DiFA: Differentiable Feature Acquisition

Aritra Ghosh and Andrew Lan

University of Massachusetts Amherst
arighosh@cs.umass.edu, andrewlan@cs.umass.edu

Abstract

Feature acquisition in predictive modeling is an important task in many practical applications. For example, in patient health prediction, we do not fully observe their personal features and need to dynamically select features to acquire. Our goal is to acquire a small subset of features that maximize prediction performance. Recently, some works reformulated feature acquisition as a Markov decision process and applied reinforcement learning (RL) algorithms, where the reward reflects both prediction performance and feature acquisition cost. However, RL algorithms only use zeroth-order information on the reward, which leads to slow empirical convergence, especially when there are many actions (number of features) to consider. For predictive modeling, it is possible to use first-order information on the reward, i.e., gradients, since we are often given an already collected dataset. Therefore, we propose differentiable feature acquisition (DiFA), which uses a differentiable representation of the feature selection policy to enable gradients to flow from the prediction loss to the policy parameters. We conduct extensive experiments on various real-world datasets and show that DiFA significantly outperforms existing feature acquisition methods when the number of features is large.

Introduction

The dominant paradigm of supervised learning assumes access to fully observed feature vectors and target values during the training of predictive models. However, access to fully observed data points is impractical in many real-world scenarios. In many applications, a model can acquire additional information (e.g., features) at an acquisition cost. Thus, the model should know when to acquire a feature and what features to acquire before making a prediction. For example, consider a doctor assessing the health of a patient: initially, the doctor may only know a few symptoms the patient is experiencing. The doctor needs to dynamically decide what additional information to ask for and further diagnostic tests to perform. Acquiring results from all possible tests is infeasible due to time and financial constraints. Therefore, the doctor needs to assess the health of the patient, dynamically acquire additional information under practical constraints, and decide on future treatment for the patient using this acquired information (Qayyum et al. 2020). A similar situation happens in education, where in online learning platforms or intelligent tutoring systems, we can test a student's knowledge on certain concepts using assessment questions and craft a better learning plan, but we need to limit the number of questions to ask to avoid overwhelming the student (Siemens 2013). Thus, dynamically acquiring relevant features is important to both minimize the feature acquisition cost and maximize prediction performance, with a significant impact on real-world applications (Shim, Hwang, and Yang 2018).

We note that the **feature acquisition** task is different from the **feature selection** or dimensionality reduction task, where the goal is to select a static subset of features for all data points to reduce the number of input variables (Guyon and Elisseeff 2003; Li et al. 2017; Cai et al. 2018). In contrast, the dynamic feature acquisition task, i.e., selecting the next feature to acquire based on past observation, has similarities to the active learning task, where we need to actively select the next data point to label. Thus, many active learning-based heuristics (such as uncertainty sampling, diversity sampling, etc.) can be extended to the task of feature acquisition (Lewis and Gale 1994; Brinker 2003; Settles 2009). Some earlier works in active feature acquisition use uncertainty sampling, expected utility gain, or variance-based approaches (Melville et al. 2004; Saar-Tsechansky, Melville, and Provost 2009; Huang et al. 2018; Gong et al. 2019; Ma et al. 2019). For example, the efficient dynamic discovery of high-value information (EDDI) method uses a probabilistic model to find the expected utility using information gained from each unobserved feature and acquire the feature with the highest value (Ma et al. 2019). EDDI uses a variational autoencoder with arbitrary conditioning (VAEAC) probabilistic model (Ivanov, Figurnov, and Vetrov 2018) to estimate a latent variable from partially observed features and compute the joint distribution of both missing features and the target variable. This joint distribution enables one to compute the expected information gain to greedily acquire the next feature based on mutual information. However, these *static* heuristics-based feature acquisition policies are neither optimal nor data-driven and cannot improve with more training data.

To address this limitation, some prior works (Shim, Hwang, and Yang 2018; Li and Oliva 2021) reformulate the feature acquisition task as a Markov decision process (MDP) where the policy selects the next feature to acquire based on the current MDP state, which is a function of the observed fea-

tures (Zubek and Dietterich 2002; Rückstieß, Osendorfer, and Smagt 2011; He, Mineiro, and Karampatziakis 2016). The joint active feature acquisition and classification (JAFA) method (Shim, Hwang, and Yang 2018) is one of the earlier methods that apply reinforcement learning (RL) for the task of feature acquisition; the JAFA method uses a deep Q-network-based (DQN) RL method (Mnih et al. 2016, 2015). However, for potentially long episodes (or a large number of acquired features), RL methods obtain sparse rewards and suffer from credit assignment problems (Li and Oliva 2021). The generative surrogate model for RL (GSMRL) method, augments JAFA with a probabilistic model to provide intermediate rewards (using expected information gain) and to provide the policy with additional side information about each unobserved feature (Li and Oliva 2021). The optimal policy learned by the GSMRL method remains the same as that by the JAFA policy; however, side information and intermediate rewards help the GSMRL method during its optimization process, resulting in improved empirical performance. Moreover, the GSMRL method uses proximal policy optimization (PPO), which has better performance than DQN on many tasks (Schulman et al. 2017).

In RL algorithms, a policy typically observes the reward after taking action for one or more steps; the reward is often assumed to be non-differentiable (Sutton and Barto 2018). Thus, many policy gradient-based RL algorithms (e.g., PPO) use zeroth-order information on the reward function when optimizing policy parameters (Williams 1992; Schulman et al. 2017). In the case of feature acquisition, the policy dynamically selects a subset of features (actions) and observes the target prediction performance (the reward). However, in contrast to applications like game playing and robot control, where we only have access to the reward itself, in feature acquisition, we can obtain **first-order** information on the reward, i.e., gradients of the predictive loss. This information offers us more information than the zeroth-order reward itself and can potentially help us improve feature acquisition.

