Learning Interpretable Queries for Explainable Image Classification with Information Pursuit

Stefan Kolek
LMU Munich

Aditya Chattopadhyay
Johns Hopkins University

Kwan Ho Ryan Chan
University of Pennsylvania

Hector Andrade-Loarca
LMU Munich

Gitta Kutyniok
LMU Munich

René Vidal
University of Pennsylvania

Abstract

Information Pursuit (IP) is an explainable prediction algorithm that greedily selects a sequence of interpretable queries about the data in order of information gain, updating its posterior at each step based on observed query-answer pairs. The standard paradigm uses hand-crafted dictionaries of potential data queries curated by a domain expert or a large language model after a human prompt. However, in practice, hand-crafted dictionaries are limited by the expertise of the curator and the heuristics of prompt engineering. This paper introduces a novel approach: learning a dictionary of interpretable queries directly from the dataset. Our query dictionary learning problem is formulated as an optimization problem by augmenting IP's variational formulation with learnable dictionary parameters. To formulate learnable and interpretable queries, we leverage the latent space of large vision and language models like CLIP. To solve the optimization problem, we propose a new query dictionary learning algorithm inspired by classical sparse dictionary learning. Our experiments demonstrate that learned dictionaries significantly outperform hand-crafted dictionaries generated with large language models.

1 Introduction

Information Pursuit (IP) [7, 5, 6] is a promising recent framework for explainable machine learning *by-design*. Given a pre-defined finite dictionary of semantic queries relevant to the task, the IP algorithm sequentially selects interpretable queries over the input data in order of information gain, updating the posterior at each step given the previously asked query-answer pairs. Once the posterior reaches a user-defined confidence level, IP stops querying the data and makes a prediction. The model explanation is a short sequence of interpretable query-answer pairs that IP selected for making a prediction.

IP has two key ingredients: (1) a mechanism that generates semantic and task-relevant queries and (2) a mechanism that answers the queries. In previous work [7, 5, 6], the query dictionary was hand-crafted in the form of manually annotated concepts, like those found in the CUB-200-2011 dataset [25], or concepts that were generated by a large language model, such as GPT-3 [4], following an elaborate human prompt. The answering mechanism followed one of two possible strategies: (a) training a model on expert concept annotations to predict the concept presence; (b) leveraging foundation models such as CLIP [20] to annotate the data with concept presence scores automatically. It is the latter mechanism that makes IP cost-efficient and scalable.

Previous work on IP [7, 5, 6], made the critical assumption that their hand-crafted dictionaries are *sufficient for the task*. However, in reality, this may not be the case. The dictionary curator may need more expertise and fail to provide a dictionary that is sufficient for making accurate predictions, or the dictionary may contain many queries that are not relevant to the task. This motivates the central question of our work:

Can we learn a data-driven query dictionary?

Paper Contributions. A learned query dictionary for IP should be (1) interpretable, (2) sufficient, and (3) yield short explanation lengths. We achieve this as follows. First, we leverage large vision models such as CLIP, providing a semantic embedding space where every point can represent a query and an associated concept. We define the answer mechanism as a thresholded dot product with the embedding of an image. The binary query answer representation ensures no information beyond a concept's presence or absence is disclosed. Since thresholded dot products are linear classifiers parametrized by a hyperplane, we cast the problem of learning a dictionary of queries as the problem of finding a set of hyperplanes in a semantic embedding space for IP. To formulate an optimization objective for learning a query dictionary, we augment IP's variational formulation with the parameters of our learnable queries. Given a query dictionary, Variational IP (V-IP) [5] learns a querier and classifier network on randomly sampled histories of query-answer pairs. The proposed query-learning algorithm alternates between dictionary and querier updates while continually optimizing the classifier. Our algorithm shares appealing connections to seminal sparse dictionary learning algorithms and outperforms hand-crafted baseline dictionaries generated by GPT-3 on three benchmark image classification datasets.

Paper Outline. In Section 2 we discuss related work. In Section 3 we introduce the IP framework [7] and its variational characterization V-IP [5]. In Section 4 we formulate a space of learnable queries and a query dictionary learning algorithm for IP. In Section 5 we compare our learned dictionary to baseline dictionaries on benchmark image classification datasets and analyze our algorithm. In Section 6 we discuss our results and the limitations of our method. Finally, we conclude in Section 7.

2 Related Work

Explainable AI has evolved along two distinct lines: *post-hoc* explanations and explainability *by-design*. The post-hoc approach [15, 21, 2, 23] treats the model as a black box, allowing curators to focus initially on maximizing performance and addressing explainability subsequently. Despite their advantages, post-hoc methods have been criticized [22, 24] for insufficient faithfulness. In contrast, the explainability by-design paradigm embeds interpretability into the core of the model design, ensuring that the explanations provided are inherently aligned with the model's internal reasoning processes. IP [7, 5, 6], which forms the foundation of our work, is part of a broader family of by-design frameworks, which include Concept Bottleneck Models (CBMs) [13, 26, 18] and Prototype Models [8, 17].

Concept Bottleneck Models (CBMs) predict targets by applying an interpretable model, such as a linear model, to an intermediate layer of human-understandable attributes. A key similarity between CBMs and IP is their reasoning through interpretable concepts. Both methodologies necessitate a task-relevant set of these concepts. In the CBM framework, as explored in [26, 18], and also in IP [6], this set of interpretable concepts is often generated using a large language model like GPT-3 [4]. However, a language model-generated set of concepts may not be optimally tailored to the task's training data. Our work addresses this gap by introducing a method to learn IP's query dictionary directly from the data, a contribution that should also interest the CBM community.

Prototype-based models, exemplified by the Prototypical Part Network (ProtoPNet) [8] and ProtoTree [17], classify images by identifying prototypical parts and synthesizing evidence from these prototypes for the final prediction. While conceptually distinct from IP, prototype models relate significantly to our work due to their data-driven approach to learning prototypes. Similarly, our work introduces a novel methodology for learning the query dictionary of interpretable concepts in IP directly from the data.

3 Background

This section reviews IP, its implementation-friendly variational formulation V-IP [5], and the baseline query dictionary for explainable image classification with IP.

3.1 Information Pursuit

We denote random variables with capital letters and their realizations as lowercase letters. Our work defines all random variables over a common sample space Ω. Let $X : \Omega \to \mathcal{X}$ and $Y : \Omega \to \mathcal{Y}$ be the input data and its ground truth label. We denote $P(Y \mid X)$ as the ground truth conditional distribution of Y given X. A *query* q is a function $q : \mathcal{X} \to \mathcal{A}$ mapping data X to an answer $q(X)$. For example, if X is an image of an animal and q represents the query "Does it have wings?", then one can define $q(X)$ as $+1$ for images of animals with wings and -1 otherwise. A *query dictionary* is a collection $\mathcal{Q} = \{q^{(i)}\}_{i=1}^{n}$ of n queries.

Given a dictionary $\mathcal{Q}$ of semantic queries, one can construct an interpretable predictor by sequentially selecting queries from $\mathcal{Q}$ such that the average number of queries needed for an accurate prediction is minimal. Solving this problem is hard [7], but IP can greedily approximate the solution. For a fixed data point x^{obs}, IP selects the queries in order of information gain:

$$q_1 := \underset{q \in \mathcal{Q}}{\arg\max}\, I\left(q\left(X\right); Y\right), \tag{1}$$

$$q_{k+1} := \underset{q \in \mathcal{Q}}{\arg\max}\, I\left(q\left(X\right); Y \mid q_{1:k}\left(x^{\text{obs}}\right)\right), \tag{2}$$

where I denotes mutual information, q_{k+1} denotes the query selected in iteration $k+1$, and

$$q_{1:k}\left(x^{\text{obs}}\right) = \left\{\left(q_i, q_i\left(x^{(obs)}\right)\right)\right\}_{i=1}^{k} \tag{3}$$

denotes the history of observed query-answer pairs. The IP algorithm terminates after τ iterations if the entropy of the posterior $P\left(Y \mid q_{1:\tau}\left(x^{\text{obs}}\right)\right)$ is below a user-defined threshold or surpasses a fixed budget of τ iterations. After IP terminates querying the input data, IP predicts the label

$$\underset{y \in \mathcal{Y}}{\arg\max}\, P\left(Y = y \mid q_{1:\tau}\left(x^{\text{obs}}\right)\right). \tag{4}$$

3.2 Variational Information Pursuit

A natural approach to selecting the most informative query in Eqs. (1) and (2) is to first learn a generative model $p(Q(X), Y)$ and then compute mutual information for each query $q \in \mathcal{Q}$. However, both intermediate steps are computationally prohibitive for high-dimensional data, such as images. To address this challenge, Variational Information Pursuit (V-IP) [5] solves an equivalent tractable variational optimization objective parameterized by neural networks. The V-IP approach defines two neural networks: (1) a classifier network

$$f_\theta : q_{1:k}\left(x^{\text{obs}}\right) \mapsto y \in \mathcal{Y} \tag{5}$$

mapping a history of k observed query-answer pairs to a class label, and (2) a querier network

$$g_\psi : q_{1:k}\left(x^{\text{obs}}\right) \mapsto q_{k+1} \in \mathcal{Q} \tag{6}$$

mapping a history of k observed query-answer pairs to a newly selected query.

