
Keyframe-Focused Visual Imitation Learning

Chuan Wen^{*1} Jierui Lin^{*2} Jianing Qian³ Yang Gao^{1,4} Dinesh Jayaraman³

Abstract

Imitation learning trains control policies by mimicking pre-recorded expert demonstrations. In partially observable settings, imitation policies must rely on observation histories, but many seemingly paradoxical results show better performance for policies that only access the most recent observation. Recent solutions ranging from causal graph learning to deep information bottlenecks have shown promising results, but failed to scale to realistic settings such as visual imitation. We propose a solution that outperforms these prior approaches by upweighting demonstration keyframes corresponding to expert action changepoints. This simple approach easily scales to complex visual imitation settings. Our experimental results demonstrate consistent performance improvements over all baselines on image-based Gym MuJoCo continuous control tasks. Finally, on the CARLA photorealistic vision-based urban driving simulator, we resolve a long-standing issue in behavioral cloning for driving by demonstrating effective imitation from observation histories. Supplementary materials and code at: <https://tinyurl.com/imitation-keyframes>.

1. Introduction

Learning controllers for complex, unmodeled agents and environments is a challenging problem. For tasks where at least one “expert” controller exists, such as a human driver for autonomous driving, imitation learning offers a simple, powerful family of solutions that exploit demonstrations provided by this expert to bootstrap control policy learning. Many imitation approaches employ a straightforward “behavioral cloning” (BC) strategy, to train policies completely “offline”, i.e., with no environmental interaction, by simply mapping expert observations to expert actions on

the demonstration data. While BC has well-documented distributional shift issues due to compounding imitation errors when executed in the environment, several effective approaches have been proposed to address them, and BC remains widely used in practice (Pomerleau, 1989; Schaal, 1999; Muller et al., 2006; Mülling et al., 2013; Bojarski et al., 2016a; Giusti et al., 2015).

We focus on the open problem of effectively extending BC to realistic partially observed settings such as driving, where the agent cannot observe all task-relevant information instantaneously. This is commonly resolved in other controller design paradigms by integrating historical information in the control policy, but this has proven challenging in BC. For over 15 years now, researchers have reported seemingly paradoxical results that show performance drops in some POMDP settings from allowing BC agents to access history information, compared to when they are restricted to instantaneous observations alone (Muller et al., 2006; Bansal et al., 2019; de Haan et al., 2019; Wang et al., 2019). Recently, Wen et al. (2020) coined the phrase “copycat problem” to describe the issue, and show that the problem is wider still: even when history information does improve BC performance, the learned policies often perform suboptimally and have room to improve if the copycat problem is correctly addressed.

Figure 1 shows a snippet of an expert driving demonstration from the imitation dataset CARLA100 (Dosovitskiy et al., 2017; Codevilla et al., 2019), where a car waiting at a red traffic light starts to move when the light turns green. We can see that the expert’s action a_t is identical to its previous action a_{t-1} , except at one moment when the light turns green (figure shows throttle). Thus, a “copycat” policy that repeated the previous action without paying any attention to the images would only commit one imitation error on the expert data. Upon execution in the environment however, since no expert would be available, it would repeat its own previous action at each step, and never move at all! Indeed, Codevilla et al. (2019) report this special case of the copycat problem as the “inertia” problem.

We study the reasons for the copycat problem and identify one key reason that can be algorithmically addressed: when expert actions are highly temporally correlated, the demonstration dataset has a very tiny fraction of important

^{*}Equal contribution ¹Institute for Interdisciplinary Information Sciences, Tsinghua University ²University of Texas at Austin ³University of Pennsylvania ⁴Shanghai Qi Zhi Institute. Correspondence to: Dinesh Jayaraman <dineshj@seas.upenn.edu>.