Contributions. In this paper, we attempt at making use of first-order gradient information to improve data-driven dynamic feature acquisition in supervised learning. We show that we can obtain a differentiable estimator of the reward function (consisting of feature acquisition cost and the log-likelihood for the prediction task) that leads to improved feature acquisition policies, especially when the number of features is large. We propose DiFA, a **Di**fferentiable **F**eature **A**cquisition framework, to *learn* a feature acquisition policy that optimizes the performance on the target prediction task. The feature acquisition policy is learned in an end-to-end manner together with the prediction model. We verify the effectiveness of DiFA through extensive experiments on several large real-world datasets. We observe that the learned acquisition policy outperforms existing policies in the prediction tasks, requiring (sometimes significantly) fewer features to reach the same predictive quality.

Methodology

We consider the supervised learning setup where each data point is associated with a feature vector $\mathbf{x} = (x_1, \cdots, x_D)$, consisting of D features, which can be real-valued or categorical, and a target variable $\mathbf{y}$, which can be real-valued ($\in \mathbb{R}$), binary ($\in \{0, 1\}$), or categorical ($\in \{0, \cdots, C-1\}$). We note that in practice, some features can be missing, i.e., unobserved, in some data points. The feature acquisition task is to acquire a certain number of features *sequentially* to maximize the performance of the supervised target variable prediction task. Thus, the feature acquisition policy Π starts with an empty feature set $\emptyset$ and acquires a set of K non-missing features ($\subset \{1, \cdots, D\}$) sequentially and use these acquired features to predict the target $\mathbf{y}$.

At time step t, we denote the $t-1$ already acquired features from previous time steps as $\boldsymbol{O}_{t-1} = \{(1), \cdots, (t-1)\}$ where (τ) denotes the acquired feature at time step τ; $\mathcal{O}_0$ is an empty set. We denote the partially observed feature set as $\mathbf{s}_{t-1} \subset \mathcal{S}$ consisting of observed feature values $x_{(1)}, \cdots, x_{(t-1)}$, the mask for acquired features $\boldsymbol{O}_{t-1}$ and any other additional side information $\mathbf{a}_{t-1}$. The policy $\Pi(\cdot)$ selects the next feature to acquire, $(t) \in \{1, \cdots, D\} \backslash (\boldsymbol{O}_{t-1} \cup \mathcal{M})$, conditioned on $\mathbf{s}_{t-1}$, where $\mathcal{M}$ are the missing features. Since we do not know the missing feature values in the original dataset, we cannot acquire these missing features during policy learning. We define a set of **valid features** as $\mathcal{V}_{t-1} = \{1, \cdots, D\} \backslash (\boldsymbol{O}_{t-1} \cup \mathcal{M})$; the policy can only acquire features from the valid feature set $\mathcal{V}_{t-1}$ at time step t. To simplify notations, we will omit the set of missing features $\mathcal{M}$ hereafter. We denote the predictive model as $f(\cdot)$ that takes as input the feature vector (which can be partially observed) and outputs the predicted target variable. The feature acquisition optimization problem (for a single data point) can be written as

$$\underset{\theta, \phi}{\text{minimize}} \sum_{t=1}^{K} \ell\Big(\mathbf{y}, f\Big(\mathbf{s}_t(\phi); \theta\Big)\Big) \tag{1}$$

$$\text{s.t. } \boldsymbol{O}_t = \boldsymbol{O}_{t-1} \cup \Pi(\mathbf{s}_{t-1}; \phi),\ \forall t \in \{1, \cdots, K\}. \tag{2}$$

Here, θ and ϕ are parameters for the prediction model $f(\cdot)$ and feature acquisition policy $\Pi(\cdot)$, respectively.

We note that there are alternative ways to formulate the feature acquisition task. Each of the features j can be associated with a different cost value c_j; the goal is to maximize predictive performance while minimizing the total cost of all acquired features. We instead formulate our methodology with a uniform cost across features to enable a fair comparison with existing methods. We defer details on the DiFA framework with varying feature acquisition costs to the supplementary material, where we illustrate that it can be treated similarly to the case with uniform feature acquisition costs.

Differentiable Feature Acquisition

We note that the feature acquisition operation $\boldsymbol{O}_t = \boldsymbol{O}_{t-1} \cup \Pi(\mathbf{s}_{t-1}; \phi)$ is non-differentiable; RL algorithms excel in these situations, which is why they are used in previous works on feature acquisition. The RL action (next feature to acquire) distribution $p(j|\mathbf{s}; \phi)$ comes from the policy $\Pi(\mathbf{s}_t; \phi)$. We can regard the negative loss (or predictive likelihood) $-\ell(\mathbf{y}, f(\mathbf{s}(\phi); \theta))$ as the reward for taking action j, $r(j; \phi)$. Many RL algorithms (e.g., PPO) use zeroth-order information

of the reward function $r(\cdot)$ when optimizing policy parameters ϕ using the score-function stochastic gradient estimator (Schulman et al. 2017; Williams 1992)

$$\nabla_\phi \mathbb{E}_{p(j|\mathbf{s};\phi)}[r(j;\phi)] = \mathbb{E}_{p(j|\mathbf{s};\phi)}[r(j;\phi)\nabla_\phi p(j|\mathbf{s};\theta)], \quad (3)$$

where we denote the expectation operator as $\mathbb{E}$. The score-function estimator is universally applicable in the absence of gradient information of the reward function $r(\cdot)$. However, in contrast to applications like game playing and robot control, where we only have access to the reward $r(\cdot)$ itself, for observable datasets (such as the task of learning feature acquisition policy), we can often obtain first-order information of the reward function

$$\nabla_\phi \mathbb{E}_{p(j|\mathbf{s};\phi)}[r(j;\phi)] = \mathbb{E}_{\epsilon\sim p(\epsilon)}[\nabla_\phi r\big(q(\epsilon;\phi)\big)], \quad (4)$$

using a known sampling path of $p(j|\mathbf{s};\phi)$ from $q(\epsilon;\phi)$ with a random variable ϵ. The latter formulation, often known as the path-wise gradient estimator, has been shown to have lower variance than zeroth order stochastic estimators (Ghadimi and Lan 2013; Mohamed et al. 2020) and has been successfully used in multiple sequential decision-making tasks (Ghosh and Lan 2021; Ghosh, Mitra, and Lan 2022).