For an input X, let S denote a randomly sampled history of query-answer pairs:

$$S = \{(q_j, q_j\left(X\right)) \mid j \in \mathcal{I}\},$$

where $\mathcal{I} \subset \{1, \ldots, |\mathcal{Q}|\}$ are query indices randomly sampled from a pre-specified distribution such as uniform sampling. Applying the querier $g_\psi(S) \in \mathcal{Q}$ yields a new query given a history of randomly drawn query-answer pairs. In V-IP, the querier is trained to select a new query that when added to the history improves the prediction of the classifier. More precisely, the querier and classifier networks are jointly trained with stochastic gradient descent (SGD) to minimize the KL divergence between the true posterior $P(Y \mid X)$ and the posterior $P_\theta(Y \mid S, A_\psi)$ predicted by the classifier f_θ, which is the distribution of the output Y given a history of query-answer pairs S and a newly added query-answer pair A_ψ chosen by the querier g_ψ based on the history S, i.e.

$$\begin{aligned}
&\min_{\theta,\psi} J_{\mathcal{Q}}(\theta, \psi) && (7)\\
&J_{\mathcal{Q}}(\theta, \psi) := \underset{X,S}{\mathbb{E}}\left[D_{KL}\left(P\left(Y \mid X\right) \| P_\theta\left(Y \mid S, A_\psi\left(X, S\right)\right)\right)\right]\\
&P_\theta\left(Y \mid S, A_\psi\left(X, S\right)\right) := f_\theta\left(S \cup A_\psi\left(X, S\right)\right)\\
&A_\psi\left(X, S\right) := \{(q_\psi, q_\psi\left(X\right))\}\\
&q_\psi := g_\psi(S).
\end{aligned}$$

In practice, the random history S is first sampled uniformly from the query dictionary (random sampling stage) and later fine-tuned using the querier g_ψ to build up the history (biased sampling stage) [5]. Mathematically, [5] prove that inference with an optimal querier and classifier network that minimize Eq. (7) gives the same sequence of queries as IP would select (see Eqs. (1) and (2)).

3.3 Establishing Baseline Query Dictionaries

As a baseline query dictionary for explainable image classification, we follow prior work [6, 18], which uses GPT-3 [4] to generate relevant concepts for every class in the dataset. For example, one of the prompts is "List the most important features for recognizing something as a ⟨class⟩". For more details, we refer to [18]. To obtain the query answers, each image in the dataset is annotated with the dot product between the normalized CLIP embedding of the image and the normalized CLIP embedding of a concept represented as text.

4 Learning a Query Dictionary

Suppose we have a space $V \subset \{q : \mathcal{X} \to \mathcal{A}\}$ of queries that we can learn for the IP dictionary. To formulate query dictionary learning as an optimization problem, we augment the V-IP objective with learnable queries for the dictionary. Let f_θ and g_ψ be classifier and querier networks that take query-answer histories from a dictionary $\mathcal{Q} = \{q^{(i)}\}_{i=1}^n \subset V$ as input. We define the query dictionary learning problem for V-IP as:

$$\min_{\mathcal{Q}=\{q^{(i)}\}_{i=1}^n \subset V} \min_{\theta,\psi} J_{\mathcal{Q}}(\theta, \psi). \tag{8}$$

A learned dictionary for IP should be (1) interpretable, (2) sufficient for the task, and (3) yield short explanation lengths. Achieving interpretability requires that the V-IP querier selects queries $q \in V$ corresponding to a semantic concept. Sufficiency and short explanation lengths require a space of queries V that is amenable to optimization, enabling the minimization of objective Eq. (8).

In the following, we begin by addressing the question of what the space of learnable queries V should be. Upon choosing V, we present an optimization algorithm to minimize the query dictionary learning objective in Eq. (8).

4.1 The Space of Learnable Queries

We leverage large vision and language models such as CLIP that provide aligned image and text embeddings

$$E_I : \mathcal{X} \to \mathcal{E} \subset \mathbb{R}^d \tag{9}$$
$$E_T : \mathcal{T} \to \mathcal{E} \subset \mathbb{R}^d, \tag{10}$$

where E_I denotes an image encoder, E_T a text encoder, and $\mathcal{E}$ a semantic embedding space. The baseline dictionary, as discussed in Section 3.3, defines the query dictionary as

$$\mathcal{Q} \coloneqq \left\{ q^{(i)} = \langle E_T(c_i), E_I(\cdot) \rangle \right\}_{i=1}^n, \tag{11}$$

where $\{c_i\}_{i=1}^n$ is a set of concepts generated by GPT-3, and the encoder pair (E_I, E_T) is the CLIP encoder pair with normalized output embeddings. Mathematically, the baseline dictionary corresponds to a subset of the dual space $\mathcal{E}^*$, i.e., the space of all linear forms on the semantic embedding space $\mathcal{E}$. A straightforward initial approach might therefore define the space of learnable queries as

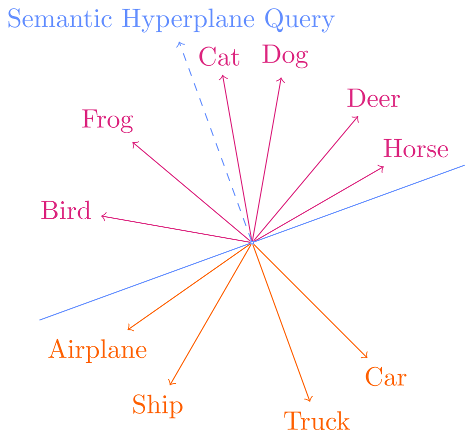


Figure 1: A hyperplane in a semantic embedding space like CLIP can effectively distinguish between high-level concepts, such as "animals versus vehicles", and defines an image query.

$$V = \mathcal{E}^* = \{q : \mathcal{X} \to \mathbb{R} \mid q = \langle w, E_I(\cdot) \rangle, \ w \in \mathcal{E}\}. \tag{12}$$

In this approach, a query is a soft linear classifier in the semantic space $\mathcal{E}$ discriminating for the presence of a concept. Since linear classifiers are parameterized by a hyperplane, we can cast the problem of learning a query dictionary as the problem of finding a set of hyperplanes in the semantic embedding space $\mathcal{E}$. Fig. 1 illustrates the notion of *semantic hyperplane queries* with an example.

Although the choice $V = \mathcal{E}^*$, is simple and natural, it overlooks a subtle yet crucial problem that compromises the interpretability of learnable queries.

4.1.1 Limitations of Soft Query Answers

Here we argue that query-answer chains using continuous-valued (soft) query answers are not interpretable, as it is often not possible to assign a semantic concept to such queries. By definition, a query $q \in V = \mathcal{E}^*$ outputs continuous-valued (soft) query answers. We describe geometrically why a single soft query answer can reveal information about the presence of multiple concepts, which makes the query difficult to interpret: consider a hyperplane query $q \in V = \mathcal{E}^*$ that is associated with some concept (see Fig. 1 for an illustration). Given an image x, a soft answer $q(x)$ reveals the exact distance of $E_I(x)$ to the hyperplane query. However, knowing the distance to one hyperplane query can also disclose positional information of $E_I(x)$ relative to another hyperplane query representing a second concept. Therefore, one soft query answer can simultaneously reveal information about the presence of multiple concepts, making it difficult to interpret the query. To retain interpretability, we propose to learn queries with hard answers, i.e., thresholded dot product answers.

4.1.2 Hard Query Answers

We define the space of learnable queries with hard answers as

$$V := \{q \mid q = \operatorname{sgn}\left(\langle w, E_I(\cdot)\rangle - b\right),\ w \in \mathcal{E}, b \in \mathbb{R}\}.$$

Mathematically, V is parameterized by elements from the dual space $\mathcal{E}^*$ and threshold parameters, i.e., $V = \mathcal{E}^* \times \mathbb{R}$. Each query is represented as a hyperplane in the semantic embedding space $\mathcal{E}$ that discriminates concept presence with a hard linear classifier. In contrast to soft query answers, hard query answers cannot reveal the presence of multiple concepts due to their binary nature.

Due to the geometric connection to hyperplanes (see Fig. 1), we refer to $q \in V = \mathcal{E}^* \times \mathbb{R}$ as a *hard semantic hyperplane query* and $q \in V = \mathcal{E}^*$ as a *soft semantic hyperplane query.* In the subsequent section, we present an optimization algorithm that learns a dictionary of hard semantic hyperplane queries for IP.

