most efficient supervised learning for the subspace manifold of interest, which retains the same feature dimensions as

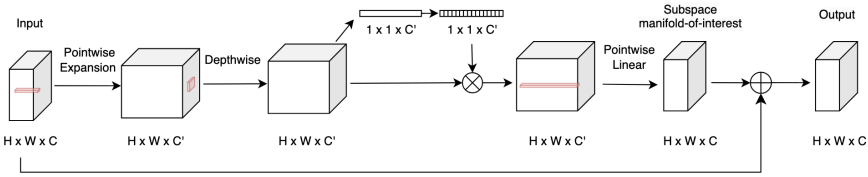

Figure 4: An Overview of SLAB. The low-dimensional manifold-of-interest feature space is of dimension H x W x C, the expanded high-dimensional feature space for dense spatial and spectral local representation learning is of dimension H x W x C'.

the input space and can be stacked with the edge extension in tree growth. In Section 5, we demonstrate the SLM/SP with SLAB for local representation learning with tremendous performance improvement over SLM/SP for decision learning design in terms of efficiency and effectiveness.

5 Experiments

5.1 SLM/SP for Pure Decision Learning with SSL feature

5.1.1 Experimental Setup

Datasets and Performance Metrics. To demonstrate the SLM/SP decision learning capability, we conduct experiments on the four image classification datasets: MNIST [27], Fashion-MNIST [42], CIFAR10 [20], and STL10 [6]. The experimental results in this section show the complex decision-learning capability of the SLM/SP tree compared to SVM and XGBoost.The four datasets are detailed as follows.

MNIST is a dataset of handwritten digits consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image of a digit representing a number from 0 to 9. The MNIST dataset is widely used for training and testing machine learning algorithms, particularly in computer vision. The MNIST dataset is often used to train and evaluate machine learning models for image classification tasks, as it provides a large, standard dataset for this purpose.

Fashion-MNIST is a dataset consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image associated with a label from 10 classes. Fashion-MNIST is intended to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms, as it shares the same image size and structure. The images in Fashion-MNIST are of higher quality and more diverse than those in the original MNIST dataset, making it a more challenging and realistic dataset for machine learning tasks.

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly selected images from each class. The training batches contain the remaining images in random order, but some may have more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

STL-10 is an image classification dataset inspired by the CIFAR-10 dataset but with some modifications. In particular, each class has fewer labeled training examples than in CIFAR-10. It consists of 10 classes of images, with a total of 5,000 images in the training set and 8,000 images in the test set. It is considered to be a more challenging and realistic dataset than some other popular image classification datasets, such as MNIST and Fashion-MNIST, due to the larger image size and the greater diversity of the images.

Image Classification Benchmarks. For performance benchmarking, we consider a couple of representative GL- and DL-based image classification methods. As mentioned earlier, GL-based methods utilize SSL feature and have three cascaded modules: 1) unsupervised representation learning, 2) supervised feature learning, and 3) supervised decision learning. Here, we compare three GL-based methods:

- GL-1: pixelhop [4] for module 1 and SVM for module 3;
- GL-2: pixelhop++ [5] for module 1 and XGBoost for module 3;
- GL-3: pixelhop++ for module 1 and SLM/SP for module 3.

For GL-2, we replace the linear classifier in Chen *et al.* [5] with the XGBoost Classifier, use DFT to select 2000 features for decision learning and set the tree depth to 5 and the tree number to 1000. For GL-3, we use DFT to select 512-D features as the input to SLM/SP and set the hidden layer neuron number to 512 and the maximum depth of the SLM/SP tree to 5. For CIFAR-10 and STL-10 datasets, we include VGG-16 [38] and LeNet-5 [26] in performance benchmarking. They are representatives of heavyweight and lightweight neural networks, respectively. We follow the standard design of VGG-16 and the settings in Chen *et al.* [5] for LeNet-5. We adopt the same hyperparameters in DL networks for CIFAR-10 and STL-10.

