Real-Time Detection of Robotic Traffic in Online Advertising

Anand Muralidhar, Sharad Chitlangia, Rajat Agarwal, Muneeb Ahmed

Amazon
{anandmur, chitshar, agrajat, munbahmd}@amazon.com

Abstract

Detecting robotic traffic at scale on online ads needs an approach that is scalable, comprehensive, precise, and can rapidly respond to changing traffic patterns. In this paper we describe SLIDR or **SLI**ce-Level **D**etection of **R**obots, a real-time deep neural network model trained with weak supervision to identify invalid clicks on online ads. We ensure fairness across different traffic slices by formulating a convex optimization problem that allows SLIDR to achieve optimal performance on individual traffic slices with a budget on overall false positives. SLIDR has been deployed since 2021 and safeguards advertiser campaigns on Amazon against robots clicking on ads on the e-commerce site. We describe some of the important lessons learned by deploying SLIDR that include guardrails that prevent updates of anomalous models and disaster recovery mechanisms to mitigate or correct decisions made by a faulty model.

1 Introduction

Online advertising has seen significant adoption among sellers and has both cost-per-click (CPC) and cost-per-mille (CPM) campaigns. In the former an advertiser is charged for a click on an ad while in the later the advertiser is charged for showing 1000 ad impressions. In this paper we are concerned with CPC campaigns that have attracted the attention of services that build sophisticated bots to click on ads. There are many motivations for designing bots to click on online ads like improving the performance of a product listing by boosting its search rankings, depleting a competitor's ad budget, modifying the CPC for an ad campaign, etc. We are concerned with robotic traffic that is fraudulent or non-human, with no value to advertisers. There are other types of robotic traffic that involve crawlers and maybe of a benign nature but are not considered here. The goal of robot detection is to build algorithms that ensure advertisers are not charged for invalid activity, and human clicks are not invalidated, with all decisions made as real-time as possible to cause minimal disruption to advertiser experience.

There have been attempts to solve this problem by creating lists of robotic entities like IP-addresses or by generating signatures of bots based on forensic analysis of the transport layer handshake between server and client. However such static lists might have low precision since an IP address might see a mixture of robotic and human traffic over a period of time. Also static lists have low recall towards emerging attacks from a new bot or when a bot evolves its signature. Instead we focus on building a solution that considers every ad click in real-time and decides if the click was made by a human or robot. This in-house approach allows us to build rich features to distinguish robots from humans and also utilize insights from customers behavior on the e-commerce site to generate signals. However we cannot design simple rules or heuristics with these features to detect bots due to the sophistication of bots, adversarial nature of problem, evolving traffic patterns, and a need to control business metrics. We adopt a ML-centric approach towards detecting robotic traffic since it allows discovery of new bot signatures, ability to tune operating points of algorithms, and extensibility in terms of adding new features. However there are a few challenges towards building ML models:

Lack of ground truth We have a large volume of ad clicks in online advertising but are unable to confidently label a click as robotic or human. Hence we need to identify proxy labels based on actions that are more likely to be performed by humans than robots.

Balance revenue and recall Any click that we incorrectly mark as robotic leads to a loss of revenue for the company while any click that we mark as human could be a potential bot that has evaded detection. We address this tension by calibrating ML models for a chosen false positive rate.

Equal protection across traffic slices Online ads are displayed on many traffic slices across a variety of devices and ad placements on the e-commerce site. The model should ensure fairness and offer similar protection across all slices of traffic. We formulate this as an offline convex optimization problem to determine the operating points of the model for each traffic slice.

Evolving bots, need fast reaction Bots are rapidly evolving with increase in their sophistication and the model needs to continuously adapt along with traffic patterns. The model should also quickly react to egregious robotic attacks that have the potential to deplete budgets of small or medium-sized ad campaigns. We have designed features that allow

the model to be nimble and deployed it for real-time detection of robotic ad clicks.

In this paper we describe SLIDR or **SLI**ce-Level **D**etection of **R**obots, a system based on a deep neural network that strives to meet the challenges described above and has been in continuous operation on advertisements shown on Amazon since 2021. The main contributions in the paper are:

- A framework for building a ML model to detect robots that click on ads in real-time, see Section 2.
- A formulation of calibration across traffic slices as a convex optimization problem, see Section 3.
- Lessons learned by deploying the system that include guardrails and disaster recovery mechanisms, see Section 5.

Section 4 evaluates the model's metrics under different scenarios. We describe the related work in Section 6 and conclude in Section 7.

2 ML Framework for Detecting Robots

2.1 Labels Based on Weak Supervision

We use weak supervision (Ratner et al. 2017) to circumvent the problem of missing labels and identify high-hurdle activities that are more likely to be performed by humans than robots. An example is purchases on the e-commerce site that require a valid credit card. We identify ad clicks that led to a purchase on the e-commerce site and label such clicks as human (0). In order to increase the density of human labels, we also identify ad clicks made by customers with the highest **RFM** score that indicates that the customer had purchased **R**ecently, **F**requently, and with a high **M**onetary value. Clicks on ads by customers with the highest **RFM** score are also marked as human (0). All other clicks are marked as robotic (1). We formulate the training of a ML model to identify robotic traffic as a binary classification problem with labels as described above. We acknowledge that there are other methods to solve this problem with unsupervised approaches that do not require explicit labels and leave the categorization of all possible approaches to robot detection on ads for a later study.