4.2 A Query Dictionary Learning Algorithm

Having motivated the choice $V = \mathcal{E}^* \times \mathbb{R}$ of learnable queries as hard semantic hyperplanes queries, we revisit the query dictionary learning problem for V-IP:

$$\min_{\mathcal{Q} \in (\mathcal{E}^* \times \mathbb{R})^n} \min_{\theta, \psi} J_{\mathcal{Q}}(\theta, \psi), \tag{13}$$

where $J_{\mathcal{Q}}(\theta, \psi)$ denotes the V-IP optimization objective for a query dictionary $\mathcal{Q}$. The most straightforward optimization strategy concurrently applies SGD to the querier, classifier, and query dictionary parameters to jointly minimize the V-IP objective in Eq. (7). However, this approach overlooks that the querier operates on a fixed query dictionary and may require several updates to adapt to a single dictionary update. Therefore, we adopt a more nuanced phased optimization process. Our final optimization algorithm relies on following three key ingredients:

1. **Batch Norm Parameterization of Queries.** This addresses how to parameterize the normal vector w and threshold b of a learnable hyperplane query. Using batch normalization [11] techniques to parameterize the hyperplane queries enhances stability and performance.

2. **Straight-Through Estimator for Query Gradients.** Employing a straight-through estimator [3] allows for effective backpropagation through the non-differentiable sign function in hyperplane queries.

3. **Phased Optimization.** Adopting a phased optimization approach allows updating the querier given a learned dictionary and conversely updating the dictionary given a learned querier.

In the following, we discuss each ingredient in detail.

4.2.1 Batch Norm Parameterization of Queries

To facilitate optimization, we employ a batch normalization [11] parameterization of our hyperplane queries. Consider a learnable vector $v \in \mathcal{E}$ in the semantic embedding space and learnable scalar values $\gamma, \beta \in \mathbb{R}$. We define the mean and standard deviation of the dot product activations as follows:

$$\begin{aligned} \mu(v) &:= \mathbb{E}_X \left\langle \frac{v}{\|v\|}, E(X) \right\rangle, \\ \sigma(v) &:= \sqrt{\mathbb{E}_X \left[\left(\left\langle \frac{v}{\|v\|}, E(X) \right\rangle - \mu(v) \right)^2 \right]}. \end{aligned} \tag{14}$$

A hyperplane query q can be reparameterized as

$$\operatorname{sgn}\left(\frac{\left\langle \frac{v}{\|v\|}, E(X) \right\rangle - \mu(v)}{\sigma(v)} \cdot \gamma + \beta \right), \tag{15}$$

where w and b, the hyperplane's normal vector and threshold, are defined as

$$w = \frac{v}{\|v\|} \cdot \frac{\gamma}{\sigma(v)} \quad \text{and} \quad b = \frac{\mu(v)}{\sigma(v)} \cdot \gamma - \beta. \tag{16}$$

The key advantage of this reparameterization is that it leads to normalized dot product activations, thereby enhancing numerical stability and facilitating faster convergence. In practice, this reparameterization is equivalent to employing a batch normalization layer (BN) [11], which during training uses empirical estimates over the current batch for $\mu(v)$ and $\sigma(v)$, and during inference, relies on a running average estimate. Our batch norm parameterized semantic hyperplane dictionary becomes:

$$\mathcal{Q} = \left\{ \operatorname{sgn}\left(\mathrm{BN}_{\gamma_i,\beta_i}\left(\left\langle \frac{v_i}{\|v_i\|}, E\left(\cdot\right) \right\rangle \right) \right) \right\}_{i=1}^{n}, \tag{17}$$

where $\{v_i, \gamma_i, \beta_i\}_{i=1}^n$ are the trainable parameters of our dictionary.

4.2.2 Straight-Through Estimator of Query Gradients

Due to the non-differentiability of the sign function, we utilize the straight-through estimator [3] to derive gradient estimates for gradient descent optimization. In the forward pass, a classifier taking query answers as input treats the binary query values

$$\operatorname{sgn}\left(\mathrm{BN}_{\gamma_i,\beta_i}\left(\left\langle \frac{v_i}{\|v_i\|}, E\left(x\right) \right\rangle \right) \right) \tag{18}$$

as constants during the forward pass but uses the gradient of the continuous surrogate function

$$\tanh\left(\mathrm{BN}_{\gamma_i,\beta_i}\left(\left\langle \frac{v_i}{\|v_i\|}, E\left(x\right) \right\rangle \right) \right) \tag{19}$$

during the backward pass.

4.2.3 Phased Optimization

We adopt the following four-stage optimization process.

- *Initialization.* We initialize the query dictionary parameters described in Eq. (17) with a unit multivariate Gaussian distribution for the semantic embedding vectors $\{v_i\}_{i=1}^n \subset \mathcal{E}$ and constant scalar parameters $\{\gamma_i = 1\}_{i=1}^n$ and $\{\beta_i = 0\}_{i=1}^n$.

- *Warm-Up Phase.* First, we focus on “warming up” the dictionary, training only the dictionary and the classifier to minimize cross-entropy on randomly sampled query-answer histories without using a querier.

Algorithm 1: Query Dictionary Learning for V-IP

```
1  Initialization
2  Initialize querier network $g_\psi$ and classifier network
     $f_\theta$. Initialize query dictionary $Q$ as random
     hyperplanes in semantic embedding space;
3  Warmup Phase
4  Train $Q$ and $f_\theta$ to minimize cross-entropy with
     random sampling without using querier;
5  while $Q$ has not converged do
6      Querier Update Phase
7      Freeze $Q$;
8      Unfreeze $g_\psi$ ;
9      Train $g_\psi$ and $f_\theta$ on V-IP objective with
         dictionary $Q$ using random sampling and
         subsequent biased sampling;
10     Dictionary Update Phase
11     Freeze $g_\psi$;
12     Unfreeze $Q$ ;
13     Train $Q$ and $f_\theta$ on V-IP objective using biased
         sampling with $g_\psi$;
14 end
```

- *Querier Update Phase.* We freeze the query dictionary and train the querier and classifier to minimize the V-IP objective using random sampling and subsequent biased sampling.

- *Dictionary Update Phase.* We freeze the querier from the previous phase and train the dictionary and classifier on the V-IP objective using biased sampling.

The algorithm is summarized in Algorithm 1.

4.3 Connections with Sparse Dictionary Learning

Our query dictionary learning algorithm (Algorithm 1) shares connections to sparse dictionary learning algorithms, such as K-SVD [1], which have been pivotal in advancing the state-of-the-art in various image and video processing applications [10, 16]. Sparse dictionary learning seeks to discover a sparse representation of input signals, represented as a linear combination of basic elements, or atoms, and the determination of the atoms themselves. These atoms, constituting the

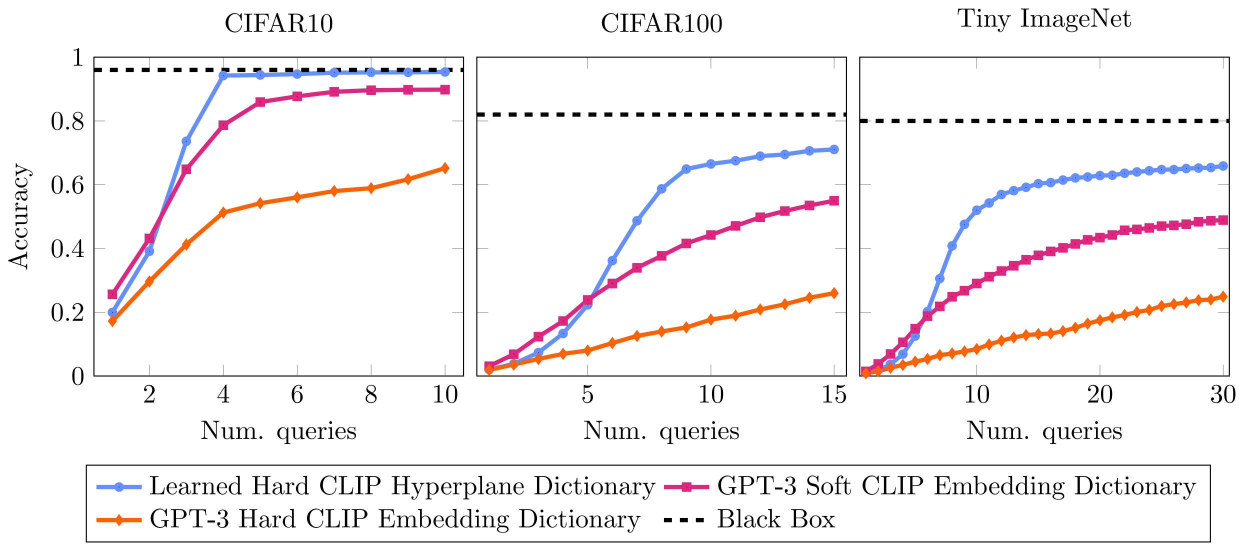


Figure 2: Comparing the test accuracy of the V-IP classifier, post-observation of k queries selected by the V-IP querier from (●) Learned Hard CLIP Hyperplane Dictionary, (■) GPT-3 Soft CLIP Embedding Dictionary, and (♦) GPT-3 CLIP Hard Embedding dictionary.

dictionary, are typically refined through an iterative process: a sparse coding phase, which computes a sparse representation of the input signal utilizing the current dictionary, and a dictionary update phase to minimize the reconstruction error between the input signal and its sparse reconstruction. The analogy to our query dictionary learning method unfolds as follows.

