Learning to Maximize Mutual Information for Dynamic Feature Selection

Ian Covert [1] **Wei Qiu** [1] **Mingyu Lu** [1] **Nayoon Kim** [1] **Nathan White** [2] **Su-In Lee** [1]

Abstract

Feature selection helps reduce data acquisition costs in ML, but the standard approach is to train models with static feature subsets. Here, we consider the *dynamic feature selection* (DFS) problem where a model sequentially queries features based on the presently available information. DFS is often addressed with reinforcement learning, but we explore a simpler approach of greedily selecting features based on their conditional mutual information. This method is theoretically appealing but requires oracle access to the data distribution, so we develop a learning approach based on amortized optimization. The proposed method is shown to recover the greedy policy when trained to optimality, and it outperforms numerous existing feature selection methods in our experiments, thus validating it as a simple but powerful approach for this problem.

1. Introduction

A machine learning model's inputs can be costly to obtain, and feature selection is often used to reduce data acquisition costs. In applications where information is gathered sequentially, a natural option is to select features adaptively based on the currently available information, rather than using a fixed feature set. This setup is known as *dynamic feature selection* (DFS),[1] and the problem has been considered by several works in the last decade (Saar-Tsechansky et al., 2009; Dulac-Arnold et al., 2011; Chen et al., 2015b; Early et al., 2016a; He et al., 2016a; Kachuee et al., 2018).

Compared to *static* feature selection with a fixed feature set (Li et al., 2017; Cai et al., 2018), DFS can offer better performance given a fixed budget. This is easy to see, because selecting the same features for all instances (e.g., all patients visiting a hospital's emergency room) is suboptimal when the most informative features vary across individuals. Although it should in theory offer better performance, DFS also presents a more challenging learning problem, because it requires learning both (i) a feature selection policy and (ii) how to make predictions given variable feature sets.

Prior work has approached DFS in several ways, though often using reinforcement learning (RL) (Dulac-Arnold et al., 2011; Shim et al., 2018; Kachuee et al., 2018; Janisch et al., 2019; Li & Oliva, 2021). RL is a natural approach for sequential decision-making problems, but current methods are difficult to train and do not reliably outperform static feature selection methods (Henderson et al., 2018; Erion et al., 2021). Our work therefore explores a simpler approach: greedily selecting features based on their conditional mutual information (CMI) with the response variable.

The greedy approach is known from prior work (Chen et al., 2015b; Ma et al., 2019), but it is difficult to use in practice because calculating the CMI requires oracle access to the data distribution (Cover & Thomas, 2012). Our focus is therefore developing a practical approximation. Whereas previous work makes strong assumptions about the data (Geman & Jedynak, 1996) or approximates the data distribution with generative models (Ma et al., 2019), we develop a flexible approach that directly predicts the optimal selection at each step. Our method is based on a variational perspective on the greedy CMI policy, and it uses a technique known as *amortized optimization* (Amos, 2022) to enable training using only a standard labeled dataset. Notably, the model is trained with an objective function that recovers the greedy policy when it is trained to optimality.

Our contributions in this work are the following:

1. We derive a variational, or optimization-based perspective on the greedy CMI policy, which shows it to be equivalent to minimizing the one-step-ahead prediction loss given an optimal classifier.

2. We develop a learning approach based on amortized optimization, where a policy network is trained to directly predict the optimal selection at each step. Rather than requiring a dataset that indicates the correct selections, our

[1]Paul G. Allen School of Computer Science & Engineering, University of Washington [2]Department of Emergency Medicine, University of Washington. Correspondence to: Ian Covert <icovert@cs.uw.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]The problem has also been referred to as *sequential feature selection*, *active feature acquisition*, and *information pursuit*.

training approach is based on a standard labeled dataset and an objective function whose global optimizer is the greedy CMI policy.

3. We propose a continuous relaxation for the inherently discrete learning objective, which enables efficient and architecture-agnostic gradient-based optimization.

Our experiments evaluate the proposed method on numerous datasets, and the results show that it outperforms many recent dynamic and static feature selection methods. Overall, our work shows that when learned properly, the greedy CMI policy is a simple and powerful approach for DFS.

2. Problem formulation

In this section, we describe the DFS problem and introduce notation used throughout the paper.

2.1. Notation

Let $\mathbf{x}$ denote a vector of input features and $\mathbf{y}$ a response variable for a supervised learning task. The input consists of d distinct features, or $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_d)$. We use the notation $s \subseteq [d] \equiv \{1, \ldots, d\}$ to denote a subset of indices and $\mathbf{x}_s = \{\mathbf{x}_i : i \in s\}$ a subset of features. Bold symbols $\mathbf{x}, \mathbf{y}$ represent random variables, the symbols x, y are possible values, and $p(\mathbf{x}, \mathbf{y})$ denotes the data distribution.

Our goal is to design a policy that controls which features are selected given the currently available information. The selection policy can be viewed as a function $\pi(x_s) \in [d]$, meaning that it receives a subset of features as its input and outputs the next feature index to query. The policy is accompanied by a predictor $f(x_s)$ that can make predictions given the set of available features; for example, if $\mathbf{y}$ is discrete then predictions lie in the probability simplex, or $f(x_s) \in \Delta^{K-1}$ for K classes. The notation $f(x_s \cup x_i)$ represents the prediction given the combined features. We initially consider policy and predictor functions that operate on feature subsets, and Section 4 proposes an implementation using a mask variable $m \in [0, 1]^d$ where the functions operate on $x \odot m$.

2.2. Dynamic feature selection

The goal of DFS is to select features with minimal budget that achieve maximum predictive accuracy. Having access to more features generally makes prediction easier, so the challenge is selecting a small number of informative features. There are multiple formulations for this problem, including versions with non-uniform feature costs and different budgets for each sample (Kachuee et al., 2018), but we focus on the setting with a fixed budget and uniform costs. Our goal is to handle predictions at inference time by beginning with no features, sequentially selecting features x_s such that $|s| = k$ for a fixed budget $k < d$, and finally making accurate predictions for the response variable y.

Given a loss function that measures the discrepancy between predictions and labels $\ell(\hat{y}, y)$, a natural scoring criterion is the expected loss after selecting k features. The scoring is applied to a policy-predictor pair (π, f), and we define the score for a fixed budget k as follows,

$$v_k(\pi, f) = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\big[\ell\big(f\big(\{\mathbf{x}_{i_t}\}_{t=1}^{k}\big), \mathbf{y}\big)\big], \tag{1}$$

where feature indices are chosen sequentially for each $(\mathbf{x}, \mathbf{y})$ according to $i_n = \pi(\{\mathbf{x}_{i_t}\}_{t=1}^{n-1})$. The goal is to minimize $v_k(\pi, f)$, or equivalently, to maximize our final predictive accuracy.

One approach is to frame this as a Markov decision process (MDP) and solve it using standard RL techniques, so that π and f are trained to optimize a reward function based on eq. (1). Several recent works have designed such formulations (Shim et al., 2018; Kachuee et al., 2018; Janisch et al., 2019; Li & Oliva, 2021). However, these approaches are difficult to train effectively, so our work focuses on a greedy approach that is easier to learn and simpler to interpret.

3. Greedy information maximization

This section first defines the greedy CMI policy, and then describes an existing approximation strategy that relies on generative models.

3.1. The greedy selection policy

As an idealized approach to DFS, we are interested in the greedy algorithm that selects the most informative feature at each step. This feature can be defined in multiple ways, but we focus on the information-theoretic perspective that the most useful feature has maximum CMI with the response variable (Cover & Thomas, 2012). The CMI, denoted as $I(\mathbf{x}_i; \mathbf{y} \mid x_s)$, quantifies how much information an unknown feature $\mathbf{x}_i$ provides about the response $\mathbf{y}$ when accounting for the current features x_s, and it is defined as the KL divergence between the joint and factorized distributions:

$$I(\mathbf{x}_i; \mathbf{y} \mid x_s) = D_{\mathrm{KL}}\big(p(\mathbf{x}_i, \mathbf{y} \mid x_s) \,||\, p(\mathbf{x}_i \mid x_s)p(\mathbf{y} \mid x_s)\big).$$

Based on this, we define the greedy CMI policy as $\pi^*(x_s) \equiv \arg\max_i I(\mathbf{x}_i; \mathbf{y} \mid x_s)$, so that features are sequentially selected to maximize our information about the response variable. We can alternatively understand the policy as performing greedy uncertainty minimization, because this is equivalent to minimizing $\mathbf{y}$'s conditional entropy at each step, or $\pi^*(x_s) = \arg\min_i H(\mathbf{y} \mid \mathbf{x}_i, x_s)$ (Cover & Thomas, 2012). For a complete characterization of this idealized approach, we also consider that the policy is paired with the Bayes classifier as a predictor, or $f^*(x_s) = p(\mathbf{y} \mid x_s)$.

Maximizing the information about $\mathbf{y}$ at each step is intuitive and should be effective in many problems. Prior work has considered the same idea, but from two perspectives that differ from ours. First, Chen et al. (2015b) take a theoretical perspective and prove that the greedy algorithm achieves performance within a multiplicative factor of the optimal policy; the proof requires specific distributional assumptions, but we find that the greedy algorithm performs well with many real-world datasets (Section 6). Second, from an implementation perspective, two works aim to provide practical approximations; however, these suffer from several limitations, so our work aims to develop a simple and flexible alternative (Section 4). In these works, Ma et al. (2019) and Chattopadhyay et al. (2022) both require a conditional generative model of the data distribution, which we discuss next.

3.2. Estimating conditional mutual information

The greedy policy is trivial to implement if we can directly calculate CMI, but this is rarely the case in practice. Instead, one option is to estimate it. We now describe a procedure to do so iteratively for each feature, assuming for now that we have oracle access to the response distributions $p(\mathbf{y} \mid \mathbf{x}_s)$ for all $s \subseteq [d]$ and the feature distributions $p(\mathbf{x}_i \mid \mathbf{x}_s)$ for all $s \subseteq [d]$ and $i \in [d]$.

At any point in the selection procedure, given the current features x_s, we can estimate the CMI for a feature $\mathbf{x}_i$ where $i \notin s$ as follows. First, we can sample multiple values for $\mathbf{x}_i$ from its conditional distribution, or $x_i^j \sim p(\mathbf{x}_i \mid x_s)$ for $j \in [n]$. Next, we can generate Bayes optimal predictions for each sampled value, or $p(\mathbf{y} \mid x_s, x_i^j)$. Finally, we can calculate the mean prediction and the mean KL divergence relative to this prediction, which yields the following CMI estimator:

$$I_i^n = \frac{1}{n}\sum_{j=1}^{n} D_{\mathrm{KL}}\Big(p(\mathbf{y} \mid x_s, x_i^j) \;||\; \frac{1}{n}\sum_{l=1}^{n} p(\mathbf{y} \mid x_s, x_i^l)\Big). \tag{2}$$

This score measures the variability among predictions and captures whether different $\mathbf{x}_i$ values significantly affect $\mathbf{y}$'s conditional distribution. The estimator can be used to select features, or we can set $\pi(x_s) = \arg\max_i I_i^n$, due to the following limiting result (see Appendix A):

$$\lim_{n\to\infty} I_i^n = I(\mathbf{y}; \mathbf{x}_i \mid x_s). \tag{3}$$

This procedure thus provides a way to identify the correct greedy selections by estimating the CMI. Prior work has explored similar ideas for scoring features based on sampled predictions (Saar-Tsechansky et al., 2009; Chen et al., 2015a; Early et al., 2016a;b), but the implementation choices in these works prevent them from performing greedy information maximization. In eq. (2), is it important that our estimator uses the Bayes classifier, that we sample features from the conditional distribution $p(\mathbf{x}_i \mid x_s)$, and that we use the KL divergence as a measure of prediction variability. However, this estimator is impractical because we typically lack access to both $p(\mathbf{y} \mid \mathbf{x}_s)$ and $p(\mathbf{x}_i \mid \mathbf{x}_s)$.

