Learning to Defer under Expert Drift

Anonymous Author
Anonymous Institution

Abstract

The *learning to defer* (L2D) framework enables safety in machine learning and AI systems by allocating difficult or critical decisions to a human expert. However, prior work on L2D assumes fixed expert reliability, overlooking temporal fluctuations in human performance due to factors such as fatigue and cognitive biases. Humans commonly exhibit such substantial, systematic performance shifts in high-stakes domains such as radiology, aviation, and driving. We propose *Expert Drift Adapted Learning-to-Defer* (EDA-L2D), a novel expert-aware L2D framework that explicitly models these temporal shifts by conditioning the deferral head on recent histories of expert outcomes via a sequence model. This history-aware approach enables adaptive deferral policies that anticipate reliability fluctuations and better allocate decisions between human and machine over time. We provide comprehensive experiments across 3 diverse tasks - image classification, medical diagnosis, and hate speech detection - showing that for a time-dependent expert, our approach consistently achieves higher performance and better deferral rates than prior L2D approaches.

1 INTRODUCTION

Hybrid intelligent (HI) systems combine human and machine intelligence to achieve goals unattainable by either alone, emphasizing complementarity over replacement (Kamar, 2016; Dellermann et al., 2019; Akata et al., 2020). One popular HI paradigm is *learning to defer* (L2D): the system learns not only to predict a label but also to decide when to hand off the decision to a human expert. The original formulation of L2D shows that accounting for the expert's strengths and biases can improve overall accuracy and fairness relative to a static rejection framework (Madras et al., 2018).

However, existing L2D systems assume that expert behavior is fixed or stationary. In practice, human performance often varies significantly over time due to fatigue (Figalová et al., 2024; Peukert et al., 2023; Pan et al., 2022; Bruni et al., 2012) and various cognitive biases (Legler et al., 2025; Urai et al., 2019; van de Wouw et al., 2024). We show real-world evidence of this effect on a widely studied computer vision task, CIFAR10, by analyzing real human label data from over 2,500 human annotators in the CIFAR10H dataset (Joshua et al., 2019). We show in Fig. 1 that there exists a slight but consistent decline in annotator accuracy over time: the more examples the annotators label, the less accurate they are at producing correct labels. This is a notable observation: CIFAR10 is a relatively simple task (image classification over only 10 classes of common objects), yet even on such a simple task, we observe annotator fatigue. This motivates developing a L2D approach that can account for these structured temporal shifts, in turn allowing it to naturally model and adapt to real human experts.

We address this challenge by proposing a novel framework for L2D: *Expert-Drift-Adapted Learning to Defer* (EDA-L2D). Our framework explicitly models nonstationarity in expert performance. We modify L2D's rejector sub-component to condition on short-horizon histories of expert decisions. This allows the system to be aware of the expert's current behavior, and modify its deferral policy accordingly. We illustrate our approach Fig. 2a. Concretely, we integrate a sequence model into the deferral head to to explicitly model expert reliability over time. We validate our approach across diverse tasks including medical image recognition, hate speech detection, and image classification. We demonstrate consistent gains in system accuracy over existing, time-agnostic L2D approaches. In summary, our contributions are as follows:

Preliminary work. Under review by AISTATS 2026. Do not distribute.

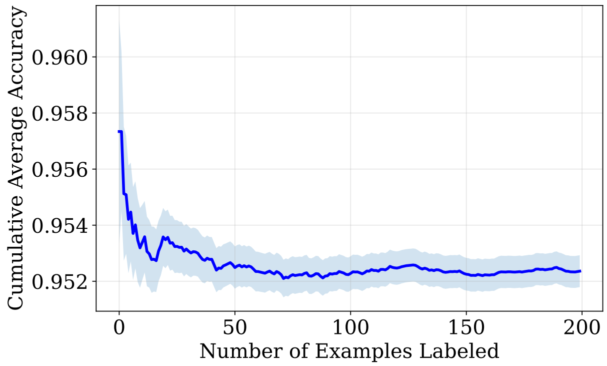


Figure 1: *Annotator Fatigue in CIFAR-10-H.* After removing attention checks, we compute the average cumulative accuracy across all 2,555 human annotators at each time-step. We observe a slight but clear downward drift in performance over time.

- We propose a novel *Expert-Drift-Adapted Learning to Defer* (EDA-L2D) that explicitly models variance in human expert performance over time, allowing the system to flexibly adapt to realistic human experts. We propose our approach in §3, including a new time-aware learning objective (§3.2) and model formulation (§3.3).
- We validate the approach with simulated, time-dependent experts on real-world tasks that require HI solutions (§5.1, §5.2, §5.3), and show that this time-variance is present in real human-annotated data (Fig.1).

2 BACKGROUND

We first describe the traditional L2D setting (Mozannar and Sontag, 2021), in which the expert's abilities are assumed to be static and time-invariant.

2.1 Data & Models

We will exclusively consider the problem of multi-class classification with a feature space $\mathcal{X}$ and the label space $\mathcal{Y} = \{1, \dots, K\}$. Given some input $x \in \mathcal{X}$, we define $\mathbb{P}(\mathrm{y} \mid x)$ as the unknown label-generating distribution and $\mathbb{P}(\mathrm{m} \mid x)$ as the unknown distribution over expert predictions $m \in \mathcal{Y}$. L2D is comprised of two sub-components: a *classifier* $h : \mathcal{X} \to \mathcal{Y}$ and a *rejector* $r : \mathcal{X} \to \{0, 1\}$, which decides whether to make a prediction using the classifier $(r(x) = 0)$ or by deferring to the expert $(r(x) = 1)$.