- The query dictionary in Algorithm 1 consists of atoms that correspond to queries for the input X, akin to how, in sparse dictionary learning, atoms are vectors in the signal space of X. This analogy was already pointed out in [6].

- The querier phase in Algorithm 1 plays the role of the sparse coding phase in dictionary learning. Indeed, [6] establishes a profound link between IP and orthogonal matching pursuit (OMP) [19], an important sparse coding technique. When employing random projections of dictionary atoms as queries, the authors demonstrate that IP closely approximates OMP.

- The dictionary update phase in Algorithm 1 refines the dictionary with the classifier to reduce the classification error post-observation of queries sampled with the current querier. This mirrors the dictionary update phase in sparse dictionary learning, which refines the dictionary to reduce the reconstruction error for the current sparse code.

In light of the inherent connections to sparse dictionary learning, we expect our query dictionary learning approach to echo the principle from classical sparse dictionary learning that learned dictionaries outperform hand-crafted ones. In the subsequent section, we verify this via experiments.

5 Experiments

This section presents numerical experiments for our proposed query dictionary learning algorithm (Algorithm 1). We perform these experiments on three benchmark image classification datasets: CIFAR10 [14], CIFAR100 [14], and Tiny ImageNet, which is a subset of ImageNet [9] featuring 200 classes, with each class comprising 500 training images and 50 test images, all downscaled to 64x64 pixels. For each dataset, we use 10% of the training data as a validation set for hyperparameter tuning. For the semantic embeddings (E_I, E_T), we use CLIP [20] with a ViT-L/14 backbone. In line with previous V-IP work [5], we use two-layered MLPs for both the querier and classifier architectures. Since MLPs can only handle fixed-sized inputs, while the querier and classifier are expected to handle variable length histories of query-answer pairs, we use masking to handle unobserved query-answer pairs. A maximum budget of queries is set for each dataset, allowing 10 queries for CIFAR10, 15 for CIFAR100, and 30 for Tiny ImageNet. The following parameters are tuned using the validation set: the learning rate for the Adam optimizer [12] for the querier, classifier, and dictionary; batch size and number of epochs in each phase of Algorithm 1; and the number of queries in the learned dictionary. Detailed implementation aspects are outlined in Appendix A.

We compare our learned dictionary, denoted as *Learned Hard CLIP Hyperplane Dictionary*, with the baseline dictionary, denoted as *GPT-3 Soft CLIP Embedding Dictionary*, and the binarized baseline dictionary, denoted as *GPT-3 Hard CLIP Embedding Dictionary*. To compute the GPT-3 Hard CLIP Embedding Dictionary, we calculate the dot product thresholds as the average dot product per query over the training set. This aligns with the expectation that dot product magnitudes cluster around higher values when a concept is present and lower otherwise.

5.1 Learned Dictionaries Perform Better

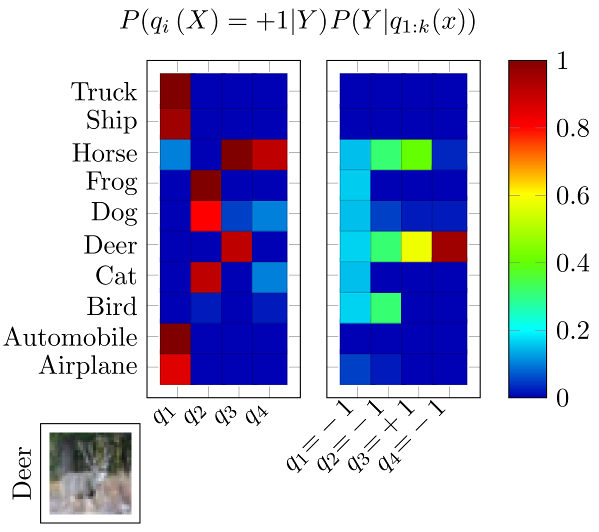
In Fig. 2, we compare the test accuracy of the V-IP classifier, post-observation of k queries selected by the V-IP querier, for our Learned Hard CLIP Hyperplane Dictionary, the baseline GPT-3 Soft CLIP Embedding Dictionary, and the GPT-3 Hard CLIP Embedding Dictionary. Our learned dictionaries surpass the GPT-3 dictionaries in terms of performance across all three benchmark datasets. For instance, on CIFAR10, the learned dictionary matches the black-box test accuracy of 95% after just four queries. In contrast, the baseline GPT-3 Soft CLIP Embedding Dictionary reaches 80% accuracy following the same number of queries.

5.2 Interpretability of Learned Dictionaries

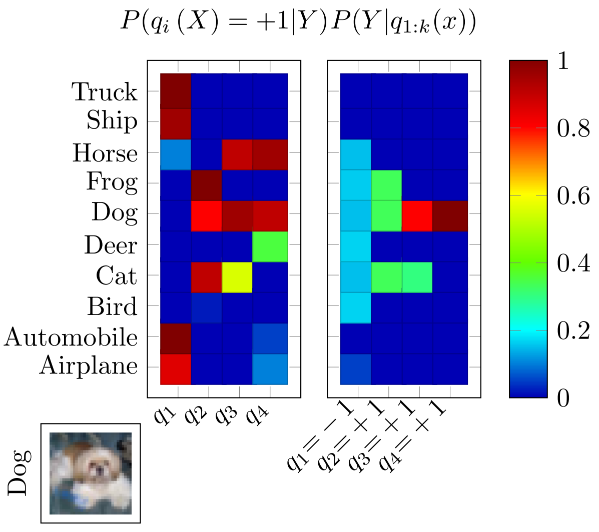
We interpret two V-IP predictions in Fig. 3 with a Learned Hard CLIP Hyperplane Dictionary trained on CIFAR10. In the figure, x denotes an image from the CIFAR10 test set (class deer in Fig. 3a and class dog in Fig. 3b). The right matrix plot shows the posterior probability matrix $\{P(Y = y \mid q_{1:k}(x))\}_{k,y}$, which is obtained by applying the classifier to the history of k query-answer pairs for image x selected in the order of the V-IP querier. The left matrix plot shows the class-conditional probability matrix $\{P(q_i(X) = +1 \mid Y = y)\}_{i,y}$ (estimated with an empirical average over the training dataset), indicating which images from which class lie on the positive side of the learned hyperplane query.

In the following, we interpret the first four queries selected by the querier that lead to the classifier confidently identifying the image in Fig. 3a as a deer: **1st Query:** This query discriminates between vehicles and animals, assigning $+1$ to the former and -1 to the latter. The deer image is correctly queried as an animal, prompting a uniform posterior increase for all animal classes. **2nd Query:** After identifying the image as an animal, the querier probes for the "smaller four-legged animals", signified by $+1$ for cats, dogs, and frogs, and -1 for other classes. The deer is correctly queried as $q_2(x) = -1$, leading the classifier to reduce the likelihood of the animal classes dog, cat, and frog while increasing the probability for bird, deer, and horse. **3rd Query:** The classifier believes the image is a bird, deer, or horse. Subsequently, the querier probes for the "hoofed animals," signified by $+1$ for deer and horses and -1 for other classes. The deer image is correctly queried as one of the hoofed animals, which leads to the classifier distributing the posterior probability between deer and horse. **4th Query:** The final query aims to distinguish between deer and horses. The query assigns $+1$ for horses and -1 for other classes. Observing $q_4(x) = -1$, the classifier conclusively identifies the image as a deer.

We interpret the dog image in Fig. 3b analogously. The first two selected queries are identical for the dog and deer images because they are both queried as animals. However, the answer to query two is $q_2(x) = +1$ since the dog image belongs to the "smaller four-legged animals". Query three probes for most of the "domesticated animals" in CIFAR10. The query answer $q_3(x) = +1$ for the dog image prompts the classifier to focus on the classes dog and cat. Finally, query four probes for deer, horse, or dog images. Since the dog image returns $q_4(x) = +1$, the classifier confidently identifies the image as a dog. Our examples demonstrate that our learned semantic hyperplane queries separate classes based on higher-level image semantics.



(a)



(b)

Figure 3: Interpreting two predictions for V-IP with the Learned Hard CLIP Hyperplane Dictionary trained on CIFAR10: (a) interpretation for image x of class deer; (b) interpretation for image x of class dog. The queries $\{q_i\}_{i=1}^4$ are selected in order by the V-IP querier for each image. For each example, we plot two matrices. The left matrix plot shows the class-conditional probability matrix $\{P(q_i(X) = +1 \mid Y = y)\}_{i,y}$ indicating which images from which class lie on the positive side of the hyperplane query. The right matrix plot shows the posterior class probability matrix $\{P(Y = y \mid q_{1:k}(x))\}_{k,y}$ to visualize the predicted class probabilities by the V-IP classifier after observing the first $1 \leq k \leq 4$ query-answer pairs that the V-IP querier selected. In particular, the first query selected by the querier probes for "animals versus vehicles". For both images, the classifier distributes the posterior among the animal classes uniformly after observing the first query-answer. For the second query, the dog, unlike the deer, is identified as one of the "smaller four-legged animals". The querier adapts. In particular, for the deer image, the third query probes whether the image is a "hoofed animal", while for the dog image the query probes for most of the "domesticated animals".