2.2 Metrics

Due to lack of confident labels on robotic clicks, we cannot reliably measure standard metrics for a classification model like precision and recall. Hence we use other metrics that are closely aligned with the business impact and advertiser experience, and also allow us to reliably compare models.

Invalidation Rate (IVR) This is defined as the fraction of clicks marked as robotic by the algorithm, i.e., IVR $=$ (CLICKS MARKED AS ROBOTIC)/(TOTAL CLICKS). IVR is indicative of recall of the model, but it cannot be viewed in isolation since a poorly-performing algorithm might incorrectly invalidate a significant volume of human clicks. Hence we always measure IVR in conjunction with False Positive Rate (defined next) to ensure that we measure recall without reducing the precision of detection.

False Positive Rate (FPR) This is defined as the fraction of human clicks invalidated by the algorithm, i.e., FPR $=$ $\frac{\text{HUMAN CLICKS MARKED AS ROBOTIC}}{\text{TOTAL HUMAN CLICKS}}$. But we cannot directly measure FPR since we do not have confident labels for human clicks. We consider ad clicks that led to a purchase as a proxy for human clicks and measure a proxy metric $\text{FPR}_{\text{PROXY}}$ defined as

$$\frac{\text{PURCHASING AD CLICKS MARKED AS ROBOTIC}}{\text{TOTAL PURCHASING AD CLICKS}}. \quad (1)$$

In rest of the paper FPR refers to $\text{FPR}_{\text{PROXY}}$, while noting that there are two underlying assumptions. First is that all purchasing clicks are human and second is that purchasing clicks are a representative sample of human clicks. We are aware of some robots that make purchases but these do not substantially change the FPR.

Robotic Coverage We define a heuristic rule to identify traffic that has a high likelihood to be robotic. If a session makes more than $\mathcal{K}$ ad clicks in a hour, then it is likely to be robotic. We choose $\mathcal{K}$ to be a large number so that humans are less likely to click on ads at that rate. We define *Robotic coverage* as the percentage of such clicks that were marked as robotic by the model. This metric can also be considered as an indicator of recall for a model.

2.3 Calibration of ML Model

A ML model's IVR cannot be held steady across days since this metric is proportional to the robotic activity in a day. For example if the robotic activity spikes due to an emergent attack, then we expect the model's IVR to proportionally increase if it detects the attack. However we can calibrate a ML model to maintain the same FPR. FPR is an important business metric and the loss of revenue for the company due to incorrect invalidation of human clicks can be upper bounded as a function of FPR as shown below.

Consider a cost-per-click (CPC) advertising program that is supply-constrained but has infinite demand, i.e., every ad placement has at least one advertiser willing to display ads on it. Let ACTUAL REVENUE be defined as the total CPC measured across all ad clicks in the program that were marked as human by the algorithm and were charged to advertisers. In this setting, we can upper bound the loss of revenue due to FPR as follows:

Lemma 2.1. *For a given FPR, the loss of revenue due to human clicks incorrectly invalidated by the algorithm is upper-bounded by* $\frac{FPR}{1 - FPR} \times$ (ACTUAL REVENUE).

Proof. Let us define MAX REVENUE as CPC added across all *human* clicks in the program. Now, in a sufficiently large advertising program that allows statistical approximations, ACTUAL REVENUE is lower bounded as

$$\geq \left(\frac{\text{Human clicks marked as valid}}{\text{Total Human clicks}}\right) * \text{MAX REVENUE} \quad (2)$$

$$= \left(1 - \frac{\text{Human clicks marked as Robot}}{\text{Total Human clicks}}\right) * \quad (3)$$

$$= \text{MAX REVENUE} \\ (1 - \text{FPR})\,\text{MAX REVENUE}, \quad (4)$$

where we have a lower bound in (2) since ACTUAL REVENUE also includes robot clicks that were incorrectly marked as human.

The loss in revenue from the lemma is bounded as

$$\text{LOSS IN REVENUE} \leq \text{MAX REVENUE} - \text{ACTUAL REVENUE}, \quad (5)$$

where the bound holds with equality if advertisers do not run out of budget and are always able to pay for the CPC on ads that were previously incorrectly invalidated. Substituting from (4) in the above bound, we get the statement of the lemma. □

Due to this close relationship between FPR and ACTUAL REVENUE, we would like to control this metric and present a systematic approach for choosing FPR across traffic slices in the next section.

3 A Neural Model for Detecting Robots

3.1 Model Architecture

We choose a variety of input features that are updated in real-time for every ad click and provide the ML model with a discriminative ability to identify robots from humans:

- **Frequency and velocity counters**: Volume and rate of clicks from a user computed over various time periods ranging from seconds to hours. These features are critical for identifying emergent robotic attacks that involve a sudden burst of clicks from a robotic entity.
- **User-entity counters**: Distinct sessions and users from an IP address in the recent past. We also compute the corresponding maximum values seen in an hour. These features help to identify IP addresses that correspond to gateways of enterprises that may have many users behind them who perform a large number of ad clicks. We do not want to inadvertently mark such IPs as robotic.
- **Time of click**: Hour-of-day and day-of-week mapped to points on the unit circle. Human activity follows the regular diurnal and weekly activity patterns but robotic activity may not conform to it and these features help to consider activity patterns over time. Also mapping these features to the unit circle ensures a smooth transition from the last hour (or day) to the first one and the model does not see a sudden jump in the feature value.
- **Logged-in**: Boolean feature that captures if the user was logged-in to their account. Logging to their account requires a valid email address and/or phone number and requires more sophistication from a robot.