5.1.2 Experimental Results and Discussion

CIFAR-10. The performance comparison of five benchmarking methods for CIFAR-10 is shown in Table 1. We can categorize them into lightweight and heavyweight two groups based on the inference complexity and the model size given in the first two rows. LeNet-5, GL-1, GL-2, and GL-3 belong to the

Table 1: Performance Comparison for CIFAR-10 with GL.

CIFAR-10					
Methods	VGG-16	LeNet-5	GL-1	GL-2	GL-3 (SLM/SP)
FLOPs	875.06 M	14.74 M	21.30 M	17.73 M	20.29 M
(Against LeNet-5)	(59.40x)	(1x)	(1.45x)	(1.20x)	(1.38x)
Model Size	138.36 M	395.01 K	1.66 M	739.89 K	4.39 M
(Against LeNet-5)	(350.27x)	(1x)	(4.20x)	(1.87x)	(11.11x)
Accuracy (%)	93.15	68.72	71.37	75.29	87.36

lightweight group while VGG-16 belongs to the heavyweight group. GL-3, which uses the proposed SLM/SP classifier, outperforms LeNet-5, GL-1, GL-2 by 18.64%, 15.99%, and 12.07%, respectively. In particular, GL-2 and GL-3 are almost identical except for the last module. GL-2 uses the XGBoost classifier, while GL-3 uses the proposed SLM/SP classifier. Their significant performance gap demonstrates the effectiveness of SLM/SP over XGBoost. The additional costs in inference complexity (FLOPs) and model sizes appear well justified. As compared to the heavyweight VGG-16 model, GL-3 is inferior by 5.79% in classification accuracy. However, GL-3 demands much fewer inference FLOPs and fewer model parameters. The savings in memory and computation are attractive for mobile and edge computing.

STL-10. STL-10 is used to study the data deficiency setting. Due to the small number of training data in STL-10, DL-based methods cannot benefit much from their large model sizes as compared to the CIFAR-10 dataset. The performance comparison of five benchmarking methods for STL-10 is shown in Table 2. GL-3 achieves the best classification accuracy among all five benchmarking methods. At the same time, its inference complexity is close to the lowest. VGG-16 can only achieve classification accuracy similar to GL-3 but with much higher inference complexity (106x) and memory requirement (33x) for the training data deficiency case. The experimental results show that GL-3 can effectively utilize the limited training data and provide the best tradeoff between efficiency and effectiveness.

Table 2: Performance Comparison for STL-10 with GL.

STL-10					
Methods	VGG-16	LeNet-5	GL-1	GL-2	GL-3 (SLM/SP)
FLOPs	875.06 M	14.74 M	76.72 M	8.16 M	8.19 M
(Against LeNet-5)	(59.40x)	(1x)	(5.20x)	(0.55x)	(0.56x)
Model Size	138.36 M	395.01 K	8.10 M	427.02 K	4.40 M
(Against LeNet-5)	(350.27x)	(1x)	(20.51x)	(1.08x)	(11.14x)
Accuracy (%)	65.75	51.89	56.48	62.07	66.15

MNIST. For MNIST and Fashion MNIST two datasets, we compare the performance of three benchmarking methods, i.e., LeNet-5, GL-2, and GL-3. MNIST is an easy dataset. The results are shown in Table 3. The classification

Table 3: Performance Comparison for MNIST with GL.

MNIST			
Methods	LeNet-5	GL-2	GL-3 (SLM/SP)
FLOPs	846.08 K	1.49 M	2.05 M
(Against LeNet-5)	(1x)	(1.76x)	(2.42x)
Model Size	61.71 K	57.82 K	1.12 M
(Against LeNet-5)	(1x)	(0.94x)	(18.15x)
Accuracy (%)	99.04	99.19	99.30

accuracy saturates at 99% for most benchmarking methods. The improved classification accuracy rates of GL-3 over LeNet-5 and GL-2 are 0.26% and 0.11%, respectively. Although the gains are relatively small, they are achieved with additional inference complexity and a larger model size.