5.3 Algorithm Analysis

This section summarizes our empirical insights on Algorithm 1, with detailed results in the Appendix B.

Importance of Warm-Up Phase. We observe the optimal performance of Algorithm 1 when the warm-up phase runs until convergence. The learned dictionaries converge after a single iteration of the warm-up phase, querier update phase, and dictionary update phase, each running until convergence.

Impact of Dictionary Initialization. Initializing the learnable dictionary with the GPT-3 baseline dictionary revealed only a mild speed-up in convergence, not a significant improvement in accuracy compared to randomly initialized dictionaries of the same size (see Appendix B.2).

Advantages of Batch Norm Parameterization. Parameterizing learnable hard semantic hyperplane queries with batch normalization techniques enhances performance, likely due to better calibration of CLIP dot products (see Appendix B.3).

Efficacy of Overcomplete Dictionaries. Our findings suggest that learned dictionaries with more queries can achieve higher accuracy, with an optimal size balancing performance and convergence speed (see Appendix B.4).

Impact of Encoder Architecture. Experiments with different CLIP backbones show significantly superior performance and faster convergence with larger backbones, like ViT-L/14, compared to smaller ones, like ViT-B/16 (see Appendix B.5).

Phased versus Joint Optimization. Jointly optimizing querier, classifier, and dictionary on the V-IP objective fares significantly worse than our phased approach in Algorithm 1 (see Appendix B.6 for a comparison).

Hard Query Answers are Necessary for Interpretability. Algorithm 1 with soft query answers reaches over 50% test accuracy after observing a single query-answer pair on CIFAR10 compared to 25% with the baseline GPT-3 Soft CLIP Embedding Dictionary (see Appendix B.7). Considering CIFAR10's class labels (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks), it is implausible that a single query corresponding to a semantic concept reveals over 50% of the class labels. Therefore, the query must correspond to multiple concepts that are difficult to disentangle or not be semantic.

6 Discussion

Our experiments show that learned dictionaries can significantly outperform hand-crafted dictionaries for IP. The wide gaps between the learned dictionaries and the baseline GPT-3 dictionaries further suggest that hand-crafted dictionaries, even by powerful pre-trained language models such as GPT-3, are far from optimal. Our query dictionary learning algorithm offers a principled method to generate interpretable and task-sufficient query dictionaries that are adapted to the training data.

Despite these advancements, our method remains limited by the lack of an automated process for translating learned hard semantic hyperplane queries into their corresponding interpretable concepts. The current approach to interpretation relies on manual effort to identify which concept activates the positive/negative side of the hyperplane query. Future work aims to develop an algorithm that automatically matches learned queries with their inherent natural language concepts. Additionally, the efficacy of our method is contingent upon the quality of features provided by the foundational model, in our case, CLIP. Tasks for which CLIP provides inadequate feature representations would inherently limit the performance of our method.

7 Conclusion

Hand-crafted query dictionaries for IP may be insufficient for real-world tasks. We addressed this problem by proposing learned data-driven query dictionaries for IP that are both sufficient and interpretable. Our work presents the first algorithm to learn a query dictionary for explainable image classification with IP. We leveraged large vision models like CLIP to formulate learnable and interpretable queries and frame query dictionary learning as an optimization problem by augmenting the variational IP objective with learnable query parameters. To minimize the objective, we introduced a phased optimization algorithm with appealing connections to classical sparse dictionary learning. Our query dictionary learning algorithm outperformed the baseline GPT-3 CLIP dictionaries on benchmark image classification datasets, echoing the principle from classical sparse dictionary learning that learned dictionaries outperform hand-crafted ones.

8 Acknowledgements

R. Vidal was supported by the Army Research Office under the Multidisciplinary University Research Initiative contract W911NF-17-1-0304, the NSF grant 2031985, the Simons Foundation Mathematical and Scientific Foundations of Deep Learning (MoDL) grant 135615, and the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE2139757. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. G. Kutyniok was supported by LMUexcellent, funded by the Federal Ministry of Education and Research (BMBF) and the Free State of Bavaria under the Excellence Strategy of the Federal Government and the Länder as well as by the Hightech Agenda Bavaria. She also acknowledges support by the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research, the Munich Center for Machine Learning (MCML) as well as the German Research Foundation under Grant DFG-SFB/TR 109 and Project C09. H. Andrade-Loarca was supported by the German Research Foundation under Grant DFG-SFB/TR109, ProjectC09.

References

[1] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006. 6

[2] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015. 2

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 5, 6, 13

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1, 2, 3

[5] Aditya Chattopadhyay, Kwan Ho Ryan Chan, Benjamin David Haeffele, Donald Geman, and Rene Vidal. Variational information pursuit for interpretable predictions. In *The Eleventh International Conference on Learning Representations*, 2022. 1, 2, 3, 7, 13

[6] Aditya Chattopadhyay, Ryan Pilgrim, and Rene Vidal. Information maximization perspective of orthogonal matching pursuit with applications to explainable ai. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 1, 2, 3, 7

[7] Aditya Chattopadhyay, Stewart Slocum, Benjamin D Haeffele, Rene Vidal, and Donald Geman. Interpretable by design: Learning predictors by composing interpretable queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7430–7443, 2022. 1, 2, 3

[8] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32, 2019. 2

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 7, 12

[10] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006. 6

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. 5, 6

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 8

[13] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pages 5338–5348. PMLR, 2020. 2

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7, 12

[15] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017. 2

[16] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse representation for color image restoration. *IEEE Transactions on image processing*, 17(1):53–69, 2007. 6

[17] Meike Nauta, Ron Van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14933–14943, 2021. 2

[18] Tuomas Oikarinen, Subhro Das, Lam Nguyen, and Lily Weng. Label-free concept bottleneck models. In *International Conference on Learning Representations*, 2023. 2, 3, 12

[19] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993. 7

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1, 7

[21] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 2

[22] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019. 2

[23] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 2

[24] Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun Preece. Sanity checks for saliency metrics. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6021–6029, 2020. 2

[25] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. *The Caltech-UCSD Birds-200-2011 Dataset.* Jul 2011. 1

[26] Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19187–19197, June 2023. 2

A Experiment Details

In the following, we outline the implementation details for the experiments in Section 5.

Datasets. Our experiments employ three datasets: CIFAR10 [14], CIFAR100 [14], and Tiny Imagenet. The latter is a subset of ImageNet [9] featuring 200 classes, with each class comprising 500 training images and 50 test images, all downscaled to 64x64 pixels. For each dataset, we use 10% of the training data as a validation set for hyperparameter tuning.

Query Dictionaries. The GPT-3 generated concept sets for the CIFAR10, CIFAR100, and Tiny ImageNet baseline dictionaries are available in the GitHub repo[1] from [18]. Note we use the concept set for ImageNet from [18] to generate the baseline query dictionary for Tiny Imagenet. As described in Section 4.1, the learnable query dictionaries are defined as

$$\mathcal{Q} = \left\{ q^{(i)} = \operatorname{sgn}\left(\mathrm{BN}_{\gamma_i,\beta_i}\left(\left\langle \frac{v_i}{\|v_i\|}, E(\cdot) \right\rangle \right) \right) \right\}_{i=1}^{n},$$

where $\{v_i\}_{i=1}^n \subset \mathbb{R}^d$, $\{\gamma_i\}_{i=1}^n \subset \mathbb{R}$, and $\{\beta_i\}_{i=1}^n$ are learnable parameters.

Query Answers. Denote the set of all query answers for a dictionary $\mathcal{Q}$ and a data point x as

$$Q(x) = \left\{ q^{(i)}(x) \right\}_{i=1}^{n}. \tag{20}$$

For the baseline GPT-3 dictionary, we compute the query answer as

$$q^{(i)}(x) = \langle E_T(c_i), E_I(x) \rangle, \tag{21}$$

where $\{c_i\}_{i=1}^n$ is the concept set generated with GPT-3 and (E_T, E_I) are the CLIP text and image encoders with unit norm embeddings. Following [18], we Z-score normalize the query answers given a training set $\{x_k\}_{k=1}^N$ as

$$q^{(i)}(x) \leftarrow \frac{q^{(i)}(x) - \hat{\mu}}{\hat{\sigma}}, \tag{22}$$

[1] https://github.com/Trustworthy-ML-Lab/Label-free-CBM/tree/main/data/concept_sets

where

$$\hat{\mu} := \frac{1}{nN}\sum_{j=1}^{n}\sum_{k=1}^{N} q^{(j)}(x_k) \tag{23}$$

$$\hat{\sigma} := \sqrt{\frac{1}{nN}\sum_{j=1}^{n}\sum_{k=1}^{N}\left(q^{(j)}(x_k)-\hat{\mu}\right)^2}. \tag{24}$$

The Z-score normalization speeds up convergence during training. Note that this normalization is employed only for the GPT-3 query dictionaries.