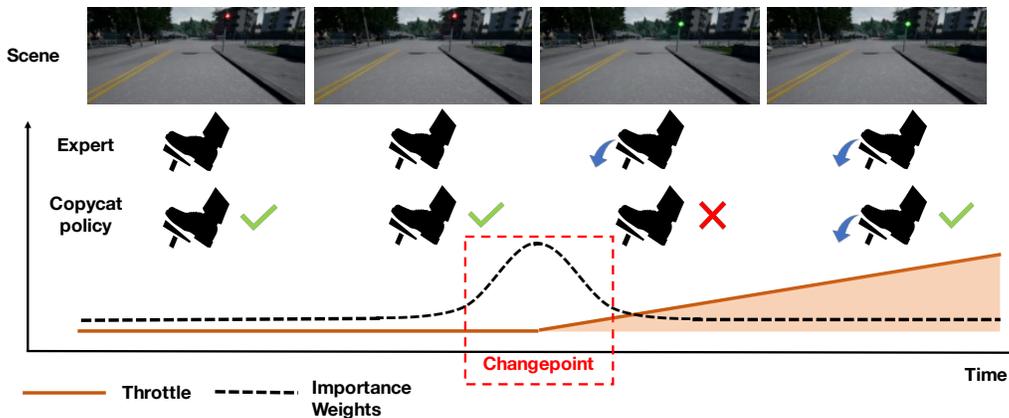


Figure 1. An instance of the copycat issue in the CARLA autonomous driving simulator. Views from the expert data show the policy waiting at a red light and then accelerating (throttle) when it turns green. A simple copycat policy is mostly correct but makes a mistake at this critical keyframe. We define a notion of changepoints to detect such keyframes and upweight them during behavior cloning.

“changepoint” samples that typically correspond to the expert responding to some external change in the observations, such as when the traffic light turns green. We then propose a simple, well-motivated metric to automatically identify these underrepresented changepoint samples in the demonstrations, and propose to upweight them in the behavioral cloning objective function for policy learning.

We evaluate our method across four varied simulated environments, ranging from robotic control from clean images, to photorealistic urban driving environments. Our experimental results validate that our method offers the most effective and scalable solution yet for tackling the copycat problem, while also being very simple to implement.

2. Related Work

Imitation Learning. Imitation learning is a powerful policy learning method that can learn complex decision behaviors from expert demonstrations (Widrow and Smith, 1964; Osa et al., 2018; Argall et al., 2009). In this paper, we focus on the widely used behavioral cloning paradigm of imitation, which directly regresses from observations to expert actions (Pomerleau, 1989; Schaal, 1999; Muller et al., 2006; Mülling et al., 2013; Bojarski et al., 2016a; Giusti et al., 2015). Like other imitation approaches, behavioral cloning must contend with distribution shift: small errors between imitator and expert policies accumulate over time leading the imitator into unfamiliar states (Ross et al., 2011). It is possible to resolve this through environmental interactions (Ho and Ermon, 2016; Brantley et al., 2020) or queryable experts (Ross et al., 2011; Sun et al., 2017; Laskey et al., 2017b; Sun et al., 2018). Our focus is on a specific well-documented problem arising due to distributional shift in partially observed imitation settings, recently coined the “copycat problem” (Wen et al., 2020). We dis-

cuss work specifically related to the copycat problem in detail in Sec 3.2, after setting the context.

Importance Weighting To Tackle Data Imbalance. Sample reweighting / resampling, a well-known technique in machine learning and statistics, has recently been shown to remain very effective at tackling long-tail problems arising from data imbalances in machine learning (Cui et al., 2019; Cao et al., 2019; Kang et al., 2020; Zhou et al., 2020). Wang et al. (2018) assume access to environmental rewards, and reweight training samples for imitation based on their corresponding value functions. While these approaches rely on “labels” such as category annotations or environmental rewards, we instead discover an unlabeled group of “changepoint” keyframes in imitation learning datasets. By identifying the scarcity of such frames as a data imbalance that causes copycat problems, we are able to propose a simple and surprisingly effective sample reweighting-based technique to alleviate them.