2.2 Classifier-Rejector Loss Function

To train our L2D model, we need to fit both the rejector and classifier models. The classifier incurs a loss of zero (correct) or one (incorrect) when it makes a prediction. Similarly, the human expert incurs the same 0-1 loss when making a prediction (i.e. $r(\boldsymbol{x}) = 1$). Combining these two 0-1 losses using the rejector model results in the combined classifier-rejector loss:

$$L_{0-1}(h, r) = \mathbb{E}_{\mathbf{x},\mathrm{y},\mathrm{m}}\left[(1 - r(\mathbf{x}))\ \mathbb{I}[h(\mathbf{x}) \neq \mathrm{y}]\ +\ r(\mathbf{x})\ \mathbb{I}[\mathrm{m} \neq \mathrm{y}]\right] \quad (1)$$

where $\mathbb{I}$ denotes an indicator function checking the prediction against the ground-truth label. When minimizing this loss, the resulting Bayes optimal classifier and rejector satisfy:

$$h^*(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}}\ \mathbb{P}(\mathrm{y} = y|\boldsymbol{x}),$$
$$r^*(\boldsymbol{x}) = \mathbb{I}\left[\mathbb{P}(\mathrm{m} = \mathrm{y}|\boldsymbol{x}) \geq \max_{y \in \mathcal{Y}} \mathbb{P}(\mathrm{y} = y|\boldsymbol{x})\right], \quad (2)$$

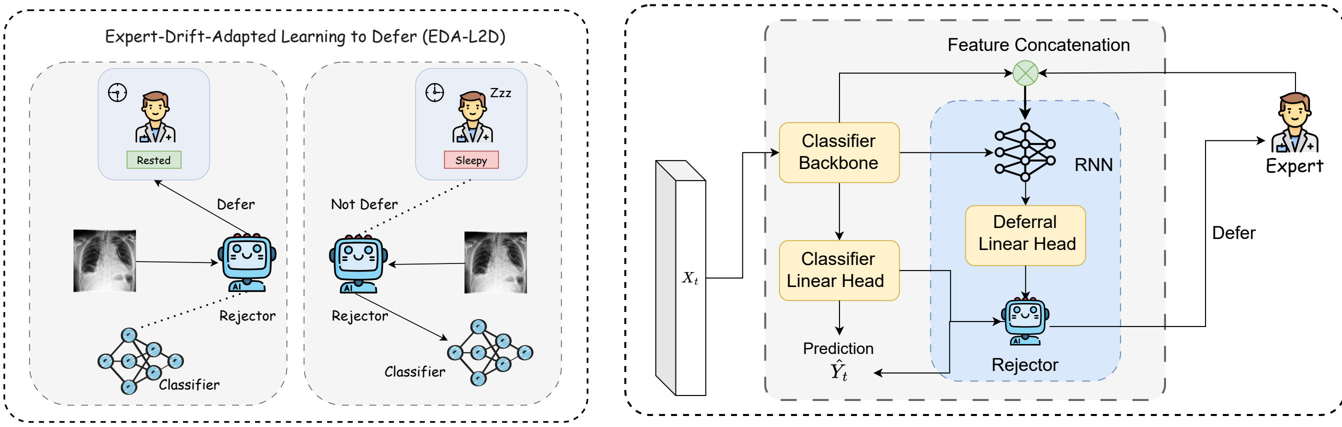
where $\mathbb{P}(\mathrm{y}|\boldsymbol{x})$ represents the label probability under the data generating process, and $\mathbb{P}(\mathrm{m} = \mathrm{y}|\boldsymbol{x})$ is the probability of the expert making the correct prediction. The expert may have knowledge not available to the classifier, so they could outperform the Bayes optimal classifier.

2.3 Softmax Surrogate Loss

A consistent surrogate loss for the above L_{0-1} loss can be derived following the formulation from Mozannar and Sontag (2021). First, we consider an augmented label space that includes both the label space $\mathcal{Y}$ and the rejection option $\perp$: $\mathcal{Y}^{\perp} := \mathcal{Y} \cup \{\perp\}$. Secondly, for a class $k \in [1, K]$, let $g_k : \mathcal{X} \mapsto \mathbb{R}$, and let $g_{\perp} : \mathcal{X} \mapsto \mathbb{R}$ denote the rejection option. We can combine these $K + 1$ with a loss resembling the cross-entropy loss for a softmax parameterization:

$$\phi_{\mathrm{SM}}(g_1, \dots, g_K, g_{\perp}; \boldsymbol{x}, y, m) = - \log\left(\frac{\exp\{g_y(\boldsymbol{x})\}}{\sum_{y' \in \mathcal{Y}^{\perp}} \exp\{g_{y'}(\boldsymbol{x})\}}\right) - \mathbb{I}[m = y]\ \log\left(\frac{\exp\{g_{\perp}(\boldsymbol{x})\}}{\sum_{y' \in \mathcal{Y}^{\perp}} \exp\{g_{y'}(\boldsymbol{x})\}}\right). \quad (3)$$

Here, the first term maximizes g_k for the true label k. The second term maximizes the rejection function $g_{\perp}$, but only when the expert's prediction is correct. At test time, the classifier takes the maximum over the classes: $\hat{y} = h(\boldsymbol{x}) = \arg\max_{k \in [1,K]} g_k(\boldsymbol{x})$. Similarly, we formulate the rejection function as $r(\boldsymbol{x}) = \mathbb{I}[g_{\perp}(\boldsymbol{x}) \geq \max_k g_k(\boldsymbol{x})]$. The minimizers $g_1^*, \dots, g_K^*, g_{\perp}^*$ of ϕ_{SM} also uniquely minimize the 0-1 loss from Equation 1, $L_{0-1}(h, r)$.



(a) Learning to Defer with a Fatigued Expert

(b) EDA-L2D System Architecture

Figure 2: (a) An example use case of our EDA-L2D framework in a setting with expert performance drift: a radiologist whose performance degrades over a workday. Our model monitors past outcomes to infer current expert reliability in real time, deferring to the expert when reliable and assigning cases to the model when the expert's performance declines. (b) Overview of our expert drift adapted L2D framework. A streaming sequence of inputs is encoded by the backbone into features. At each timestep, the feature vector is concatenated with the expert's previous prediction and passed to an RNN module. The classifier head outputs class logits and the defer head outputs a deferral score; a deferral gate then assigns the instance to either the expert or the classifier. Because the expert's competence drifts over time, this design makes the L2D system time-aware and adaptive to temporal shifts.

