Towards Efficient and Domain-Agnostic Evasion Attack with High-Dimensional Categorical Inputs

Hongyan Bao¹, Yufei Han², Yujun Zhou¹, Xin Gao¹, Xiangliang Zhang⁵,¹,*
¹King Abdullah University of Science and Technology
²INRIA
³University of Notre Dame
hongyan.bao@kaust.edu.sa, yfhan.hust@gmail.com, yujun.zhou@kaust.edu.sa, xin.gao@kaust.edu.sa, xzhang33@nd.edu

Abstract
Our work targets at searching feasible adversarial perturbation to attack a classifier with high-dimensional categorical inputs in a domain-agnostic setting. This is intrinsically a NP-hard knapsack problem where the exploration space becomes explosively larger as the feature dimension increases. Without the help of domain knowledge, solving this problem via heuristic method, such as Branch-and-Bound, suffers from exponential complexity, yet can bring arbitrarily bad attack results. We address the challenge via the lens of multi-armed bandit based combinatorial search. Our proposed method, namely FEAT, treats modifying each categorical feature as pulling an arm in multi-armed bandit programming. Our objective is to achieve highly efficient and effective attack using an Orthogonal Matching Pursuit (OMP)-enhanced Upper Confidence Bound (UCB) exploration strategy. Our theoretical analysis bounding the regret gap of FEAT guarantees its practical attack performance. In empirical analysis, we compare FEAT with other state-of-the-art domain-agnostic attack methods over various real-world categorical data sets of different applications. Substantial experimental observations confirm the expected efficiency and attack effectiveness of FEAT applied in different application scenarios. Our work further hints the applicability of FEAT for assessing the adversarial vulnerability of classification systems with high-dimensional categorical inputs.

Introduction
Adversarial evasion attacks have been witnessed in many real-world data analytical applications (Goodfellow, Shlens, and Szegedy 2014; Cartella et al. 2021; Suciu, Coull, and Johns 2019; Stringhini, Kruegel, and Vigna 2010; Imam and Vassilakis 2019), including text processing (Yang et al. 2020; Papernot et al. 2016) and image recognition (Goodfellow, Shlens, and Szegedy 2014; Szegedy et al. 2013). Despite the flourish efforts on evasion attacks with continuous inputs, such as image and video contents (Goodfellow, Shlens, and Szegedy 2014; Szegedy et al. 2013; Carlini and Wagner 2018; Biggio, Nelson, and Laskov 2012), much less attention has been paid to explore the adversarial threat against on machine learning systems with categorical inputs. Categorical data exist prevalently in real-world trust-critical applications, like cyber attack detection (Shu et al. 2020; Wang 2017; van Ede et al. 2022) and medical diagnosis. For instance, detecting cyber attacks usually depends on categorical behavioral signatures of the target IT infrastructures, including malware execution traces, malicious network communication logs, and system event logs (van Ede et al. 2022; Pendlebury et al. 2019). Machine Learning-based medical diagnosis is often conducted by combining and encoding qualitative results of various medical tests. Unlike continuous measurements such as pixel intensities, each categorical feature is valued with mutually exclusively category values. These optional category values have no intrinsic ordering. Conducting adversarial perturbations on categorical features is therefore in nature an NP-hard knapsack problem (Wang et al. 2020). On one hand, popular gradient-guided evasion attack methods against deep learning models (Goodfellow, Shlens, and Szegedy 2015) become infeasible as computing gradients directly over categorical variables is not applicable. On the other hand, classic heuristic search solutions, e.g., Branch-and-Bound and trial-and-error methods, suffer from high complexity and lack guaranteed quality of the derived attack results, which can lead to arbitrarily bad attack performances. It is therefore difficult to define a computationally efficient strategy to produce effective adversarial perturbations over categorical inputs.