Thus, the key idea in learning a data-driven differentiable feature acquisition policy $\Pi(\cdot)$ is that we need to pass the gradient of the loss function $\ell(\cdot,\cdot)$ through the feature acquisition policy. For RL algorithms in (3), only the action distribution needs to be differentiable; we do not need to back-propagate gradients from the reward value (negative loss). However, if we want to use (4), we need to back-propagate (i) from the loss $\ell(\mathbf{y}, f(\mathbf{s}_t(\phi);\theta))$ to the input of the prediction model $\mathbf{s}_t$, $\frac{\partial \ell}{\partial \mathbf{s}_t}$, (ii) from $\mathbf{s}_t$ to the feature mask $\mathcal{O}_t$, $\frac{\partial \mathbf{s}_t}{\partial \mathcal{O}_t}$, and (iii) from $\mathcal{O}_t$ to the policy parameter ϕ, $\frac{\partial \mathcal{O}_t}{\partial \phi}$. Back-propagation from the loss $\ell(\mathbf{y}, f(\mathbf{s}_t(\phi);\theta))$ to the input of the prediction model $\mathbf{s}_t$ (i) is straight-forward. To back-propagate from the input $\mathbf{s}_t$ to the feature mask $\mathcal{O}_t$ (ii), we need to make sure the input representation $\mathbf{s}_t$ is a differentiable function of $\mathcal{O}_t$; we will detail the feature representation in the following subsections. To back-propagate from the feature mask $\mathcal{O}_t$ to the policy parameters ϕ, we need to approximate the non-differentiable feature acquisition operation, which we detail next.

Feature Acquisition Policy

For gradient backpropagation, we use a sparse mask vector $\mathbf{z}_{t-1} \in \{0,1\}^M$ to encode the indices of the features present in the acquired feature set $\boldsymbol{O}_{t-1}$ at time t, with $\mathbf{z}_{t-1,j} = 1$ if and only if the feature j has been acquired $j \in \mathcal{O}_{t-1}$. This vector $\mathbf{z}_{t-1}$ has a one-to-one correspondence with the acquired feature set $\mathcal{O}_{t-1}$ and we will use them interchangeably. The policy can use an arbitrary non-linear feature acquisition model $g(\cdot) : \mathcal{S} \to \mathbb{R}^D$ with parameters ϕ to score each of the available features that has not been acquired yet, and to acquire a single feature $\mathbf{w}_t \in \{0,1\}^D \cap \Delta^{D-1}$ for the next step. We use the Gumbel-softmax operator (ρ) (Jang, Gu, and Poole 2016) followed by straight-through approximation (Bengio, Léonard, and Courville 2013) to randomly sample a single feature $\mathbf{w}_t$ from the output of $g(\cdot)$ as the feature acquired by the policy. The updated acquired feature set $\mathbf{z}_t$ at time step $t+1$ is given by

$$\mathbf{z}_t = \mathbf{z}_{t-1} + \mathbf{w}_t\Big(\rho\big(g(\mathbf{s}_{t-1};\phi)\big)\Big) \quad (5)$$

The Gumbel softmax operation on vector $g(\cdot) \in \mathbb{R}^D$ is defined as (Jang, Gu, and Poole 2016)

$$\rho_j(\cdot) = \frac{\exp\big((g_j(\cdot)+\epsilon_j)/\tau\big)}{\sum_{k\in\mathcal{V}_{t-1}} \exp\big((g_k(\cdot)+\epsilon_k)/\tau\big)},\ \forall j \in \mathcal{V}_{t-1} \quad (6)$$

where $\epsilon_1, \cdots, \epsilon_D$ are i.i.d random variable drawn from the standard Gumbel distribution. The single feature acquisition operation is defined as

$$\mathbf{w}_t\big(g(\mathbf{s}_t;\phi)\big) = \text{one-hot}\Big(\operatorname*{argmax}_{j\in\mathcal{V}_{t-1}} \rho_j(\cdot)\Big).$$

The selection operation, $\mathbf{w}_t\big(\rho(\cdot)\big)$, is differentiable w.r.t. ϕ with the approximation, so we have

$$\frac{\partial \ell}{\partial \mathbf{w}_t\big(\rho_j(\cdot)\big)} \approx \frac{\partial \ell}{\partial \rho_j(\cdot)},\ \forall j \in \mathcal{V}_{t-1}. \quad (7)$$

We note that one can easily compute gradient of the output of the Gumbel-softmax operator $\rho(\cdot)$ w.r.t. policy parameters ϕ using the reparameterization trick (Kingma and Welling 2013). Empirically, straight-through Gumbel-softmax-based gradient estimators have lower variance than score function-based estimators in (3) (Jang, Gu, and Poole 2016), which have been applied in prior work to learn feature acquisition policies (Shim, Hwang, and Yang 2018; Li and Oliva 2021).

Feature Representation and Imputation Model

We also need to represent the feature vector $\mathbf{s}_t$ such that we can compute $\frac{\partial \mathbf{s}_t}{\partial \mathbf{z}_t}$ and in turns $\frac{\partial \ell}{\partial \mathbf{z}_t}$, $\frac{\partial \ell}{\partial \phi}$. To this end, we consider an optional missing value imputation model $h(\cdot;\psi)$. The imputation model h uses acquired feature values $x_{(1)}, \cdots, x_{(t)}$, and the mask for the acquired features, $\mathbf{z}_t$, to impute the missing feature values $\mathbf{x}_{\setminus \boldsymbol{O}_t}$. We consider the imputation model to be fixed throughout the feature acquisition training process. Our framework is agnostic to the imputation model h and mean/zero imputation is equally applicable in our case; we denote the imputed feature values as $\hat{\mathbf{x}} := (\hat{x}_1, \cdots, \hat{x}_D)$. In particular, we use VAEAC (Ivanov, Figurnov, and Vetrov 2018) to learn the missing features (potentially including the target also) conditioned on observed features $\mathbf{x}_{\mathcal{O}_t}$, similar to the EDDI method (Ma et al. 2019).