We note that this is not an exhaustive list of features for a ML model for robot detection. These features are used to train a DNN to solve a classification problem based on the labels defined in Section 2.1. The network has three fully-connected hidden layers with ReLU activation. The third hidden layer is connected to a sigmoid output. Since the dataset is imbalanced with very few human (0) labels, we weigh the i-th training sample by $w_i = C/N_i$, where N_i is total clicks in $(\text{hour}_i; \text{day}_i; \text{logged-in}_i; \text{label}_i)$ bucket and C is a constant added for numerical stability. L2 regularization is applied on all layers except the first one. The network is optimized to minimize weighted binary cross-entropy loss and we use Adam as the optimizer. The network trains on all ad clicks in a week for one epoch.

3.2 Single-threshold Calibration

The DNN model described above outputs a score for every ad click and the score can be interpreted as the probability of the click being robotic. As part of model calibration, we need to choose a threshold above which all output scores will be considered as robotic. For each choice of output threshold, we can compute the IVR and FPR for the model as described in Section 2.2. We vary the model's output threshold and plot IVR against FPR in Figure 1a. We would like to choose high IVR to improve recall but this would result in high FPR and a drop in the company's revenue. An ideal compromise is to fix the FPR above the *knee* of the curve since any increase in IVR beyond the knee comes at a much higher incremental cost in FPR. An example operating point corresponding to FPR_A is depicted in Figure 1a.

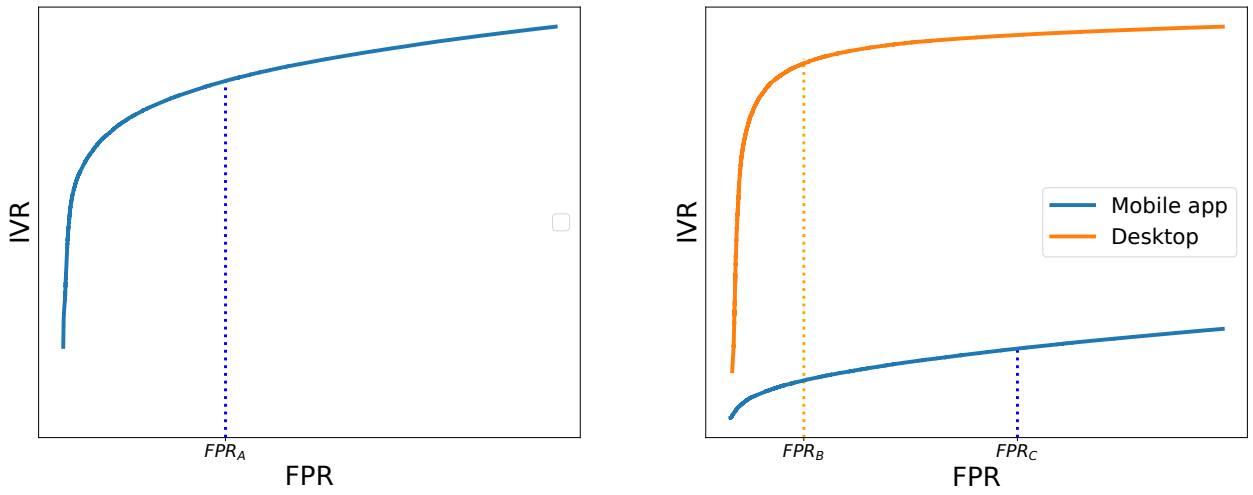
We calibrated the model at this operating point and analyzed its impact on two traffic slices corresponding to ad clicks from desktop or mobile app devices on different types of ad placements on the e-commerce site. The choice of overall FPR = FPR_A corresponds to a threshold for model output and results in FPR calibration points of FPR_B and FPR_C for the two traffic slices as shown in Figure 1b. But this approach is clearly sub-optimal since it has resulted in under-calibration for desktop users while over-calibration for ad clicks from the mobile app. At this operating point, we are not invalidating bots in the desktop slice while we are incurring extra FPR and losing revenue in the mobile app slice. This can also lead to a difference in performance of ad campaigns on traffic slices.

3.3 SLIDR: Slice-level Detection of Robots

If we divide traffic into multiple slices, then there are two possible approaches towards choosing FPR operating points (and corresponding thresholds) for each slice:

1. **Individually optimize slices**: Choose the FPR at the knee of the IVR-FPR curve for each traffic slice. But this approach does not allow us to control the overall FPR which is a critical business metric.
2. **Joint optimization**: Pick an overall FPR budget and jointly optimize across all slices.

Next we describe (2) in more detail. Let the IVR and FPR for the i-th slice be defined as IVR_i and FPR_i respectively. Then the overall $\text{FPR} = \sum_{i=0}^{N} \text{FPR}_i$, where N is the total number of traffic slices. Similarly we can compute the overall IVR. Let the overall FPR be limited to $\text{FPR}_{\max}$. If we maximize IVR subject to $\text{FPR} \leq \text{FPR}_{\max}$, then an optimal solution may choose $\text{FPR}_i = \text{IVR}_i = 0$ for some slice, which is unacceptable since it will provide no protection against robots in the i-th slice. Hence we bound the robotic coverage R_i (see Section 2.2) in i-th slice to ensure a certain recall



(a) IVR-FPR curve of full traffic

(b) IVR-FPR curves of traffic slices.