The current study in solving the adversarial attack problem over categorical inputs falls into two groups. First, domain-specific knowledge is applied to narrow down the combinatorial perturbation space, and used as constraints to preserve semantic/function integrity of the perturbed instances (Li et al. 2020; Zhang et al. 2020; Gao et al. 2018; Li et al. 2018; Jin et al. 2020; Samanta and Mehta 2017; Papernot et al. 2016; Ma et al. 2018a; Wang et al. 2020; Suciu, Coull, and Johns 2019; Narodytska and Kasiviswanathan 2017; Croce and Hein 2019; Pierazzi et al. 2020). Such domain-specific dependency limits the adaptive potential of the attack method across different applications. Moreover, domain-specific knowledge may not be always readily available. For example, the threat settings of cyber attacks vary drastically across different attack techniques and IT system architectures (van Ede et al. 2022). Encoding domain-specific contexts of various intrusion incidents require expensive investigation overheads on a case-by-case basis. Besides, system threats may stay unknown to security analysts when an attack is delivered. It is impossible to define domain-specific rules for the zero-day attack events. The absence of a principled and domain-agnostic adversarial attack protocol...
makes it difficult to provide an attack-as-a-service pipeline to evaluate the adversarial vulnerability of different trust-critical applications. Second, forward stepwise greedy search (FSGS) has been adopted in (Ebrahimi et al. 2018; Wang et al. 2020) as a domain-agnostic method to generate feasible adversarial modifications to categorical data. Domain-specific constraints over the feasible adversarial modifications can be used as a plug-in to FSGS. However, the greedy search method induces prohibitively expensive computational cost as the number of categorical features and/or the optional category values in the target input become large. The intense overheads prevent the adversary from organizing efficient attacks and/or makes it inapplicable to assess the adversarial vulnerability of a target machine learning system in practices.

To address the limits of current study, we propose an orthogonal matching pursuit (OMP)-boosted multi-armed bandit search to deliver a fast and effective adversarial attack in a high-dimensional combinatorial search space, named as FEAT hereafter. FEAT adopts orthogonal matching pursuit (Elenberg et al. 2018; Wang et al. 2020) to identify the most sensitive categorical features to perturb in each round of the iterative attack process. Over the selected candidate features, FEAT considers modifying each categorical feature as triggering an arm in a multi-armed bandit game. Exploring the feasible combinations of categorical feature modifications can thus be guided with Upper Confidence Bound (UCB)-driven exploration in a computationally efficient way. The advantages of FEAT are summarized as follows.

• **Computationally-economic attack with high-dimensional categorical inputs.** The computational complexity of FEAT is linear to the number of modified features. In contrast, the complexity of the state-of-the-art domain-agnostic attack methods proposed by (Qi et al. 2019) and (Wang et al. 2020) grow as a geometric series of the number of modified features. Empirically FEAT costs 1/10 – 1/3 of the overheads compared to the state-of-the-art domain-agnostic and domain-specific attack methods, while requiring less features to modify to deliver highly successful attacks.

• **Theoretical guaranteed attack performance.** We set up an upper bound of the expected regret of FEAT in our analysis. It applies to a general Lipschitz-smooth deep learning-based target classifier with categorical inputs, which guarantees the attack performance of FEAT in general attack scenarios.

• **Domain-agnostic adaption to various different applications.** We evaluate FEAT over 4 categorical data sets collected from various real-world applications. The empirical observations confirm FEAT is well adapted to different application domains and show its consistently superior attack effectiveness and efficiency, comparing to the state-of-the-art domain-agnostic and domain specific attack baseline methods. The results also reconcile with the theoretical guarantee to the success of FEAT attacking general classifiers.

**Related Work**

(Wang et al. 2020; Qi et al. 2019; Yang et al. 2020; Ebrahimi et al. 2018) proposed to adopt forward stepwise greedy search (FSGS) based methods in generating discrete adversarial samples in a domain-agnostic way. FSGS is an iterative process. In each iteration, it considers all possible combination of each candidate categorical feature with the subsets of the adversarially modified features in previous rounds. FSGS then chooses the candidate feature that can achieve the largest marginal gain of the attack objective. Though FSGS plays as a domain-agnostic attack method, it can also use additional domain-specific constraints to reduce the size of feasible feature modifications. However, the bottleneck of FSGS is that its computational cost grows as a geometric series of the number of the modified features. It becomes prohibitively expensive as the dimension of the target discrete instance is high.

Domain-specific adversarial attack mostly target at text classifiers (Papernot et al. 2016; Miyato, Dai, and Goodfellow 2016; Samanta and Mehta 2017; Yang et al. 2018; Gao et al. 2018; Li et al. 2018; Jia and Liang 2017; Jin et al. 2020). (Gao et al. 2018) developed scoring functions to evaluate the importance of each word in a sentence and proposed to modify the top-ranked words identified by the scoring functions. Similarly, (Papernot et al. 2016) selected the word to replace where the variation of the word’s embedding vector is best aligned to the gradient direction of the target model. In contrast, (Jia and Liang 2017) proposed to insert distraction sentences into a target text sample with a human-involved loop to fool a reading comprehension system. (Samanta and Mehta 2017) added linguistic constraints over the pool of candidate-replacing words. Recently, TextBugger (Li et al. 2018) used typo-based perturbation for each word to get the candidates of feasible modifications over each word. TextFooler (Jin et al. 2020) used the similarity of word embedding to select the candidates of each words to attack. These methods depend on semantic/syntactic rules to shrink the feasible set of text modifications. Besides, they adopt trial-and-error search to explore possible text modifications. They lack the guarantee to the solution quality to the knapsack based discrete evasion attack problem. Their attack performances, i.e., the success of attack complying to the attack budget constraint, may thus vary drastically over different target inputs.