3 EXPERT-DRIFT-ADAPTED LEARNING TO DEFER

This section introduces our *Expert-Drift-Adapted Learning to Defer* (EDA-L2D) framework, which extends L2D to account for non-stationarity in expert performance. Accordingly, we derive the corresponding softmax surrogate loss for this setting. We also propose multiple parameterizations, including one that uses a recurrent neural network to incorporate short-horizon temporal context into the deferral policy.

3.1 Setting: Non-Stationary Expert

We consider the same setting as Section 2—L2D for multi-class classification—except now we assume the expert is non-stationary. Thus, instead of a fixed distribution $\mathbb{P}(\mathrm{m} \mid x)$, the expert's predictions are generated by $\mathbb{P}_t(\mathrm{m}_t \mid x)$, with the subscript $t \in \mathcal{T}$ denoting a time index. Note that we are *assuming the classification task itself is stationary*, i.e. $\mathrm{y} \sim \mathbb{P}(\mathrm{y} \mid x)$, which still has no time dependence.

To ground the notation in an example, consider an L2D system for radiology. The problem of forming a diagnosis or other prediction from the medical image itself is a static task, since we assume the distribution of patients and the mechanism that govern their health is not changing over time (or is at least changing so slowly as to be negligible, e.g. drift in hospital patient demographics). Yet the radiologist who is paired with the L2D system will change, at the very least becoming more tired and distracted over the course of a 10+ hour shift. The expert drift may also be non-monotonic: perhaps her performance degrades over the course of a morning, but after lunch and a mid-day walk, the expert feels rejuvenated for a few more hours.

Returning to the technical details, our EDA-L2D system will have a classifier that is defined just as above: $h : \mathcal{X} \to \mathcal{Y}$. Again, the classifier is time-invariant since the prediction task itself is not time dependent. The difference, however, arises in the rejector: $r : \mathcal{X} \times \mathcal{T} \mapsto \{0, 1\}$, meaning the rejector is a function of both the feature space $\mathcal{X}$ and time $\mathcal{T}$. While in the simplest case $\mathcal{T}$ would be a scalar time index, we could also formulate $\mathcal{T}$ as a feature space that describes the current state of the expert (e.g. tabular biomarkers).

3.2 Learning Objective and Bayes Solutions

The learning objective is similar to that in Section 2 (equation 1), except now it takes the form of a summation over time:

$$L_{0-1}^{\mathcal{T}}(h, r) = \sum_{t \in \mathcal{T}} \mathbb{E}_{\mathbf{x}, \mathrm{y}, \mathrm{m}_t} \Big[(1 - r(\mathbf{x}, t))\, \mathbb{I}[h(\mathbf{x}) \neq \mathrm{y}] + \; r(\mathbf{x}, t)\, \mathbb{I}[\mathrm{m}_t \neq \mathrm{y}] \Big]. \tag{4}$$

As the distribution of y does not depend on time, the classifier's Bayes solution is just as before: $h^*(\boldsymbol{x}) =$

$\arg\max_{y\in\mathcal{Y}}\ \mathbb{P}(\mathrm{y} = y \mid \boldsymbol{x})$. Yet the Bayes optimal rejector does change, becoming time-dependent like so:

$$r^*(\boldsymbol{x}, t) = \mathbb{I}\left[\mathbb{P}_t(\mathrm{m}_t = \mathrm{y} \mid \boldsymbol{x}) \ \geq\ \max_{y\in\mathcal{Y}} \mathbb{P}(\mathrm{y} = y \mid \boldsymbol{x})\right].$$

The intuition is the same as above—compare the chance of expert correctness vs classifier confidence for the modal label—yet now the expert correctness term is time dependent, $\mathbb{P}_t(\mathrm{m}_t = \mathrm{y} \mid \boldsymbol{x})$.

3.3 Implementations of the Softmax Surrogate Loss

The derivation for the corresponding softmax surrogate loss follows fairly directly from Mozannar and Sontag (2021)'s original derivation. Yet we consider two distinct parameterizations of the underlying model. The first is a 'per time step' version that treats the L2D problem independently at each time step. We consider this the most naive extension of traditional L2D that yet still satisfies our motivating setting of expert drift. We then go on to describe how to use a recurrent neural network (RNN) to parameterize the rejector, which allows the expert's previous predictions to help inform the deferral policy. For all implementations, we assume we have access to a training set $\mathcal{D} = \left\{\{(\boldsymbol{x}_{t,n}, y_{t,n}, m_{t,n})\}_{n=1}^{N}\right\}_{t\in\mathcal{T}}$, which consists of N feature-label-expert-prediction triplets at each of $|\mathcal{T}|$ time steps. Again, we emphasize that only the underlying distribution of $m_{t,n}$ is time-varying, and the time indices on $\boldsymbol{x}_{t,n}, y_{t,n}$ are there merely to 'synchronize' the feature-label pairs with the expert predictions.

Per-Time-Step Implementation and Loss The softmax surrogate in equation 3 can be naturally extended to the drifting-expert setting by instantiating a traditional L2D model at each time step. Specifically, we define a model at each time step and train in locally for that time step via the surrogate:

$$\begin{aligned}\phi^t_{\mathrm{SM}}(g_1,..., g_K, g^t_\perp; \boldsymbol{x}, y, m_t) = \\ -\log\frac{\exp\{g_y(\boldsymbol{x})\}}{\sum_{y'\in\mathcal{Y}^\perp}\exp\{g_{y'}(\boldsymbol{x})\}} \\ -\mathbb{I}[m_t = y]\log\frac{\exp\{g^t_\perp(\boldsymbol{x})\}}{\sum_{y'\in\mathcal{Y}^\perp}\exp\{g_{y'}(\boldsymbol{x})\}}.\end{aligned} \tag{5}$$

Notice that $g^t_\perp$ is specific to the current time step, but $g_1, \ldots, g_K$ could be shared across all time steps (since they encode the classifier). As mentioned above, we consider this a naive implementation and will primarily use it to benchmark our second approach, described below. Having $g^t_\perp(\mathbf{x})$ only be a function of $\mathbf{x}$ means that it ignores patterns that could be detected by having information about the expert's behavior at previous time steps. Moreover, this implementation has the clear drawback that it cannot generalize to time steps that were not seen in the training set. For instance, if all training sets had $|\mathcal{T}| = 10$, denoting that the expert worked a 10-hour shift. Then it is not clear how to deploy and use the model when the expert could for a longer period of time than 10 hours.