In practice, we would instead require learned substitutes for each distribution. For example, we can use a a classifier that approximates $f(x_s) \approx p(\mathbf{y} \mid x_s)$ and a generative model that approximates samples from $p(\mathbf{x}_i \mid \mathbf{x}_s)$. Similarly, Ma et al. (2019) propose jointly modeling $(\mathbf{x}, \mathbf{y})$ with a conditional generative model, which is implemented via a modified VAE (Kingma et al., 2015). This approach is limited for several reasons, including (i) the difficulty of training an accurate conditional generative model, (ii) the challenge of modeling mixed continuous/categorical features (Ma et al., 2020; Nazabal et al., 2020), and (iii) the slow CMI estimation process. In our approach, which we discuss next, we bypass all three of these challenges by directly predicting the best selection at each step.

4. Proposed method

We now introduce our approach, a practical approximation of the greedy policy trained using amortized optimization. Unlike prior work that estimates the CMI as an intermediate step, we develop a variational perspective on the greedy policy, which we then leverage to train a network that directly predicts the optimal selection given the current features.

4.1. A variational perspective on CMI

For our purpose, it is helpful to recognize that the greedy policy can be viewed as the solution to an optimization problem. Section 3 provides a conventional definition of CMI as a KL divergence, but this is difficult to integrate into an end-to-end learning approach. Instead, we now consider the one-step-ahead prediction achieved by a policy π and predictor f, and we determine the behavior that minimizes their loss. Given the current features x_s and a selection $i = \pi(x_s)$, the expected one-step-ahead loss is:

$$\mathbb{E}_{\mathbf{y}, \mathbf{x}_i \mid x_s}\Big[\ell\big(f(x_s \cup \mathbf{x}_i), \mathbf{y}\big)\Big]. \tag{4}$$

The variational perspective we develop here consists of two main results regarding this expected loss. The first result relates to the predictor, and we show that the loss-minimizing predictor can be defined independently of the policy π. We formalize this in the following proposition for classification tasks, and our results can also be generalized to regression tasks (see proofs in Appendix A).

Proposition 1. *When $\mathbf{y}$ is discrete and ℓ is cross-entropy loss, eq.* (4) *is minimized for any policy π by the Bayes classifier, or $f^*(x_s) = p(\mathbf{y} \mid x_s)$.*

This property requires that features are selected without knowledge of the remaining features or the response variable, which is a valid assumption for DFS, but not in scenarios where selections are based on the full feature set (Chen et al., 2018; Yoon et al., 2018; Jethani et al., 2021). Now, assuming that we use the Bayes classifier f^* as a predictor, our second result concerns the selection policy. As we show next, the loss-minimizing policy is equivalent to making selections based on CMI.

Proposition 2. *When* $\mathbf{y}$ *is discrete,* ℓ *is cross-entropy loss and the predictor is the Bayes classifier* f^**, eq.* (4) *is minimized by the greedy CMI policy, or* $\pi^*(x_s) = \arg\max_i \; I(\mathbf{y}; \mathbf{x}_i \mid x_s)$.

With this, we can see that the greedy CMI policy defined in Section 3 is equivalent to minimizing the one-step-ahead prediction loss. Next, we exploit this variational perspective to develop a joint learning procedure for a policy and predictor network.

4.2. An amortized optimization approach

Instead of estimating each feature's CMI to identify the next selection, we now develop an approach that directly predicts the best selection at each step. The greedy policy implicitly requires solving an optimization problem for each selection, or $\arg\max_i I(\mathbf{y}, \mathbf{x}_i; x_s)$, but since we lack access to this objective, we now formulate an approach that directly predicts the solution. Following a technique known as amortized optimization (Amos, 2022), we do so by casting our variational perspective on CMI from Section 4.1 as an objective function to be optimized by a learnable network.

First, because it facilitates gradient-based optimization, we now consider that the policy outputs a *distribution* over feature indices. With slight abuse of notation, this section lets the policy be a function $\pi(x_s) \in \Delta^{d-1}$, which generalizes the previous definition $\pi(x_s) \in [d]$. Using this stochastic version of the policy, we can now formulate our objective function as follows.

Let the selection policy be parameterized by a neural network $\pi(\mathbf{x}_s; \phi)$ and the predictor by a neural network $f(\mathbf{x}_s; \theta)$. Let $p(\mathbf{s})$ represent a distribution with support over all subsets, or $p(s) > 0$ for all $|s| < d$. Then, our objective function $\mathcal{L}(\theta, \phi)$ is defined as

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{p(\mathbf{x},\mathbf{y})} \mathbb{E}_{p(\mathbf{s})} \Big[\mathbb{E}_{i \sim \pi(\mathbf{x}_\mathbf{s}; \phi)} \big[\ell\big(f(\mathbf{x}_\mathbf{s} \cup \mathbf{x}_i; \theta), \mathbf{y}\big) \big] \Big]. \tag{5}$$

Intuitively, eq. (5) represents generating a random feature set $\mathbf{x}_s$, sampling a feature index according to $i \sim \pi(\mathbf{x}_s; \phi)$, and then measuring the loss of the prediction $f(\mathbf{x}_s \cup \mathbf{x}_i; \theta)$. Our objective thus optimizes for individual selections and predictions rather than the entire trajectory, which lets us build on Proposition 1-2. We describe this as an implementation of the greedy approach because it recovers the greedy CMI selections when it is trained to optimality. In the classification case, we show the following result under a mild assumption that there is a unique optimal selection.

Theorem 1. *When* $\mathbf{y}$ *is discrete and* ℓ *is cross-entropy loss, the global optimum of eq.* (5) *is a predictor that satisfies* $f(x_s; \theta^*) = p(\mathbf{y} \mid x_s)$ *and a policy* $\pi(x_s; \phi^*)$ *that puts all probability mass on* $i^* = \arg\max_i I(\mathbf{y}; \mathbf{x}_i \mid x_s)$.

If we relax the assumption of a unique optimal selection, the optimal policy $\pi(\mathbf{x}_s; \phi^*)$ simply splits probability mass among the best indices. A similar result holds in the regression case, where we can interpret the greedy policy as performing conditional variance minimization.

Theorem 2. *When* $\mathbf{y}$ *is continuous and* ℓ *is squared error loss, the global optimum of eq.* (5) *is a predictor that satisfies* $f(x_s; \theta^*) = \mathbb{E}[\mathbf{y} \mid x_s]$ *and a policy* $\pi(x_s; \phi^*)$ *that puts all probability mass on* $i^* = \arg\min_i \mathbb{E}_{\mathbf{x}_i | x_s}[\mathrm{Var}(\mathbf{y} \mid \mathbf{x}_i, x_s)]$.

Proofs for these results are in Appendix A. We note that the function class for each model must be expressive enough to contain their respective optima, and that the result holds for any $p(\mathbf{s})$ with support over all subsets.

This approach has two key advantages over the CMI estimation procedure from Section 3.2. First, we avoid modeling the feature conditional distributions $p(\mathbf{x}_i \mid \mathbf{x}_s)$ for all (s, i). Modeling these distributions is a difficult intermediate step, and our approach instead aims to directly output the optimal index. Second, our approach is faster because each selection is made in a single forward pass: selecting k features using the procedure from Ma et al. (2019) requires $\mathcal{O}(dk)$ scoring steps, but our approach requires only k forward passes through the policy network $\pi(\mathbf{x}_s; \phi)$.

Furthermore, compared to a policy trained by RL, the greedy approach is easier to learn. Our training procedure can be viewed as a form of reward shaping (Sutton et al., 1998; Randløv & Alstrøm, 1998), where the reward accounts for the loss after each step and provides a strong signal about whether each selection is helpful. In comparison, observing the reward only after selecting k features provides a comparably weak signal to the policy network (see eq. (1)). RL methods generally face a challenging exploration-exploitation trade-off, but learning the greedy policy is simpler because it only requires finding the locally optimal choice at each step.

4.3. Training with a continuous relaxation

Our objective in eq. (5) yields the correct greedy policy when it is perfectly optimized, but $\mathcal{L}(\theta, \phi)$ is difficult to optimize by gradient descent. In particular, gradients are difficult to propagate through the policy network given a sam-

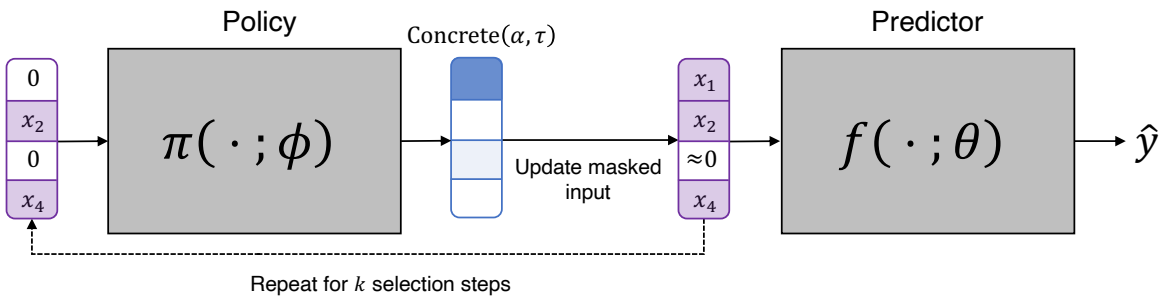

Figure 1. Diagram of our training approach. Left: features are selected by making repeated calls to the policy network using masked inputs. Right: predictions are made after each selection using the predictor network. Only solid lines are backpropagated through when performing gradient descent.

pled index $i \sim \pi(\mathbf{x}_s; \phi)$. The REINFORCE trick (Williams, 1992) is one way to get stochastic gradients, but high gradient variance can make it ineffective in many problems. There is a robust literature on reducing gradient variance in this setting (Tucker et al., 2017; Grathwohl et al., 2018), but we propose using a simple alternative: the Concrete distribution (Maddison et al., 2016).

An index sampled according to $i \sim \pi(x_s; \phi)$ can be represented by a one-hot vector $m \in \{0, 1\}^d$ indicating the chosen index, and with the Concrete distribution we instead sample an *approximately* one-hot vector in the probability simplex, or $m \in \Delta^{d-1}$. This continuous relaxation lets us calculate gradients using the reparameterization trick (Maddison et al., 2016; Jang et al., 2016). Relaxing the subset $s \subseteq [d]$ to a continuous vector also requires relaxing the policy and predictor functions, so we let these operate on a masked input x, or the element-wise product $x \odot m$. To avoid ambiguity about whether features are zero or masked, we can also pass the mask as a model input.

Training with the Concrete distribution requires specifying a temperature parameter $\tau > 0$ to control how discrete the samples are. Previous works have typically trained with a fixed temperature or annealed it over a pre-determined number of epochs (Chang et al., 2017; Chen et al., 2018; Balın et al., 2019), but we instead train with a sequence of τ values and perform early stopping at each step. This removes the temperature and number of epochs as important hyperparameters to tune. Our training procedure is summarized in Figure 1, and in more detail by Algorithm 1.

There are also several optional steps that we found can improve optimization:

- Parameters can be shared between the predictor and policy networks $f(\mathbf{x}; \theta), \pi(\mathbf{x}, \phi)$. This does not complicate their joint optimization, and learning a shared representation in the early layers can in some cases help the networks optimize faster (e.g., for image data).
- Rather than training with a random subset distribution $p(\mathbf{s})$, we generate subsets using features selected by the current policy $\pi(\mathbf{x}; \phi)$. This allows the models to focus on subsets likely to be encountered at inference time, and it does not affect the globally optimal policy/predictor: gradients are not propagated between selections, so both eq. (5) and this sampling approach treat each feature set as an independent optimization problem, only with different weights (see Appendix D).
- We pre-train the predictor $f(\mathbf{x}; \theta)$ using random subsets before jointly training the policy-predictor pair. This works better than optimizing $\mathcal{L}(\theta, \phi)$ from a random initialization, because a random predictor $f(\mathbf{x}; \theta)$ provides no signal to $\pi(\mathbf{x}; \phi)$ about which features are useful.