Shortcut Learning. With the increasingly widespread use of machine learning, researchers have begun to pay attention to various intriguing errors and quirks, particularly with deep neural networks (DNNs). DNN image classifiers often classify images based on irrelevant backgrounds rather than foregrounds (Beery et al., 2018) and object textures rather than shapes (Geirhos et al., 2019). Geirhos et al. (2020) recently surveyed several such phenomena, identifying them as instances of “shortcuts”: models that are *easy* to learn, perform well on the data they were trained on, but then fail to generalize to the real world. We view the copycat problem as another instance of the shortcut learning problem, identify conditions that lead to its emergence in imitation learning, and make progress towards alleviating it.

3. Preliminaries

We are interested in learning control policies in settings that can be modeled as partially observed Markov decision processes (POMDP). In POMDPs, the environment at time t provides to the agent a reward r_t , and an observation o_t which only partially represents its true state. To account for this missing state information, it is common practice (Murphy, 2000; Schulman et al., 2017) to augment the current observation o_t with the last H observations to form the “observation history” $\tilde{o}_t = [o_{t-H}, \dots, o_t]$. Optimal controllers that maximize the sum of environmental rewards¹ $R = \sum_t r_t$ must rely on this observation history \tilde{o}_t rather than solely on o_t .

3.1. Behavioral Cloning

While the above point about observation histories also holds for control policies synthesized through reinforcement learning or other approaches, we are interested in policies trained via imitation learning. In particular, we focus on the widely used behavioral cloning (BC) paradigm, which reduces imitation to simple supervised learning to mimic expert actions. Specifically, an expert policy π_e , such as a human demonstrator, generates training demonstrations $\mathcal{D} = \{(o_t, a_t)\}_{t=1}^N$. The goal of BC is to train a parameterized policy $\pi_\theta(\tilde{o}_t) = \hat{a}_t$ that estimates the expert’s action at time t . To do this, BC methods typically minimize the following mean squared error (MSE) loss on \mathcal{D} :

$$\arg \min_{\theta} \text{MSE}_{\mathcal{D}}(\theta) = \frac{1}{N} \sum_{t=0}^N (\pi_{\theta}(\tilde{o}_t) - a_t)^2. \quad (1)$$

3.2. The “Copycat” Problem in Behavioral Cloning

As mentioned above, optimal controllers typically require historical information to account for partial observability. Therefore, we would expect BC policies with access to the observation history \tilde{o}_t (“BC-OH”) to perform better than those that map a single observation o_t to a_t (“BC-SO”). Yet, in practice, many prior works (Muller et al., 2006; Bansal et al., 2019; de Haan et al., 2019; Wen et al., 2020; Codevilla et al., 2019; Wang et al., 2019) report that BC-OH performs poorly compared to BC-SO. In particular, BC-OH produces better (lower) values of the BC loss in Eq (1) on both training and validation data from expert demonstrations, but performs poorly when actually executed in the environment. In recent attempts to deal with this issue, it has variously been identified as the “copycat” problem (Wen et al., 2020), the “inertia” problem (Codevilla et al., 2019), and “causal confusion” (de Haan et al., 2019): an imitator exploits the strong temporal correlation of expert actions to learn policies that predict a_t purely as a function of previous actions a_{t-1}, a_{t-2}, \dots . Wen et al. (2020) make two

important observations that widen the scope of the copycat problem: (1) Even when history information does improve BC performance as we would expect, the learned policies often perform suboptimally and have room to improve if the copycat problem is correctly addressed. (2) Even when past actions are not explicitly available as input, the imitator commonly learns to recover them from the observation history \tilde{o}_t and manifest the copycat problem.

4. Method

We now analyze the copycat problem and identify its key causes. Motivated by this analysis, we then propose a simple approach that aims to resolve the problem by reweighting training data samples based on the temporal characteristics of expert action sequences.

4.1. What Causes the Copycat Problem?

We argue that the copycat problem arises from (A) strong temporal correlation among expert actions, (B) misalignment between environmental reward and the imitation objective, and (C) the difficulty of learning truly optimal policies that fit the expert data. First, temporal correlation makes it possible for a “copycat policy” $\psi(a_{t-1}, a_{t-2}, \dots)$ that relies purely on expert action histories to produce low MSE for predicting expert actions a_t on expert demonstrations (training as well as held-out data) without accessing environmental observations at all. Second, the well-documented distributional shift problem in imitation (Ross et al., 2011), compounded by misalignment between the MSE objective and the true environment reward R , means that $\psi(\cdot)$ yields low rewards upon environmental execution. And finally, it is difficult to learn a “good” policy that correctly identifies and relies on the causes of expert actions among the observations. This means that the copycat policy offers an excellent “shortcut” (Geirhos et al., 2020) to the BC learner. We expand further on these intuitions below.