In addition to the imputed values for the missing features, the probabilistic VAEAC model can provide side information on features, which is used by the GSMRL method (Li and Oliva 2021). In particular, we consider the mean μ_j, standard deviation σ_j, and utility (expected information gain to the target) u_j for each missing feature j. The distribution of the target variable, $\mathbf{y}$, is given by $\mathcal{N}(\mu_\mathbf{y}, \sigma_\mathbf{y})$ for real-valued features and $\text{Cat}(p_1, \cdots, p_C)$ for categorical features where $\mathcal{N}$ and Cat represent Normal and Categorical distribution, respectively. We represent the target distribution using a single vector $\hat{\mathbf{y}} = (\mu_\mathbf{y}, \sigma_\mathbf{y})$ (or $(p_1, \cdots, p_C)$) and concatenate with the auxiliary information from the features. Thus, the auxiliary information vector is

$$\mathbf{a} = (\ \underbrace{\mu_1, \sigma_1, u_1}_{\text{side info on feature 1}}\ , \cdots,\ \underbrace{\mu_D, \sigma_D, u_D}_{\text{side info on feature D}}\ , \hat{\mathbf{y}}), \quad (8)$$

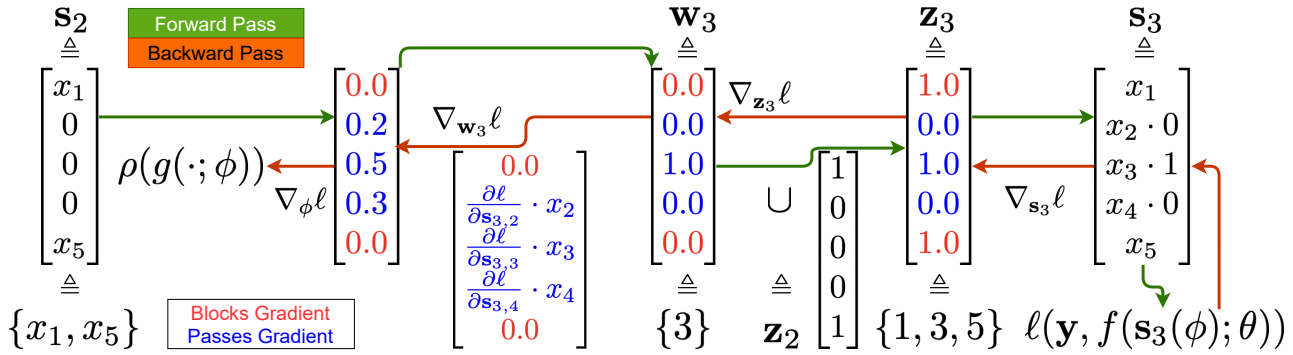

Figure 1: Visualization of DiFA's gradient calculation at time step 3, i.e., selecting the third feature. Observed features are $\{1, 5\}$ and the policy acquires feature 3 at this step; the gradient flows through valid features $\mathcal{V}_2 = \{2, 3, 4\}$. For simplification, we do not show feature mask $\mathbf{z}_t$, side information $\mathbf{a}_t$, and the imputed values $\hat{\mathbf{x}}$ when constructing $\mathbf{s}_t$ in (9).

where the expected information gain to the target is

$$u_j = \mathbb{H}(\mathbf{y}|\mathbf{x}_{\boldsymbol{O}}) - \mathbb{E}_{p(x_j|\mathbf{x}_{\boldsymbol{O}})}\mathbb{H}(\mathbf{y}|x_j, \mathbf{x}_{\boldsymbol{O}}),$$
$$= \mathbb{H}(x_j|\mathbf{x}_{\boldsymbol{O}}) - \mathbb{E}_{p(\mathbf{y}|\mathbf{x}_{\boldsymbol{O}})}\mathbb{H}(x_j|\mathbf{y}, \mathbf{x}_{\boldsymbol{O}}).$$

We denote the shannon entropy operator as $\mathbb{H}$. We note that the auxiliary information is only defined for features that have not yet been acquired; otherwise, they are assumed to be 0 for $j \in \boldsymbol{O}_t$.

We represent the feature vector at time step t as

$$\mathbf{s}_t = \Big(\underbrace{\mathbf{x} \odot \mathbf{z}_t + \hat{\mathbf{x}} \odot (1 - \mathbf{z}_t)}_{1} \oplus \underbrace{\mathbf{z}_t}_{2} \oplus \underbrace{\mathbf{a}_t}_{3} \Big), \quad (9)$$

where $\odot$ and $\oplus$ are element-wise multiplication and concatenation operators on vectors, respectively. The first term, $\mathbf{x} \odot \mathbf{z}_t + \hat{\mathbf{x}} \odot (1 - \mathbf{z}_t)$, uses the acquired feature values and imputes the rest using an optional imputation model h. The second term, $\mathbf{z}_t$, provides information on the acquired feature set. Together, these two terms provide full information on the state of the partially observed sample. The third auxiliary term, $\mathbf{a}_t$ provides side information on features; previous research found that they are useful in the presence of sparse information (Li and Oliva 2021). We use the feature vector $\mathbf{s}_t$ as input to both the feature acquisition model g and the prediction model f.

Gradients In (9), features that are not acquired yet, are imputed with the imputation model $h(\cdot; \psi)$; in case of mean imputation, we simply use the mean values. However, note that $\mathbf{s}_t$ is differentiable w.r.t. $\mathbf{z}_t$ (or $\mathcal{O}_t$) and even when a feature j is not acquired, the actual value of the that feature x_j affects the gradient computation. Recall that the imputation model is fixed during the training process and the gradient does not flow through the auxiliary information vector $\mathbf{a}_t$ in (9). Thus, for simplicity, we will omit this auxiliary vector. We rewrite the mask vector in (5) as

$$\mathbf{z}_t = \Big((\mathbf{z}_{t-1,1} + \mathbf{w}_{t,1}), \cdots, (\mathbf{z}_{t-1,D} + \mathbf{w}_{t,D}) \Big). \quad (10)$$