Fashion MNIST. The results are shown in Table 4. Its classification accuracy is around 90% for most benchmarking methods. The improved classification accuracy rates of GL-3 over LeNet-5 and GL-2 are 2.46% and 0.83%, respectively. The inference complexity of all three methods is comparable. Although the model size of GL-3 is larger than those of LeNet-5 and GL-2, it is still relatively small (i.e., 1.25M). It can be well deployed in mobile and edge devices.

Table 4: Performance Comparison for Fashion-MNIST with GL.

Fashion MNIST			
Methods	LeNet5	GL-2	GL-3 (SLM/SP)
FLOPs	3.58 M	2.69 M	3.05 M
(Against LeNet-5)	(1x)	(0.75x)	(0.85x)
Model Size	194.56 K	233.03 K	1.25 M
(Against LeNet-5)	(1x)	(1.20x)	(6.41x)
Accuracy (%)	89.74	91.37	92.20

5.2 SLM/SP for Decision Learning with SLAB

5.2.1 Experimental Setup

Datasets and Performance Metrics. To demonstrate the SLM/SP with Efficient Local Feature Learning, we conduct experiments on the four image classification datasets: MNIST [27], CIFAR10, CIFAR100 [20] and Tiny-Imagenet [25]. The MNIST and CIFAR10 datasets are detailed in Section 5.1, and the CIFAR-100 and Tiny-Imagenet are detailed as follows.

CIFAR-100 is introduced by Krizhevsky and Hinton [20] along with CIFAR-10. It is a subset of the Tiny Images Dataset and consists of 60,000 32x32 color images. The dataset is divided into 100 classes, each containing 600 images. These 100 classes are further grouped into 20 superclasses. Each image in the dataset comes with two labels: a fine label, the class it belongs to, and a coarse label, the superclass it belongs to. There are 500 training images and 100 testing images per fine class. The CIFAR-100 dataset is widely used for benchmarking in the field of machine learning, particularly for tasks related to image classification. It provides a challenging test bed due to its fine-grained classification tasks and the relatively small size of the images. We use CIFAR-100 to show the capability of the proposed classification framework for higher class numbers, and it is a direct comparison with the CIFAR10 dataset since the input images are from similar domains.

Tiny-Imagenet was proposed by Le and X. Yang [25]. It is a subset of the original ImageNet dataset [8]. It consists of 100,000 images across 200 classes, with each class containing 500 training images, 50 validation images, and 50 test images. The images are downsized to a resolution of 64x64 pixels, which makes the dataset more challenging for information extraction and image classification tasks. This dataset has since been used in numerous benchmarks. We use Tiny-Imagenet is a more challenging image classification task with more class numbers than CIFAR-100 and higher input image resolution.

Image Classification Benchmarks. As discussed in Section 4.2, for experimental setup of the SLM/SP with SLAB for local representation learning (SLM/SP-SLAB), we utilize the optimization in SLAB for major global to local representation learning at each node in the SLM/SP tree, and SSL for unsupervised rich decorrelated spectral information via Saab transform.

For the overall SLAB setting, we utilize a 3x3 filer size for the depth-wise convolution, set 0.0625 as the squeeze ratio for the SE block, and set the pointwise expansion ratio to 6. For the optimization, we apply Adams optimizer and set the learning rate as 1e-3 and weight decay as 1e-6. We set the learning rate scheduler as a step scheduler, and the learning rate drops to 0.1x every 50 epochs. We set the local optimization for each dataset as 100 epochs and the global optimization as 200 epochs.

