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

Hongyan Bao[1], Yufei Han[2], Yujun Zhou[1], Xin Gao[1], Xiangliang Zhang[3,1,*]

[1]King Abdullah University of Science and Technology
[2]INRIA
[3]University 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; Zang 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*

[*]Corresponding author.

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 **f**ast and **e**ffective **a**dversarial a**t**tack 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 $\mathbf{x} = \{x_1, x_2, x_3, ..., x_N\}$ 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., $\mathbf{e}_i^j \in R^D, j = 1, 2, ..., M$. We introduce binary indicators $\mathbf{b} = \{b_i^j\}$, i=1,2,...,N, j=1,2,...,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 $\mathbf{x}$ can then be represented by stacking the embedding vectors of each categorical variable x_i as an R^{N*M*D} tensor with $\mathbf{x}_{\{i,j,:\}} = b_i^j \mathbf{e}_i^j$.

With this setting, the adversarial perturbation over $\mathbf{x}$ is to modify $\mathbf{b}$ to $\hat{\mathbf{b}}$. $\hat{b}_i^j = b_i^j$ 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 $\hat{b}_i^j = 1$, given $b_i^j = 0, \forall j = 1, ..., m$. *Deletion* is to let $\hat{b}_i^j = 0$, given $b_i^j = 1$. *Substitution* is to let $\hat{b}_i^j = 1, \hat{b}_i^{j'} = 0$, given $b_i^j = 0$, $b_i^{j'} = 1, j \neq j'$. A modified instance $\hat{\mathbf{x}}$ can thus be written as $\hat{\mathbf{x}}_{\{i,j,:\}} = \hat{b}_i^j \mathbf{e}_i^j$. The classifier f outputs decision confidence 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 $\mathbf{x}$, the goal of evasion attack is to increase the misclassification risk of f over $\mathbf{x}$, i.e., making $f_{y_K}(\mathbf{x},\hat{\mathbf{b}})$ as low as possible and $f_{y_k}(\mathbf{x},\hat{\mathbf{b}})$ of any of the k except K (*non-targeted attack*) as high as possible simultaneously. The combinatorial optimization problem of evasion attack is defined below:

Definition 1 *$f : \mathbf{x} \to y$ denotes a classifier with categorical inputs $\mathbf{x}$. The adversary aims to maximize the misclassification confidence f complying the constraint of the attack budget ε, i.e. the maximum number of modified categorical variables in $\mathbf{x}$.*

$$\hat{\mathbf{b}}^* = \underset{\hat{\mathbf{b}}, |\mathit{diff}(\mathbf{b},\hat{\mathbf{b}})| \leq \varepsilon}{\operatorname{argmax}} \; f_{y_k}(\hat{\mathbf{x}}_{\{i,j,:\}} = \hat{b}_i^j \mathbf{e}_i^j), \quad y_k \neq y_K \tag{1}$$

where $\mathbf{x}_{\{i,j,:\}} = b_i^j \mathbf{e}_i^j$ and $\hat{\mathbf{x}}_{\{i,j,:\}} = \hat{b}_i^j \mathbf{e}_i^j$ are the unperturbed and the adversarially tuned instance.

```
Algorithm 1: FEAT: Fast and Effective Adversarial aTtack
Input: The input $\mathbf{x}$ to perturb, the trained model $f_y$,
    the attack budget $\varepsilon$, the time limit $T_L$,
    the number of features to select $L$,
    the number of UCB loops $\tau$;
Output: the chain of features selected to attack $S$;
 1: $S_0 \leftarrow \emptyset$, $\hat{\mathbf{x}} \leftarrow \mathbf{x}$
 2: while $len(S_t) \leq \varepsilon$ and TimeCost $< T_L$ do
 3:     $\text{grad}_i \leftarrow$ The gradient $\nabla f_y(\hat{x}_i)$ for each feature $\hat{x}_i$
 4:     Weight of each feature $w_i = \text{grad}_i / \sum_{j=1}^{N} \text{grad}_j$
 5:     Select top-$L$ features based on $w_i$ from $N$ features
 6:     for $l = 1,2,...,L$, do
 7:         $\hat{x}_l \leftarrow$ the $l$-th selected feature
 8:         $G_{l,t^0} = \max f_{y_k}(\hat{\mathbf{x}}_{l,t^0}) - f_{y_K}(\mathbf{x}) + \Lambda$
 9:     end for
10:     for $t = 1,2,...,\tau$ do
11:         Update $\bar{\mu}_{l,t-1}$ and $\bar{\delta}_{l,t-1}^2$ in Eq.(3)
12:         $I_t = \underset{l \in \{1,2...,L\}}{\operatorname{argmax}} B_{l,(t-1)_l,t-1}$
13:         $S_t \leftarrow S_{t-1} \cup I_t$
14:         Modify $I_t$ in $\hat{\mathbf{x}}$
15:     end for
16: end while
```