We note that $\mathbf{w}_{t,j}$ is 1 for the newly acquired feature and 0 otherwise. Moreover, the selection process is defined over valid features $\mathcal{V}_{t-1}$ (features not acquired till the last time step and not missing in the original data point, $\mathbf{x}$ i.e., $\in \{1, \cdots, D\} \backslash \boldsymbol{O}_{t-1} \cup \mathcal{M}$). Thus, the gradient flows only through $\mathbf{w}_{t,j}$ where $j \in \mathcal{V}_{t-1}$. $\mathbf{w}_{t,j}$ has effect on the first two terms in (9): the feature values $\mathbf{x} \odot \mathbf{z}_t + \hat{\mathbf{x}} \odot (1 - \mathbf{z}_t) \in \mathbb{R}^D$ (actual or imputed) and the feature mask $\mathbf{z}_t \in \mathbb{R}^D$. Thus, we have these two co-ordinates for each feature j as

$$\mathbf{s}_{t,j} = \hat{x}_j + (x_j - \hat{x}_j) \cdot (\mathbf{z}_{t-1,j} + \mathbf{w}_{t,j}(\phi)),$$
$$\mathbf{s}_{t,D+j} = \mathbf{z}_{t-1,j} + \mathbf{w}_{t,j}(\phi), \ \forall j \in \{1, \cdots, D\}.$$

For a valid feature $j \in \mathcal{V}_{t-1}$, we have

$$\frac{\partial \ell}{\partial \mathbf{w}_{t,j}} = \frac{\partial \ell}{\partial \mathbf{s}_{t,D+j}} + \frac{\partial \ell}{\partial \mathbf{s}_{t,j}} \cdot (x_j - \hat{x}_j). \quad (11)$$

Intuitively, on the one hand, if the imputed feature value $\hat{x}_j$ is close to the true feature value x_j (in the case of a good imputation model) or the gradient w.r.t. the feature value $\mathbf{s}_{t,j}$ is small (in the case of an unimportant feature), the weight $\mathbf{w}_{t,j}$ should remain similar. On the other hand, if the imputed feature value is far from the true feature value and the absolute value of the gradient w.r.t. to $\mathbf{s}_{t,j}$ is large, the weight for the feature $\mathbf{w}_{t,j}$ should be increased (or decreased) when $\frac{\partial \ell}{\partial \mathbf{s}_{t,j}} \cdot (x_j - \hat{x}_j) < 0$ (or > 0). We note that even when a feature j is not acquired at step t, we can still compute the gradient w.r.t. the weight $\mathbf{w}_{t,j}$ and pass it to the policy parameters. In contrast, score-function estimators in (3) have non-zero gradient only a feature j that is acquired, i.e., $\mathbf{w}_{t,j} = 1$:

$$\frac{\partial \ell}{\partial \mathbf{w}_{t,j}} = -\ell \log \nu_j(\cdot; \phi) \cdot \mathbb{1}[\mathbf{w}_{t,j} = 1], \quad (12)$$

where $\nu_j(\cdot; \phi)$ represents the probability (softmax output) of the acquired feature using a reinforcement learning policy with parameter ϕ. We summarize DiFA's training process in Algorithm 1. We visualize DiFA training process for a single time step in Figure 1.

Dealing with Spatial Features We encode the feature representation $\mathbf{s}_t$ differently for image datasets. The reason is that image datasets are particularly vulnerable when spatial features are provided as a single vector, even when flattened

Algorithm 1: DiFA training process

Input (optional) Imputation model $h(\cdot;\psi)$.
Initialize parameters ϕ, θ, learning rate η.
while not converged **do**
  Randomly sample data point(s) $(\mathbf{x}, \mathbf{y})$.
  Initialize $\mathcal{O}_0 = \emptyset$ for each data points.
  for $t \in 1 \cdots K$ **do**
    Sample feature $j \sim \Pi(\mathbf{s}_{t-1}; \phi)$ and $\mathcal{O}_t \leftarrow \mathcal{O}_{t-1} \cup j$.
    Update feature mask $\mathbf{z}_t$ and representation $\mathbf{s}_{\mathcal{O}_t}$ using (5), and (9).
    Compute loss $\ell\Big(\mathbf{y}, f\big(\mathbf{s}_{\boldsymbol{O}_t}(\phi); \theta\big)\Big)$ and gradients $\nabla_\theta \ell$ and $\nabla_\phi \ell$ using (11) and (7).
    Take gradient steps for ϕ and θ.
  end for
end while

$\mathbf{s}_t$ provides full information to both the feature selection policy and the prediction model. We encode a partially observed image $\mathbf{x} \in \mathbb{R}^{c \times H \times W}$ with c channels into dimension $\mathbb{R}^{5c \times H \times W}$ where each of the 5 chunks represent the feature value $x_j \cdot \mathbf{z}_{\cdot,j} + \hat{x}_j \cdot (1 - \mathbf{z}_{\cdot,j})$ (observed or imputed), the observation mask $\mathbf{z}_{\cdot,j}$, the mean μ_j, standard deviation σ_j, and utility u_j for each of the c channel features. For the mean imputation scheme, there will be only two chunks containing the feature values and the observation masks. Using these spatial representations for image datasets, we can leverage 2-D convolutional layers often used in image classification models. We provide specific network architecture choices for different datasets in their respective experimental section.

Dealing with Categorical Features In (9), we assume the features are real-valued when masking the feature d using $x_d \odot \mathbf{z}_{t,d}$ operation. Even though the masked value is 0 for features that are not yet acquired, this representation lets us compute the gradient w.r.t. $\mathbf{z}_{t,d}$ which we can pass to the policy parameters. To handle categorical features, we first convert them into dense features using embedding layers before applying the masking operations. We represent categorical features as $\mathcal{E}(x_d) \odot \mathbf{z}_{t,d}$ where $\mathcal{E}$ is the embedding layer and $\odot$ multiplies the scalar $\mathbf{z}_{t,d}$ with each dimension in the vector $\mathcal{E}(x_d)$. Thus, although unobserved categorical features are represented using zero vectors, we can compute the gradient w.r.t. $\mathbf{z}_{t,d}$. We handle $\hat{x}_d$ for categorical features similarly, where we sample the most likely feature value using the imputation model h. The $\frac{\partial \ell}{\partial \mathbf{s}_{t,j}} \cdot (x_j - \hat{x}_j)$ term in (11) for valid categorical features becomes $\sum_l \frac{\partial \ell}{\partial \mathbf{s}_{t,j,l}} \cdot (\mathcal{E}(x_j)_l - \mathcal{E}(\hat{x}_j)_l)$, where l is the embedding dimension, and the additional subscript $\mathbf{s}_{t,j,l}$ can be viewed as multiple co-ordinates for the embedding of the categorical feature.