Figure 1: Performance curves for the model calibrated by choosing a single threshold across traffic slices.

of robots in the slice. The optimization problem reduces to:

$$\max \sum_{i=0}^{N} \text{IVR}_i, \quad \text{s.t.} \quad \sum_{i=0}^{N} \text{FPR}_i \leq \text{FPR}_{\max} \quad (6)$$
$$R_i > R_{\min}, \forall i.$$

The robotic coverage R_i monotically increases with FPR_i and the second constraint can be replaced with an equivalent constraint $\text{FPR}_i > \text{FPR}_{i,\min}$ where $\text{FPR}_{i,\min}$ is the lowest FPR for the i-th slice when its robotic coverage exceeds $R_{\min}$. Next, we can approximate each slice's IVR-FPR curve by a quadratic $y = ax^2 + bx + c$, where a, b, c are determined with linear regression. Concave shape of the IVR-FPR curve ensures that this quadratic approximation always results in $a < 0$, due to which the quadratic is strictly concave. We can rewrite the above optimization problem with this approximation for all slices:

$$\max \sum_{i=0}^{N} a_i x_i^2 + b_i x_i + c_i, \quad \text{s.t.} \quad \sum_{i=0}^{N} \text{x}_i \leq \text{x}_{\max} \quad (7)$$
$$x_i > x_{i,\min}, \forall i,$$

where x_i, $x_{i,\min}$, and $x_{\max}$ represent FPR_i, $\text{FPR}_{i,\min}$, and $\text{FPR}_{\max}$ respectively, and a_i, b_i, c_i are constants corresponding to the quadratic approximation for i-th slice. Now (7) is a standard convex optimization problem and can be solved with a package like (Fu, Narasimhan, and Boyd 2020).

In a typical e-commerce site, ad placements will be across search pages or product description pages while the ad clicks originate from devices corresponding to desktop, mobile browser, and mobile app. In the production SLIDR model, we consider traffic slices by considering various combinations across ad placements and devices. The optimization problem in (6) needs to be solved once in an offline setting to determine the FPR operating points for all slices. Subsequently we adjust the operating points periodically or when we notice any deviations in the traffic patterns in various slices.

4 Evaluation

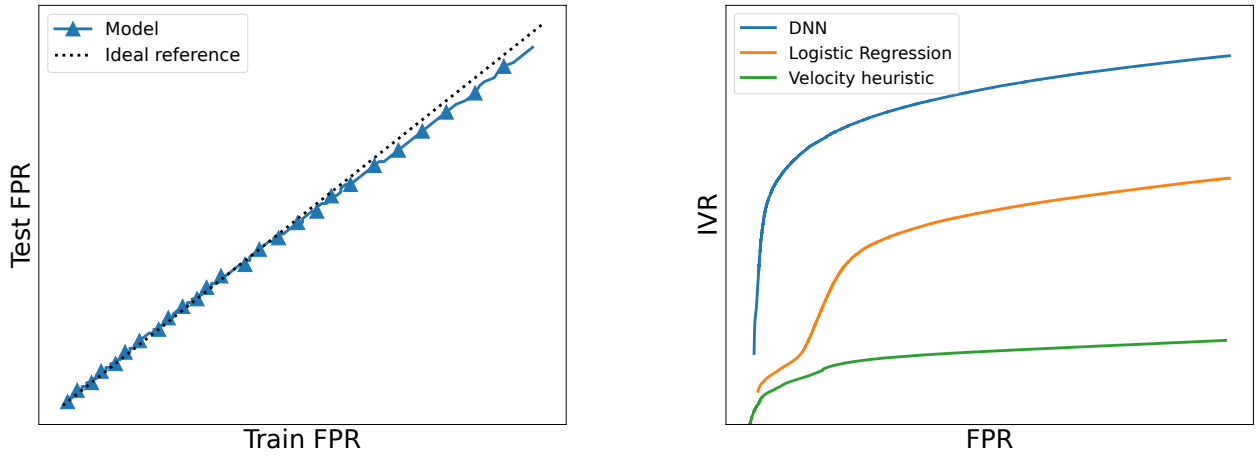
In this section, we check the generalization ability of the model and systematically evaluate the weak labelling scheme and importance of sample weights.

4.1 Model Generalization

To study the generalization of the model, we choose a value for the train FPR, determine the operating point for the model, and compute the test FPR incurred by the model at this threshold. Note that train FPR is calibrated on training data of one week used for training the model while the test FPR is computed on the next week's data which was not seen by the model. Also we perform this experiment for single-threshold calibration where we pick the same threshold across all traffic slices for the sake of simplicity. In an ideal scenario, the model's test FPR should exactly match the train FPR and a drift would indicate a lack of generalization of the model. As depicted in Figure 2a, our model deviates very little from the ideal reference with no generalization error, indicating the stability of our model.

4.2 Comparison with Baselines

We compare our neural model for detecting robots against a heuristic and a logistic regression model. We define a heuristic based on velocity of clicks received from an entity in a set time window. Clicks originating from an entity that have high velocities are likely to be robotic in nature. We also compare our neural model against a logistic regression



(a) Comparison of train and test FPR

(b) Model comparison against baselines

Figure 2: Evaluation of the neural model's generalization and comparison against baselines

model that uses the same features and training setup (including the sample weights and the combination of weak labels).