**Preliminaries**

Let \( x = \{ x_1, x_2, x_3, \ldots, x_N \} \) denote a discrete input instance with \( N \) categorical features. Each \( x_i \) may take any of \( M (M \geq 1) \) categorical values. We cast each optional category value of a discrete feature \( x_i \) to a \( D \)-dimensional pre-trained embedding vector, e.g., \( e_i^j \in R^D, j = 1, 2, \ldots, M \). We introduce binary indicators \( b = \{ b_i^j \}, i = 1, 2, \ldots, N, j = 1, 2, \ldots, M \), where \( b_i^j = 1 \) when the \( i \)-th categorical feature \( x_i \) takes the \( j \)-th categorical value of \( x_i \), and \( b_i^j = 0 \) otherwise. One instance \( x \) can then be represented by stacking the embedding vectors of each categorical variable \( x_i \) as an \( R^{N \times M \times D} \) tensor with \( x_{(i,j)} = b_i^j e_i^j \).

With this setting, the adversarial perturbation over \( x \) is to modify \( b \) to \( \hat{b} \). \( \hat{b}_i^j = b_i^j \) denotes \( x_i \) is not perturbed. Otherwise, \( \hat{b}_i^j \neq b_i^j \) indicates the corresponding feature \( x_i \) is changed. Depending on the type of attacks, i.e., insertion, deletion or substitution, \( \hat{b}_i^j \) can be valued in different ways. **Insertion** is to let \( \hat{b}_i^j = \hat{b}_i^j, b_i^j = 0, \forall j = 1, \ldots, m. \) **Deletion** is to let \( \hat{b}_i^j = 0, \ldots, m. \)
given \( b'_{i,j} = 1 \). 
Substitution is to let \( b'_{i,j} = 1, b'_{i,j} = 0 \), given \( b'_{i,j} = 0, b'_{i,j} = 1 \). 
A modified instance \( \mathbf{x} \) can thus be written as \( \mathbf{x}_{\{i,j\}} = b'_{i,j} \mathbf{e}'_{i,j} \). 
The classifier \( f \) outputs decision confidence as \( f_{y_k}(k = 1,2,3,...,K) \) with respect to different class labels. 
Without loss of generality, let \( y_k \) denote the true class label of \( x \) and all the other \( y_k \) \( k = \{1,...,K-1\} \) are the potential targets of an evasion attack. 
Given an input \( x \), the goal of evasion attack is to increase the misclassification risk of \( x \). 
Inheriting the terms used in Eq.1, the Upper Confidence Bound (UCB) score of each candidate is analogous to selecting \( \Lambda = 1 \) discrete feature can be computed following Eq.(3): 
\[
\Lambda = 1 \text{ rewards. In practices, we set } \Lambda \text{ added to Eq.2 to ensure the non-negativeness of the received rewards. Similarly, taking an action to modify the cat-
\]
\[
\text{The parameter } \alpha \text{ is tunable to make a trade-off between exploration and exploitation of the search for discrete feature perturbations. On the other hand, an extremely small } \alpha \text{ drives the search to lean more towards the highly sensitive features. Modifying any of these features can cause drastic variation of } f \text{'s decision. We traverse different choices of } \alpha \text{ in FEAT to empirically observe the impact of } \alpha \text{ over the attack performance. According to Theorem.2 in (Aubidert, Munos, and Szepesvári 2007), choosing the UCB score as in Eq.3 ensures that the event of drawing sub-optimal candidate features in the attack process has a decreasingly smaller probability after increasingly more iterations of search. The adversary then conducts the UCB-guided exploration of feasible discrete perturbations to avoid exhaustive search over all the possible combinations of the candidate categorical features.}
\]
The popular heuristic search methods, e.g., the standard UCB and Thompson Sampling (TS), share similar computational complexity according to (Agrawal and Goyal 2013; Auer, Cesa-Bianchi, and Fischer 2002). In the high-dimensional feature space, applying these methods directly is prone to fast increasing of the computational cost as the number of the categorical features (noted as \( N \)) and/or the number of the optional categorical values per feature (noted as \( M \)) become higher (Agrawal and Goyal 2013; Auer, Cesa-Bianchi, and Fischer 2002). Besides, the regret of TS does not scale polynomially in the feature dimension. TS can perform strictly worse than random choice in the high dimensional case (Zhang and Combes 2021). The complexity bottleneck of both meth-
ods motivates us to adopt a more efficient strategy adapted to the high-dimensional search problem. Previously, (Wang, Audibert, and Munos 2008) chooses to randomly sample a subset of features to perform the UCB subroutine to reduce the cost. However, blindly sampling subsets of features may miss effective feature perturbations that bring large variation of the classifier’s output, eventually hurting the attack performance, as shown in (Wang, Audibert, and Munos 2008).