5. Related work

Prior work has frequently addressed DFS using RL. For example, Dulac-Arnold et al. (2011); Shim et al. (2018); Janisch et al. (2019); Li & Oliva (2021) optimize a reward based on the final prediction accuracy, and Kachuee et al. (2018) use a reward that accounts for prediction uncertainty. RL is a natural approach for sequential decision-making problems, but it can be difficult to optimize in practice: RL requires complex training routines, is slow to converge, and is highly sensitive to its initialization (Henderson et al., 2018). As a result, RL-based DFS does not reliably outperform static feature selection, as shown by Erion et al. (2021) and confirmed in our experiments.

Several other approaches include imitation learning (He et al., 2012; 2016a) and iterative feature scoring methods (Melville et al., 2004; Saar-Tsechansky et al., 2009; Chen et al., 2015a; Early et al., 2016b;a). Imitation learning casts DFS as supervised classification, whereas our training approach bypasses the need for an oracle policy. Most existing feature scoring techniques are greedy methods, like ours, but they use scoring heuristics that are unrelated to maximizing CMI (see Section 3.2). Two feature scoring methods are specifically designed to calculate the CMI, but they suffer from important practical limitations: both Ma et al. (2019)

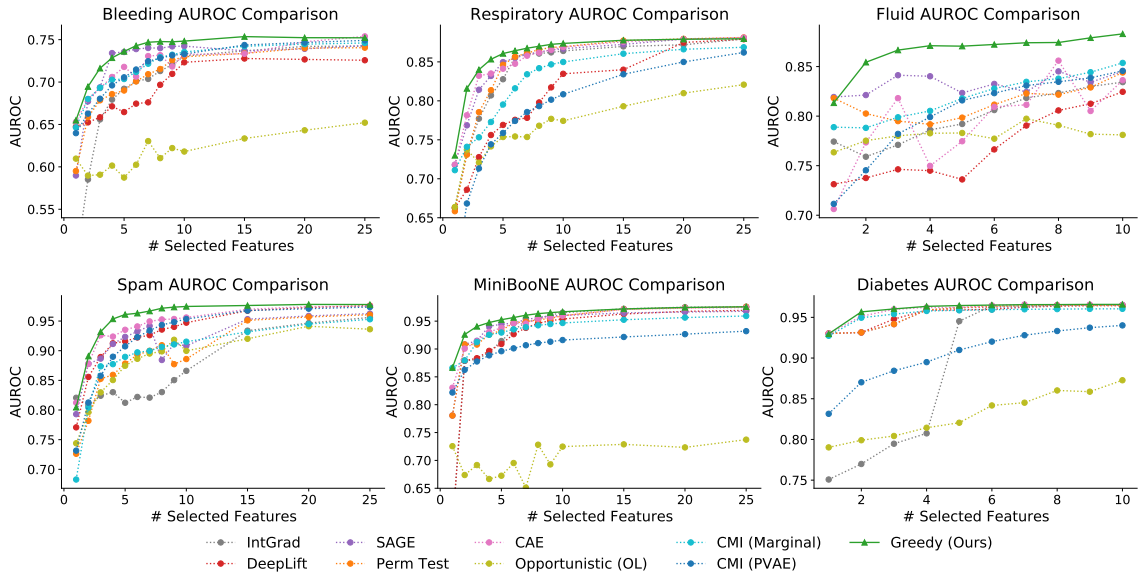

Figure 2. Evaluating the greedy approach on six tabular datasets. The results for each method are the average across five runs.

and Chattopadhyay et al. (2022) rely on difficult-to-train generative models, which can lead to inaccurate CMI estimation. Our approach is simpler, faster and more flexible, because the selection logic is contained within a policy network that avoids the need for generative modeling.[2]

Static feature selection is a long-standing problem (Guyon & Elisseeff, 2003; Cai et al., 2018). There are no default approaches for neural networks, but one option is ranking features by local or global importance scores (Breiman, 2001; Shrikumar et al., 2017; Sundararajan et al., 2017; Covert et al., 2020). In addition, several prior works have leveraged continuous relaxations to learn feature selection strategies by gradient descent: for example, Chang et al. (2017); Balın et al. (2019); Yamada et al. (2020); Lee et al. (2021); Covert et al. (2022) perform static feature selection, and Chen et al. (2018); Jethani et al. (2021) perform instance-wise feature selection given access to all the features. Our work uses a similar continuous relaxation for optimization, but in the DFS context, where our method learns a selection policy rather than a static selection layer.

Finally, several works have examined greedy feature selection algorithms from a theoretical perspective. For example, Das & Kempe (2011); Elenberg et al. (2018) show that weak submodularity implies near-optimal performance for static feature selection. More relevant to our work, Chen et al. (2015b) find that the related notion of adaptive submodularity (Golovin & Krause, 2011) does not not hold in the DFS setting, but the authors provide performance guarantees under specific distributional assumptions.

[2]Concurrently, Chattopadhyay et al. (2023) proposed a similar approach to predict the optimal selection at each step.

6. Experiments

We now demonstrate the use of our greedy approach on several datasets. We first explore tabular datasets of various sizes, including four medical diagnosis tasks, and we then consider two image classification datasets. Several of the tasks are natural candidates for DFS, and the remaining ones serve as useful tasks to test the effectiveness of our approach. Code for reproducing our experiments is available online: `https://github.com/iancovert/dynamic-selection`.

We evaluate our method by comparing to both dynamic and static feature selection methods. We also ensure consistent comparisons by only using methods applicable to neural networks. As static baselines, we use permutation tests (Breiman, 2001) and SAGE (Covert et al., 2020) to rank features by their importance to the model's accuracy, as well as per-prediction DeepLift (Shrikumar et al., 2017) and IntGrad (Sundararajan et al., 2017) scores aggregated across the dataset. We then use a supervised version of the Concrete Autoencoder (CAE, Balın et al. 2019), a state-of-the-art static feature selection method. As dynamic baselines, we use two versions of the CMI estimation procedure described in Section 3.2. First, we use the PVAE generative model from Ma et al. (2019) to sample unknown features, and second, we instead sample unknown features from their marginal distribution; in both cases, we use a classifier trained with random feature subsets to make predictions. Finally, we also use the RL-based Opportunistic Learning (OL) approach (Kachuee et al., 2018). Appendix C provides more information about the baseline methods.

Table 1. AUROC averaged across budgets of 1-10 features (with 95% confidence intervals).

		Spam	MiniBooNE	Diabetes	Bleeding	Respiratory	Fluid
Static	IntGrad	82.84 ± 0.68	89.10 ± 0.33	88.91 ± 0.24	66.70 ± 0.27	81.10 ± 0.04	79.94 ± 0.94
	DeepLift	90.16 ± 1.24	88.62 ± 0.30	95.42 ± 0.13	67.75 ± 0.49	76.05 ± 0.35	76.96 ± 0.56
	SAGE	89.70 ± 1.10	92.64 ± 0.03	95.43 ± 0.01	71.34 ± 0.19	82.92 ± 0.26	83.27 ± 0.53
	Perm Test	85.64 ± 3.58	92.19 ± 0.15	95.46 ± 0.02	68.89 ± 1.06	81.56 ± 0.28	81.35 ± 1.04
	CAE	92.28 ± 0.27	92.76 ± 0.41	95.91 ± 0.07	70.69 ± 0.57	83.10 ± 0.45	79.40 ± 0.86
Dynamic	Opportunistic (OL)	85.94 ± 0.20	69.23 ± 0.64	83.07 ± 0.82	60.63 ± 0.55	74.44 ± 0.42	78.13 ± 0.31
	CMI (Marginal)	86.57 ± 1.54	92.21 ± 0.40	95.48 ± 0.05	70.57 ± 0.46	79.62 ± 0.62	81.97 ± 0.93
	CMI (PVAE)	89.01 ± 1.40	88.94 ± 1.25	90.50 ± 5.16	70.17 ± 0.74	74.12 ± 3.50	80.27 ± 1.02
	Greedy (Ours)	**93.91 ± 0.17**	**94.46 ± 0.12**	**96.03 ± 0.02**	**72.64 ± 0.31**	**84.48 ± 0.08**	**86.59 ± 0.25**

6.1. Tabular datasets

We first applied our method to three medical diagnosis tasks derived from an emergency medicine setting. The tasks involve predicting a patient's bleeding risk via a low fibrinogen concentration (bleeding), whether the patient requires endotracheal intubation for respiratory support (respiratory), and whether the patient will be responsive to fluid resuscitation (fluid). See Appendix B for more details about the datasets. In each scenario, gathering all possible inputs at inference time is challenging due to time and resource constraints, thus making DFS a natural solution.

We use fully connected networks for all methods, and we use dropout to reduce overfitting (Srivastava et al., 2014). Figure 2 (top) shows the results of applying each method with various feature budgets. The classification accuracy is measured via AUROC, and the greedy method achieves the best results for nearly all feature budgets on all three tasks. Among the baselines, several static methods are sometimes close, but the CMI estimation method is rarely competitive (Ma et al., 2019). Additionally, OL provides unstable and weak results. The greedy method's advantage is often largest when selecting a small number of features, and it usually becomes narrower once the accuracy saturates.

Next, we conducted experiments using three publicly available tabular datasets: spam classification (Dua & Graff, 2017), particle identification (MiniBooNE) (Roe et al., 2005) and diabetes diagnosis (Miller, 1973). The diabetes task is a natural application for DFS and was used in prior work (Kachuee et al., 2018). We again tested various numbers of features, and Figure 2 (bottom) shows plots of the AUROC for each feature budget. On these tasks, the greedy method is once again most accurate for nearly all numbers of features. Table 1 summarizes the results via the mean AUROC across $k = 1, \ldots, 10$ features, further emphasizing the benefits of the greedy method across all six datasets. Appendix E shows larger versions of the AUROC curves (Figure 4 and Figure 5), as well as plots demonstrating the variability of selections within each dataset.

The results with these datasets reveal that, perhaps surprisingly, dynamic methods can be outperformed by static methods. Interestingly, this point was not highlighted in prior works where strong static baselines were not tested (Kachuee et al., 2018; Janisch et al., 2019). For example, OL is not competitive on these datasets, and the two versions of the CMI estimation approach are not consistently among the top baselines. Dynamic methods are in principle capable of performing better, so the sub-par results from these methods underscore the difficulty of learning both a selection policy and a prediction function that works for multiple feature sets. In these experiments, our approach is the only dynamic method to do both successfully.

6.2. Image classification datasets

Next, we considered two standard image classification datasets: MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky et al., 2009). Our goal is to begin with a blank image, sequentially reveal multiple pixels or patches, and ultimately make a classification using a small portion of the image. Although this is not an obvious use case for DFS, it represents a challenging problem for our method, and similar tasks were considered in several earlier works (Karayev et al., 2012; Mnih et al., 2014; Early et al., 2016a; Janisch et al., 2019).

For MNIST, we use fully connected architectures for both the policy and predictor, and we treat pixels as individual features; we therefore have $d = 784$. For CIFAR-10, we use a shared ResNet backbone (He et al., 2016b) for the policy and predictor networks, and each network uses its own output head. The 32×32 images are coarsened into $d = 64$ patches of size 4×4, so the selector head generates logits corresponding to each patch, and the predictor head generates probabilities for each class.