As depicted in Figure 2b, we find that our neural model is vastly superior to both the baselines considered. The velocity based heuristic suffers from low recall and hence, has significantly lower IVR at a set FPR. Although, the logistic regression model uses the same features, it has significantly less capacity as compared to the neural model. This sugggests that the neural model is able to learn more sophisticated patterns as compared to the baselines.

4.3 Effect of Using Multiple Weak Labels

As described in Section 2.1, we define human labels based on ad clicks that led to a purchase as well as clicks from customers with a high RFM score. To understand the importance of using a combination of both these weak labels, we train and evaluate a model that does not assign human labels to clicks from high RFM customers and only marks ad clicks that led to a purchase as human (0). Figure 3a has the IVR v/s FPR curves for models trained with and without human labels based on RFM signal. We find that to achieve a similar IVR level as the model trained with both the weak labels, the model trained with labels based only on purchases needs to significantly increase its FPR. Further, we find that excluding human labels based on the RFM signal increases the instability of FPR across hours, see Figure 3b. The model trained with human labels based only on purchases has a significantly higher variance in FPR across hours than the model trained with dense labels. For both the plots in Figure 3, we consider single-threshold calibration to simplify the exposition. Hence, we see that a combination of weak labels based on ad clicks that led to purchases as well as ad clicks from high-RFM customers is essential and enables the model to better disambiguate between robots and humans.

4.4 Effect of Removing Sample Weights

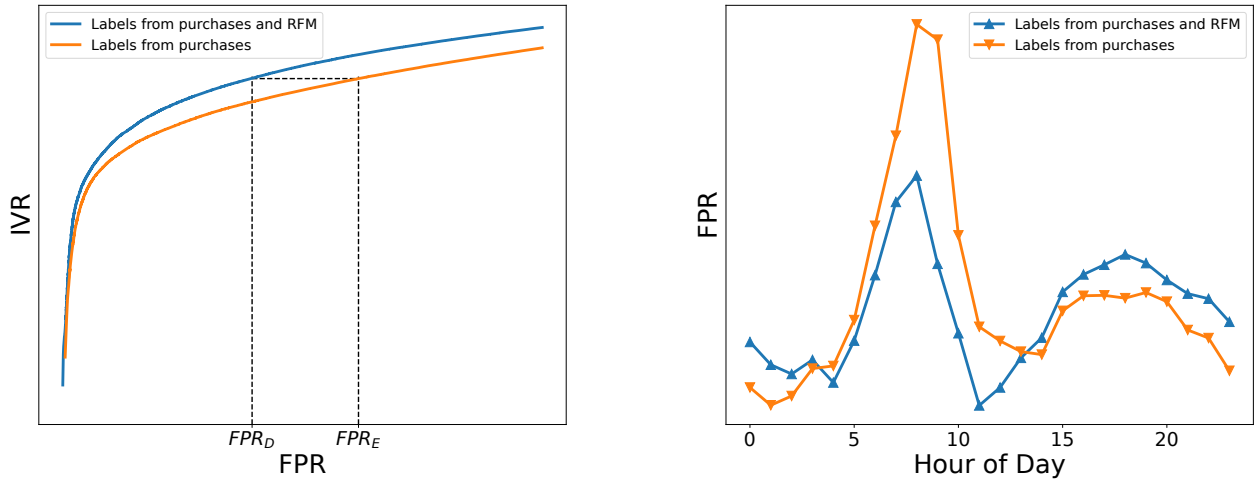
We check the importance of sample weights by training a model without them, i.e., all data points in the training set are given equal importance. Similar to the comparison methodology described in Section 4.3, we plot the IVR v/s FPR of the model at various operating points in Figure 4. We find the model trained with sample weights to have a superior performance than the model without sample weights. For a similar IVR level, the model trained without sample weights incurs almost double the FPR as shown in Figure 4a. Sample weights allow the model to proportionally weigh data points across various traffic slices in the training data. In particular, early morning hours, weekdays, and non-logged-in slices are examples of sparse traffic slices that need to be compensated for by sample weighing. For instance, we see that the model trained without sample weights has a high variance in FPR across hours, see Figure 4b.

5 Deployment in Production

5.1 System Design

The production setup consists of a real-time component for feature computations and low-latency model inference, an offline component for periodic model retraining and calibration, and a control plane for model management, see Figure 5.

Real-time Component Static features are stored in a Read-only cache that is ideal for bulk key-value updates and fast lookups, whereas dynamic features which update with every click are stored in a Redis cache that supports both low latency and high throughput read and write operations. The inference service reads features for every click, runs model inference on a fleet of NVIDIA T4 GPU instances on AWS, and uses vanilla TensorFlow Serving containers operating



(a) IVR v/s FPR in the test period

(b) FPR across various hours of the day

Figure 3: Compare models trained with and without human labels based on high-RFM clicks.

behind a Network Load Balancer that auto-scales the service to handle traffic surges. The inference service has p99.9 latency below 5ms to meet the real-time constraints.