In the proposed FEAT method, we aim to optimize the balance between the exploration coverage and the computational overheads by boosting the UCB-guided search with an orthogonal matching pursuit (OMP)-based feature ranking strategy (Buchbinder et al. 2014; Wang et al. 2020). Each iteration of the search is composed by first conducting the OMP computing to rank the candidate categorical features according to their influence over the classifier’s decision given the current input instance (see Algorithm 1 Line 3-5) and then performing $\tau$ rounds of UCB search as the inner iterations over the top $L$ influential features selected by the OMP computation (see Algorithm 1 Line 6-15). $\tau$ is a tunable parameter, adjusting the number of search rounds within the selected top $L$ features. We choose $\tau$ empirically to 1/3 of the attack budget, which presents consistently good attack success rate with low attack budget cost.

To perform the OMP-based feature ranking, we relax the binary indicators $b_i$ attached to each categorical feature $x_i$ to be continuous and valued within the range $[0,1]$. We then take the gradient of $f(x,b)$ with respect to $b$, denoted as $\nabla_b(f(x,b))$. The selected candidate categorical features are those with the largest gradient magnitudes $||\nabla_b(f(x,b))||$. We reason the rationality of using the OMP-boosted UCB search by establishing the theoretical study first in Theorem 1. The theoretical study shows the OMP-based ranking can select highly influential features over the classifier’s decision. Furthermore, we build the regret bound of the OMP-boosted UCB search of FEAT in Theorem 2, which further explains the merit of the OMP-ranking in enhancing the efficiency of UCB search. The regret analysis in Theorem 2 also provides a theoretical guarantee to the attack performance of FEAT against a general classifier with categorical inputs. We state the two theorems in the followings. Via the theoretical study, we discuss further how the the designed balance between exploration and exploitation is achieved in FEAT.

The reward distribution may drift during the attack process, which poses a challenge of non-stationary rewards to the practitioners of FEAT. We adapt the UCB-based search to the scenario from two aspects. **On the hand.** The OMP operation of FEAT restricts the search range to the potentially sensitive features. The rewards of those sensitive features remain relatively more stable than the rest features within a few iterations of search. **On the other hand,** FEAT re-initialise the UCB-based search every $\tau$ rounds and recompute the OMP-based feature ranking. Via this way, the UCB-based search is conducted only within the consecutive $\tau$ inner iterations. The reward distribution of the selected $L$ sensitive features can be considered approximately stationary within the $\tau$ inner iterations. Though lack of proof, FEAT provides an empirically feasible environment for the use of the UCB-based search.

### The Indicator of Feature Importance

**Theorem 1 Gradient as an Indicator.** Let $b$ indicate the category value assignment of an unperturbed data instance $x$. $b$ and $b'$ indicate the two different sets of the modification over the same unperturbed input $x$. We further assume $|\text{diff}(b,b')| \leq \zeta |\text{diff}(b,b')| \leq \zeta$. Given a smooth target classifier $f$ with a finite Lipschitz constant, $f_{y_k}(x)$ denotes the decision output of $f$ over any incorrect class label, i.e. $y_k \neq y_k$. Let $\nabla f_{y_k}(b)$ denote the elements of $\nabla f_{y_k}(x,b)$ corresponding to the difference between $b$ and $b'$, where $\nu = \text{diff}(b,b')$.

$$|f_{y_k}(x,b') - f_{y_k}(x,b)| \leq \max\left\{ \frac{1}{2m_k \zeta} \left| \nabla f_{y_k}(x,b) \right|^2, \frac{\left| \nabla f_{y_k}(x,b) \right|_2 + M_k \zeta}{\zeta} \right\}$$

where $m_k$ and $M_k, \zeta$ are the local strong convexity factor and local Lipschitz constant of the target classifier $f$ around the unperturbed input $x$.