Loss for RNN-Based Model We next consider a more sophisticated implementation that uses an RNN-based parameterization to incorporate information about the expert's previous predictions. Here we use a surrogate loss defined across all time steps:

$$\begin{aligned}&\phi^{\mathcal{T}}_{\mathrm{SM}}(g_1, ..., g_K, g_\perp; \{\boldsymbol{x}_t, y_t, m_t\}_{t\in\mathcal{T}}) = \\ &\sum_{t\in\mathcal{T}} -\log\frac{\exp\{g_{y_t}(\boldsymbol{x}_t)\}}{\exp\{g_\perp(\boldsymbol{x}_T, t)\} + \sum_{y'\in\mathcal{Y}}\exp\{g_{y'}(\boldsymbol{x}_t)\}} \\ &-\mathbb{I}[m_t = y]\log\frac{\exp\{g_\perp(\boldsymbol{x}_t, t)\}}{\exp\{g_\perp(\boldsymbol{x}_T, t)\} + \sum_{y'\in\mathcal{Y}}\exp\{g_{y'}(\boldsymbol{x}_t)\}}.\end{aligned} \tag{6}$$

where here $g_\perp(\boldsymbol{x}_t, t)$ is parameterized with an RNN so that it can leverage information from previous time steps. While we have left the time feature t vague, this will contain the expert's previous prediction m_{t-1} when it is available. We next describe the neural architecture in more detail.

RNN Architecture The RNN first takes as input the feature vector $\mathbf{x}_t$, producing a hidden representation $\boldsymbol{f}_t = \psi_0(\boldsymbol{x}_t)$. For instance, if $\mathbf{x}_t$ is an image, then $\phi(\cdot)$ could be a convolutional NN. The feature representation is then passed into a recurrent module: $\boldsymbol{z}_t = \psi_1\big(\boldsymbol{z}_{t-1}, \boldsymbol{f}_t, \hat{m}_{t-1}\big)$. During training we use teacher forcing for $\hat{m}_{t-1} \in \{0, 1\}$ (expert correct/incorrect at $t-1$); at test time it is computed from realized routing and the expert's actual outcome. Lastly, the g-functions found in Equation 6 are produced by a final linear transformation: $g_k(\boldsymbol{x}_t) = \boldsymbol{w}_k^\top \boldsymbol{z}_t \quad (k \in 1, \ldots, K])$ and $g_\perp(\mathbf{x}_t) = \boldsymbol{w}_\perp^\top \boldsymbol{z}_t$. The predicted label is chosen via $\arg\max_k g_k(\mathbf{x}_t)$. $g_\perp(\boldsymbol{x}_t)$ quantifies the preference for deferral, with the system deferring if $g_\perp(\boldsymbol{x}_t) \geq \max_k g_k(\mathbf{x}_t)$.

The intuition behind this formulation is that the classifier and rejector should not operate solely on the current input but instead incorporate temporal context that reflects the expert's evolving reliability. By conditioning the recurrent state $\boldsymbol{z}_t$ on both the encoded input $\boldsymbol{f}_t$ and the previous correctness indicator $\hat{m}_{t-1}$, the system adaptively tracks fluctuations in expert performance. The classifier head then focuses on predicting the label from this time-dependent state, while the deferral head estimates whether the expert should be trusted at the current step.

4 RELATED WORK

Learning to Defer (L2D) Early work on classifiers that can choose to reject or abstain instead of making a prediction (Chow, 1957) inspired more recent work on how to model what happens after the classifier abstains (Madras et al., 2018), with Mozannar and Sontag (2021) deriving the first consistent surrogate loss for the problem. This work has been expanded in recent years to address several limitations, including mis-calibration (Verma and Nalisnick, 2022; Cao et al., 2023), underfitting (Narasimhan et al., 2022), realizable consistency (Mozannar et al., 2023), sample complexity (Charusaie et al., 2022), data scarcity (Hemmer et al., 2023), and deferral to multiple experts (Verma et al., 2023). However, these approaches do not consider the 'messiness' of modeling *human* experts—as we do—and could just as well be applied to a black-box API.

L2D for Sequences and Populations L2D has also been extended to the difficult settings of sequential tasks and encountering novel experts. Regarding the former, *Sequential Learning-to-Defer* (Joshi et al., 2023) incorporates temporal dynamics by framing deferral as a sequential decision-making problem, employing model-based reinforcement learning. This work, however, is primarily concerned with the sequential nature of the prediction task itself and not robustness to expert drift, which is our primary concern. Regarding the latter, *Learning to Defer to a Population* (Tailor et al., 2024) uses meta-learning to adapt to never-before-seen experts at test-time. Their motivation is to be robust to novel experts (with the novelty not being a function of time), whereas we are concerned about modeling the temporal drift of a known expert.

Temporal Shifts in Human Performance In many high-stakes domains, human performance shift over time (due to fatigue or other cognitive factors) are well-documented and widely studied. Fatigue is extremely common in high-stakes domains and long-horizon tasks, such as radiology (Bruni et al., 2012), automated driving (Figalová et al., 2024), air traffic control (Peukert et al., 2023), and flying planes (Pan et al., 2022). Beyond affecting surface-level performance on these tasks, fatigue physically affects the brain, causing reduced alertness and vigilance that can be measured directly, e.g. using EEG signals (Peukert et al., 2023). We extend L2D to be adaptive to these variations in human performance.

5 EXPERIMENTS

We evaluate our proposed EDA-L2D framework through a progression of controlled simulations. We train the two variants of EDA-L2D (per-step and RNN-based) on three benchmarks spanning distinct modalities and annotation regimes: **CIFAR-10** (image classification), **CheXpert** (chest X-ray classification) and **HateSpeech** (text moderation). Our primary baseline is "Native L2D", meaning that we apply a traditional L2D system that has no awareness of time, collapsing all training data and presenting it to the model without an information about time ordering.