Offline Component The pre-training stage has Spark jobs that join feature logs from the real-time component with the labels and static features to construct the training dataset. This stage is also responsible for stringent quality checks on the training data. The model is then trained on an SageMaker based GPU cluster, which outputs the model artifacts to a S3 location. The post-training stage calibrates and compares various metrics between the current model in production and the latest trained model and decides if the current model should be replaced by the new one.

Control Plane Post training, the model artifacts are saved in a S3 location and replicated to a shared file system (Amazon EFS), where it gets unpacked, deployed, and ready to be served. The details of the model such as a unique identifier and thresholds are maintained in a configuration file. The real-time module appends the model identifier to the inference service endpoint and makes a POST REST call to get the probability score. It then compares the probability score with the threshold for the traffic slice, to make a decision whether the ad click is robotic or human.

5.2 Guardrails Against Anomalous Behavior

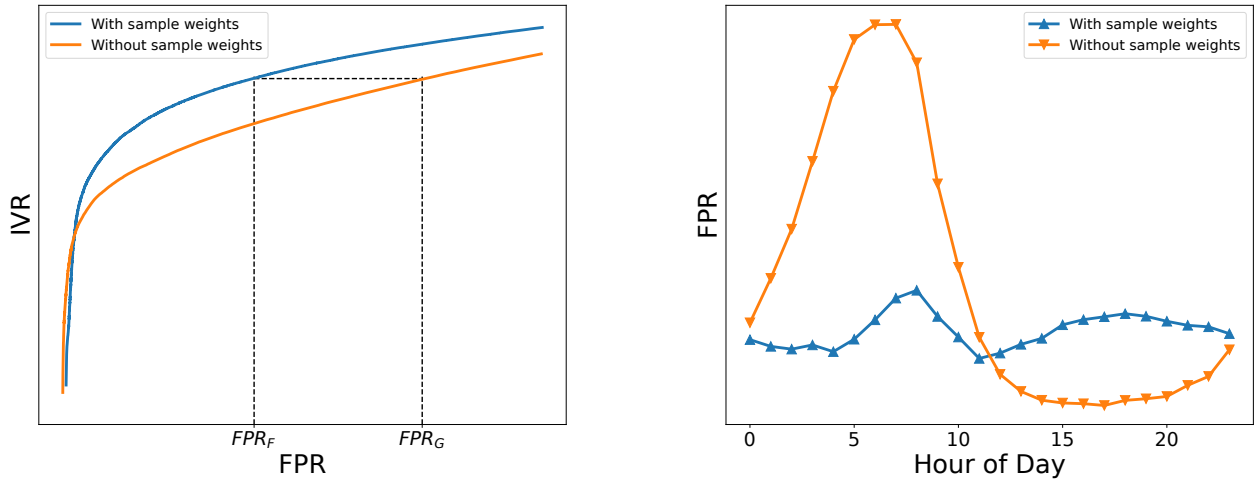
An anomalous behavior in the SLIDR system can result in significant business impact by over or under-invalidating traffic. We have observed two kinds of problems that can lead to such events: corruption of training data and poorly trained models.

Label and Feature Quality Checks The labels and features for SLIDR are generated through hourly Spark jobs that can sometimes fail due to infrastructure-related issues or missing upstream datasets. In particular if human (0) labels are missing for a particular hour in the training data, which can happen if the upstream dataset containing purchases was not accessible for that hour, then the model mistakenly learns that a particular hour in a day has zero human traffic. This causes the model's IVR to sharply increase for the affected hour and can potentially cause a huge revenue loss. To prevent this, we have guardrails to check that every hour x day across the training period of a week contains a minimum number of clicks with a certain human label density. We ignore any training samples that have null values for a feature and have a guardrail on the total number of such samples. If any of these guardrails fails, then model training will not proceed for that instance.

Check Metrics for Model Promotions Post-training of a new model, we need to decide whether to replace the existing model in production or continue with it. The new model needs to pass many checks to qualify for promotion:

- The new model's metrics like AUROC and log-loss at the end of training should lie within specific bands.
- The new model should have a higher IVR in validation data than the production model though both are calibrated at the same FPR, which indicates that the new model has higher recall of robots.
- The new model's IVR on most recent hours should be within a certain percentage of the IVR of the production model, to ensure that the update does not cause any drastic changes in production metrics.

If any of these checks are not met, then the new model does not get promoted.



(a) IVR v/s FPR in the test period

(b) FPR across various hours of the day

Figure 4: Compare models trained with and without sample weights.

5.3 Disaster Recovery Mechanisms

In spite of multiple guardrails, we have faced scenarios wherein an anomalous model got promoted in production. We have built two mechanisms to deal with this scenario as outlined below.

Quick Rollbacks The model in production has an identifier and thresholds for various traffic slices that are saved in a configuration file. The real-time module has listeners attached to this configuration file and imports the new model whenever its updated. When we observe sharp metric deviations from a model, we have a mechanism to manually update the configuration file and roll back the model identifer to a previous stable version, thereby mitigating any issues.

Replay Tool We can replay traffic through a previous stable model and publish decisions, albeit with a delay. Some of the features in SLIDR like velocity of ad clicks are sensitive to time differences between two clicks from a user. The replay infrastructure reads clicks from a chosen time period and passes them through the feature computation module while ensuring that the timestamps are accurately handled to simulate real-time behavior.