### The Expected Regret Bound of FEAT

**Definition 2** For one feature $l$ out of the top-ranked $L$ features, the expected regret of perturbing $l$ is

$$\Delta_l = \mu^* - \mu_l \quad (\mu^* = \max_{1 \leq i \leq L} \mu_i)$$

where $\mu_l$ and $\mu^*$ are the expected and optimal reward received by changing $l$.

**Theorem 2** Perturbing highly sensitive features helps shrink the regret bound of FEAT. Let $\Delta_l > 0$ and $\delta_l^2 > 0$ be the expected regret and the expected variance of the rewards received by modifying each of the top-$L$ candidate features. The expected regret bound of FEAT after $T$ iterations can be given as:

$$\mathbb{E}[R_T] \leq \sum_{l=1}^{L} \left[ 8 \frac{\delta_l^2}{\mu_l} + 2 \log T + \frac{\alpha}{\alpha - 2} \Delta_l \right],$$

where $\mathbb{E}[R_T] = \sum_{l=1}^{L} \mathbb{E}[T_l] \Delta_l$. $T_l$ is the number of times that feature $l$ is selected after $T$ iterations.
Empirically, we observe that the variances $\delta^2_l$ of the received rewards for each top-ranked sensitive features remain low in the attack process. It is possible that these highly sensitive features play important roles in classification. Any modification over such features (e.g., switching the category value of these features) thus produces consistently large change to the classification output (i.e., obtaining consistently high rewards). As a result, the variance of the reward obtained by perturbing these features is low.

According to Theorem 2, as $\delta^2_l$ is generally small, the lower regret for each selected feature ($\Delta_i$) in the attack leads to a lower regret bound of FEAT. Intuitively, FEAT tends to search for feasible perturbation within the highly sensitive features via the OMP-based feature ranking step. These features have a significantly lower $\Delta_i$ compared to the rest. Perturbing these features then help FEAT reduce the expected regret, which guarantees in theory a more successful attack after $T$ iterations of exploration. Compared to randomly sampling subsets of features in (Auer, Cesa-Bianchi, and Fischer 2002; Wang, Audibert, and Munos 2008), FEAT is destined to achieve a better balance between narrowing down the search range to save the computational overheads and maintaining the attack effectiveness.

**Experimental Evaluation**

We use 4 categorical datasets with high-dimensional combinatorial search space ($N$ and/or $M$ are large) to measure the effectiveness and efficiency of FEAT. They are collected from diverse real-world applications, including text classification, cyber intrusion and digital health service.

**Yelp-5 (Yelp)** (Asghar 2016). The Yelp-5 dataset was obtained from the Yelp Review Dataset Challenge in 2015. We use 650K training and 50K testing samples with the classes from 1 star to 5 stars for training and testing a classifier. Each word is encoded as categorical feature with a 300-dimensional embedding vector.

**Intrusion Prevention System Dataset (IPS)** (Wang et al. 2020). Collected by a cyber-security vendor, the IPS dataset contains 242,467 instances of network attack reports. Each is composed by a sequence of 20 incident logs and categorised into any of the 3 threatening levels (‘common’, ‘intermediate’ and ‘urgent’). At each log, the adversary may choose to replace it with 1,103 candidate logs. We randomly select 80% of the IPS data for training and rest for testing.

**Windows PE Malware Detection (PEDec)**. PEDec consists of dynamic analysis reports of 20,000 benignwares and 10,972 PE malwares. The malware samples of 152 families are randomly selected from those submitted to VirusTotal between 2018 and 2020. Each of the executables is classified as benign or malicious by more than 21 antivirus engines. In our work, each executable of the dataset is encoded into a binary feature vectors with 5000 signatures selected by human experts. We randomly select 80% of each dataset for training and others for testing.

**Electronic Health Records (EHR)** (Ma et al. 2018b). The EHR dataset consists of time-ordered medical visit records of 7,314 patients. Each patient has from 4 to 200 medical visits. Each visit record is composed by a subset of 4,130 categorical ICD9 diagnosis codes. Each diagnosis code represents occurrence of a disease, a symptom, or an abnormal finding. Using the historical EHR data of patients, our target is to predict whether a patient will suffer heart failure disease in the future. We randomly select 80% of the EHR data for training and others for testing. Each EHR data instance is organized as a tensor $x \in \mathbb{R}^{2000 \times 4130 \times 70}$ with each of the 4130 diagnosis codes projected to a 70-dimensional embedding vector. For the patients with less than 200 visits, we pad the empty observations by setting the corresponding $b_i = 0$.