For MNIST, we apply 5x5 Saab filters to the input 28x28 grayscale images and utilize DFT to select the most discriminated 16 greedily channels with the lowest spatial global average DFT loss, output 28x28x16 feature map as the global SSL representation. Then we apply SLM/SP to the global representation for joint local representation and decision learning. For SLAB, we set the channel number of the manifold-of-interest subspace feature as 64. To compare with the results in Section 5.1, we demonstrate the results with three learned SLM/SP tree architectures, named SLM-SLAB-tiny, SLM-SLAB-small, and SLM-SLAB-large, by settings the maximum depth of the SLM/SP tree to 3 for tiny, 4 for small, and 5 for large, respectively. We set the hidden layer

dimension for the inner node module as 0.5x of the input dimension. The batch size is set to 256.

For CIFAR10 and CIFAR100, similar to MNIST, we apply 5x5 Saab filters to the input 32x32x3 color images and DFT to greedily select 32 channels, yielding 32x32x32 global SSL feature map for the SLM/SP tree. For SLAB, we set the channel number of the manifold-of-interest subspace feature as 64 for CIFAR10 and 128 for CIFAR100. We set the hidden layer dimension for the inner node module as 0.5x of the input dimension. Linear classifiers are adopted for the leaf node modules. The batch size is set to 256.

For Tiny-Imagenet, we apply SSL to extract a 64x64x32 feature map for SLM/SP tree. For SLAB, we set the channel number of the manifold-of-interest subspace feature as 256. We set the hidden layer dimension for the inner node module as 0.5x of the input dimension. Linear classifiers are adopted for the leaf node modules. The batch size is set to 64.

5.2.2 Experimental Results and Discussion

MNIST. For MNIST, we compare the performance of benchmarking methods in Section 4.2 along with the SLM-SLAB. The results are shown in Table 5. The classification accuracy saturates at 99% for all the benchmarking methods. We compare the results from the three proposed designs, SLM-SLAB-tiny, SLM-SLAB-small, and SLM-SLAB-large, with previous results. The module has the most lightweight model size and inference among the benchmark methods for SLM-SLAB-tiny and still outperforms the LeNet-5 and GL series. The SLM-SLAB-tiny utilizes 0.3x of the model parameters and 0.76x FLOPs compared to LeNet-5, and the improved classification accuracy rate over LeNet-5 is 0.31%. The SLM-SLAB-tiny also outperforms GL-3 with SLM/SP for decision learning by 0.05%, with more than 50x smaller model size. Figure 5 shows the SLM/SP tree architecture discovered on MNIST for the SLM-SLAB-tiny setting and the probability of each class reaching each node in the SLM/SP tree. The hierarchical clustering effect among the ten classes after the tree growth and subspace learning process is the foundation of the efficient and high-performance classification achieved with SLM/SP. For SLM-SLAB-small, the performance can be further pushed to another 0.12% with 1.6x model size and FLOPs, which is still fewer than 0.5x of the previously the most lightweight LeNet-5 model. The best performance of MNIST is achieved with SLM-SLAB-large, with 2.25x model size and 4.5x FLOPs compare to LeNet-5, the SLM-SLAB-large achieves 0.52% accuracy improvement, which is significant considering the hard cases and saturated accuracy.

CIFAR-10. The performance comparison of five benchmarking methods for CIFAR-10 is shown in Table 6. We propose SLM-SLAB-small and SLM-

Table 5: Performance Comparison for MNIST with SLM-SLAB.

MNIST			
Methods	FLOPs	Model Size	Accuracy (%)
LeNet-5	846.08 K	61.71 K	99.04
GL-1	14.23 M	3.128 M	99.09
GL-2	1.51 M	104.28 K	99.19
GL-3	2.05 M	1.12 M	99.30
SLM-SLAB-tiny	645.62 K	18.5 k	99.35
SLM-SLAB-small	1.14 M	30.5 K	99.47
SLM-SLAB-large	3.8 M	139.35 K	99.56