Experiments

In this section, we detail our experimental setup, model implementation, and experimental results on real-world public datasets. The number of actions, i.e., the feature dimension D, is key to learning a feature acquisition policy; large values of D can be challenging to handle in practice. Prior work mostly used synthetic datasets and smaller real-world datasets such as UCI datasets and the MNIST dataset down-sampled to $D = 16 \times 16$ (Li and Oliva 2021). The reason for this choice is computational: the time complexity of the EDDI method grows as $\mathcal{O}(D^2)$ and the RL methods used for JAFA and GSMRL methods are empirically slow to converge when the number of actions (D) increases (Freeman et al. 2021). We show in our experiments that DiFA, due to its differentiable nature, is more scalable to much larger datasets, such as CIFAR where the feature dimension D is 3072, than existing methods.

Dataset	K	EDDI	JAFA	GSMRL	DiFA
MNIST	16	0.2474	0.3019	0.8794	**0.9038**
	32	0.5306	0.9694	0.9641	**0.9701**
Fashion-MNIST	32	0.1035	0.8507	0.8435	**0.8713**
	64	0.1111	0.8758	0.878	**0.9069**
SVHN	64	0.2962	0.902	0.8726	**0.9107**
	128	0.5489	0.9186	0.9057	**0.9445**
CIFAR10	64	0.1802	0.5651	0.5485	**0.5712**
	128	0.2646	0.6751	0.6597	**0.7072**
Grid	4	0.691	0.7889	**0.793**	0.792
	6	0.7448	**0.9007**	0.8979	0.8985
	8	0.7922	**0.968**	0.9658	0.9633
Parkinson (-MSE)	6	-0.9272	-0.6586	**-0.6523**	-0.6595
	8	-0.8844	-0.6342	-0.6202	**-0.6038**
	10	-0.8354	-0.5971	-0.5967	**-0.5831**
PhysioNet-Mortality (F1)	4	0.3384	**0.4806**	0.4735	0.4805
	8	0.374	0.4968	**0.4989**	0.4952
	12	0.4103	0.5084	0.5056	**0.51**

Table 1: Average metrics (accuracy is the default metric on the first five datasets) for all models. We defer the standard deviation numbers in Figure 2 to the supplementary material.

Setup and Baselines

We compare our method, DiFA, with the JAFA (Shim, Hwang, and Yang 2018), GSMRL (Li and Oliva 2021), and EDDI methods (Ma et al. 2019). We repeat each experiment **five** times with different random seeds and list/plot the **mean** and the **standard deviation** (std) numbers; we list the **p-values** in the supplementary material. We implement all methods in PyTorch and run our experiments in a single NVIDIA 2080Ti GPU. Our implementation will be publicly available at https://github.com/arghosh/DiFA.

We take a unifying approach for preprocessing, network architecture, and training process for all the methods considered in this paper. The reason is that we observe that minor changes in preprocessing, network architecture, training process, and even the base RL algorithm can cause significant changes in the performance of the same method. Since we use both image and static datasets, we start with the overall setup first. We detail the network architecture specifics of the dataset, in their respective subsections. We list the hyperparameters in the supplementary material.

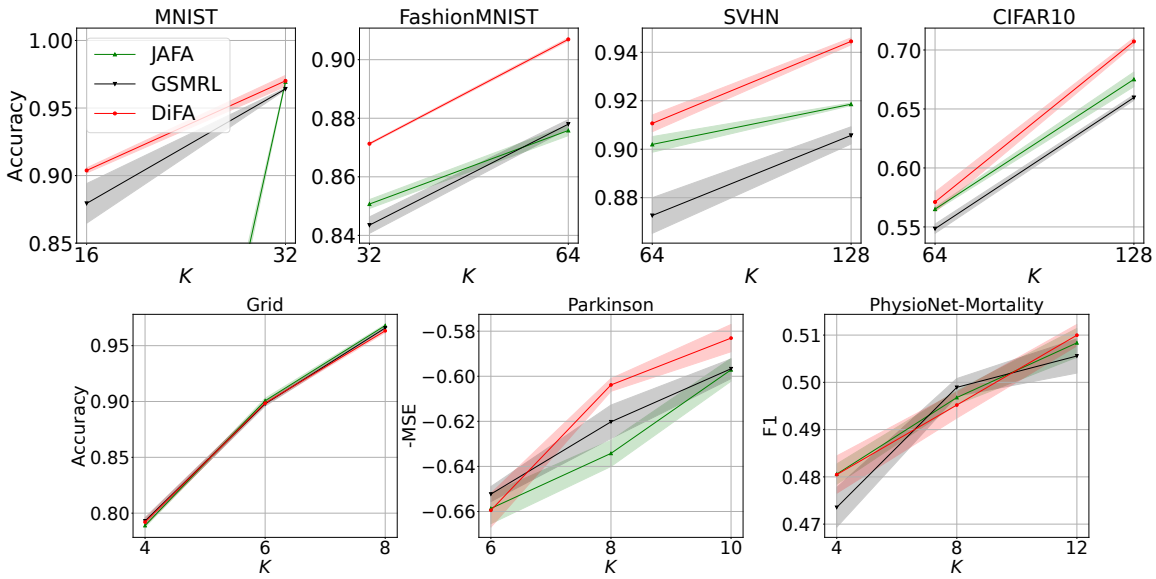

Figure 2: Visualization of average metrics (lines) and standard deviations (error bars) for learnable feature acquisition policies.

Preprocessing and the imputation model: All existing methods and our method, except JAFA, use an imputation model. We use the exact *same* imputation model based on VAEAC for all of these methods. DiFA uses the imputation model to augment side information and to impute the missing input feature values in a differentiable fashion in (9). EDDI uses the imputation model to impute the missing features. GSMRL uses the imputation model to augment side information and leave the missing feature values as 0. JAFA does not use the imputation model and keeps the missing feature value as 0. We perform standard Gaussian normalization for all the real-valued features.