Figure 3 shows our method's accuracy for different feature budgets. For MNIST, we use the previous baselines but exclude the CMI estimation method due to its computational cost: it becomes slow when evaluating many candidate features. We observe a large benefit for our method, particularly when making a small number of selections. Our greedy method reaches nearly 90% accuracy with just 10 pixels, which is roughly 10% higher than the best baseline and con-

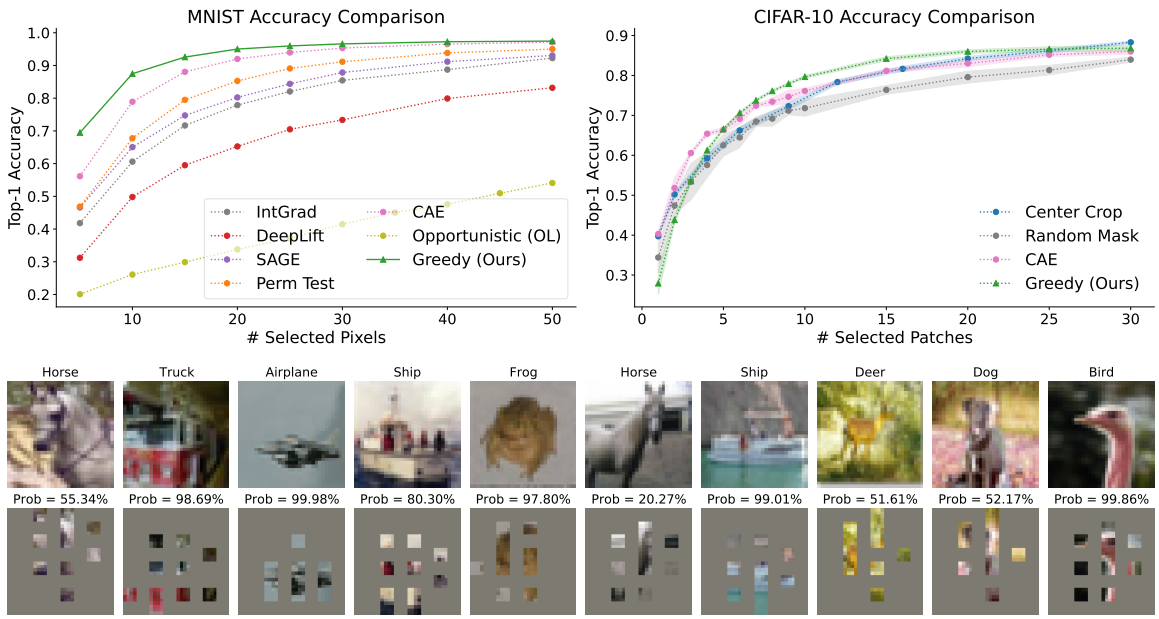

Figure 3. Greedy feature selection for image classification. Top left: accuracy comparison on MNIST with results averaged across five runs. Top right: accuracy comparison on CIFAR-10 with 95% confidence intervals. Bottom: example selections and predictions for the greedy method with 10 out of 64 patches for CIFAR-10 images.

siderably higher than prior work (Balın et al., 2019; Yamada et al., 2020; Covert et al., 2020). OL yields the worst results, and it also trains slowly due to the large number of states.

For CIFAR-10, we omit several baseline comparisons due to their computational cost. We use the CAE, which is our most competitive static baseline, as well as two simple baselines: center crops and random masks of various sizes. For each method, we plot the mean and 95% confidence intervals determined from five trials. Our greedy approach is slightly less accurate with a very small number of patches, but it reaches significantly higher accuracy when using 6-20 patches. Finally, Figure 3 (bottom) also shows qualitative examples of our method's predictions after selecting 10 out of 64 patches, and Appendix E shows similar plots with different numbers of patches.

7. Conclusion

In this work, we explored a greedy algorithm for dynamic feature selection (DFS) that selects features based on their CMI with the response variable. We proposed an approach to approximate this policy by directly predicting the optimal selection at each step, and we conducted experiments that show our method outperforms a variety of existing feature selection methods, including both dynamic and static baselines. Future work on this topic may include incorporating non-uniform features costs or determining the ideal feature budget on a per-sample basis; from a theoretical perspective, characterizing the greedy algorithm's performance outside of our fixed-budget case is another interesting topic for future work (Chen et al., 2015b). Finally, future work may also explore architectures that are well-suited to processing partial inputs, particularly for structured data like images.

Acknowledgements

We thank Samuel Ainsworth, Kevin Jamieson, Mukund Sudarshan and the Lee Lab for helpful discussions. This work was funded by NSF DBI-1552309 and DBI-1759487, NIH R35-GM-128638 and R01-NIA-AG-061132.

References

National health and nutrition examination survey, 2018. URL `https://www.cdc.gov/nchs/nhanes`.

Amos, B. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*, 2022.

Balın, M. F., Abid, A., and Zou, J. Concrete autoencoders: Differentiable feature selection and reconstruction. In *International Conference on Machine Learning*, pp. 444–453. PMLR, 2019.

Breiman, L. Random forests. *Machine Learning*, 45(1): 5–32, 2001.

Cai, J., Luo, J., Wang, S., and Yang, S. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018.

Chang, C.-H., Rampasek, L., and Goldenberg, A. Dropout feature ranking for deep learning models. *arXiv preprint arXiv:1712.08645*, 2017.

Chattopadhyay, A., Slocum, S., Haeffele, B. D., Vidal, R., and Geman, D. Interpretable by design: Learning predictors by composing interpretable queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Chattopadhyay, A., Chan, K. H. R., Haeffele, B. D., Geman, D., and Vidal, R. Variational information pursuit for interpretable predictions. *arXiv preprint arXiv:2302.02876*, 2023.

Chen, J., Song, L., Wainwright, M., and Jordan, M. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pp. 883–892. PMLR, 2018.

Chen, S., Choi, A., and Darwiche, A. Value of information based on decision robustness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015a.

Chen, Y., Hassani, S. H., Karbasi, A., and Krause, A. Sequential information maximization: When is greedy near-optimal? In *Conference on Learning Theory*, pp. 338–363. PMLR, 2015b.

Cover, T. and Thomas, J. *Elements of Information Theory*. Wiley, 2012. ISBN 9781118585771.

Covert, I., Lundberg, S. M., and Lee, S.-I. Understanding global feature contributions with additive importance measures. *Advances in Neural Information Processing Systems*, 33:17212–17223, 2020.

Covert, I., Lundberg, S. M., and Lee, S.-I. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22:209–1, 2021.

Covert, I., Gala, R., Wang, T., Svoboda, K., Lee, S.-I., et al. Predictive and robust gene selection for spatial transcriptomics. *bioRxiv*, 2022.

Das, A. and Kempe, D. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*, 2011.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Dulac-Arnold, G., Denoyer, L., Preux, P., and Gallinari, P. Datum-wise classification: a sequential approach to sparsity. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 375–390. Springer, 2011.

Early, K., Fienberg, S. E., and Mankoff, J. Test time feature ordering with FOCUS: Interactive predictions with minimal user burden. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 992–1003, 2016a.

Early, K., Mankoff, J., and Fienberg, S. E. Dynamic question ordering in online surveys. *arXiv preprint arXiv:1607.04209*, 2016b.

Elenberg, E. R., Khanna, R., Dimakis, A. G., and Negahban, S. Restricted strong convexity implies weak submodularity. *The Annals of Statistics*, 46(6B):3539–3568, 2018.

Erion, G., Janizek, J. D., Hudelson, C., Utarnachitt, R. B., McCoy, A. M., Sayre, M. R., White, N. J., and Lee, S.-I. CoAI: Cost-aware artificial intelligence for health care. *medRxiv*, 2021.

Feng, J. and Simon, N. Sparse-input neural networks for high-dimensional nonparametric regression and classification. *arXiv preprint arXiv:1711.07592*, 2017.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059. PMLR, 2016.

Geman, D. and Jedynak, B. An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):1–14, 1996.

Golovin, D. and Krause, A. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182, 2003.

He, H., Daumé III, H., and Eisner, J. Cost-sensitive dynamic feature selection. In *ICML Inferning Workshop*, 2012.

He, H., Mineiro, P., and Karampatziakis, N. Active information acquisition. *arXiv preprint arXiv:1602.02181*, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016b.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with Gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Janisch, J., Pevnỳ, T., and Lisỳ, V. Classification with costly features using deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3959–3966, 2019.

Jethani, N., Sudarshan, M., Aphinyanaphongs, Y., and Ranganath, R. Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations. In *International Conference on Artificial Intelligence and Statistics*, pp. 1459–1467. PMLR, 2021.

Kachuee, M., Goldstein, O., Kärkkäinen, K., Darabi, S., and Sarrafzadeh, M. Opportunistic learning: Budgeted cost-sensitive learning from data streams. In *International Conference on Learning Representations*, 2018.

Kachuee, M., Karkkainen, K., Goldstein, O., Zamanzadeh, D., and Sarrafzadeh, M. Cost-sensitive diagnosis and learning leveraging public health data. *arXiv preprint arXiv:1902.07102*, 2019.

Karayev, S., Baumgartner, T., Fritz, M., and Darrell, T. Timely object recognition. *Advances in Neural Information Processing Systems*, 25, 2012.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems*, 28, 2015.

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., et al. Captum: A unified and generic model interpretability library for PyTorch. *arXiv preprint arXiv:2009.07896*, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, C., Imrie, F., and van der Schaar, M. Self-supervision enhanced feature selection with correlated gates. In *International Conference on Learning Representations*, 2021.

Lemhadri, I., Ruan, F., and Tibshirani, R. Lassonet: Neural networks with feature sparsity. In *International Conference on Artificial Intelligence and Statistics*, pp. 10–18. PMLR, 2021.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45, 2017.

Li, Y. and Oliva, J. Active feature acquisition with generative surrogate models. In *International Conference on Machine Learning*, pp. 6450–6459. PMLR, 2021.

Ma, C., Tschiatschek, S., Palla, K., Hernandez-Lobato, J. M., Nowozin, S., and Zhang, C. EDDI: Efficient dynamic discovery of high-value information with partial VAE. In *International Conference on Machine Learning*, pp. 4234–4243. PMLR, 2019.

Ma, C., Tschiatschek, S., Turner, R., Hernández-Lobato, J. M., and Zhang, C. VAEM: a deep generative model for heterogeneous mixed type data. *Advances in Neural Information Processing Systems*, 33:11237–11247, 2020.

Maddison, C. J., Mnih, A., and Teh, Y. W. The Concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Melville, P., Saar-Tsechansky, M., Provost, F., and Mooney, R. Active feature-value acquisition for classifier induction. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pp. 483–486. IEEE, 2004.

Miller, H. W. Plan and operation of the health and nutrition examination survey, United States, 1971-1973. *DHEW publication no. (PHS)-Dept. of Health, Education, and Welfare (USA)*, 1973.

Mnih, V., Heess, N., Graves, A., et al. Recurrent models of visual attention. *Advances in Neural Information Processing Systems*, 27, 2014.

Mosesson, M. W. Fibrinogen and fibrin structure and functions. *Journal of Thrombosis and Haemostasis*, 3(8): 1894–1904, 2005.

Nazabal, A., Olmos, P. M., Ghahramani, Z., and Valera, I. Handling incomplete heterogeneous data using VAEs. *Pattern Recognition*, 107:107501, 2020.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. 2017.

Randløv, J. and Alstrøm, P. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, pp. 463–471. Citeseer, 1998.

Roe, B., Yand, H., Zhu, J., Lui, Y., Stancu, I., et al. Boosted decision trees, an alternative to artificial neural networks. *Nucl. Instrm. Meth. A*, 543:577–584, 2005.

Saar-Tsechansky, M., Melville, P., and Provost, F. Active feature-value acquisition. *Management Science*, 55(4): 664–684, 2009.