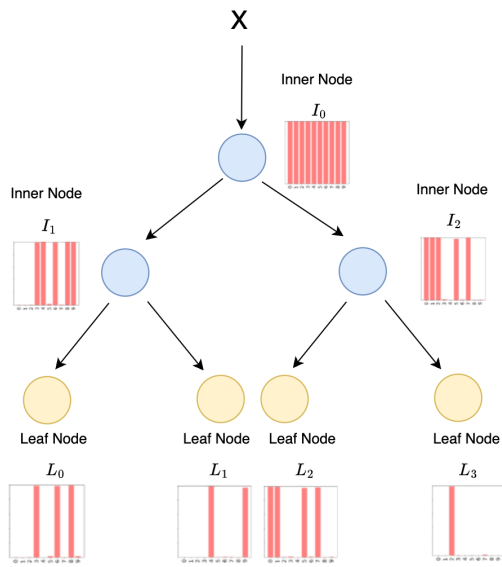

Figure 5: SLM/SP tree architecture discovered on MNIST for SLM-SLAB-tiny setting. With inner nodes routing the samples starting from the root, the probability of each class reaching each node in the SLM/SP tree is illustrated with the red histogram.

SLAB-large for the CIFAR-10 dataset and compare the results with the methods in Section 5.1. With GL-3, which uses the proposed SLM/SP classifier, there is already a significant performance gap with previous GL and LeNet-5, which demonstrates the effectiveness of SLM/SP for decision learning over XGBoost. The SLM-SLAB method further pushes the performance to the next level by 3.37% and 4.44%, with the small and large settings, respectively. With the proposed SLAB for efficient local representation learning, the overall parameter distribution is more efficient, and the SLM-SLAB-small module

Table 6: Performance Comparison for CIFAR10 with SLM-SLAB.

CIFAR10			
Methods	FLOPs	Model Size	Accuracy (%)
LeNet-5	14.74 M	395.01 K	68.72
GL-1	21.30 M	1.66 M	71.37
GL-2	17.83 M	1.47 M	75.29
GL-3	20.29 M	4.39 M	87.36
SLM-SLAB-small	37.62 M	1.31 M	90.73
SLM-SLAB-large	60.08 M	4.89 M	91.80

uses 0.3x the number of parameters to achieve performance improvement above. While introducing the convolutions in the SLAB, the filters need to apply to each spatial location of the feature map, which yields 1.85x FLOPs compare to GL-3. SLM-SLAB-large is proposed to match the model size with GL-3 for a more direct comparison. With similar model size and 3x FLOPs, SLM-SLAB-large achieves efficient module design with the best accuracy.

CIFAR-100. The performance benchmarking results are summarized in Table 7. With the increased number of classes in the dataset, the GL features fail to represent the data distribution with the fine class labeling. For example, with the GL-2 setting same as CIFAR-10, the model only achieves 25.24% test accuracy with model size as 3.73 M, which is higher than the compared mobilenetv2 and SLM-SLAB listed in Table 7. Hence, we compare our SLM-SLAB method with representative deep-learning methods, i.e. VGG-16, Resnet-18, and mobilenetv2. We utilize the VGG-16, Resnet-18 and mobilenetv2 as straightforward, effective, and efficient CNN designs for image classification. Compared to VGG-16, SLM-SLAB outperforms it by 0.18% while utilizing 25x fewer model parameters and 18x fewer FLOPs, demonstrating the method's high performance and lightweight design. Compare to mobilenetv2, SLM-SLAB has less 1.32% accuracy with 1.8x fewer model parameters and 1.8x fewer FLOPs. Resnet-18 achieves comparable performance in accuracy, with 8.4x larger model parameters utilized for optimization compared to SLM-SLAB. In summary, the results demonstrate that SLM-SLAB enables lightweight inference with its performance comparable with straightforward and efficient deep learning models.

Tiny-Imagenet. The performance benchmarking results are summarized in Table 8. Similar to CIFAR100, we compare the proposed SLM-SLAB with representative deep-learning methods, i.e. VGG-16, Resnet-18, and mobilenetv2. With the distribution of Tiny-Imagenet among 200 classes, the task is much more challenging than CIFAR-100. With the 43x model parameters and 20x FLOPs, the straightforward CNN design achieves the best accuracy among the listed methods. Similar to CIFAR-100, our SLM-SLAB

Table 7: Performance Comparison for CIFAR100 with SLM-SLAB.