For each of IPS, EHR and PEDec datasets, we choose randomly 80% of the dataset to train the target classifier. The rest 20% of the data instances are used to test attack performances. On Yelp data, we choose 650k text instances for training the target text classification model. The rest 50k instances are used for performing different attacks and evaluating the attack performances. For PEDec dataset, we adopt a simple CNN model composed of one convolution layer followed by two linear layers. The rest datasets contain sequential instances, we thus apply standard LSTM as the classifier. Without loss of generality, we use ReLu activation function in both the CNN and LSTM classifier with the dropout module. We conduct all the experiments on Linux server with 2 GPUS (GeForce 1080Ti) and 16-core CPU (Intel Xeon). Implementations of the experiments are available at https://github.com/xmudinfc/FEAT.

**Performance Benchmark**

We include three state-of-the-art domain-agnostic attack methods as the baselines.

**FSGS** (Elenberg et al. 2018). FSGS is a greedy search-based method. In each round, it traverse the combination of each candidate feature with each subset of the already modified features. The candidate feature bringing the highest value of the attack objective function is selected to modify. FSGS only needs to query the target classifier $f$ to obtain the decision confidence. It is thus a black-box attack method.

**OMPGS** (Wang et al. 2020). It is a white-box extension of FSGS by adopting the OMP-based ranking to constraint the greedy search within the top-ranked features in each round of the attack process.

**GradAttack** (Qi et al. 2019). GradAttack is a white-box evasion attack method originally proposed to generate adversarial text samples. It treats each word in a sentence as a categorical feature and uses gradients of the word embeddings to select feasible candidate words to attack. However, since it doesn’t evaluate the combination patterns composed by the candidate words and the subset of words already modified, GradAttack requires to change much more features to deliver attacks than FSGS and OMPGS (Wang et al. 2020).

**FEAT-B** is a variant of FEAT. It randomly samples $L$ out of the total $M$ features to conduct the UCB-guided search as in (Wang, Audibert, and Munos 2008). Compared to FEAT, FEAT-B does not use the OMP computing to select the top influential features. The purpose of involving FEAT-B, as well as OMPGS, into the comparative study is to demonstrate the necessity of combing the OMP-based feature ranking and the UCB-based search together in FEAT.

---

1http://www.icd9data.com/
With the similar level of We focus on showing FEAT, as an universally applicable we also show the averaged running time needed to attack one TextFooler expensive. Similarly, GradAttack is expensive to conduct if of the confidence computing operations required to deliver L becomes large. In contrast, the complexity of FEAT and changed features. In the high-dimensional problem with large number of the top-ranked candidate features selected by OMP.

In Table. 1, \( t \) denotes the overall number of iterations required to reach successful attack. \( N \) and \( M \) are the number of features of \( x \) and category values of each \( x_i \). \( L \) denotes the number of selected features to explore in GradAttack, OMPGS, FEAT-B and FEAT. In each iteration, FSGS and OMPGS only pick one feature to perturb. Therefore \( T \) equals to the total number of changed features for these two methods. OMPGS, GradAttack and FEAT all use gradient information to shrink the size of the candidate features. Then, \( L \) in the three methods denotes the number of the top-ranked candidate features selected by OMP.

As given in Table. 1, the cost of FSGS and OMPGS increases as the sum of a geometric sequence of the number of changed features. In the high-dimensional problem with large \( N \) and/or \( M \), applying FSGS and OMPGS is prohibitively expensive. Similarly, GradAttack is expensive to conduct if \( M \) becomes large. In contrast, the complexity of FEAT and FEAT-B is significantly lower, as \( L \ll N \) (only the top \( L \) ranked candidate features are considered) and it grows linearly as the number of the attack iterations increases. FEAT-B reduces to the standard UCB if \( L \) equals to \( N \). To initialize the exploration, the standard UCB needs to draw each candidate categorical feature at least once. In the high-dimensional case, perturbing each feature once can induce an expensive overhead of \( O(NM) \) to query the variation of the decision output of the target classifier by changing each feature. Compared to the standard UCB, the cost of FEAT is significantly lower.

**Overall Performance.** The results in Table. 2 illustrate that FEAT achieves generally both highly efficient computation (low Runtime and low No.query) and effective attack (high SR), comparing with the other baseline attack methods on Yelp-5, IPS, EHR and PEDec. We organize the detailed comparison results with suitable attack budgets (the maximum number of the modified features) on each dataset. We highlight the performance metrics of the proposed FEAT with bold fonts in the followings.