Suppose we train an optimal copycat policy $\psi^*(\cdot)$ on the training dataset through behavioral cloning as:

$$\psi^* = \arg \min_{\psi} \frac{1}{N} \sum_{t=0}^N (\psi(a_{t-1}, a_{t-2}, \dots) - a_t)^2. \quad (2)$$

Suppose further that the expert data has a fraction ϵ_{CP} of “changepoint” frames for which a_t is not predicted well by the optimal copycat policy $\psi^*(\cdot)$. For convenience, we will assume these samples all suffer from uniform copycat error equal to 1, so that the training MSE of $\psi^*(\cdot)$ is the changepoint fraction ϵ_{CP} . Low ϵ_{CP} corresponds to low-MSE copycats. This relates to A above.

Next, we turn our attention to the reward-optimal policy parameters θ^{R*} corresponding to the policy within the model class, that yields the highest environmental reward R . Ob-

¹ignoring discount factors for simplicity

serve that, in general, θ^{R^*} does not fit the expert data perfectly. In other words, it produces a non-zero training error $\text{MSE}_{\mathcal{D}}(\theta^{R^*}) > 0$. This happens due to the misalignment issue (**B** above), and optimization difficulties, model class mismatch or noisy demonstrations (**C** above).

When we synthesize the above observations, a clear-cut domain for the copycat problem begins to emerge. BC learners will always prefer the copycat solution ψ^* over the reward-optimal parameters if:

$$\text{MSE}_{\mathcal{D}}(\theta^{R^*}) > \epsilon_{CP}, \quad (3)$$

or in other words, the BC training loss is lower for the copycat ψ than it is for $\pi_{\theta^{R^*}}$.² Note that we operate in data-rich settings without overfitting, so that all the above statements about training errors also hold for validation errors. So, to restate, copycat problems occur when the changepoint fraction is lower than the error of the reward-optimal imitator.

While the above argument is not fully rigorous or comprehensive,³ it yields strong intuitions for the factors that make copycat problems more likely in POMDP imitation: **(1)** the higher the temporal correlation among expert actions, the more infrequent the changepoints (i.e., lower ϵ_{CP}), and **(2)** the harder the imitation setup (such as high-dimensional observations or noisy demonstrations), the higher the value of $\text{MSE}_{\mathcal{D}}(\theta^{R^*})$. In both cases, the copycat-producing inequality in Eq (3) becomes more likely to hold.

These observations directly motivate our approach. We assume fixed standard datasets, architectures, and optimizers in this paper, so we cannot address **(2)** above. However, we can artificially inflate the changepoint fraction ϵ_{CP} simply by upweighting changepoints when setting up the BC objective, to address **(1)**. This is the crux of our approach, which we describe in more detail next.

4.2. Reweighted Behavioral Cloning Objective

In datasets with high temporal correlation among expert samples, the natural changepoint fraction ϵ_{CP} is very low, which makes copycat issues more likely, as we have argued above. However, we can effectively upsample these changepoints by shifting to a *weighted* version of the behavioral cloning objective in Eq (1):

$$\theta^* = \arg \min_{\theta} \sum_{t=0}^N w_t (\pi_{\theta}(\tilde{o}_t) - a_t)^2, \quad (4)$$

²Since ψ is typically a very simple function, we implicitly make the assumption that the learning algorithm can easily find parameters θ such that $\pi_{\theta}(\cdot) = \psi^*(\cdot)$.

³In particular, Eq (3) does not rule out that that may be other parameter vectors $\theta \neq \theta^{R^*}$ that yield higher reward and produce lower error than the copycat $\psi^*(\cdot)$.