Shim, H., Hwang, S. J., and Yang, E. Joint active feature acquisition and classification with variable-size set encoding. *Advances in Neural Information Processing Systems*, 31, 2018.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pp. 3145–3153. PMLR, 2017.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Subcommittee, A., Group, I. A. W., et al. Advanced trauma life support (ATLS®): the ninth edition. *The Journal of Trauma and Acute Care Surgery*, 74(5):1363–1366, 2013.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

Sutton, R. S., Barto, A. G., et al. Introduction to reinforcement learning. 1998.

Tank, A., Covert, I., Foti, N., Shojaie, A., and Fox, E. B. Neural Granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4267–4279, 2021.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, J., and Sohl-Dickstein, J. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 30, 2017.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

Yamada, Y., Lindenbaum, O., Negahban, S., and Kluger, Y. Feature selection using stochastic gates. In *International Conference on Machine Learning*. PMLR, 2020.

Yoon, J., Jordon, J., and van der Schaar, M. INVASE: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2018.

A. Proofs

In this section, we re-state and prove our main theoretical results. We begin with our proposition regarding the optimal predictor for an arbitrary policy π.

Proposition 1. *When* $\mathbf{y}$ *is discrete and* ℓ *is cross-entropy loss, eq.* (4) *is minimized for any policy* π *by the Bayes classifier, or* $f^*(x_s) = p(\mathbf{y} \mid x_s)$.

Proof. Given the predictor inputs x_s, our goal is to determine the prediction that minimizes the expected loss. Because features are selected sequentially by π with no knowledge of the non-selected values, there is no other information to condition on; for the predictor, we do not even need to distinguish the order in which features were selected. We can therefore derive the optimal prediction $\hat{y} \in \Delta^{K-1}$ for a discrete response $\mathbf{y} \in [K]$ as follows:

$$\begin{aligned} f^*(x_s) &= \arg\min_{\hat{y}} \ \mathbb{E}_{\mathbf{y}|x_s}\big[\ell(\hat{y}, \mathbf{y})\big] \\ &= \arg\min_{\hat{y}} \ \sum_{i \in \mathcal{Y}} p(\mathbf{y} = i \mid x_s) \log \hat{y}_i \\ &= \arg\min_{\hat{y}} \ D_{\mathrm{KL}}\big(p(\mathbf{y} \mid x_s) \mid\mid \hat{y}\big) + H(\mathbf{y} \mid x_s) \\ &= p(\mathbf{y} \mid x_s). \end{aligned}$$

In the case of a continuous response $\mathbf{y} \in \mathbb{R}$ with squared error loss, we have a similar result involving the response's conditional expectation:

$$\begin{aligned} f^*(x_s) &= \arg\min_{\hat{y}} \ \mathbb{E}_{\mathbf{y}|x_s}\big[(\hat{y} - \mathbf{y})^2\big] \\ &= \arg\min_{\hat{y}} \ \mathbb{E}_{\mathbf{y}|x_s}\big[(\hat{y} - \mathbb{E}[\mathbf{y} \mid x_s])^2\big] + \mathrm{Var}(\mathbf{y} \mid x_s) \\ &= \mathbb{E}[\mathbf{y} \mid x_s]. \end{aligned}$$

□

Proposition 2. *When* $\mathbf{y}$ *is discrete,* ℓ *is cross-entropy loss and the predictor is the Bayes classifier* f^**, eq.* (4) *is minimized by the greedy CMI policy, or* $\pi^*(x_s) = \arg\max_i \ I(\mathbf{y}; \mathbf{x}_i \mid x_s)$.

Proof. Following eq. (4), the policy network's selection $i = \pi(x_s)$ incurs the following expected loss with respect to the distribution $p(\mathbf{y}, \mathbf{x}_i \mid x_s)$:

$$\begin{aligned} \mathbb{E}_{\mathbf{y},\mathbf{x}_i|x_s}\big[\ell(f^*(x_s \cup \mathbf{x}_i), \mathbf{y})\big] &= \mathbb{E}_{\mathbf{y},\mathbf{x}_i|x_s}\big[\ell(p(\mathbf{y} \mid \mathbf{x}_i, x_s), \mathbf{y})\big] \\ &= \mathbb{E}_{\mathbf{x}_i|x_s}\Big[\mathbb{E}_{\mathbf{y}|\mathbf{x}_i,x_s}[\ell(p(\mathbf{y} \mid \mathbf{x}_i, x_s), \mathbf{y})]\Big] \\ &= \mathbb{E}_{\mathbf{x}_i|x_s}\big[H(\mathbf{y} \mid \mathbf{x}_i, x_s)\big] \\ &= H(\mathbf{y} \mid x_s) - I(\mathbf{y}; \mathbf{x}_i \mid x_s). \end{aligned}$$

Note that $H(\mathbf{y} \mid x_s)$ is a constant that does not depend on i. When identifying the index that minimizes the expected loss, we therefore have the following result:

$$\arg\min_i \ \mathbb{E}_{\mathbf{y},\mathbf{x}_i|x_s}\big[\ell(f^*(x_s \cup \mathbf{x}_i), \mathbf{y})\big] = \arg\max_i \ I(\mathbf{y}; \mathbf{x}_i \mid x_s).$$

In the case of a continuous response with squared error loss and an optimal predictor given by $f^*(x_s) = \mathbb{E}[\mathbf{y} \mid x_s]$, we have a similar result:

$$\begin{aligned}
\mathbb{E}_{\mathbf{y},\mathbf{x}_i|x_s}\big[(f^*(x_s \cup \mathbf{x}_i) - \mathbf{y})^2\big] &= \mathbb{E}_{\mathbf{y},\mathbf{x}_i|x_s}\big[(\mathbb{E}[\mathbf{y} \mid \mathbf{x}_i, x_s] - \mathbf{y})^2\big] \\
&= \mathbb{E}_{\mathbf{x}_i|x_s}\Big[\mathbb{E}_{\mathbf{y}|\mathbf{x}_i,x_s}[(\mathbb{E}[\mathbf{y} \mid \mathbf{x}_i, x_s] - \mathbf{y})^2]\Big] \\
&= \mathbb{E}_{\mathbf{x}_i|x_s}[\mathrm{Var}(\mathbf{y} \mid \mathbf{x}_i, x_s)].
\end{aligned}$$

When we aim to minimize the expected loss, our selection is therefore the index that yields the lowest expected conditional variance:

$$\arg\min_i \ \mathbb{E}_{\mathbf{x}_i|x_s}[\mathrm{Var}(\mathbf{y} \mid \mathbf{x}_i, x_s)].$$

□

Next, we also prove the limiting result presented in eq. (3), which states that $I_i^n \to I(\mathbf{y}; \mathbf{x}_i \mid x_s)$.

Proof. The conditional mutual information $I(\mathbf{y}; \mathbf{x}_i \mid x_s)$ is defined as follows (Cover & Thomas, 2012):

$$\begin{aligned}
I(\mathbf{y}; \mathbf{x}_i \mid x_s) &= D_{\mathrm{KL}}\big(p(\mathbf{x}_i, \mathbf{y} \mid x_s) \mid\mid p(\mathbf{x}_i \mid x_s)p(\mathbf{y} \mid x_s)\big) \\
&= \mathbb{E}_{\mathbf{y},\mathbf{x}_i|x_s}\Big[\log \frac{p(\mathbf{y}, \mathbf{x}_i \mid x_s)}{p(\mathbf{x}_i \mid x_s)p(\mathbf{y} \mid x_s)}\Big].
\end{aligned}$$

Rearranging terms, we can write this as an expected KL divergence with respect to $\mathbf{x}_i$:

$$\begin{aligned}
I(\mathbf{y}; \mathbf{x}_i \mid x_s) &= \mathbb{E}_{\mathbf{x}_i|x_s}\mathbb{E}_{\mathbf{y}|x_s,\mathbf{x}_i}\Big[\log \frac{p(\mathbf{y}, \mathbf{x}_i \mid x_s)}{p(\mathbf{x}_i \mid x_s)p(\mathbf{y} \mid x_s)}\Big] \\
&= \mathbb{E}_{\mathbf{x}_i|x_s}\mathbb{E}_{\mathbf{y}|x_s,\mathbf{x}_i}\Big[\log \frac{p(\mathbf{y} \mid \mathbf{x}_i, x_s)}{p(\mathbf{y} \mid x_s)}\Big] \\
&= \mathbb{E}_{\mathbf{x}_i|x_s}\Big[D_{\mathrm{KL}}\big(p(\mathbf{y} \mid \mathbf{x}_i, x_s) \mid\mid p(\mathbf{y} \mid x_s)\big)\Big]
\end{aligned}$$

Now, when we sample multiple values $x_i^1, \ldots, x_i^n \sim p(\mathbf{x}_i \mid x_s)$ and make predictions using the Bayes classifier, we have the following mean prediction as n becomes large:

$$\lim_{n\to\infty} \frac{1}{n}\sum_{j=1}^{n} p(\mathbf{y} \mid x_s, x_i^j) = \mathbb{E}_{\mathbf{x}_i|x_s}\big[p(\mathbf{y} \mid \mathbf{x}_i, x_s)\big] = p(\mathbf{y} \mid x_s).$$

Calculating the mean KL divergence relative to this prediction, we arrive at the following result:

$$\lim_{n\to\infty} \ I_i^n = \mathbb{E}_{\mathbf{x}_i|x_s}\Big[D_{\mathrm{KL}}\big(p(\mathbf{y} \mid \mathbf{x}_i, x_s) \mid\mid p(\mathbf{y} \mid x_s)\big)\Big] = I(\mathbf{y}; \mathbf{x}_i \mid x_s).$$

□

Theorem 1. *When* $\mathbf{y}$ *is discrete and* ℓ *is cross-entropy loss, the global optimum of eq.* (5) *is a predictor that satisfies* $f(x_s; \theta^*) = p(\mathbf{y} \mid x_s)$ *and a policy* $\pi(x_s; \phi^*)$ *that puts all probability mass on* $i^* = \arg\max_i I(\mathbf{y}; \mathbf{x}_i \mid x_s)$.