For the learned dictionaries, we cannot perform Z-score normalization because the CLIP dot product changes throughout training. Instead we employ a BatchNorm layer, i.e.,

$$q^{(i)}(x) = \operatorname{sgn}\left(\mathrm{BN}_{\gamma_i,\beta_i}\left(\left\langle \frac{v_i}{\|v_i\|}, E(x)\right\rangle\right)\right). \tag{25}$$

To obtain a gradient estimate, we apply the straight-through trick: in the forward pass of the classifier and querier, we show the binary answers returned by the sign function, while in the backward pass, we use

$$q^{(i)}(x) = \tanh\left(\mathrm{BN}_{\gamma_i,\beta_i}\left(\left\langle \frac{v_i}{\|v_i\|}, E(x)\right\rangle\right)\right). \tag{26}$$

Representing and Updating the History. In V-IP, the input to the classifier f_θ and querier g_ψ is a query-answer pair history S of variable length. Following the original V-IP work [5], we represent the history S for a sample x, as a binary mask M and $\mathcal{Q}(x) \odot M$, where $\odot$ denotes the Hadamard product. If history S contains the i-th query-answer pair, then $M_i = 1$, otherwise $M_i = 0$.

Suppose $S^{(k)}$ denotes a history of k query-answer pairs for x and $M^{(k)}$ is the associated binary mask, then we can update the history with a new query using the querier g_ψ:

$$M^{(k+1)} = M^{(k)} + g_\psi(S^{(k)}), \tag{27}$$

$$S^{(k+1)} = S^{(k)} + \mathcal{Q}(x) \odot g_\psi(S^{(k)}). \tag{28}$$

In particular, g_ψ returns a one-hot encoding to select the next query. We use a straight-through softmax layer on the query logits to backpropagate gradients through g_ψ.

Classifier and Querier Architecture. The classifier and querier architecture is a two-layer, fully connected neural network. Fig. 4 illustrates the architecture with a diagram. The size of the query dictionary only affects the final output dimension of the querier g_ψ, while the number of class labels only affects the final output dimension of the classifier f_θ. We never share the weights between the classifier and the querier networks. We apply a softmax layer to the class and query logits to obtain probabilities for each class and query, respectively. During training, we employ a straight-through softmax [3] with temperature parameter T for the querier output. We linearly decay the temperature T for every experiment from 1.0 to 0.2 for all epochs. We find V-IP training is not particularly sensitive to the annealing scheme for T.

Training Details of Query Dictionary Learning for V-IP. All hyperparameters were tuned using the validation set.

- **CIFAR10:** We sample histories of no more than 10 query-answer pairs. We used a dictionary size of 50. We used 50 warm-up epochs, 10 random sampling epochs followed by 50 biased sampling epochs in the querier phase, and 100 epochs in the dictionary phase. We used a learning rate of 0.001 for the Adam optimizer with a batch size equal to 512.
- **CIFAR100:** We sample histories of no more than 30 query-answer pairs. We used a dictionary size of 892 (the same size as the baseline GPT-3 dictionary). We used 300 warm-up epochs, 20 random sampling epochs followed by 80 biased sampling epochs in the querier phase, and 200 epochs in the dictionary phase. We used a learning rate of 0.001 for the Adam optimizer with a batch size equal to 512.
- **Tiny ImageNet:** We sample histories of no more than 30 query-answer pairs. We used a dictionary size of 1000. We used 400 warm-up epochs, 20 random sampling epochs followed by 80 biased sampling epochs in the querier phase, and 200 epochs in the dictionary phase. We used a learning rate of 0.001 for the Adam optimizer with a batch size equal to 512.

Training Details of Classic V-IP with GPT-3 Dictionaries. All hyperparameters were tuned using the

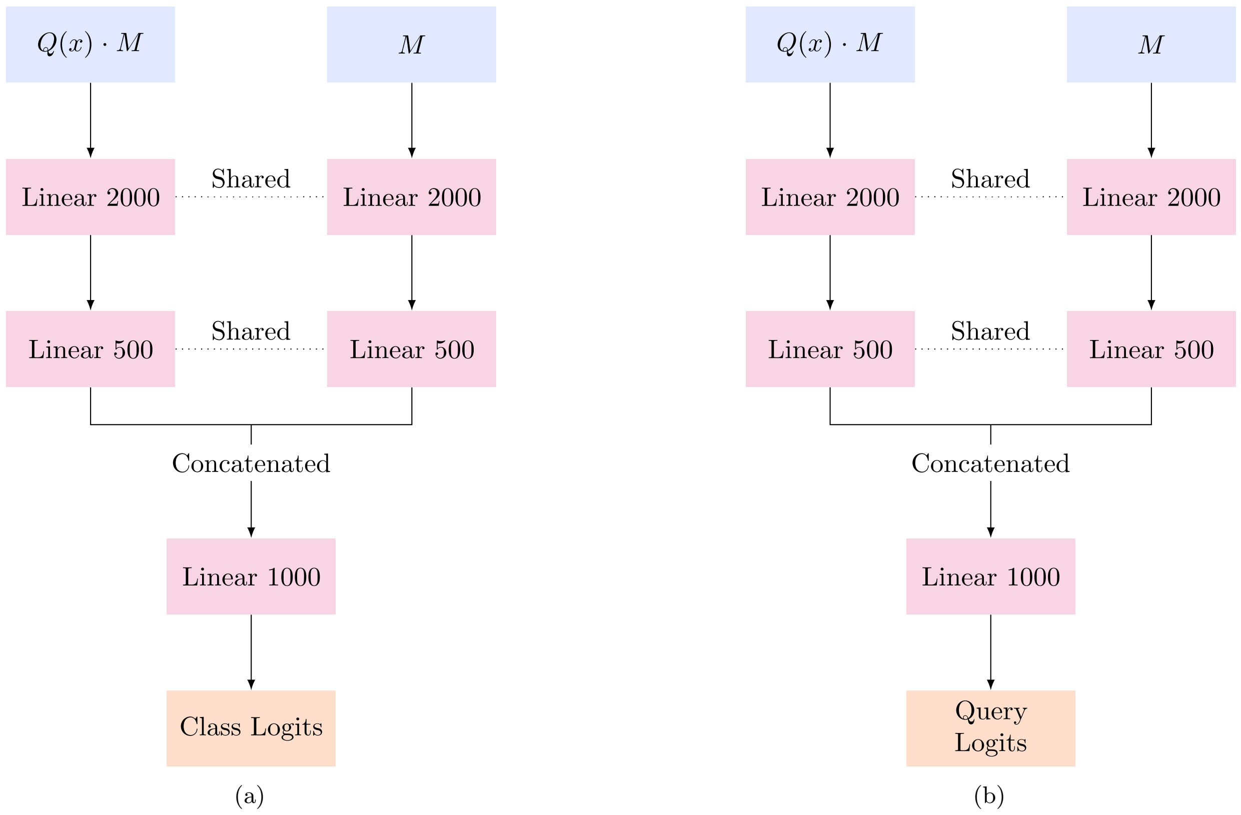


Figure 4: Diagram of the neural network architecture for (a) classifier f_θ and (b) querier g_ψ. "Shared" indicates that two linear layers share weights. "Concatenated" implies the output from previous layers is concatenated. Every arrow $\rightarrow$ before the concatenation and after the input layer applies a LayerNorm, followed by ReLU. In the forward pass of g_ψ, we convert the query logits into a one-hot encoding with a straight-through softmax.

validation set.

- **CIFAR10:** We sample histories of no more than 10 query-answer pairs. We use 300 random sampling epochs followed by 100 biased sampling epochs. The learning rate for the Adam optimizer is 0.0001 with a batch size equal to 512.

- **CIFAR100:** We sample histories of no more than 30 query-answer pairs. We use 400 random sampling epochs followed by 200 biased sampling epochs. The learning rate for the Adam optimizer is 0.0001 with a batch size equal to 512.

- **Tiny ImageNet:** We sample histories of no more than 30 query-answer pairs. We use 400 random sampling epochs followed by 300 biased sampling epochs. The learning rate for the Adam optimizer is 0.001 with a batch size equal to 512.

We employed these hyperparameters for the soft and hard query answers.

Training Details of Black Box Baseline. For the black box baseline, we use a MLP and a dropout layer following the CLIP image encoder. The exact architecture is outlined in Table 1. We train the MLP for 100 epochs with a batch size equal to 512 and use the Adam optimizer with learning rate 0.001 and weight decay 0.0001.

Layer Type	Output Dimension
CLIP Encoder	768
Dropout	768
Fully Connected	2000
Fully Connected	1000
Fully Connected	500
Fully Connected	400
Fully Connected	300
Linear	# Classes

Table 1: Layer types and their output dimension for the black box baseline. he first layer is a dropout layer with parameter $p = 0.5$ and takes the CLIP features as input. We use LayerNorm and ReLU after fully connected layers. The final layer outputs the class logits.