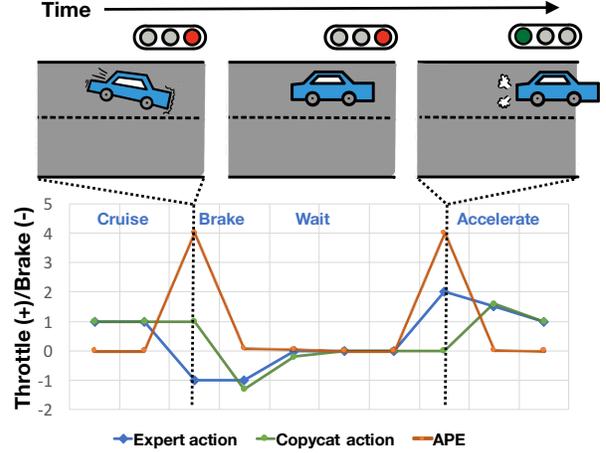


Figure 2. Action prediction error (APE) computation (Equation (5)) as a function of expert and copycat actions in a traffic light setting similar to Figure 1. APE peaks align with keyframes.

where w_t is the weight for each data sample. With the right weighting scheme, the reweighted MSE error of the copycat policy ϵ_{CP} would rise and therefore making the condition in Eq (3) more difficult to meet, alleviating the copycat problem.

4.3. Action Prediction Error (APE)

What would an appropriate sample weighting scheme look like? Since the copycat problem arises from exploiting temporal correlation among expert actions, we must up-weight and emphasize those keyframe samples where this correlation breaks down. Identifying such samples amounts to a type of changepoint detection in the expert action sequence. While many generic changepoint and keyframe detection approaches have been proposed for time series or video (van den Burg and Williams, 2020; Sheng et al., 2019), in our specialized setting, the most appropriate choice is a changepoint detection score that is closely tied to the copycat policy defined in Eq (2), as foreshadowed in Sec 4.1. Specifically, we first train a small MLP copycat policy network ψ^* with the training objective of Eq (2) — recall that the only inputs to this policy are the past actions $[a_{t-1}, a_{t-2}, \dots]$. Then, we use its prediction error for each training sample to set the sample weight w_t in the reweighted BC objective of Eq (4). These weights need only be computed once, before training the BC policy $\pi_{\theta}(\tilde{o}_t)$.

In more detail, for every training sample $(\tilde{o}_t, a_t) \sim \mathcal{D}$, we define the “action prediction error” (APE) as the squared error of the copycat policy ψ^* with respect to expert actions:

$$\text{APE}_t = (\psi^*(a_{t-1}, a_{t-2}, \dots) - a_t)^2. \quad (5)$$

Figure 2 shows a schematic. To avoid copycat overfitting

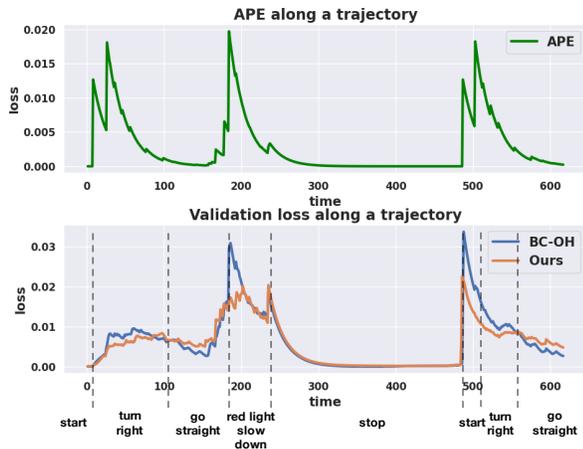


Figure 3. Action prediction error (APE) and behavioral cloning loss of BC-OH and our approach along a validation driving trajectory in CARLA. Dotted lines segment the trajectory into different annotated phases. APE is well-aligned with BC-OH errors.

when working with small datasets, the APE can instead be computed through cross-validation, always training copycat policies and measuring their errors on disjoint data.