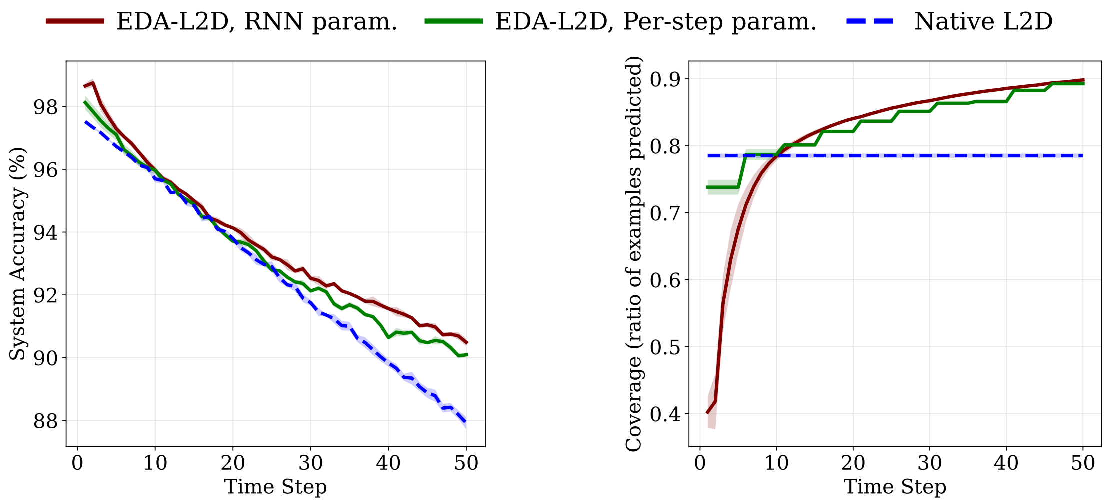
5.1 CIFAR-10

Task and Data. We study standard image classification on CIFAR-10 (Krizhevsky, 2009). The dataset contains 60,000 color images at 32×32 resolution from ten object categories (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck), with 50,000 training images and 10,000 test images. Given an input image x, the goal is to output either a predicted class label or defer the decision to the expert. For our experiments, we first arrange the 50,000 training images into sequences and perform a 90/10 split into training and validation sets at the sequence level, ensuring that each sequence appears in exactly one set. The official 10,000-image test set is used for final evaluation.

Synthetic Expert In this experiment, we constructed a simple synthetic expert to provide a controlled baseline. Following the setup of Mozannar and Sontag (2021), we assume that the expert initially has full knowledge of all classes, i.e., K=10. The expert's accuracy is then designed to decrease linearly over time from 100% accuracy at the beginning to 50% accuracy at T=50. At each timestep, the expert's prediction is sampled according to a Bernoulli trial with the corresponding time-dependent accuracy, thereby generating stochastic outputs that reflect the intended accuracy trajectory.

Classifier and EDA-L2D architecture. For the classifier, we adopt WRN-28-4 with standard CIFAR-10 normalization. For deferral, we take the features output by the classifier, concatenate them with a one-step indicator of whether the expert was correct at t-1 and a normalized timestep indicator, and pass this concatenated vector through a 1-layer GRU with 256 hidden units. The GRU output is then fed to a fully connected deferral head that produces a single defer logit.

Training. Training proceeds in two stages. First, the backbone are optimized with cross-entropy, using SGD (momentum 0.9, weight decay 5×10^{-4}), cosine-annealed learning rate, batch size 128, and no dropout. The model minimizes cross-entropy for 200 epochs; this yields 90.27% test accuracy on CIFAR-10. We then



(a)

(b)

Figure 3: Plot of system accuracy (a) and coverage(proportion of classifier taken examples) (b) comparing our methods with the baseline on the CIFAR-10 test set across 50 time steps. We evaluate using a toy expert whose accuracy linearly decays from 100% to 50%. Error bars represent standard deviations over 10 runs.

fine-tune the entire network end-to-end using the L2D softmax surrogate loss: the classifier backbone continues training with SGD using cosine annealing over 25000 total iterations, while GRU and the two classifier heads use Adam driven by a custom cosine-then-hold schedule implemented via LambdaLR (cosine anneal for the first 25 epochs, then hold for the remaining 25 of a 50-epoch run).

Results. In this experiment we considered a simple expert model with linearly decaying accuracy from 100% to 50%. Since the classifier achieves an accuracy of 90.47%, we would expect fewer samples to be deferred to the expert once the expert's performance falls below the classifier's. Our results confirm this intuition: as shown in Figure 3b, at time t=10 our RNN-based EDA-L2D and the per-step EDA-L2D adapt their deferral rate accordingly. Moreover, Figure 3a demonstrates that EDA-L2D achieves higher overall system accuracy in both settings. These findings indicate that our method makes more appropriate allocations both before and after the crossing point, correctly assigning more samples to the expert when its accuracy is higher and shifting them to the classifier once the expert's performance declines.