B Algorithm Analysis

In this section, we analyze Algorithm 1. In particular, we perform an ablation study on the algorithm and study the effect of hyperparameters such as dictionary size and CLIP backbone size. The analysis uses the CIFAR10 and CIFAR100 datasets.

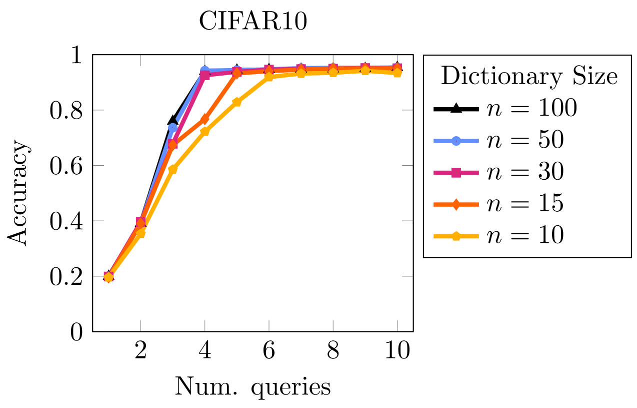


Figure 5: Comparing the test accuracy of the V-IP classifier, post-observation of k queries selected by the V-IP querier from learned dictionaries of size (▲) 100, (●) 50, (■) 30, (♦) 15, and (⬟) 10.

B.1 Importance of Warm-Up Phase

We compare Algorithm 1 with and without warm-up phase. In Fig. 7, we show that running a warm-up phase can boost accuracy. In particular, for CIFAR10, we achieve 73% accuracy after three observed queries with warm-up phase compared to 66% without the warm-up phase. For CIFAR100, the accuracy gap is even more pronounced. Note that without a warm-up phase we run the querier phase on the randomly initialized query dictionary. The dictionary phase subsequently refines the dictionary on queries sampled with the trained querier. However, the querier was tuned on a randomly initialized dictionary and is therefore likely suboptimal. The warm-up phase ensures the querier phase runs with a trained dictionary.

B.2 Impact of Dictionary Initialization

In Fig. 8, we compare our method with random dictionary initialization versus GPT-3 dictionary initialization. In particular, the GPT-3 dictionary initialization uses $v_i = E_T(c_i)$, $\gamma_i = 1$, and $\beta_i = 0$ in

$$\mathcal{Q} = \left\{ \operatorname{sgn}\left(\mathrm{BN}_{\gamma_i,\beta_i}\left(\left\langle \frac{v_i}{\|v_i\|}, E(\cdot) \right\rangle \right) \right) \right\}_{i=1}^{n}, \tag{29}$$

where $\{c_i\}_{i=1}^n$ are the GPT-3 generated concepts for the task. We find that the GPT-3 initialization speeds up convergence, however it does not improve accuracy compared to random initialization. In fact, the random initialization performs marginally better. This suggests one can learn query dictionaries from scratch without relying on special initializations.

B.3 Advantages of Batch Norm Parameterization

In Fig. 9, we compare the BatchNorm parameterziation to the standard dictionary parameterization

$$\mathcal{Q} = \left\{ q^{(i)} = \operatorname{sgn}\left(\left\langle \frac{v_i}{\|v_i\|}, E(\cdot) \right\rangle \cdot \gamma_i - \beta_i \right) \right\}_{i=1}^{n}, \tag{30}$$

where $\{v_i\}_{i=1}^n \subset \mathbb{R}^d$, $\{\gamma_i\}_{i=1}^n \subset \mathbb{R}^d$, and $\{\beta_i\}_{i=1}^n \subset \mathbb{R}^d$ are trainable parameters. We find that the BatchNorm parameterization significantly improves accuracy.

B.4 Efficacy of Overcomplete Dictionaries

In Fig. 5, we compare Algorithm 1 with different dictionary sizes on CIFAR10. In general, we find that dictionaries with more queries can achieve higher accuracy.

However, larger dictionaries take longer to converge and accuracy stops growing at some dictionary size. Hence, an optimal dictionary size is small enough to convergence quickly and sufficiently large to achieve optimal accuracy.

B.5 Impact of Encoder Architecture

In Fig. 10, we compare our method with the larger ViT-L/14 CLIP backbone and the smaller ViT-B/16 CLIP backbone. We find that our method performs vastly better with the larger CLIP backbone, which underscores the importance of a strong semantic embedding in our framework.

B.6 Phased versus Joint Optimization

We compare our phased approach in Algorithm 1 to the naive joint optimization approach, where SGD is applied concurrently to the classifier, querier, and dictionary parameters. Fig. 11 shows, that joint optimization performs worse than phased optimization. In particular, for CIFAR10, we achieve 95% accuracy after just four queries using phased optimization compared to 83% with joint optimization. For CIFAR100, the accuracy gap is even more prominent. For joint optimization on CIFAR10, we train for 200 random sampling epochs and 200 biased sampling epochs until convergence. For joint optimization on CIFAR100, we train for 300 random sampling epochs and 300 biased sampling epochs until convergence.

B.7 Hard Query Answers are Necessary for Interpretability

In Section 4.1.1, we argued geometrically why using learned hyperplane queries with soft answers compromises interpretability. To validate this experimentally, we run Algorithm 1 with soft query answers on CIFAR10. In particular, we use the learnable query dictionary

$$\mathcal{Q} = \left\{ q^{(i)} = \left\langle \frac{v_i}{\|v_i\|}, E_I(\cdot) \right\rangle \right\}_{i=1}^{n}, \tag{31}$$

where $\{v_i\}_{i=1}^n \subset \mathbb{R}^d$ are trainable parameters. As illustrated in Fig. 6, Algorithm 1 with soft query answers reaches over 50% test accuracy after observing a single query-answer pair on CIFAR10 compared to 25% with the baseline GPT-3 Soft CLIP Embedding Dictionary. Considering CIFAR10's class labels (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks), it is implausible that a single query corresponding to a semantic concept reveals over 50% of the class labels. Therefore, the query must correspond to multiple concepts that are difficult to disentangle or not be semantic.

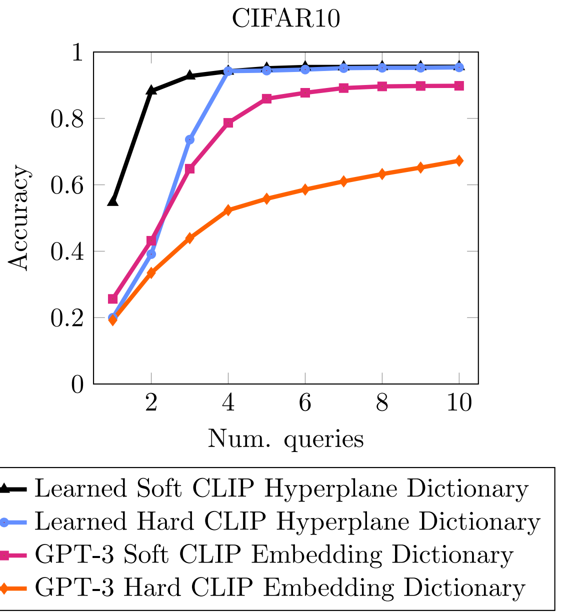


Figure 6: Comparing the test accuracy of the V-IP classifier, post-observation of k queries selected by the V-IP querier from (▲) Learned Soft CLIP Hyperplane Dictionary, (●) Learned Hard CLIP Hyperplane Dictionary, (■) GPT-3 Soft CLIP Embedding Dictionary, and (♦) GPT-3 CLIP Hard Embedding dictionary. The Learned Soft CLIP Hyperplane Dictionary achieves over 50% test accuracy after observing a single query. It is implausible that a single query corresponding to a semantic concept reveals over 50% of the CIFAR10 class labels. Hence, soft query answers for learned query dictionaries compromise interpretability.

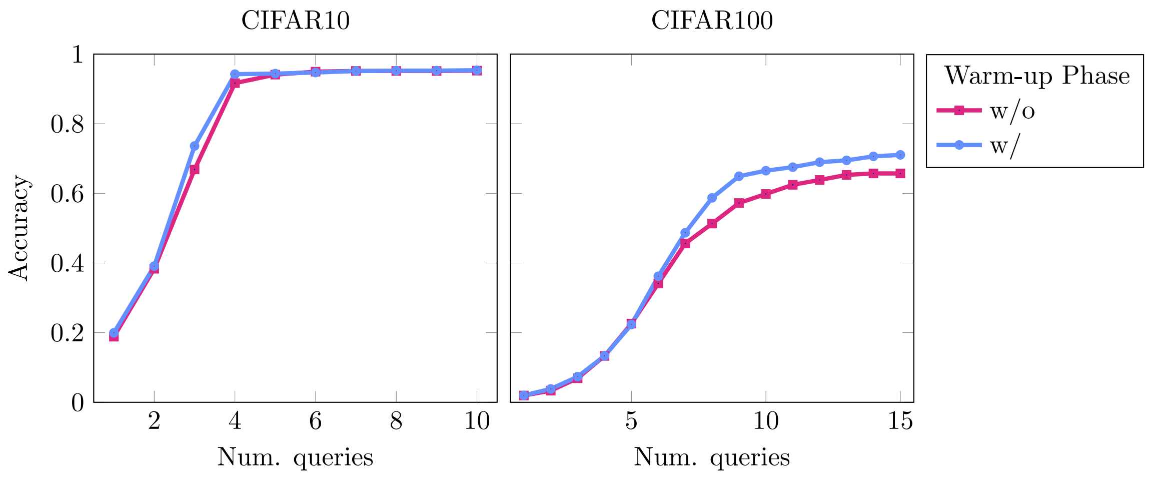


Figure 7: We plot test accuracy of V-IP classifier after observing k query-answer pairs selected by the querier from a learned dictionary (●) with warm-up phase and (■) without warm-up phase.