Proof. We first consider the predictor network $f(\mathbf{x}_s;\theta)$. When the predictor is given the feature values x_s, it means that one index $i \in s$ was chosen by the policy according to $\pi(x_{s\setminus i};\phi)$ and the remaining indices $s \setminus i$ were sampled from $p(\mathbf{s})$. Because $\mathbf{s}$ is sampled independently from $(\mathbf{x},\mathbf{y})$, and because $\pi(x_{s\setminus i};\phi)$ is not given access to $(\mathbf{x}_{[d]\setminus s},\mathbf{x}_i,\mathbf{y})$, the predictor's expected loss must be considered with respect to the distribution $\mathbf{y} \mid x_s$. The globally optimal predictor $f(x_s;\theta^*)$ is thus defined as follows, regardless of the selection policy $\pi(x_s;\phi)$ and which index i was selected last:

$$f(x_s;\theta^*) = \arg\min_{\hat{y}} \ \mathbb{E}_{\mathbf{y}|x_s}\big[\ell(\hat{y},\mathbf{y})\big] = p(\mathbf{y} \mid x_s).$$

The above result follows from our proof for Proposition 1. Now, given the optimal predictor $f(x_s;\theta^*)$, we can define the globally optimal policy by minimizing the expected loss for a fixed input x_s. Denoting the probability mass placed on each index $i \in [d]$ as $\pi_i(x_s;\phi)$, where $\pi(x_s;\phi) \in \Delta^{d-1}$, the expected loss is the following:

$$\begin{aligned}\mathbb{E}_{i\sim\pi(x_s;\phi)}\mathbb{E}_{\mathbf{y},\mathbf{x}_i|x_s}\big[\ell(f(x_s\cup\mathbf{x}_i;\theta^*),\mathbf{y})\big] &= \sum_{i\in[d]}\pi_i(x_s;\phi)\mathbb{E}_{\mathbf{y},\mathbf{x}_i|x_s}\big[\ell\big(f(x_s\cup\mathbf{x}_i;\theta^*),\mathbf{y}\big)\big]\\ &= \sum_{i\in[d]}\pi_i(x_s;\phi)\mathbb{E}_{\mathbf{x}_i|x_s}[H(\mathbf{y}\mid\mathbf{x}_i,x_s)].\end{aligned}$$

The above result follows from our proof for Proposition 2. If there exists a single index $i^* \in [d]$ that yields the lowest expected conditional entropy, or

$$\mathbb{E}_{\mathbf{x}_{i^*}|x_s}[H(\mathbf{y}\mid\mathbf{x}_{i^*},x_s)] < \mathbb{E}_{\mathbf{x}_i|x_s}[H(\mathbf{y}\mid\mathbf{x}_i,x_s)] \quad \forall i \neq i^*,$$

then the optimal predictor must put all its probability mass on i^*, or $\pi_{i^*}(x_s;\phi^*) = 1$. Note that the corresponding feature $\mathbf{x}_{i^*}$ has maximum conditional mutual information with $\mathbf{y}$, because we have

$$I(\mathbf{y};\mathbf{x}_{i^*}\mid x_s) = \underbrace{H(\mathbf{y}\mid x_s)}_{\text{Constant}} - \mathbb{E}_{\mathbf{x}_{i^*}|x_s}[H(\mathbf{y}\mid\mathbf{x}_{i^*},x_s)].$$

To summarize, we derived the global optimum to our objective $\mathcal{L}(\theta,\phi)$ by first considering the optimal predictor $f(\mathbf{x}_s;\theta^*)$, and then considering the optimal policy $\pi(\mathbf{x}_s;\phi^*)$ when we assume that we use the optimal predictor. □

Theorem 2. *When* $\mathbf{y}$ *is continuous and* ℓ *is squared error loss, the global optimum of eq.* (5) *is a predictor that satisfies* $f(x_s;\theta^*) = \mathbb{E}[\mathbf{y}\mid x_s]$ *and a policy* $\pi(x_s;\phi^*)$ *that puts all probability mass on* $i^* = \arg\min_i \mathbb{E}_{\mathbf{x}_i|x_s}[\mathrm{Var}(\mathbf{y}\mid\mathbf{x}_i,x_s)]$.

Proof. Our proof follows the same logic as our proof for Theorem 1. For the optimal predictor given an arbitrary policy, we have:

$$f(x_s;\theta^*) = \arg\min_{\hat{y}} \ \mathbb{E}_{\mathbf{y}|x_s}\big[(\hat{y}-\mathbf{y})^2\big] = \mathbb{E}[\mathbf{y}\mid x_s].$$

Then, for the policy's expected loss, we have:

$$\mathbb{E}_{i\sim\pi(x_s;\phi)}\mathbb{E}_{\mathbf{y},\mathbf{x}_i|x_s}\big[\big(f(x_s\cup\mathbf{x}_i;\theta^*)-\mathbf{y}\big)^2\big] = \sum_{i\in[d]}\pi_i(x_s;\phi)\mathbb{E}_{\mathbf{x}_i|x_s}[\mathrm{Var}(\mathbf{y}\mid\mathbf{x}_i,x_s)].$$

If there exists an index $i^* \in [d]$ that yields the lowest expected conditional variance, then the optimal policy must put all its probability mass on i^*, or $\pi_{i^*}(x_s;\phi^*) = 1$. □

B. Datasets

The datasets used in our experiments are summarized in Table 2. Three of the tabular datasets and the two image classification datasets are publicly available, and the three emergency medicine tasks were privately curated from the Harborview Medical Center Trauma Registry.

Table 2. Summary of datasets used in our experiments.

Dataset	# Features	# Feature Groups	# Classes	# Samples
Fluid	224	162	2	2,770
Respiratory	112	35	2	65,515
Bleeding	121	44	2	6,496
Spam	58	–	2	4,601
MiniBooNE	51	–	2	130,064
Diabetes	45	–	3	92,062
MNIST	784	–	10	60,000
CIFAR-10	1,024	64	10	60,000

B.1. MiniBooNE and spam classification

The spam dataset includes features extracted from e-mail messages to predict whether or not a message is spam. Three features describes the usage of capital letters in the e-mail, and the remaining 54 features describe the frequency with which certain key words or characters are used. The MiniBooNE particle identification dataset involves distinguishing electron neutrinos from muon neutrinos based on various continuous features (Roe et al., 2005). Both datasets were obtained from the UCI repository (Dua & Graff, 2017).

B.2. Diabetes classification

The diabetes dataset was obtained from from the National Health and Nutrition Examination Survey (NHANES) (NHA, 2018), an ongoing survey designed to assess the well-being of adults and children in the United States. We used a version of the data pre-processed by Kachuee et al. (2018; 2019) that includes data collected from 1999 through 2016. The input features include demographic information (age, gender, ethnicity, etc.), lab results (total cholesterol, triglyceride, etc.), examination data (weight, height, etc.), and questionnaire answers (smoking, alcohol, sleep habits, etc.). An expert was also asked to suggest costs for each feature based on the financial burden, patient privacy, and patient inconvenience, but we assume uniform feature costs in our experiments. Finally, the fasting glucose values were used to define three classes based on standard threshold values: normal, pre-diabetes, and diabetes.

B.3. Image classification datasets

The MNIST and CIFAR-10 datasets were downloaded using PyTorch (Paszke et al., 2017). We used the standard train-test splits, and we split the train set to obtain a validation set with the same size as the test set (10,000 examples).

B.4. Emergency medicine datasets

The emergency medicine datasets used in this study were gathered over a 13-year period (2007-2020) and encompass 14,463 emergency department admissions. We excluded patients under the age of 18, and we curated 3 clinical cohorts commonly seen in pre-hospitalization settings. These include 1) pre-hospital fluid resuscitation, 2) emergency department respiratory support, and 3) bleeding after injury. These datasets are not publicly available due to patient privacy concerns.

Pre-hospital fluid resuscitation We selected 224 variables that were available in the pre-hospital setting, including dispatch information (injury date, time, cause, and location), demographic information (age, sex), and pre-hospital vital signs (blood pressure, heart rate, respiratory rate). The outcome was each patient's response to fluid resuscitation, following the Advanced Trauma Life Support (ATLS) definition (Subcommittee et al., 2013).

Emergency department respiratory support In this cohort, our goal is to predict which patients require respiratory support upon arrival in the emergency department. Similar to the previous dataset, we selected 112 pre-hospital clinical features including dispatch information (injury date, time, cause, and location), demographic information (age, sex), and pre-hospital vital signs (blood pressure, heart rate, respiratory rate). The outcome is defined based on whether a patient received respiratory support, including both invasive (intubation) and non-invasive (BiPap) approaches.

Bleeding In this cohort, we only included patients whose fibrinogen levels were measured, as this provides an indicator for bleeding or fibrinolysis (Mosesson, 2005). As with the previous datasets, demographic information, dispatch information, and pre-hospital observations were used as input features. The outcome, based on experts' opinion, was defined by whether an individual's fibrinogen level is below 200 mg/dL, which represents higher risk of bleeding after injury.

C. Baselines

This section provides more details on the baseline methods used in our experiments (Section 6).

C.1. Global feature importance methods

Two of our static feature selection baselines, permutation tests and SAGE, are *global feature importance methods* that rank features based on their role in improving model accuracy (Covert et al., 2021). In our experiments, we ran each method using a single classifier trained on the entire dataset, and we then selected the top k features depending on the budget.

When running the permutation test, we calculated the validation AUROC while replacing values in the corresponding feature column with random draws from the training set. When running SAGE, we used the authors' implementation with automatic convergence detection (Covert et al., 2020). To handle held-out features, we averaged across 128 sampled values for the six tabular datasets, and for MNIST we used a zeros baseline to achieve faster convergence.

C.2. Local feature importance methods

Two of our static feature selection baselines, DeepLift and Integrated Gradients, are *local feature importance methods* that rank features based on their importance to a single prediction. In our experiments, we generated feature importance scores for the true class using all examples in the validation set. We then selected the top k features based on their mean absolute importance. We used a mean baseline for Integrated Gradients (Sundararajan et al., 2017), and both methods were run using the Captum package (Kokhlikyan et al., 2020).

C.3. Differentiable feature selection

Our last static feature selection baseline is the Concrete autoencoder (CAE) from Balın et al. (2019). The method was originally proposed to perform unsupervised feature selection by reconstructing the full input vector, but we changed the prediction target to use it in a supervised fashion. The authors propose training with an exponentially decayed temperature over a hand-tuned number of epochs, but we used an approach similar to our own method: we trained with a sequence of temperature values, performing early stopping using the validation loss for each one, and we returned the features chosen after training with the final temperature.

We tried a similar method proposed by Yamada et al. (2020), but this method requires tuning a penalty parameter to achieve the desired number of features, and we found that it gave similar performance in our experiments on MNIST. Among methods that learn to select features within a neural network, there are several others that do so using group sparse penalties (Feng & Simon, 2017; Tank et al., 2021; Lemhadri et al., 2021); we tested the LassoNet approach from Lemhadri et al. (2021) and found that it was not competitive on MNIST. For simplicity, we present results only for the supervised CAE.

C.4. CMI estimation

Our experiments use two versions of the CMI estimation approach described in Section 3.2. Both are inspired by the EDDI method introduced by Ma et al. (2019), but a key difference is that we do not jointly model $(\mathbf{x}, \mathbf{y})$ within the same conditional generative model: we instead separately model the response with a classifier $f(\mathbf{x}_s) \approx p(\mathbf{y} \mid \mathbf{x}_s)$ and the features with a generative model of $p(\mathbf{x}_i \mid \mathbf{x}_s)$. This partially mitigates one challenge with this approach, which is working with mixed continuous/categorical data (i.e., we do not need to jointly model categorical response variables).

For the first version of this approach, we train a PVAE as a generative model (Ma et al., 2019). The encoder and decoder both have two hidden layers, the latent dimension is set to 16, and we use 128 samples from the latent posterior to approximate $p(\mathbf{x}_i \mid x_s) = \int p(\mathbf{x}_i \mid \mathbf{z})p(\mathbf{z} \mid x_s)$. We use Gaussian distributions for both the latent and decoder spaces, and we generate samples using the decoder mean, similar to the original approach (Ma et al., 2019). In the second version, we bypass the need for a generative model with a simple approximation: we sample features from their marginal distribution, which is equivalent to assuming feature independence.

C.5. Opportunistic learning

Kachuee et al. (2018) proposed Opportunistic Learning (OL), an approach to solve DFS using RL. The model consists of two networks analogous to our policy and predictor: a Q-network that estimates the value associated with each action, where actions correspond to features, and a P-network responsible for making predictions. When using OL, we use the same architectures as our approach, and OL shares network parameters between the P- and Q-networks.

The authors introduce a utility function for their reward, shown in eq. (6), which calculates the difference in prediction uncertainty as approximated by MC dropout (Gal & Ghahramani, 2016). The reward also accounts for feature costs, but we set all feature costs to $c_i = 1$:

$$r_i = \frac{||Cert(x_s) - Cert(x_s \cup x_i)||}{c_i} \tag{6}$$

To provide a fair comparison with the remaining methods, we made several modifications to the authors' implementation. These include 1) preventing the prediction action until the pre-specified budget is met, 2) setting all feature costs to be identical, and 3) supporting pre-defined feature groups as described in Appendix D.4. When training, we update the P-, Q-, and target Q-networks every $1 + \frac{d}{100}$ experiences, where d is the number of features in a dataset. In addition, the replay buffer is set to store the $1000d$ most recent experiences, and the random exploration probability is decayed so that it eventually reaches a value of 0.1.