On Yelp-5 data, within the same attack budget, FEAT obtains very close SR level to those of the two domain-agnostic attack methods, FSGS and OMPGS. At the same time, FEAT’s No.query and Runtime are significantly lower than those of FSGS and OMPGS, showing much higher efficiency for attack. The two greedy search methods (FSGS and OMPGS) exhaustively evaluate the combination of every candidate feature and the features that have been modified in previous iterations. In contrast, FEAT avoids the exhaustive search by balancing exploring rarely modified features and exploiting the features that show consistently high influence to the classifier’s output in the search. The results validates that FEAT maintains attack effectiveness, while running in a much more efficient way.

Compared to the domain-specific baselines (TextFooler and TextBugger), FEAT achieves 10% to 40% higher SR than those of TextFooler (0.97 v.s. 0.88), TextBugger (0.96 v.s. 0.64) and GradAttack (0.96 v.s. 0.78) on one hand. On the other hand, FEAT’s Runtime is 33% and 11% of those of TextFooler and TextBugger respectively, while achieving higher SR. This indicates faster attack speed using FEAT. No.query of TextFooler and TextBugger is lower than that of FEAT. The reason is they incrementally modify words and evaluate the corresponding attack effects. On the contrary, FEAT evaluates all the candidate words at the initial step of the search, which is the origin of the increased computational overheads. However, via the initial per-word evaluation, FEAT can conduct the exploration in a more comprehensive way, which helps FEAT achieve much higher SR.

SR of FEAT-B performs worse than baselines, e.g. the two greedy search-based methods, as randomly selecting features to explore is likely to miss influential features thus cause ineffective perturbation. The comparison between FEAT-B and FSGS/OMPGS shows that locating highly influential features to perturb is the key-to-success of attack. Nevertheless, FEAT-B always has orders of magnitude lower Runtime and No.query compared to FSGS. Both downsampling of the candidate feature set and conducting the UCB search help FEAT-B avoid exhaustive search in FSGS. The result implies the benefit of heuristically shrinking down the search range in the high-dimensional feature space. The key question to deliver simultaneously fast and effective attack is thus how to identify the most influential / sensitive features, where the UCB
The proposed FEAT method explores how to deliver both effective and computationally efficient domain-agnostic adversarial attack in a high-dimensional categorical feature space. FEAT first conducts the orthogonal matching pursuit-based feature ranking to narrow down the search range to the most sensitive features. After that, FEAT performs a MAB-driven combinatorial search over the shrinked set of candidate features. Through this way, FEAT maintains the effectiveness of the adversarial perturbation, while boosting the search efficiency to reach a fast yet successful attack. The comprehensive cross-application evaluation shows the superior domain-agnostic adaptivity of FEAT to different applications than the other state-of-the-art baselines, which makes FEAT a generally applicable tool to assess the adversarial risk of different applications with high-dimensional categorical inputs. However, FEAT still needs soft decision scores of the target classifier to evaluate different search paths. Perturbation-based defense, e.g. differential privacy, may help mitigate the attack. We will thus focus on the threat model with only hard labels accessible.

**Discussion and Conclusion**

The research reported in this paper was partially supported by funding from King Abdullah University of Science and Technology (KAUST).

---

**Table 2:** Attack performances evaluated on efficiency and effectiveness metrics: The attack time limit $T_L=1000$ sec.

### (a) The results on Yelp-5 data

<table>
<thead>
<tr>
<th>Attack Type &amp; Algo.</th>
<th>Runtime (sec) ↓</th>
<th>No.query ↓</th>
<th>SR ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Specific</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextBugger</td>
<td>1.18</td>
<td>23</td>
<td>0.64</td>
</tr>
<tr>
<td>TextFooler</td>
<td>0.42</td>
<td>167</td>
<td>0.88</td>
</tr>
<tr>
<td>Black Box</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GradAttack</td>
<td>0.15</td>
<td>10000</td>
<td>0.78</td>
</tr>
<tr>
<td>OMPGS</td>
<td>1.25</td>
<td>7000</td>
<td>0.96</td>
</tr>
<tr>
<td>FEAT</td>
<td>0.14</td>
<td>887</td>
<td>0.97</td>
</tr>
</tbody>
</table>