6 Related Work

6.1 Fraud Detection

Fraud Detection as a problem in machine learning has been widely studied especially in the context of credit card fraud with datasets such as (Yeh and Lien 2018) serving as common benchmarks. In the context of online advertising, fraud detection has certain unique characteristics as previously mentioned (evolving landscape, lack of labels, etc.). There have been various works which rely on non-ML based approaches such as rules, signature-based measures, etc. Rule-based methods rely on certain characteristics to identify patterns that differentiate fraud activity from human activity. (Stone-Gross et al. 2011) is an example where a set of rules based on common aggregate statistics with tunable thresholds was developed to identify anomalous traffic. Some previous works have also used data mining to identify more nuanced patterns (Oentaryo et al. 2014). Although rules have an advantage of being interpretable and can have high precision, they have a low recall by not being able to model complex patterns. Additionally they are not flexible enough to adapt to an adversarial and evolving landscape.

There have been various learning-based approaches to identify click fraud as well. TalkingData, a leading big data service platform in China, conducted a click adfraud detection challenge on Kaggle (Talking Data 2017). Top solutions in this competition utilized variants of tree based models such as Gradient Boosting Machines, Bagged Trees, etc. For example, (Mouawi et al. 2018) proposed and compared three models; KNN, SVMs and Artificial Neural Networks on aggregate features such as distinct IPs per publisher to identify click fraud. (Thejas et al. 2019) utilized GANs to produce adversarial data to boost performance of a supervised Neural Network model.

6.2 Weak Supervision

Weak Supervision deals with creating supervised models with labels that are noisy and limited. In the past, various solutions have suggested that a relatively good model can be constructed with a combination of such weak labels. To the best of our knowledge, we are not aware of any work that has utilized weak labels to build models that have been deployed at scale to combat click fraud detection. Weak Supervision has been used in less adversarial settings such as Fake News Detection on Twitter (Helmstetter and Paulheim

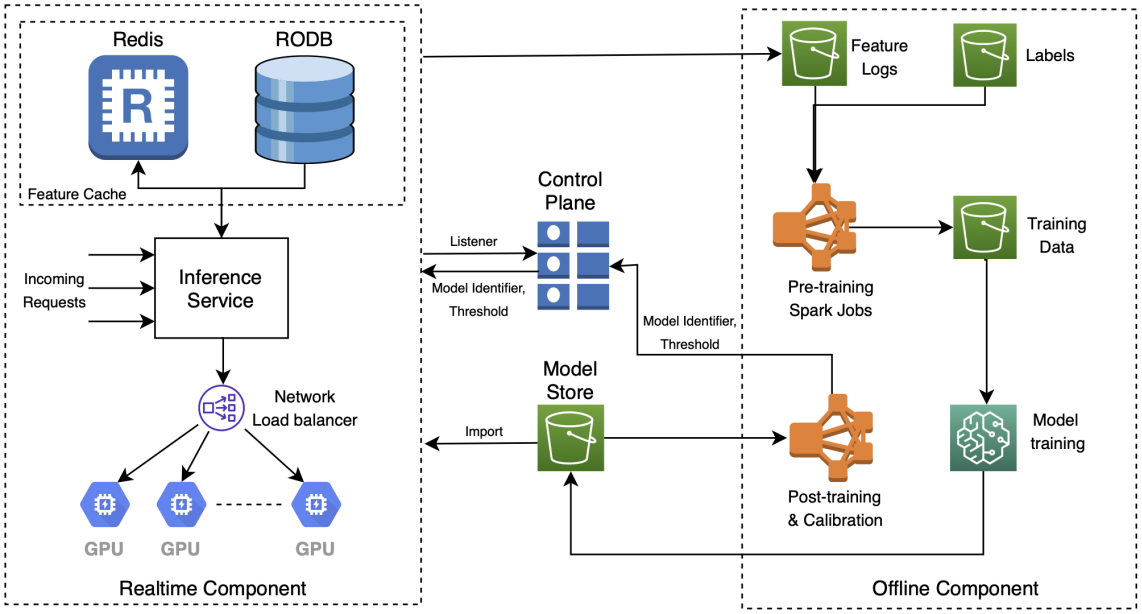


Figure 5: Deployment Diagram

2018) or elsewhere (Yuan et al. 2020). Weak Labels have also been used in computer vision for identifying image-level labels for tasks such as Object Detection on Microsoft COCO (Oquab et al. 2015) and in scenarios where highly precise labels are very difficult to obtain such as computation pathology (Campanella et al. 2019) and remote sensing (Han et al. 2015). Weak labels act as a replacement for highly precise labels since they can be obtained relatively easily. Snorkel (Ratner et al. 2017), was designed with a similar philosophy and is a system to create training datasets with weak labels and enables users to write labelling functions based on heuristics.

6.3 Optimizing ML Models on Subsets of Data

Slice-level Optimization Often in practical settings, datasets have critical slices on which performance needs to be maintained. Optimizing for an overall objective may not necessarily perform well on these individual slices. (Chen et al. 2019) studied this problem where they introduced they concept of "slicing" functions which identify the data subsets and developed an approach where the model learns expert slice-level representations and these expert representations are combined through attention mechanisms. (Wang et al. 2021) proposed an enhancement over the slice-based learning model by utilizing a mixture-of-attentions to learn slice-aware dual-attentive representations. In our setting, the dataset slices are relatively strongly defined (examples are desktop or mobile-app) and hence we do not require the concept of slicing functions. We find that optimizing decision thresholds of our model for particular slices does not deplete overall model performance and at the same time calibrates the model well for particular slices, similar to (Chen et al. 2019).