CIFAR100			
Methods	FLOPs	Model Size	Accuracy (%)
VGG-16	666.34 M	34.01 M	64.48
Resnet-18	148.76 M	11.23M	65.61
MobileNetv2	68.4 M	2.36M	65.98
SLM-SLAB (Ours)	37.65 M	1.34 M	64.66

Table 8: Performance Comparison for Tiny-Imagenet with SLM-SLAB.

Tiny-Imagenet			
Methods	FLOPs	Model Size	Accuracy (%)
VGG-16	2.56 G	40.71 M	38.75
Resnet-18	595.24 M	11.28 M	25.90
MobileNetv2	253.60 M	2.57 M	33.13
SLM-SLAB (Ours)	125.58 M	0.94 M	32.12

achieves similar results compared to mobilenetv2 with 2.73x fewer model parameters and 2x fewer FLOPs. Our method outperforms the recent-18 model by 6.22%, with 12x fewer model parameters and 4.7x fewer FLOPs. The results demonstrate that our proposed SLM-SLAB achieves a lightweight model and inference compared to deep learning methods' efficient and effective designs.

6 Conclusion and Future Work

A novel image classification framework based on a tree-based classifier called Subspace Learning Machine with Soft Partitioning (SLM/SP) is proposed in this work. SLM/SP adopts the SDT data structure, modulated design, and adaptive topology. It learns an adaptive tree structure with local greedy subspace partitioning. SLAB is proposed for the SLM/SP tree edge for efficient local representation learning. With the proposed SLM/SP, the optimal weights at each edge and node can be obtained by optimizing a differentiable loss function using the mini-batch SGD method. The SLM/SP classifier enables efficient training, high classification accuracy, and small model size and can be highly competitive with DL networks against image classification datasets.

In this work, we experimented with SLM/SP on datasets with class numbers up to 200. As the number of classes increases, the SLM/SP tree grows deeper to achieve higher accuracy. This, in turn, requires an exponentially increasing number of training samples. It remains an open problem to achieve the

efficiency and effectiveness of SLM/SP for more diversified datasets with high-class numbers such as ImageNet and ImageNet-1k [8].

The SLM/SP method has the potential to contribute to green AI by providing a foundational classification model to be applied to various applications. For instance, in the realm of image-based classification, SLM/SP may be useful for tasks such as fake image detection [2, 3, 52, 53, 54], disease classification [32], and face gender classification [36]. Additionally, its ability to handle high-dimensional inputs could make it a viable solution for point cloud classification, segmentation, and registration [46, 47], as well as regression problems such as image generation [1, 28, 29, 51] and object tracking [48, 49, 50]. While we are excited about the potential of SLM/SP, its practical applications are still largely unexplored. It will be interesting to improve SLM/SP further and explore its applications for a wide range of classification and regression problems in the future.

Acknowledgments

This material is based on research sponsored by the US DEVCOM Army Research Laboratory (ARL) under contract number W911NF2020157. The US. The government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the US Army Research Laboratory (ARL) or the US Government. Computation for the work was supported by the University of Southern California Center for Advanced Research Computing (CARC).

References

[1] Z. Azizi, C.-C. J. Kuo, *et al.*, "PAGER: Progressive Attribute-Guided Extendable Robust Image Generation", *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[2] H.-S. Chen, S. Hu, S. You, C.-C. J. Kuo, *et al.*, "Defakehop++: An enhanced lightweight deepfake detector", *APSIPA Transactions on Signal and Information Processing*, 11(2), 2022.

[3] H.-S. Chen, M. Rouhsedaghat, H. Ghani, S. Hu, S. You, and C.-C. J. Kuo, "Defakehop: A light-weight high-performance deepfake detector", in *2021 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2021, 1–6.