The Algorithm Design of FEAT

Our design of FEAT illustrated in **Algorithm** 1 is inspired by the analogy between Multi-Armed Bandit (MAB)-based combinatorial search and the attack problem given in Definition.1. Finding *one categorical feature x_i in the input instance $\mathbf{x}$ to perturb* is analogous to selecting *one arm to pull in an MAB game*. Each arm is characterized by the distribution of the received rewards. Similarly, taking an action to modify the category value of one discrete feature can also cause the variation of the decision output of the target classifier f as a feedback.

More specifically, FEAT defines an iterative MAB search in the discrete feature space to solve the knapsack optimization problem in Eq.1 (see **Algorithm** 1 **Line 6-15**). Given a categorical feature x_l of $\mathbf{x}$, t_l denotes the number of times when x_l is selected to perturb after t iterations of the MAB-driven search.Inheriting the terms used in Eq.1, the reward of modifying each x_l in current t^c iteration (noted as G_{l,t^c} in Eq(2)) is defined as the maximum gap m_f between the classifier's output over any wrong label k and the correct label K by modifying x_l.

$$G_{l,t^c} = \max f_{y_k}(\hat{\mathbf{x}}_{l,t^c}) - f_{y_K}(\mathbf{x}) + \Lambda \tag{2}$$

where $\hat{\mathbf{x}}_{l,t^c}$ denotes the adversarially perturbed input instance at the current iteration t^c with x_l changed. A constant Λ is added to Eq.2 to ensure the non-negativeness of the received rewards. In practices, we set $\Lambda = 1$. In each iteration, the Upper Confidence Bound (UCB) score of each candidate discrete feature can be computed following Eq.(3):

$$B_{l,t_l,t} = \bar{\mu}_{l,t} + \sqrt{\frac{\alpha \bar{\delta}_{l,t_l}^2 * \log t}{t_l}} + \frac{\log t}{t_l} \tag{3}$$

where $\bar{\mu}_{l,t} \overset{\text{def}}{=} \frac{1}{t}\sum_{t^c=1}^{t} G_{l,t^c}$ and $\bar{\delta}_{l,t}^2 \overset{\text{def}}{=} \frac{1}{t}\sum_{t^c=1}^{t}(G_{l,t^c} - \bar{\mu}_{l,t})^2$ are the empirical mean and variance of the obtained rewards (increase of the classification confidence produced by f) by changing x_l after t iterations of search. *In each iteration, the adversary chooses the candidate feature with the highest UCB score as the target to perturb.* The parameter α is tunable to make a trade-off between exploration and exploitation of the search for discrete feature perturbations. **On the one hand**, a larger α extends the exploration covering more new candidate features that have never been tried before. **On the other hand**, an extremely small α drives the search to lean more towards the highly sensitive features. Modifying any of these features can cause drastic variation of f's decision. We traverse different choices of α in FEAT to empirically observe the impact of α over the attack performance.According to Theorem.2 in (Audibert, 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 methods 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 optimise 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 τ 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**). τ is a tunable parameter, adjusting the number of search rounds within the selected top L features. We choose τ 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^j attached to each categorical feature x_i to be continuous and valued within the range $[0,1]$. We then take the gradient of $f(\mathbf{x},\mathbf{b})$ with respect to $\mathbf{b}$, denoted as $\nabla_{\mathbf{b}}(f_y(\mathbf{b}))$. The selected candidate categorical features are those with the largest gradient magnitudes $\|\nabla_{\mathbf{b}}(f_y(\mathbf{b}))\|$. 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 practices of FEAT. We adapt the UCB-based search to the scenario from two aspects. **On one 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 τ rounds and recompute the OMP-based feature ranking. Via this way, the UCB-based search is conducted only within the consecutive τ inner iterations. The reward distribution of the selected L sensitive features can be considered approximately stationary within the τ 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* $\mathbf{b}$ *indicate the category value assignment of an unperturbed data instance* $\mathbf{x}$. $\hat{\mathbf{b}}$ *and* $\hat{\mathbf{b}}'$ *indicate the two different sets of the modification over the same unperturbed input* $\mathbf{x}$. *We further assume* $|\mathit{diff}(\mathbf{b},\hat{\mathbf{b}})| \leq \zeta, |\mathit{diff}(\mathbf{b},\hat{\mathbf{b}}')| \leq \zeta, |\mathit{diff}(\hat{\mathbf{b}},\hat{\mathbf{b}}')| \leq \zeta, \zeta \geq 1$. *Given a smooth target classifier* f *with a finite Lipschitz constant,* $f_{y_k}(\mathbf{x})$ *denotes the decision output of* f *over any incorrect class label, i.e.* $y_k \neq y_K$ *in Definition.1. Let* $\nabla f_y(\mathbf{x},\hat{\mathbf{b}})_\nu$ *denote the elements of* $\nabla f_y(\mathbf{x},\hat{\mathbf{b}})$ *corresponding to the difference between* $\hat{\mathbf{b}}$ *and* $\hat{\mathbf{b}}'$*, where* $\nu = \mathit{diff}(\hat{\mathbf{b}},\hat{\mathbf{b}}')$.