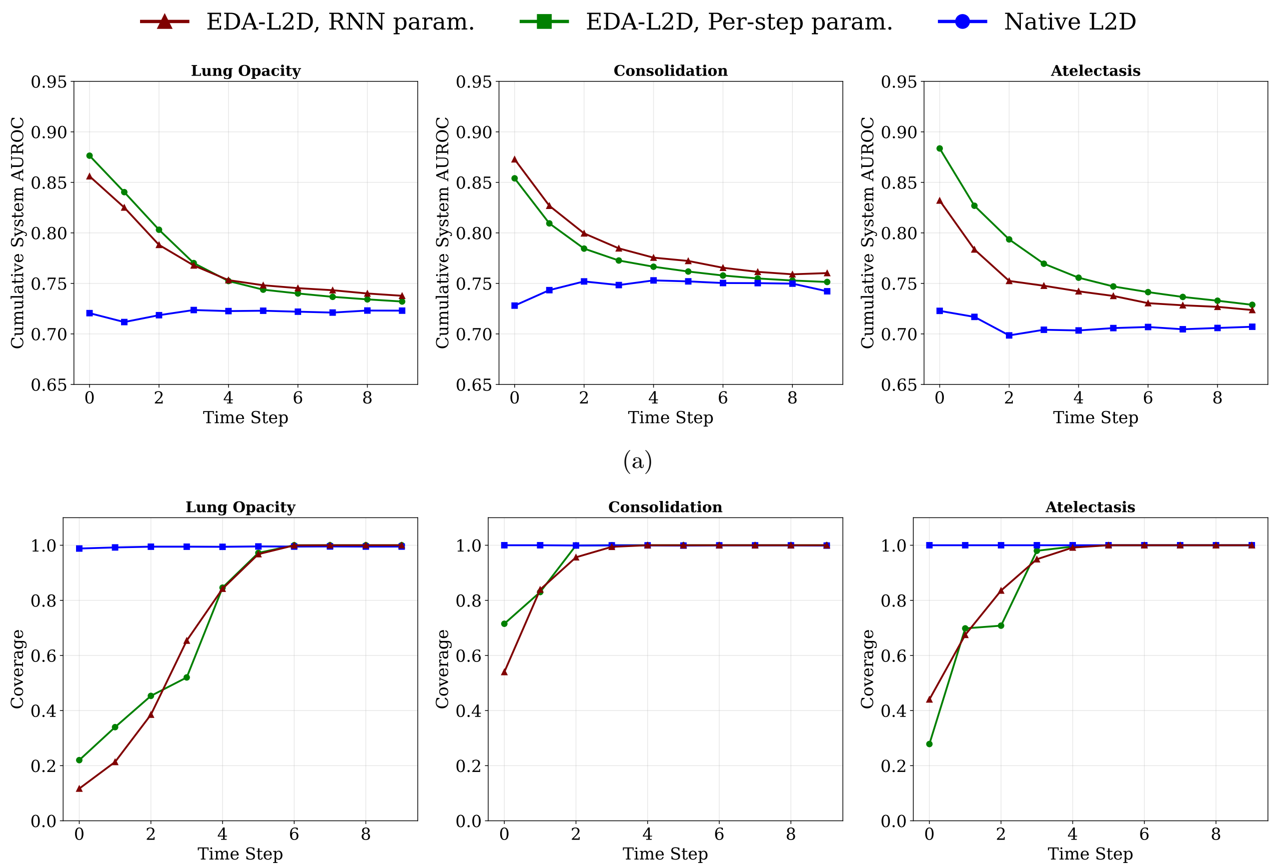
5.2 CheXpert

Task. We study chest X-ray classification on CheXpert(Irvin et al., 2019) with the standard 14 observations, framing the problem as *per-task, per-timestep* binary decisions over image sequences of length T. At each timestep $t \in \{1, \ldots, T\}$ and for each task $k \in \{1, \ldots, 14\}$, the system must either output a class prediction or defer to an external expert.

Expert. Following Mozannar and Sontag (2021), we simulate a class-dependent, time-varying expert with two accuracy levels, p and q. A designated confounding class receives a lower base accuracy, while all other classes receive a higher one. Over time, both accuracies decay linearly at fixed rates, with a small positive floor to avoid vanishing performance. In our experiments, we set the base accuracies to p=1.0 (non-confounding) and q=0.7 (confounding), apply decay rates of d=0.05 and δ=0.035, and T=10. The expert label is generated as Bernoulli(q_t) if the example is confounding and Bernoulli(p_t) otherwise.

Data. We use the downsampled CheXpert release and keep 14 one-vs-rest targets and a binary mask that suppresses tasks marked uncertain or missing under CheXpert's label encoding. All images are converted to three channels, normalized, and resized to ImageNet-compatible resolution; training augmentation uses random resized crops, horizontal flips, and random rotations up to 15°. Dataset splits are patient-disjoint with an 80/10/10 train/validation/test partition.

Classifier and EDA-L2D architecture. Following Irvin et al. (2019), we use a DenseNet-121 backbone initialized with ImageNet pretraining. We add two heads to the classifier backbone. The first is a per-class classifier that takes the CNN feature at time step t and outputs two logits (negative vs. positive). The second is a one-layer LSTM with 1 024 hidden units



(a)

(b)

Figure 4: Plot of system accuracy (a) and coverage(ratio of classifier-taken examples) (b) comparing our methods with the baseline on the CheXpert dataset. For each timestep t, we evaluate on sequences built from a 10% patient-ID subset, using a fair expert whose accuracy decays linearly from 100% to 50%.

that takes the CNN feature together with the expert's previous predictions and produces one deferral logit per class. At inference, for each class, we form a three-way distribution—negative (no disease), positive (diseased), and defer—and we defer to the expert whenever the defer score exceeds the larger of the two class scores; otherwise, we use the classifier's prediction.

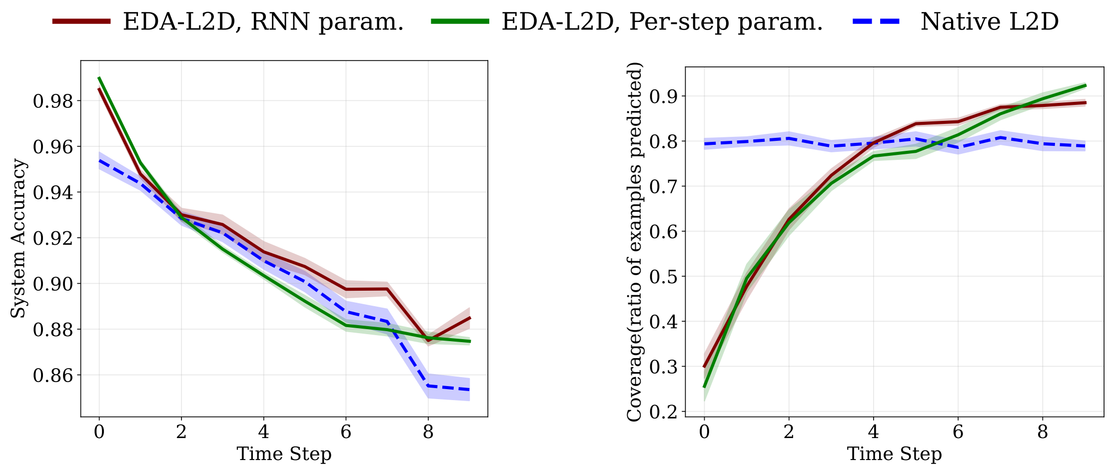
Training. We train the model in two stages over a total of four epochs. In the first stage, we pretrain the CNN and classification head with standard cross-entropy for three epochs, averaging the loss over samples. In the second stage, we fine-tune the entire network end-to-end for one epoch with L_{CE}, accumulating losses across each sequence and normalizing by its length T. All four epochs use Adam (learning rate 1×10^{-4}, weight decay 1×10^{-5}) with a Reduce-on-Plateau scheduler.