Prediction model and the acquisition policy: For all methods, we pre-train the prediction model for a fixed number of epochs. We use the same network architecture for all methods. We select a subset of the features to be masked uniformly random across all features. Since the distribution of the uniform random mask is different from the mask for the selected features, we further jointly fine-tune the prediction model along with the feature selection policy with a smaller learning rate. The EDDI method takes $\sim$ 7 days ($\mathcal{O}(D^2)$) just to perform inference on the test set of the CIFAR10 dataset. Thus, for the EDDI method, we fixed the pre-trained prediction model and perform inference without further training. The RL policy for the JAFA and GSMRL methods and the differentiable policy for the DiFA method have the exact same network architecture.

Base RL algorithm: JAFA and GSMRL train the feature selection policy using RL-based algorithms. The former uses Double DQN (Hasselt 2010) while the latter uses PPO (Schulman et al. 2017). Given the success of PPO methods for stabilizing RL training, we use PPO algorithms for both of these methods. In fact, we observe that JAFA performs significantly better than previously reported (Li and Oliva 2021) when using PPO as the base RL algorithm. Since we use a fixed number of acquired features for all methods, we do not use any feature acquisition costs; instead, we use negative log-likelihood as the reward at the end of each RL episode. Moreover, for the GSMRL method, we provide intermediate rewards following prior work (Li and Oliva 2021).

Image Classification

We use the following four datasets for image classification experiments: MNIST ($C = 10$, $n = 70K$, $c = 1$, $D = 1 \times 28 \times 28$) (LeCun et al. 1998), FashionMNIST ($C = 10$, $n = 70K$, $c = 1$, $D = 1 \times 28 \times 28$) (Xiao, Rasul, and Vollgraf 2017), SVHN ($C = 10$, $n = 100K$, $c = 3$, $D = 3 \times 32 \times 32$) (Netzer et al. 2011), and CIFAR10 ($C = 10$, $n = 70K$, $c = 3$, $D = 3 \times 32 \times 32$) (Krizhevsky, Hinton et al. 2009), where C, D, n, and c are the number of classes, the image dimension (channels, height, and width), number of samples, and number of input channels, respectively.

We use the ResNet18 architecture (He et al. 2016) as the prediction model for all of these image datasets. We change the kernel size to 3 for the first convolutional layer, following prior work (Chen et al. 2020). JAFA and EDDI use raw pixel values as features (true value or masked value) and the feature mask in their prediction models, whereas DiFA and GSMRL augment the prediction model with side information (mean, standard deviation, and expected information gain). Thus, the first convolutional layer of ResNet18 has $2c$ and $5c$ channels for these two cases, respectively, depending on whether side information is present. We also use a ResNet-style architecture with five convolutional layers as the model for the feature selection policy. For DiFA and GSMRL, the side information $\mathbf{a}_t$ uses estimated logits $\hat{\mathbf{y}}$ from the imputation model. We concatenate the output logits $\hat{\mathbf{y}}$ with the output from the convolutional layers; JAFA does not use any side information, and the convolutional layers use spatial features $\in \mathbb{R}^{2c \times H \times W}$. We finally use a single hidden layer

for the actor layer and the critic layer for RL-based methods, JAFA and GSMRL. We also use a ResNet-style architecture for the VAEAC imputation model. We use five convolutional layers to encode the image, five deconvolutional layers to decode the image, and two linear layers to compute the latent variables.

Results and Discussion In Table 1, we list the mean accuracies across all random seeds for various numbers of total acquired features (K) on the MNIST, FashionMNIST, SVHN, and CIFAR10 datasets. The static feature selection policy using a heuristic criterion, EDDI, does not perform well, with a performance that falls way below other methods on every dataset and for all values of K. In Figure 2, we plot the means (as lines) and stds (as error bars) of the remaining models (without EDDI) to visualize the statistical significance of the results; we list p-value numbers in the supplementary material.

On the MNIST dataset, DiFA outperforms other methods by 0.5%-2%. JAFA performs slightly better than GSMRL for $K = 64$; however, we observe that JAFA converges to suboptimal solutions for $K = 16$ for all 5 random seeds. We believe that the reason is that the auxiliary information provided by the imputation module is more helpful when the number of total acquired features, K, is smaller. We observe a higher variance in the MNIST dataset, especially for small values of K. On the FashionMNIST dataset, DiFA outperforms other methods by 2%-3% for all values K. Interestingly, DiFA with $K = 32$ reaches the same predictive performance as GSMRL and JAFA with $K = 64$, which means a 50% sample efficiency improvement on the FashionMNSIT dataset. JAFA and GSMRL perform similarly on the FashionMNSIT dataset as well. On the SVHN dataset, DiFA outperforms other methods by 1%-4%. Similar to the FashionMNIST dataset, we observe that DiFA requires 50% fewer features to reach similar predictive performance on the SVHN dataset. On the CIFAR10 dataset, DiFA outperforms other methods by 1%-6%. Overall, on all the image classification datasets, DiFA outperforms other methods; in most cases, the performance gain is statistically significant, as evident in Figure 2. We also observe that EDDI performs worse than other methods by a large margin. JAFA and GSMRL perform relatively the same for larger values of K.

Other Supervised Tasks

We use two other supervised datasets from the UCI repository (Asuncion and Newman 2007): the Grid dataset (a binary classification task with $n = 10K$, $D = 12$) and the Parkinson dataset (a regression task with $n = 5K$, $D = 16$) where n and D are the number of data points and feature dimension, respectively. We use negative mean squared error (MSE) as the metric for the regression task. We also experiment with the in-hospital mortality task (an imbalanced binary classification task with $n = 12K$, $D = 41$) from the PhysioNet 2012 challenge (Goldberger et al. 2000). Since the PhysioNet dataset is highly class-imbalanced, we use weighted cross-entropy loss for feature selection policy and prediction model learning, following prior work (Li and Oliva 2021). Moreover, we use the F1 score as the evaluation metric for the PhysioNet dataset. Note that PhysioNet contains on average 10 unobserved features per data point; we select features from only the observed features. The prediction model has three fully connected layers with skip connections, dropout regularizer, and LeakyReLU activation function. The input layer maps real-valued features to a dense vector and concatenates the embeddings of the categorical features to another dense vector. Finally, we pass these two dense vectors along with the mask and auxiliary side information to the prediction model. The feature selection policy model has the same network architecture as the prediction model. The VAEAC imputation model uses a skip connection with three fully connected layers to encode the features and three fully connected layers to decode the latent variables. The input layer of the VAEAC model concatenates two dense latent vectors, one for real-valued features, and one for categorical features, along with the mask, similar to the prediction model.