[4] Y. Chen and C.-C. J. Kuo, "Pixelhop: A successive subspace learning (ssl) method for object recognition", *Journal of Visual Communication and Image Representation*, 70, 2020, 102749.

[5] Y. Chen, M. Rouhsedaghat, S. You, R. Rao, and C.-C. J. Kuo, "Pixelhop++: A small successive-subspace-learning-based (ssl-based) model for image classification", in *2020 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2020, 3294–8.

[6] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning", in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, 215–23.

[7] A. Criminisi and J. Shotton, *Decision forests for computer vision and medical image analysis*, Springer Science & Business Media, 2013.

[8] 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, 248–55.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale", *arXiv preprint arXiv:2010.11929*, 2020.

[10] H. Fu, Y. Yang, Y. Liu, J. Lin, E. Harrison, V. K. Mishra, and C.-.-C. J. Kuo, "Acceleration of subspace learning machine via particle swarm optimization and parallel processing", in *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, IEEE, 2022, 1019–24.

[11] H. Fu, Y. Yang, V. K. Mishra, and C.-C. J. Kuo, "Classification via Subspace Learning Machine (SLM): Methodology and Performance Evaluation", in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, 1–5.

[12] H. Fu, Y. Yang, V. K. Mishra, and C.-C. J. Kuo, "Subspace learning machine (slm): Methodology and performance", *arXiv preprint arXiv:2205.05296*, 2022.

[13] 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, 770–8.

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications", *arXiv preprint arXiv:1704.04861*, 2017.

[15] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, 7132–41.

[16] Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, and A. Criminisi, "Decision forests, convolutional networks and the models in-between", *arXiv preprint arXiv:1603.01250*, 2016.

[17] O. Irsoy, O. T. Yildiz, and E. Alpaydin, "Budding trees", in *2014 22nd international conference on pattern recognition*, IEEE, 2014, 3582–7.

[18] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm", *Neural computation*, 6(2), 1994, 181–214.

[19] P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulo, "Deep neural decision forests", in *Proceedings of the IEEE international conference on computer vision*, 2015, 1467–75.

[20] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images", *tech. rep.*, Toronto, Ontario: University of Toronto, 2009.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, 25, 2012.

[22] C.-C. J. Kuo and A. M. Madni, "Green learning: Introduction, examples and outlook", *Journal of Visual Communication and Image Representation*, 2022, 103685.

[23] C.-C. J. Kuo, M. Zhang, S. Li, J. Duan, and Y. Chen, "Interpretable convolutional neural networks via feedforward design", *Journal of Visual Communication and Image Representation*, 2019.

[24] D. Laptev and J. M. Buhmann, "Convolutional decision trees for feature learning and segmentation", in *Pattern Recognition: 36th German Conference, GCPR 2014, Münster, Germany, September 2-5, 2014, Proceedings 36*, Springer, 2014, 95–106.

[25] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge", *CS 231N*, 7(7), 2015, 3.

[26] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition", *Neural computation*, 1(4), 1989, 541–51.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, 86(11), 1998, 2278–324.

[28] X. Lei, G. Zhao, and C.-C. J. Kuo, "NITES: A Non-Parametric Interpretable Texture Synthesis Method", in *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, IEEE, 2020, 1698–706.

[29] X. Lei, G. Zhao, K. Zhang, and C.-C. J. Kuo, "TGHop: an explainable, efficient, and lightweight method for texture generation", *APSIPA Transactions on Signal and Information Processing*, 10, 2021, e17.

[30] A. Léon and L. Denoyer, "Policy-gradient methods for Decision Trees.", in *ESANN*, 2016.

[31] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation", in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, 663–70.

[32] X. Liu, F. Xing, C. Yang, C.-C. J. Kuo, S. Babu, G. El Fakhri, T. Jenkins, and J. Woo, "Voxelhop: Successive subspace learning for als disease classification using structural mri", *IEEE journal of biomedical and health informatics*, 26(3), 2021, 1128–39.