$$\begin{aligned}|f_{y_k}(\mathbf{x},\hat{\mathbf{b}'}) - f_{y_k}(\mathbf{x},\hat{\mathbf{b}})| \leq \quad & \max\{\frac{1}{2m_{k,\zeta}}\|\nabla f_{y_k}(\mathbf{x},\hat{\mathbf{b}})_\nu\|_2^2, \\ & \|\nabla f_{y_k}(\mathbf{x},\hat{\mathbf{b}})_\nu\|_2 + M_{k,\Omega_\zeta}|\zeta|/2\}\end{aligned} \tag{4}$$

where $m_{k,\zeta}$ *and* M_{k,Ω_ζ} *are the local strong convexity factor and local Lipschitz constant of the target classifier* f *around the unperturbed input* $\mathbf{x}$.

As corroborated by Theorem 1, the magnitude of each element in $\nabla_{\mathbf{b}}(f_y(\mathbf{b}))$ provides a bounded estimator to the marginal contribution of attacking the corresponding categorical feature. The top-ranked candidate features with the highest gradient magnitudes $\|\nabla_{\mathbf{b}}(f_y(\mathbf{b}))\|$ are more likely to be the most sensitive features with respect to the adversarial perturbation, compared to those at the tail of the ranking list. We therefore use the OMP-based ranking strategy to narrow down the search regime within the selected top-ranked and potentially sensitive features. Perturbing more sensitive features can produce higher adversarial risk over categorical inputs, according to Theorem.2 in (Bao et al. 2022). The rationality of integrating this feature ranking step is thus to encourage the UCB-guided search to concentrate more on manipulating the sensitive features, which is more likely to cause larger change to the classifier's output with only a few feature changed.

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*

$$\triangle_l = \mu^* - \mu_l \quad (\mu^* = \max_{1 \leq l \leq L} \mu_l) \tag{5}$$

where μ_l *and* μ^* *are the expected and optimal reward received by changing* l.

Theorem 2 ***Perturbing highly sensitive features helps shrink the regret bound of FEAT.*** *Let* $\triangle_l > 0$ *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:*

$$\mathbb{E}[R_T] \leq \sum_{l=1}^{L}[8(\frac{\delta_l^2}{\triangle_l}+2)\log T + \frac{\alpha}{\alpha-2}\triangle_l], \tag{6}$$

where $\mathbb{E}[R_T] \overset{def}{=} \sum_{l=1}^{L}\mathbb{E}[T_l]\triangle_l$. T_l *is the number of times that feature* l *is selected after* T *iterations.*

Empirically, we observe that the variances δ_l^2 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 δ_l^2 is generally small, the lower regret for each selected feature ($\triangle_l$) 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 $\triangle_l$ 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[1]. 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 $\mathbf{x} \in R^{200*4130*70}$ 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 $b_i^j = 0$.

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/xnudinfc/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 an 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.*

[1] http://www.icd9data.com/

Attack Algo.	**Computational Complexity**
FSGS	$\sum_{t=0}^{T}((N-t)*M*2^t)$
GradAttack	$T*\sum_{k=0}^{L}(C_L^k*M^k)$
OMPGS	$\sum_{t=0}^{T}(L*2^t)$
FEAT-B	$L*M+T$
FEAT	$(L*M+\tau)*T$

Table 1: Complexity of the domain-agnostic attack methods

We also involve two domain-specific attack methods **TextFooler** (Jin et al. 2020) and **TextBugger** (Li et al. 2018) into the test over the text data of Yelp-5. They have been popularly adopted in various attacks against text classification. We focus on showing FEAT, as an universally applicable method, can achieve similar applicability to the text-formatted categorical data, compared to these specially designed attack methods for NLP tasks based on semantic integrity/similarity-based knowledge. Demonstrating FEAT as a novel attack against text classification is beyond our scope.