Results and Discussion In Table 1, we list the mean metrics across all random seeds for various numbers of acquired features (K) for the Grid, Parkinson, and PhysioNet 2012 Challenge datasets. Similar to image datasets, the static policy, EDDI, does not perform well in any setting. In Figure 2, we plot the means (as line plot) and stds (as error bars) for the remaining models to visualize the statistical significance of the results. We observe that JAFA, GSMRL, and DiFA perform similarly on the UCI datasets (Grid and Parkinson) and PhysioNet datasets since they are relatively small. In Figure 2, we see that only on the Parkinson dataset, for $K = \{8, 10\}$, DiFA has a statistically significant advantage over other methods. In every other case, there is no clear winner among all methods. We emphasize that this result is as expected since RL methods are asymptotically optimal; since these datasets are small, for small values of K, RL can often learn the optimal feature selection policy, unlike on larger image datasets where they often cannot do so.

Conclusions and Future Work

In this paper, we proposed DiFA, a differentiable feature acquisition method that uses first order information on the prediction loss reward to optimize the feature acquisition policy parameters. Through extensive experiments on real-world image classification and supervised learning datasets, we demonstrated that DiFA can significantly outperform existing reinforcement learning-based methods and informativeness-based heuristics for feature acquisition. DiFA expands the horizon of feature selection to scenarios where the number of features is an order of magnitude larger than what existing methods can manage. There are numerous avenues for future work, including extending DiFA to temporal and spatio-temporal data. For temporal datasets (e.g., videos), we need to dynamically decide whether to acquire some features in each time step, which requires changing the objective in (2) to handle temporal dynamics in the data. We also note that on datasets where the number of features is very large, such as ImageNet ($D \sim 150K$), feature acquisition remains challenging. We believe that differentiable physics engines combined with DiFA can be a promising solution (Freeman et al. 2021; de Avila Belbute-Peres et al. 2018).

References

Asuncion, A.; and Newman, D. 2007. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Brinker, K. 2003. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, 59–66.

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

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

de Avila Belbute-Peres, F.; Smith, K.; Allen, K.; Tenenbaum, J.; and Kolter, J. Z. 2018. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31.

Freeman, C. D.; Frey, E.; Raichuk, A.; Girgin, S.; Mordatch, I.; and Bachem, O. 2021. Brax–A Differentiable Physics Engine for Large Scale Rigid Body Simulation. *arXiv preprint arXiv:2106.13281*.

Ghadimi, S.; and Lan, G. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4): 2341–2368.

Ghosh, A.; and Lan, A. 2021. BOBCAT: Bilevel Optimization-Based Computerized Adaptive Testing. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2410–2417. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Ghosh, A.; Mitra, S.; and Lan, A. 2022. DiPS: Differentiable Policy for Sketching in Recommender Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 6703–6712.

Goldberger, A. L.; Amaral, L. A.; Glass, L.; Hausdorff, J. M.; Ivanov, P. C.; Mark, R. G.; Mietus, J. E.; Moody, G. B.; Peng, C.-K.; and Stanley, H. E. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation*, 101(23): e215–e220.

Gong, W.; Tschiatschek, S.; Nowozin, S.; Turner, R. E.; Hernández-Lobato, J. M.; and Zhang, C. 2019. Icebreaker: Element-wise efficient information acquisition with a bayesian deep latent gaussian model. *Advances in neural information processing systems*, 32.

Guyon, I.; and Elisseeff, A. 2003. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar): 1157–1182.

Hasselt, H. 2010. Double Q-learning. *Advances in neural information processing systems*, 23.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Huang, S.-J.; Xu, M.; Xie, M.-K.; Sugiyama, M.; Niu, G.; and Chen, S. 2018. Active feature acquisition with supervised matrix completion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1571–1579.

Ivanov, O.; Figurnov, M.; and Vetrov, D. 2018. Variational autoencoder with arbitrary conditioning. *arXiv preprint arXiv:1806.02382*.

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

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

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

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

Lewis, D. D.; and Gale, W. A. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, 3–12. Springer.

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

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

Ma, C.; Tschiatschek, S.; Palla, K.; Hernandez-Lobato, J. M.; Nowozin, S.; and Zhang, C. 2019. EDDI: Efficient Dynamic Discovery of High-Value Information with Partial VAE. In *International Conference on Machine Learning*, 4234–4243. PMLR.

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

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937. PMLR.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Mohamed, S.; Rosca, M.; Figurnov, M.; and Mnih, A. 2020. Monte Carlo Gradient Estimation in Machine Learning. *J. Mach. Learn. Res.*, 21(132): 1–62.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading Digits in Natural Images with

Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

Qayyum, A.; Qadir, J.; Bilal, M.; and Al-Fuqaha, A. 2020. Secure and robust machine learning for healthcare: A survey. *IEEE Reviews in Biomedical Engineering*, 14: 156–180.

Rückstieß, T.; Osendorfer, C.; and Smagt, P. v. d. 2011. Sequential feature selection for classification. In *Australasian joint conference on artificial intelligence*, 132–141. Springer.

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

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Settles, B. 2009. Active learning literature survey. University of Wisconsin-Madison Department of Computer Sciences.

Shim, H.; Hwang, S. J.; and Yang, E. 2018. Joint active feature acquisition and classification with variable-size set encoding. *Advances in neural information processing systems*, 31: 1368–1378.

Siemens, G. 2013. Learning analytics: The emergence of a discipline. *American Behavioral Scientist*, 57(10): 1380–1400.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

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

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

Zubek, V. B.; and Dietterich, T. G. 2002. Pruning Improves Heuristic Search for Cost-Sensitive Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 19–26.