D. Training approach and hyperparameters

This section provides more details on our training approach and hyperparameter choices.

D.1. Training pseudocode

Algorithm 1 summarizes our training approach. Briefly, we select features by drawing a Concrete sample using policy network's logits, we calculate the loss based on the subsequent prediction, and we then update the mask for the next step using a discrete sample from the policy's distribution. We implemented this approach using PyTorch (Paszke et al., 2017) and PyTorch Lightning.[3]

Algorithm 1: Training pseudocode

Input: Data distribution $p(\mathbf{x}, \mathbf{y})$, budget $k > 0$, learning rate $\gamma > 0$, temperature $\tau > 0$
Output: Predictor model $f(\mathbf{x}; \theta)$, policy model $\pi(\mathbf{x}; \phi)$
initialize $f(\mathbf{x}; \theta), \pi(\mathbf{x}; \phi)$
while *not converged* **do**
  sample $x, y \sim p(\mathbf{x}, \mathbf{y})$
  initialize $\mathcal{L} = 0$, $m = [0, \ldots, 0]$
  for $j = 1$ **to** k **do**
    calculate logits $\alpha = \pi(x \odot m; \phi)$, sample $G_i \sim$ Gumbel for $i \in [d]$
    set $\tilde{m} = \max\big(m, \mathrm{softmax}(G + \alpha, \tau)\big)$ `// update with Concrete`
    set $m = \max\big(m, \mathrm{softmax}(G + \alpha, 0)\big)$ `// update with one-hot`
    update $\mathcal{L} \leftarrow \mathcal{L} + \ell\big(f(x \odot \tilde{m}; \theta), y\big)$
  end
  update $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}, \quad \phi \leftarrow \phi - \gamma \nabla_\phi \mathcal{L}$
end
return $f(\mathbf{x}; \theta), \pi(\mathbf{x}; \phi)$

One notable difference between Algorithm 1 and our objective $\mathcal{L}(\theta, \phi)$ in the main text is the use of the policy $\pi(\mathbf{x}; \phi)$ for generating feature subsets. This differs from eq. (5), which generates feature subsets using a subset distribution $p(\mathbf{s})$. The key shared factor between both approaches is that there are separate optimization problems over each feature set that are effectively treated independently. For each feature set x_s, the problem is the one-step-ahead loss, and it incorporates both the policy and predictor as follows:

$$\mathbb{E}_{i \sim \pi(\mathbf{x_s}; \phi)} \big[\ell\big(f(\mathbf{x_s} \cup \mathbf{x}_i; \theta), \mathbf{y}\big) \big]. \tag{7}$$

The problems for each subset do not interact: during optimization, the selection given x_s is based only on the immediate change in the loss, and gradients are not propagated through multiple selections as they would be for an RL-based solution. In solving these multiple problems, the difference is simply that eq. (5) weights them according to $p(\mathbf{s})$, whereas Algorithm 1 weights them according to the current policy $\pi(\mathbf{x}, \phi)$.

We find that incorporating the current policy when generating feature sets is important to achieve good performance. As an ablation, we tested how much our method's performance changes when we instead generate training examples $(\mathbf{x_s}, \mathbf{y})$ at random rather than using the current policy: using the MNIST dataset, we find that using random subsets leads to a significant drop in performance (Table 3).

D.2. Model selection

One detail not shown in Algorithm 1 that we alluded to in the main text is our approach for decaying the Concrete distribution's temperature parameter τ. We train with a sequence of relatively few temperature values, using the validation loss to perform early stopping with each value. To perform model selection, we separately calculate the validation loss using a temperature value of zero, which more accurately represents the model's usage at inference time; we eventually return the version of the model that performed best on this zero-temperature loss, chosen across all training temperatures.

[3] `https://www.pytorchlightning.ai`

Table 3. Ablation experiment using MNIST.

# Features	5	10	15	20	25	30	40	50
Ours	0.695	0.875	0.926	0.950	0.960	0.966	0.973	0.975
Ablation	0.578	0.757	0.807	0.819	0.838	0.850	0.869	0.883

D.3. Hyperparameters

Our experiments with the six tabular datasets used fully connected architectures with dropout in all layers (Srivastava et al., 2014). The dropout probability is set to 0.3, the networks have two hidden layers of width 128, and we performed early stopping using the validation loss. For our method, the predictor and policy were separate networks with identical architectures. When training models with the features selected by static methods, we reported results using the best model from multiple training runs based on the validation loss. We did not perform any additional hyperparameter tuning due to the large number of models being trained.

For MNIST, we used fully connected architectures with two layers of width 512 and the dropout probability set to 0.3. Again, our method used separate networks with identical architectures. For CIFAR-10, we used a shared ResNet backbone (He et al., 2016b) consisting of several residually connected convolutional layers. The classification head consists of global average pooling and a linear layer, and the selection head consisted of a transposed convolution layer followed by a 1×1 convolution, which outputs a grid of logits with size 8×8. Our CIFAR-10 networks are trained using random crops and random horizontal flips as augmentations.

D.4. Feature grouping

All of the methods used in our experiments were designed to select individual features, but this is undesirable when using categorical features with one-hot encodings. Each of our three emergency medicine tasks involve such features, so we extended each method to support feature grouping.

SAGE and permutation tests are trivial to extend to feature groups: we simply removed groups of features rather than individual features when calculating importance scores. For DeepLift and Integrated Gradients, we used the summed importance within each group, which preserves each method's additivity property. For the method based on Concrete Autoencoders, we implemented a generalized version of the selection layer that operates on feature groups. We also extended OL to operate on feature groups by having actions map to groups rather than individual features.

Finally, for our method, we parameterized the policy network $\pi(\mathbf{x}; \phi)$ so that the number of outputs is the number of groups g rather than the total number of features d (where $g < d$). When applying masking, we first generate a binary mask $m \in [0, 1]^g$, and we then project the mask into $[0, 1]^d$ using a binary group matrix $G \in \{0, 1\}^{d \times g}$, where $G_{ij} = 1$ if feature i is in group j and $G_{ij} = 0$ otherwise. Thus, our masked input vector is given by $x \odot (Gm)$.

E. Additional results

This section provides several additional experimental results. First, Figure 4 and Figure 5 show the same results as Figure 2 but larger for improved visibility. Next, Figure 6 though Figure 11 display the feature selection frequency for each of the tabular datasets when using the greedy method. The heatmaps in each plot show the portion of the time that a feature (or feature group) is selected under a specific feature budget. These plots reveal that our method is indeed selecting different features for different samples.

Finally, Figure 12 displays examples of CIFAR-10 predictions given different numbers of revealed patches. The predictions generally become relatively accurate after revealing only a small number of patches, reflecting a similar result as Figure 3. Qualitatively, we can see that the policy network learns to select vertical stripes, but the order in which it fills out each stripe depends on where it predicts important information may be located.

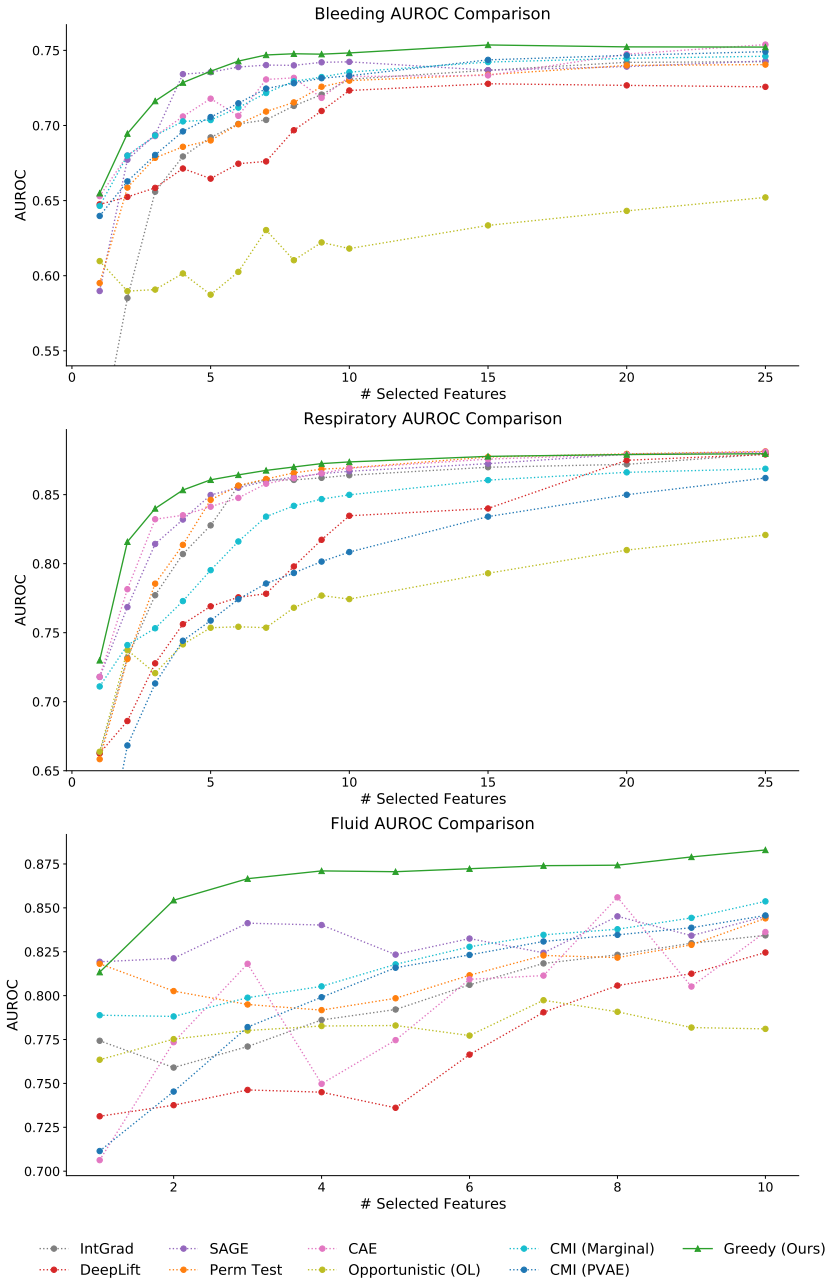

Figure 4. AUROC comparison on the three emergency medicine diagnosis tasks.