Results In Fig. 4, we compare our method with the baseline L2D approach and the per-step EDA-L2D on the CheXpert dataset with our synthetic expert. On the Lung Opcacity and Ateletasis tasks, our approach clearly outperforms native L2D and performs on par with the per-step EDA-L2D; on the Consolidation task, RNN-based EDA-L2D outperforms both the per-step EDA-L2D and native L2D.

5.3 Hate Speech

We study detection of abusive content on Twitter posts using the Davidson et al. (2017) corpus of 24,783 English tweets annotated into three mutually exclusive classes: *hate speech*, *offensive but not hate*, and *neither*.

Expert. We follow prior work that uses the Twitter-AAE lexical model to probabilistically identify tweets written in African-American English (AAE) and binarize group membership with a 0.5 threshold(Blodgett et al., 2016). We instantiate a synthetic fair expert whose accuracy is identical across demographics. At timestep t, the expert's correctness probability decays linearly from 1.0 to 0.5 over the sequence. For each



(a) (b)

Figure 5: Plot of system accuracy (a) and coverage(ratio of classifier–taken examples) (b) comparing our methods with the baseline on the HateSpeech dataset across 10 time steps. We simulate using a toy expert whose accuracy decays linearly from 100% to 50% over 10 timesteps. Results are averaged over 10 runs, with error bars showing standard deviations.

tweet, we sample a Bernoulli variable to decide whether the expert outputs the ground-truth label; if not, the expert predicts uniformly at random among the remaining classes. We construct sequences of length T=10 for evaluation.

Classifier architecture. Our hate-speech classifier models are lightweight TextCNNs. Tweets are tokenized with spaCy and mapped to a 25k vocabulary; embeddings are initialized with GloVe-6B (100d) and fine-tuned. The encoder applies three parallel convolutional filters of widths 3, 4, and 5 (300 channels each) over the embedding matrix, followed by ReLU and max-over-time pooling. The pooled features are concatenated, passed through dropout (0.5), and fed to a linear classifier.

EDA-L2D architecture. We employ a TextCNN encoder to obtain a per-timestep feature representation for each tweet in a length-T sequence. To model temporal dependencies and expert reliability, we concatenate a binary indicator of the expert's correctness at the previous timestep to the CNN feature and feed the resulting vector into a single-layer LSTM with 256 hidden units. The TextCNN encoder feeds a fully connected classification head that outputs three logits (hate, offensive, neither), while the LSTM feeds a fully connected deferral head that outputs a single defer logit. Concatenating these yields a four-way prediction over hate, offensive, neither, defer. At inference, we defer to the expert when the defer logit exceeds the maximum of the three class logits; otherwise, the classifier's prediction is returned.

Results. From Fig. 5a, we observe that our model trained on the HateSpeech dataset demonstrates a clear advantage. In terms of system accuracy, our method consistently outperforms both the native L2D and per-step EDA-l2D approach, with an average margin of 1.26% and 0.70%, respectively. Moreover, from Fig. 5b, we observe that our method identifies more favorable timesteps than the per-step model at which the system coverage surpasses that of the general baseline, resulting in higher system accuracy at those points.

6 CONCLUSION & FUTURE WORK

We propose *Expert-Drift-Adapted Learning to Defer (EDA-L2D)*, a framework that enables improved generalization to real human experts by explicitly modeling temporal drift in their performance—variance that is common and caused by a wide variety of factors, such as fatigue and cognitive biases. We achieve this by explicitly incorporating time and prediction history into the L2D model, allowing the model to learn representations of the expert that vary with each time-step and dynamically adapt to the expert's behavior to achieve improved deferral decisions and greater human-AI team performance. In future work, we aim to extend the framework to capture complex cognitive biases, such as the availability heuristic (Tversky and Kahneman, 1973), gambler's fallacy (Kovic and Kristiansen, 2019), delay discounting (Kurth-Nelson et al., 2012), and the recency bias (Turvey and Freeman, 2012).

References

Zeynep Akata, Dan Balliet, Maarten de Rijke, Frank Dignum, Virginia Dignum, A. E. Eiben, Antske Fokkens, Davide Grossi, Koen V. Hindriks, Holger Hoos, Haley Hung, Catholijn Jonker, Christof Monz, Mark Neerincx, Frans Oliehoek, Henry Prakken, Stefan Schlobach, Linda C. van der Gaag, Frank van Harmelen, Herke van Hoof, M. Birna van Riemsdijk, Aimee van Wynsberghe, Rineke Verbrugge, Bart Verheij, Paul Vossen, and Max Welling. A research agenda for hybrid intelligence: Augmenting human intellect with collaborative, adaptive, responsible, and explainable artificial intelligence. *Computer*, 53 (8):18–28, 2020. doi: 10.1109/MC.2020.2996587. URL https://www.computer.org/csdl/magazine/co/2020/08/09153877/1lUB5gL2CnS.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. Demographic dialectal variation in social media: A case study of african-american english, 2016. URL https://arxiv.org/abs/1608.08868.

Silvio G Bruni, Eric Bartlett, and Eugene Yu. Factors involved in discrepant preliminary radiology resident interpretations of neuroradiological imaging studies: a retrospective analysis. *American Journal of Roentgenology*, 198(6):1367–1374, 2012.

Yuzhou Cao, Hussein Mozannar, Lei Feng, Hongxin Wei, and Bo An. In defense of softmax parametrization for calibrated and consistent learning to defer, 2023. URL https://arxiv.org/abs/2311.01106.