Fairness in ML Models Recently, ML models have been shown to learn certain biases that are particular to certain subsets in data and were learnt as a result of an overall learning framework (Corbett-Davies and Goel 2018). Issues and problems surrounding this have been studied under the umbrella of *Fairness in Machine Learning Systems*. Note that fairness usually defines these data subsets around human attributes such as gender, ethnicity, etc., but our data subsets (traffic slices) are defined from a standpoint of the types of devices that send traffic and placements for ads. Through the optimization framework described in this paper and slice-level calibration, we ensure that the SLIDR system avoids biases and offers comparable protection to traffic across various slices. Recently (Salvador et al. 2021) followed a similar approach of slicing datasets and identified sensitive subgroups through clustering of image features and calibrated a face verification model on the FPR incurred on these subgroups.

7 Concluding Remarks

We presented SLIDR – a neural network based model for large-scale realtime robot detection across traffic slices. We showed that the model trained using hand-crafted features with multiple weak labels, when combined with sample weighing and traffic slice level calibration can be highly effective in detecting click robots on online ads. In the future, we plan to make the model more feature-rich with end-to-end learned representations for users, IPs, UserAgents, search queries, etc. With the increased number and types of features, we plan to experiment with advanced neural architectures like cross-networks that can effectively capture feature interactions in tabular data.

References

Campanella, G.; Hanna, M. G.; Geneslaw, L.; Miraflor, A.; Werneck Krauss Silva, V.; Busam, K. J.; Brogi, E.; Reuter, V. E.; Klimstra, D. S.; and Fuchs, T. J. 2019. Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. *Nature medicine*, 25(8): 1301–1309.

Chen, V.; Wu, S.; Ratner, A. J.; Weng, J.; and Ré, C. 2019. Slice-based learning: A programming model for residual learning in critical data slices. *Advances in neural information processing systems*, 32.

Corbett-Davies, S.; and Goel, S. 2018. The measure and mismeasure of fairness: A critical review of fair machine learning. *arXiv preprint arXiv:1808.00023*.

Fu, A.; Narasimhan, B.; and Boyd, S. 2020. CVXR: An R Package for Disciplined Convex Optimization. *Journal of Statistical Software*, 94: 1–34.

Han, J.; Zhang, D.; Cheng, G.; Guo, L.; and Ren, J. 2015. Object Detection in Optical Remote Sensing Images Based on Weakly Supervised Learning and High-Level Feature Learning. *IEEE Transactions on Geoscience and Remote Sensing*, 53(6): 3325–3337.

Helmstetter, S.; and Paulheim, H. 2018. Weakly Supervised Learning for Fake News Detection on Twitter. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 274–277.

Mouawi, R.; Awad, M.; Chehab, A.; Hajj, I. H. E.; and Kayssi, A. 2018. Towards a Machine Learning Approach for Detecting Click Fraud in Mobile Advertizing. In *2018 International Conference on Innovations in Information Technology (IIT)*, 88–92.

Oentaryo, R.; Lim, E.-P.; Finegold, M.; Lo, D.; Zhu, F.; Phua, C.; Cheu, E.-Y.; Yap, G.-E.; Sim, K.; Nguyen, M. N.; et al. 2014. Detecting click fraud in online advertising: a data mining approach. *The Journal of Machine Learning Research*, 15(1): 99–140.

Oquab, M.; Bottou, L.; Laptev, I.; and Sivic, J. 2015. Is object localization for free? - Weakly-supervised learning with convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 685–694.

Ratner, A.; Bach, S. H.; Ehrenberg, H.; Fries, J.; Wu, S.; and Ré, C. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, 269. NIH Public Access.

Salvador, T.; Cairns, S.; Voleti, V.; Marshall, N.; and Oberman, A. 2021. FairCal: Fairness Calibration for Face Verification. *arXiv preprint arXiv:2106.03761*.

Stone-Gross, B.; Stevens, R.; Zarras, A.; Kemmerer, R.; Kruegel, C.; and Vigna, G. 2011. Understanding Fraudulent Activities in Online Ad Exchanges. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, 279–294. New York, NY, USA: Association for Computing Machinery. ISBN 9781450310130.

Talking Data. 2017. TalkingData AdTracking Fraud Detection Challenge. https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection. Accessed: 2022-11-28.

Thejas, G.; Boroojeni, K. G.; Chandna, K.; Bhatia, I.; Iyengar, S.; and Sunitha, N. 2019. Deep learning-based model to fight against ad click fraud. In *Proceedings of the 2019 ACM southeast conference*, 176–181.

Wang, C.; Lee, S.; Park, S.; Li, H.; Kim, Y.-B.; and Sarikaya, R. 2021. Learning slice-aware representations with mixture of attentions. *arXiv preprint arXiv:2106.02363*.

Yeh, I.; and Lien, C. 2018. UCI Machine Learning Repository: Default of Credit Card Clients Data Set. https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients. Accessed: 2022-11-28.

Yuan, C.; Ma, Q.; Zhou, W.; Han, J.; and Hu, S. 2020. Early detection of fake news by utilizing the credibility of news, publishers, and users based on weakly supervised learning. *arXiv preprint arXiv:2012.04233*.