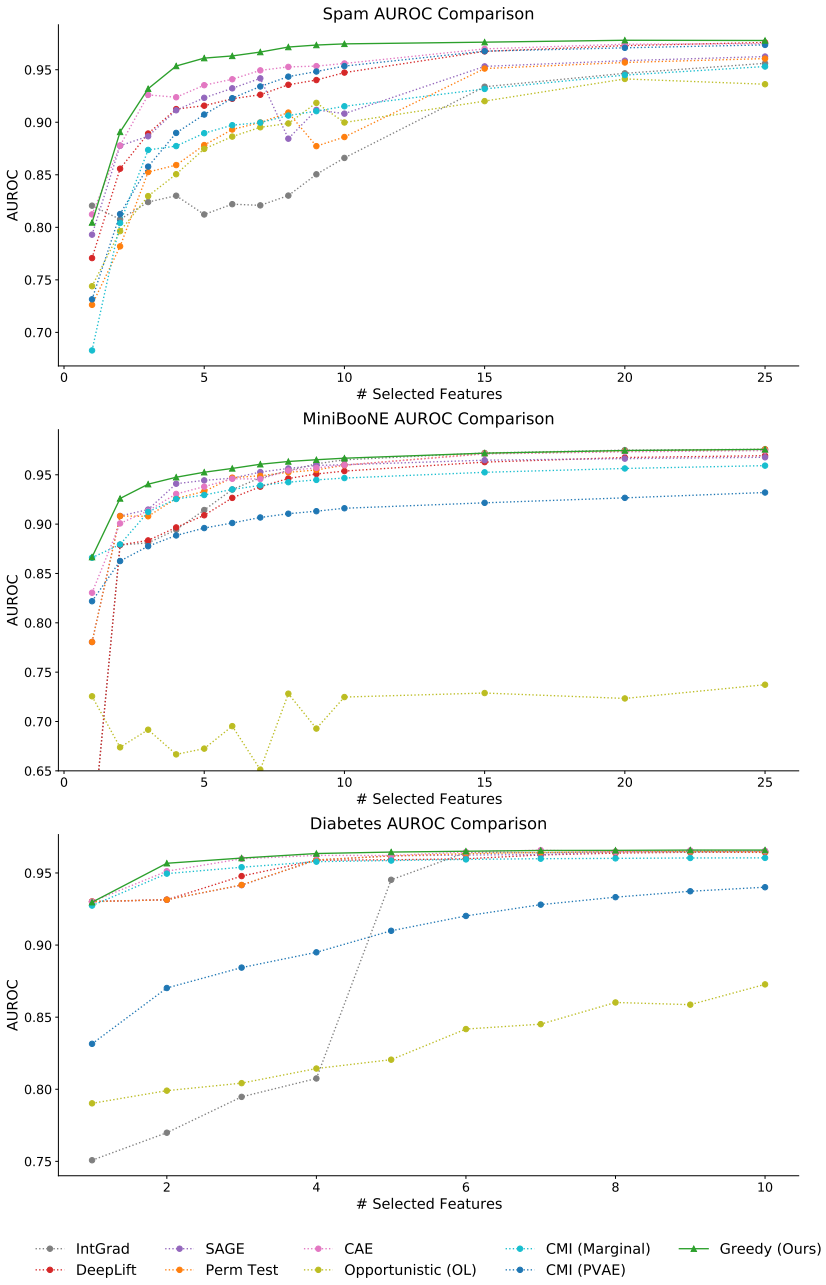

Figure 5. AUROC comparison on the three public tabular datasets.

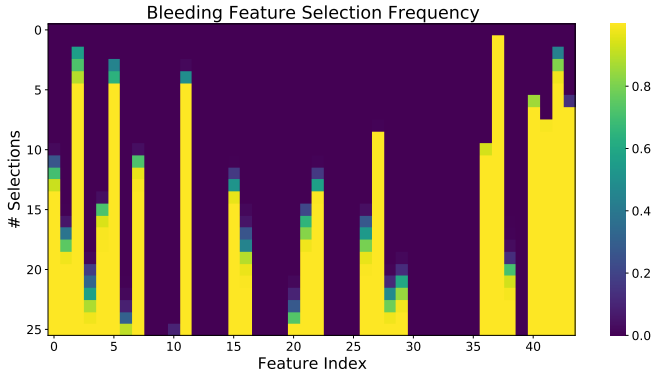

Figure 6. Feature selection frequency for our greedy approach on the bleeding dataset.

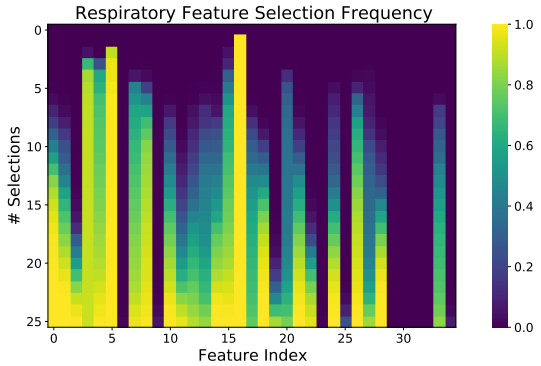

Figure 7. Feature selection frequency for our greedy approach on the respiratory dataset.

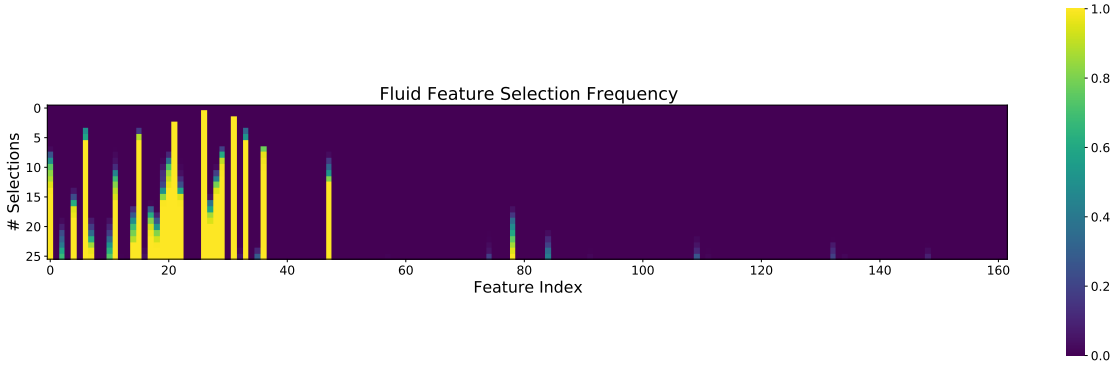

Figure 8. Feature selection frequency for our greedy approach on the fluid dataset.

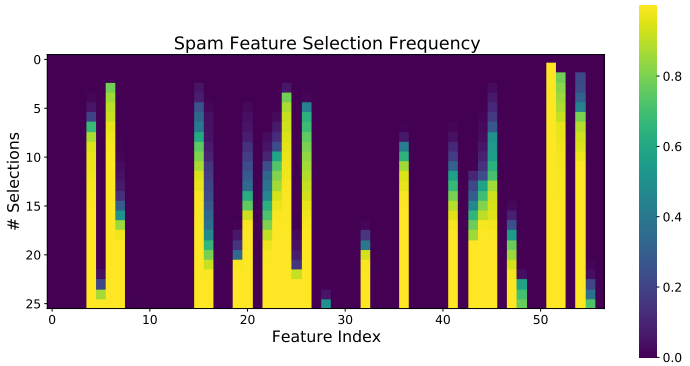

Figure 9. Feature selection frequency for our greedy approach on the spam dataset.

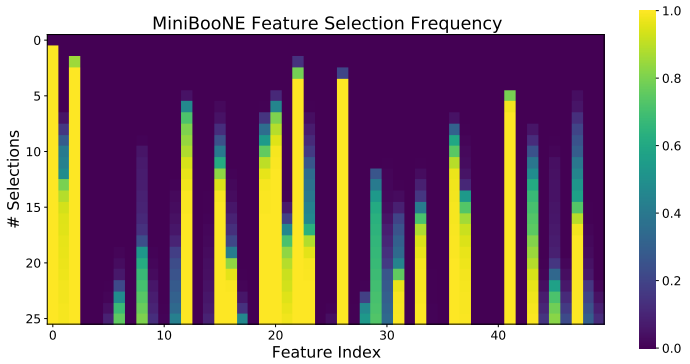

Figure 10. Feature selection frequency for our greedy approach on the MiniBooNE dataset.

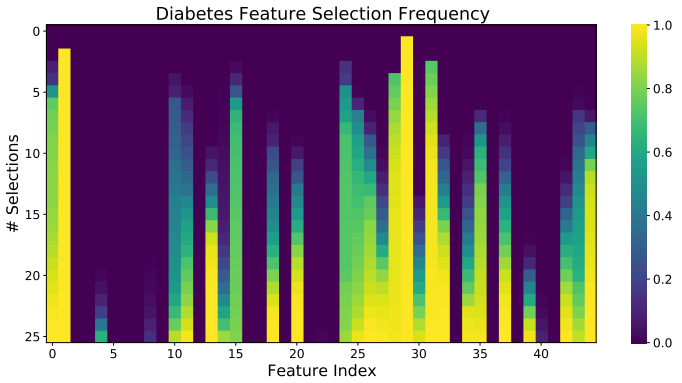

Figure 11. Feature selection frequency for our greedy approach on the diabetes dataset.

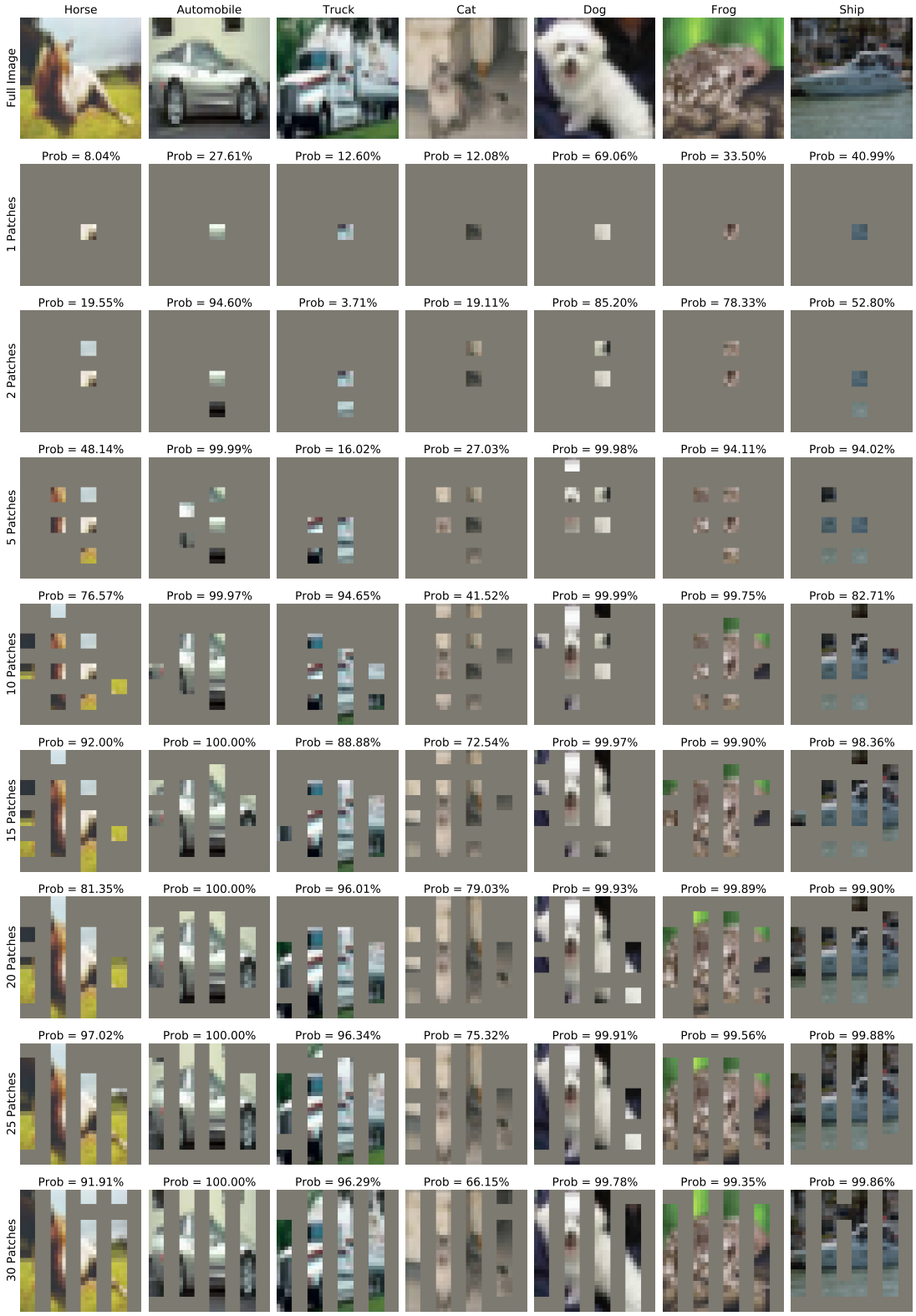

Figure 12. CIFAR-10 predictions with different numbers of patches revealed by our approach.