Mohammad-Amin Charusaie, Hussein Mozannar, David Sontag, and Samira Samadi. Sample efficient learning of predictors that complement humans, 2022. URL https://arxiv.org/abs/2207.09584.

C. K. Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, EC-6(4):247–254, 1957. doi: 10.1109/TEC.1957.5222035.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language, 2017. URL https://arxiv.org/abs/1703.04009.

Dominik Dellermann, Philipp Ebel, Matthias Söllner, and Jan Marco Leimeister. Hybrid intelligence. *Business & Information Systems Engineering*, 61(5):637–643, 2019. doi: 10.1007/s12599-019-00595-2. URL https://link.springer.com/article/10.1007/s12599-019-00595-2.

Nikol Figalová, Hans Joachim Bieg, Michael Schulz, Jürgen Pichen, Martin Baumann, Lewis Chuang, and Olga Pollatos. Fatigue and mental underload further pronounced in l3 conditionally automated driving: Results from an eeg experiment on a test track. *ArXiv preprint arXiv:2405.18114*, 2024. L3 supervisor sleepiness increases with prolonged monitoring; EEG measures show drift in alertness.

Patrick Hemmer, Lukas Thede, Michael Vössing, Johannes Jakubik, and Niklas Kühl. Learning to defer with limited expert predictions, 2023. URL https://arxiv.org/abs/2304.07306.

Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison, 2019. URL https://arxiv.org/abs/1901.07031.

Shalmali Joshi, Sonali Parbhoo, and Finale Doshi-Velez. Learning-to-defer for sequential medical decision-making under uncertainty. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2109.06312.

C. Peterson Joshua M. Battleday Ruairidh L. Griffiths Thomas and Olga Russakovsky. Human uncertainty makes classification more robust. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9617–9626, 2019. doi: 10.1109/ICCV.2019.00971. URL https://arxiv.org/abs/1908.07086.

Ece Kamar. Directions in hybrid intelligence: Complementing ai systems with human intelligence. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, 2016. URL https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/HybridIntelligence.pdf. Position paper outlining research directions for human-AI systems.

Marko Kovic and Silje Kristiansen. The gambler's fallacy fallacy (fallacy). *Journal of Risk Research*, 22 (3):291–302, 2019. doi: 10.1080/13669877.2017.1378248. URL https://doi.org/10.1080/13669877.2017.1378248.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. Technical Report.

Zeb Kurth-Nelson, Warren Bickel, and A. David Redish. A theoretical account of cognitive effects in delay discounting. *European Journal of Neuroscience*, 35 (7):1052–1064, 2012. doi: https://doi.org/10.1111/j.1460-9568.2012.08058.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1460-9568.2012.08058.x.

Eric Legler, Darío Cuevas Rivera, Sarah Schwöbel, Ben J. Wagner, and Stefan Kiebel. Cognitive computational model reveals repetition bias in a sequential decision-making task. *Communications Psychology*, 3(1):92, 2025. doi: 10.1038/s44271-025-00271-0.

David Madras, Toniann Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer, 2018. URL https://arxiv.org/abs/1711.06664.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert, 2021. URL https://arxiv.org/abs/2006.01862.

Hussein Mozannar, Hunter Lang, Dennis Wei, Prasanna Sattigeri, Subhro Das, and David Sontag. Who should predict? exact algorithms for learning to defer to humans, 2023. URL https://arxiv.org/abs/2301.06197.

Harikrishna Narasimhan, Wittawat Jitkrittum, Aditya K Menon, Ankit Rawat, and Sanjiv Kumar. Post-hoc estimators for learning to defer to an expert. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 29292–29304. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/bc8f76d9caadd48f77025b1c889d2e2d-Paper-Conference.pdf.

T. Pan et al. Research on the identification of pilots' fatigue status based on fnirs signals and deep models. *Aerospace*, 9(3):173, 2022. doi: 10.3390/aerospace9030173. Identifies pilot fatigue using brain signals; performance drift with fatigue.

M. Peukert et al. Subjective and objective fatigue dynamics in air traffic control. *Journal of Sleep Research*, 2023. Showing increase in subjective fatigue and drop in vigilance toward end of evening shifts for air traffic controllers.

Dharmesh Tailor, Aditya Patra, Rajeev Verma, Putra Manggala, and Eric Nalisnick. Learning to defer to a population: A meta-learning approach, 2024. URL https://arxiv.org/abs/2403.02683.

B.E. Turvey and J.L. Freeman. Jury psychology. In V.S. Ramachandran, editor, *Encyclopedia of Human Behavior (Second Edition)*, pages 495–502. Academic Press, San Diego, second edition edition, 2012. ISBN 978-0-08-096180-4. doi: https://doi.org/10.1016/B978-0-12-375000-6.00216-0. URL https://www.sciencedirect.com/science/article/pii/B9780123750006002160.

Amos Tversky and Daniel Kahneman. Availability: A heuristic for judging frequency and probability. *Cognitive Psychology*, 5(2):207–232, 1973. ISSN 0010-0285. doi: https://doi.org/10.1016/0010-0285(73)90033-9. URL https://www.sciencedirect.com/science/article/pii/0010028573900339.

Anne E. Urai, Jan Willem de Gee, Konstantinos Tsetsos, and Tobias H. Donner. Choice history biases subsequent evidence accumulation. *eLife*, 8:e46331, 2019. doi: 10.7554/eLife.46331.

Didrika S. van de Wouw, Ryan T. McKay, and Nicholas Furl. Biased expectations about future choice options predict sequential economic decisions. *Communications Psychology*, 2(1):119, 2024. doi: 10.1038/s44271-024-00172-8.

Rajeev Verma and Eric Nalisnick. Calibrated learning to defer with one-vs-all classifiers, 2022. URL https://arxiv.org/abs/2202.03673.

Rajeev Verma, Daniel Barrejón, and Eric Nalisnick. Learning to Defer to Multiple Experts: Consistent Surrogate Losses, Confidence Calibration, and Conformal Ensembles. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, 2023.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]

   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]