Evaluation metrics. We measure the average number of the confidence computing operations required to deliver successful attack on one input instance (noted as **No.query**) as the metric of computational cost of the attack. In addition, we also show the averaged running time needed to attack one instance, noted as **Runtime** and measured in seconds. Both metrics indicate the computational efficiency level of each attack method. To evaluate the effectiveness of the attack, we use the attack success rate (**SR**) over all the testing instances. With the similar level of **SR**, if **Runtime** and **No.query** of an attack method are significantly lower than the other opponents, it indicates that this attack method could attack the high-dimensional instances faster.

Attack Performance on Four Datesets

Computational Complexity Analysis. We compare the computational complexity of all the involved domain-agnostic attack methods by counting the number of times evaluating the decision confidence $f(\hat{\mathbf{x}})$ during attacking one input instance. In Table. 1, T denotes the overall number of iterations required to reach successful attack. N and M are the number of features of $\mathbf{x}$ 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. **??** 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 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*

(a) The results on Yelp-5 data

Yelp-5		**Budget = 6**		
Attack Type & Algo.		**Runtime** (sec) ↓	**No.query** ↓	**SR** ↑
Domain Specific	TextBugger	1.18	23	0.64
	TextFooler	0.42	167	0.88
Black Box	FSGS	0.52	24000	0.97
	FEAT-B	0.14	2057	0.90
White Box	GradAttack	0.15	10000	0.78
	OMPGS	1.25	7000	0.96
	FEAT	**0.14**	**887**	**0.97**

(b) The results on IPS data

IPS		**Budget = 5**		
Attack Type & Algo.		**Runtime** (sec) ↓	**No.query** ↓	**SR** ↑
Black Box	FSGS	136	37000	0.80
	FEAT-B	19.5	2500	0.74
White Box	GradAttack	21.2	2100	0.59
	OMPGS	1.99	127	0.77
	FEAT	**0.28**	**111**	**0.92**

(c) The results on PEDec data

PEDec		**Budget = 14**		
Attack Type & Algo.		**Runtime** (sec) ↓	**No.query** ↓	**SR** ↑
Black Box	FSGS	435	213256	0.88
	FEAT-B	3.65	9959	0.87
White Box	GradAttack	3.51	18563	0.67
	OMPGS	360	27758	0.80
	FEAT	**2.89**	**5923**	**0.91**

(d) The results on EHR data

EHR		**Budget = 6**		
Attack Type & Algo.		**Runtime** (sec) ↓	**No.query** ↓	**SR** ↑
Black Box	FSGS	482	58000	0.84
	FEAT-B	167	7108	0.94
White Box	GradAttack	2.34	204	0.94
	OMPGS	27.5	35	0.94
	FEAT	**0.35**	**20**	**0.94**

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

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.

Discussion and Conclusion

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 candidate 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.

Acknowledgements

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

References

Agrawal, S.; and Goyal, N. 2013. Further Optimal Regret Bounds for Thompson Sampling. In Carvalho, C. M.; and Ravikumar, P., eds., *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, 99–107. Scottsdale, Arizona, USA: PMLR.

Asghar, N. 2016. Yelp dataset challenge: Review rating prediction. *arXiv preprint arXiv:1605.05362*.

Audibert, J.-Y.; Munos, R.; and Szepesvári, C. 2007. Tuning Bandit Algorithms in Stochastic Environments. In Hutter, M.; Servedio, R. A.; and Takimoto, E., eds., *Algorithmic Learning Theory*, 150–165. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-75225-7.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2): 235–256.

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

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 1467–1474.

Buchbinder, N.; Feldman, M.; Naor, J.; and Schwartz, R. 2014. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, 1433–1452. SIAM.

Carlini, N.; and Wagner, D. 2018. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *SPW*.

Cartella, F.; Anunciacao, O.; Funabiki, Y.; Yamaguchi, D.; Akishita, T.; and Elshocht, O. 2021. Adversarial Attacks for Tabular Data: Application to Fraud Detection and Imbalanced Data. *arXiv preprint arXiv:2101.08030*.

Croce, F.; and Hein, M. 2019. Sparse and Imperceivable Adversarial Attacks. In *ICCV*, 4723–4731.