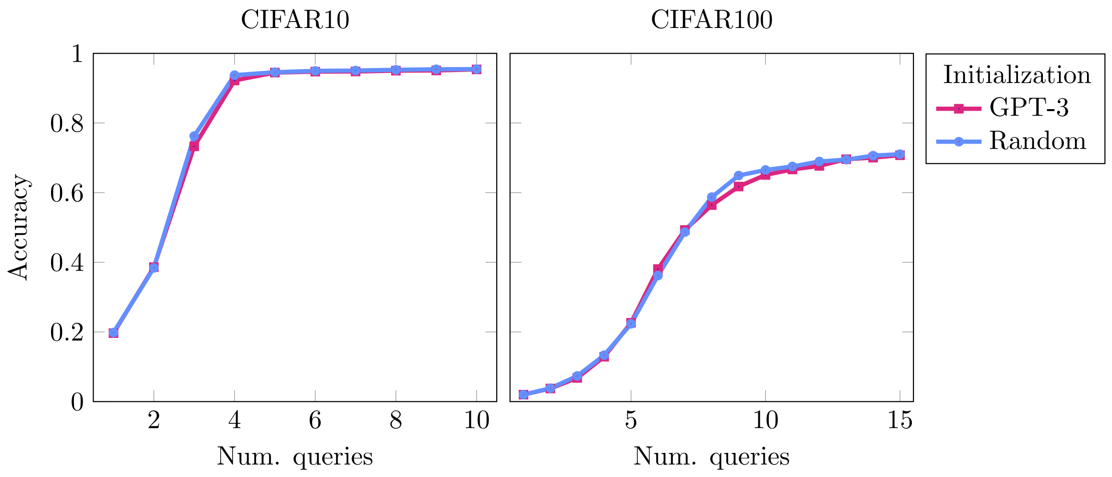


Figure 8: We plot test accuracy of V-IP classifier after observing k query-answer pairs selected by the querier from a learned dictionary with (●) random initialization and (■) GPT-3 initialization.

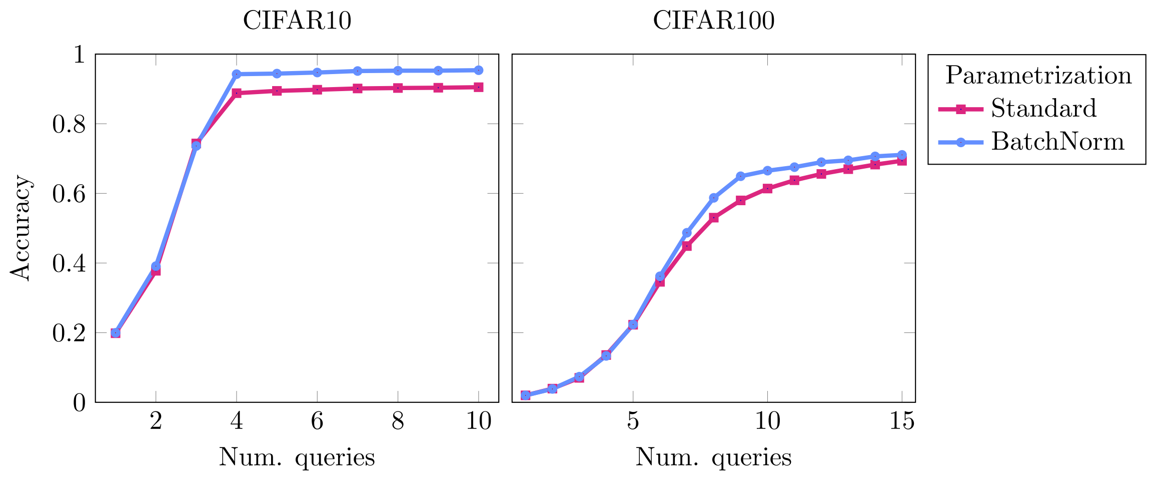


Figure 9: We plot test accuracy of V-IP classifier after observing k query-answer pairs selected by the querier from a learned dictionary (●) with BatchNorm parameterization and (■) standard parameterization.

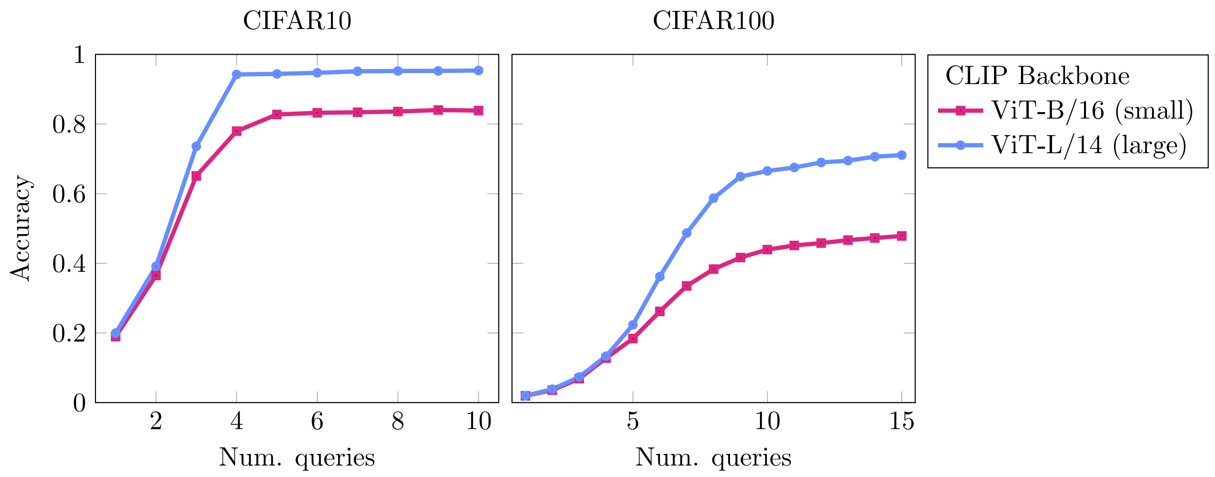


Figure 10: We plot test accuracy of V-IP classifier after observing k query-answer pairs selected by the querier from a learned dictionary with CLIP backbone (●) ViT-L/14 (large) and (■) ViT-B/16 (small).

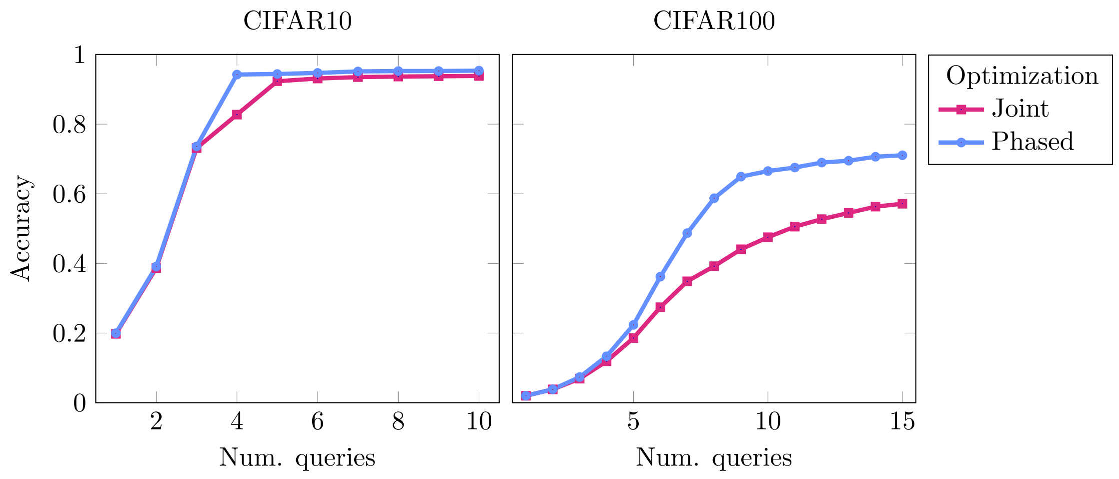


Figure 11: We plot test accuracy of V-IP classifier after observing k query-answer pairs selected by the querier from a learned dictionary optimized with (●) phased optimization and (■) joint optimization.