### (b) The results on IPS data

<table>
<thead>
<tr>
<th>Attack Type &amp; Algo.</th>
<th>Runtime (sec) ↓</th>
<th>No.query ↓</th>
<th>SR ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black Box</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSFG</td>
<td>136</td>
<td>37000</td>
<td>0.80</td>
</tr>
<tr>
<td>FEAT-B</td>
<td>19.5</td>
<td>2500</td>
<td>0.74</td>
</tr>
<tr>
<td>White Box</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GradAttack</td>
<td>21.2</td>
<td>2100</td>
<td>0.59</td>
</tr>
<tr>
<td>OMPGS</td>
<td>1.99</td>
<td>127</td>
<td>0.77</td>
</tr>
<tr>
<td>FEAT</td>
<td>0.28</td>
<td>111</td>
<td>0.92</td>
</tr>
</tbody>
</table>

### (c) The results on PEDec data

<table>
<thead>
<tr>
<th>Attack Type &amp; Algo.</th>
<th>Runtime (sec) ↓</th>
<th>No.query ↓</th>
<th>SR ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black Box</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSFG</td>
<td>435</td>
<td>213256</td>
<td>0.88</td>
</tr>
<tr>
<td>FEAT-B</td>
<td>3.65</td>
<td>9959</td>
<td>0.87</td>
</tr>
<tr>
<td>White Box</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GradAttack</td>
<td>3.51</td>
<td>18563</td>
<td>0.67</td>
</tr>
<tr>
<td>OMPGS</td>
<td>360</td>
<td>27758</td>
<td>0.80</td>
</tr>
<tr>
<td>FEAT</td>
<td>2.89</td>
<td>5923</td>
<td>0.91</td>
</tr>
</tbody>
</table>

### (d) The results on EHR data

<table>
<thead>
<tr>
<th>Attack Type &amp; Algo.</th>
<th>Runtime (sec) ↓</th>
<th>No.query ↓</th>
<th>SR ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black Box</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSFG</td>
<td>482</td>
<td>58000</td>
<td>0.84</td>
</tr>
<tr>
<td>FEAT-B</td>
<td>167</td>
<td>7108</td>
<td>0.94</td>
</tr>
<tr>
<td>White Box</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GradAttack</td>
<td>2.34</td>
<td>204</td>
<td>0.94</td>
</tr>
<tr>
<td>OMPGS</td>
<td>27.5</td>
<td>35</td>
<td>0.94</td>
</tr>
<tr>
<td>FEAT</td>
<td>0.35</td>
<td>20</td>
<td>0.94</td>
</tr>
</tbody>
</table>

search is performed. The OMP-boosted UCB search of FEAT answers this question and addresses the balance between the attack efficiency and effectiveness.

On IPS and PEDec data, we can observe that FEAT consistently achieves the highest SR, significantly higher than the secondly ranked baseline. Meanwhile, Runtime and No.query of FEAT remain to be the lowest among all the attack methods. The results confirm the benefit of balancing exploration and exploitation in FEAT. On Yelp-5, IPS and PEDec data, GradAttack’s SR is less than 80% of that of FEAT. The reason is GradAttack requires more features to modify than the attack budget to deliver successful attacks over the testing inputs. Therefore, GradAttack is terminated when the number of modified features reaches the attack budget even before before it achieves successful attacks on the testing samples. On EHR data, with the similar level of the SR performance, FEAT obtains lower Runtime and No.query than other methods. Because of the sensitivity of the specific features in EHR, GradAttack and OMPGS can use the gradient of features to evaluate the importance of the features and select the most sensitive features very fast compared with the type of back-box adversarial attack. It is worth noting that the OMPGS and FEAT both use the OMP-based feature ranking to shrink the search range, the superior attack effectiveness and efficiency of FEAT confirm the merit of conducting the query-efficient UCB search over the top-ranked features, instead of the exhaustive greedy search in OMPGS. The total computational overheads of OMPGS and FEAT are composed of the cost for the OMP computation and the query evaluating the classifier’s output. OMPGS (greedy-based attack) needs to conduct significantly more OMP operations in each iteration than FEAT. Hence we can observe a much larger gap regarding the runtime measurement between OMPGS and FEAT, compared to that regarding the query number.
References


Bao, H.; Han, Y.; Zhou, Y.; and Zhang, X. 2022. Towards understanding the robustness against evasion attack on categorical inputs. In ICLR.


Goodfellow, I.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In ICLR.


Li, L.; Ma, R.; Guo, Q.; Xue, X.; and Qiu, X. 2020. Bert-attack: Adversarial attack against BERT using BERT. In EMNLP.