Ebrahimi, J.; Rao, A.; Lowd, D.; and Dou, D. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. In *ACL*.

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

Gao, J.; Lanchantin, J.; Soffa, M. L.; and Qi, Y. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, 50–56. IEEE.

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

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Imam, N. H.; and Vassilakis, V. G. 2019. A survey of attacks against twitter spam detectors in an adversarial environment. *Robotics*, 8(3): 50.

Jia, R.; and Liang, P. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Jin, D.; Jin, Z.; Zhou, J. T.; and Szolovits, P. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 8018–8025.

Li, J.; Ji, S.; Du, T.; Li, B.; and Wang, T. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

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

Ma, F.; Gao, J.; Suo, Q.; You, Q.; Zhou, J.; and Zhang, A. 2018a. Risk prediction on electronic health records with prior medical knowledge. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1910–1919.

Ma, F.; Gao, J.; Suo, Q.; You, Q.; Zhou, J.; and Zhang, A. 2018b. Risk Prediction on Electronic Health Records with Prior Medical Knowledge. In *KDD*.

Miyato, T.; Dai, A. M.; and Goodfellow, I. 2016. Adversarial Training Methods for Semi-Supervised Text Classification. In *ICLR*.

Narodytska, N.; and Kasiviswanathan, S. 2017. Simple Black-Box Adversarial Attacks on Deep Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1310–1318.

Papernot, N.; McDaniel, P.; Swami, A.; and Harang, R. 2016. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, 49–54. IEEE.

Pendlebury, F.; Pierazzi, F.; Jordaney, R.; Kinder, J.; and Cavallaro, L. 2019. TESSERACT: Eliminating experimental bias in malware classification across space and time. In *USENIX Security*.

Pierazzi, F.; Pendlebury, F.; Cortellazzi, J.; and Cavallaro, L. 2020. Intriguing Properties of Adversarial ML Attacks in the Problem Space. *2020 IEEE Symposium on Security and Privacy*, 1332–1349.

Qi, L.; Wu, L.; P, C.; A, D.; Dhillon, I.; and Witbrock, M. 2019. Discrete Attacks and Submodular Optimization with Applications to Text Classification. In *SysML*.

Samanta, S.; and Mehta, S. 2017. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*.

Shu, K.; Mahudeswaran, D.; Wang, S.; Lee, D.; and Liu, H. 2020. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big data*, 8(3).

Stringhini, G.; Kruegel, C.; and Vigna, G. 2010. Detecting spammers on social networks. In *Proceedings of the 26th annual computer security applications conference*, 1–9.

Suciu, O.; Coull, S. E.; and Johns, J. 2019. Exploring adversarial examples in malware detection. In *2019 IEEE Security and Privacy Workshops (SPW)*, 8–14. IEEE.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

van Ede, T.; Aghakhani, H.; Spahn, N.; Bortolameotti, R.; Cova, M.; Continella, A.; van Steen, M.; Peter, A.; Kruegel, C.; and Vigna, G. 2022. DeepCASE: Semi-Supervised Contextual Analysis of Security Events. In *IEEE S&P*.

Wang, W. Y. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Wang, Y.; Audibert, J.-y.; and Munos, R. 2008. Algorithms for Infinitely Many-Armed Bandits. In Koller, D.; Schuurmans, D.; Bengio, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.

Wang, Y.; Han, Y.; Bao, H.; Shen, Y.; Ma, F.; Li, J.; and Zhang, X. 2020. Attackability Characterization of Adversarial Evasion Attack on Discrete Data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1415–1425.

Yang, P.; Chen, J.; Hsieh, C.; Wang, J.; and Jordan, M. I. 2018. Greedy Attack and Gumbel Attack: Generating Adversarial Examples for Discrete Data. *ArXiv*, abs/1805.12316.

Yang, P.; Chen, J.; Hsieh, C.-J.; Wang, J.-L.; and Jordan, M. I. 2020. Greedy Attack and Gumbel Attack: Generating Adversarial Examples for Discrete Data. *J. Mach. Learn. Res.*, 21(43): 1–36.

Zang, Y.; Qi, F.; Yang, C.; Liu, Z.; Zhang, M.; Liu, Q.; and Sun, M. 2020. Word-level Textual Adversarial Attacking as Combinatorial Optimization. In *ACL*.

Zhang, R.; and Combes, R. 2021. On the Suboptimality of Thompson Sampling in High Dimensions. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 8345–8354. Curran Associates, Inc.