[33] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning", *IEEE transactions on pattern analysis and machine intelligence*, 34(4), 2011, 791–804.

[34] Z. Mei, Y.-C. Wang, X. He, and C.-C. J. Kuo, "GreenBIQA: A Lightweight Blind Image Quality Assessment Method", in *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2022, 1–6.

[35] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?", *Vision research*, 37(23), 1997, 3311–25.

[36] M. Rouhsedaghat, Y. Wang, X. Ge, S. Hu, S. You, and C.-C. J. Kuo, "Facehop: A light-weight low-resolution face gender classification method", in *International Conference on Pattern Recognition*, Springer, 2021, 169–83.

[37] J. Shotton, T. Sharp, P. Kohli, S. Nowozin, J. Winn, and A. Criminisi, "Decision jungles: Compact and rich models for classification", *Advances in neural information processing systems*, 26, 2013.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014.

[39] A. Suárez and J. F. Lutsko, "Globally optimal fuzzy decision trees for classification and regression", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12), 1999, 1297–311.

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, 1–9.

[41] R. Tanno, K. Arulkumaran, D. Alexander, A. Criminisi, and A. Nori, "Adaptive neural trees", in *International Conference on Machine Learning*, PMLR, 2019, 6166–75.

[42] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms", *arXiv preprint arXiv:1708.07747*, 2017.

[43] H. Xiao and G. Xu, "Neural decision tree towards fully functional neural graph", *Unmanned Systems*, 8(03), 2020, 203–10.

[44] Y. Yang, H. Fu, and C.-C. J. Kuo, "Design of supervision-scalable learning systems: Methodology and performance benchmarking", *Journal of Visual Communication and Image Representation*, 96, 2023, 103925.

[45] Y. Yang, W. Wang, H. Fu, C.-C. J. Kuo, *et al.*, "On supervised feature selection from high dimensional feature spaces", *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[46] M. Zhang, Y. Wang, P. Kadam, S. Liu, and C.-C. J. Kuo, "Pointhop++: A lightweight learning model on point sets for 3d classification", in *2020 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2020, 3319–23.

[47] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo, "PointHop: An Explainable Machine Learning Method for Point Cloud Classification", *IEEE Transactions on Multimedia*, 2020.

[48] Z. Zhou, H. Fu, S. You, C. C. Borel-Donohue, and C.-C. J. Kuo, "UHP-SOT: An Unsupervised High-Performance Single Object Tracker", in *2021 International Conference on Visual Communications and Image Processing (VCIP)*, IEEE, 2021, 1–5.

[49] Z. Zhou, H. Fu, S. You, and C.-C. J. Kuo, "Gusot: Green and unsupervised single object tracking for long video sequences", in *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2022, 1–6.

[50] Z. Zhou, H. Fu, S. You, C.-C. J. Kuo, *et al.*, "UHP-SOT++: An Unsupervised Lightweight Single Object Tracker", *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[51] Y. Zhu, S. Suri, P. Kulkarni, Y. Chen, J. Duan, and C.-C. J. Kuo, "An interpretable generative model for handwritten digits synthesis", in *2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2019, 1910–4.

[52] Y. Zhu, X. Wang, H.-S. Chen, R. Salloum, and C.-C. J. Kuo, "A-PixelHop: A Green, Robust and Explainable Fake-Image Detector", in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, 8947–51.

[53] Y. Zhu, X. Wang, H.-S. Chen, R. Salloum, and C.-C. J. Kuo, "Green Steganalyzer: A Green Learning Approach to Image Steganalysis", *arXiv preprint arXiv:2306.04008*, 2023.

[54] Y. Zhu, X. Wang, R. Salloum, H.-S. Chen, C.-C. J. Kuo, *et al.*, "RGGID: A Robust and Green GAN-Fake Image Detector", *APSIPA Transactions on Signal and Information Processing*, 11(2), 2022.