Samples with high APE_t are more likely to be changepoints. Since we would like to upweight changepoints, we set the sample weights w_t in Eq (4) to be monotonic non-decreasing functions of APE_t :

$$w_t = f(APE_t). \quad (6)$$

Figure 3 shows a plot of the APE and the MSE loss for BC-OH along a validation trajectory, for a driving policy in a photorealistic image-based driving environment. Sample-wise BC-OH errors align very well with the APE, which are the copycat errors, verifying the existence of the copycat issue. We evaluate setting $f(\cdot)$ to softmax and step functions in our experiments. Plugging this back into Eq (4), all that remains is to train the BC policy by solving:

$$\theta^* = \arg \min_{\theta} \sum_{t=0}^N f(APE_t) (\pi_{\theta}(\tilde{o}_t) - a_t)^2. \quad (7)$$

Algorithm 1 summarizes our complete approach. Intuitively, our approach amounts to focusing the behavioral cloning objective on the demonstration frames where copycat policies fail, so that the BC learner becomes less likely to discover such copycat policies.

4.4. Implementation Details

Our copycat policy network ψ^* is a two-layer MLP. For $f(\cdot)$, we experiment with softmax and step functions. The softmax function is applied within each training mini-batch, i.e., $w_i = \frac{e^{\tau APE_i}}{\sum_j e^{\tau APE_j}}$; the temperature τ is a hyperparameter. The step function assigns a constant weight

Algorithm 1 Keyframe-Focused Visual Imitation Learning

- 1: **Input:** Expert demonstrations $\mathcal{D} = \{(\tilde{o}_t, a_t)\}$.
- 2: Train an optimal copycat policy MLP ψ^* on \mathcal{D} (Eq (2)).
- 3: Compute APE_t for each sample in \mathcal{D} (Eq (5)).
- 4: Compute the sample weights w_t (Eq (6)).
- 5: Optimize the imitation policy neural net π_{θ} to minimize the reweighted behavioral cloning objective (Eq(4)).
- 6: **Return** π_{θ}

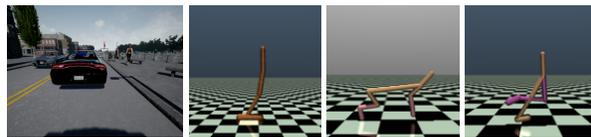


Figure 4. The four environments used in our experiments: CARLA, Hopper, HalfCheetah and Walker2d.

$w_i = W$ to samples in the top THR percentile of APE and $w_i = 1$ otherwise; W and THR are hyper-parameters. All hyperparameters were set through a simple grid search; see Supp for details. All policies using observation histories are trained by stacking image observations along the channels dimension. Architectural details are environment-specific and discussed in Sec 5.

Our method introduces barely any computational overheads over baseline behavioral cloning (BC-OH). At test time, our method is exactly identical to BC-OH. At training time, the only extra steps are training the copycat policy ψ^* and calculating the sample weights, before following the BC-OH training procedure. Since the inputs to ψ^* are only the action histories, rather than the visual observations, this all amounts to a fast data preprocessing step (less than 15 mins even on our largest and most complex environments). Further, once this is completed, any number of policies may be trained on that data with zero additional overhead.

5. Experimental Setup

We now comprehensively evaluate our approach on a photorealistic driving simulator, CARLA (Dosovitskiy et al., 2017), and three image-based OpenAI Gym MuJoCo robotics environments.

CARLA. CARLA is a photorealistic urban driving simulator with varying road and traffic conditions. It has recently emerged as a standard testbed for visual imitation learning, through the publicly available 100-hour CARLA100 driving dataset (Codevilla et al., 2019). This dataset is generated by a PID expert controller with access to simulator states. We use the hardest CARLA100 benchmark, *NoCrash-Dense*, which has the most pedestrians and traffic. We set history size $H = 6$. For each method, we train three policies from random initializations, and evaluate each policy three times

to account for environmental stochasticity. We measure the mean and standard deviation of four metrics: *%success*, *#collision*, *%progress* and *avg. speed*. *%success* is the number of test episodes correctly completed by the agent, *#collision* counts the times the agent crashes into pedestrians, vehicles and other obstructions, *%progress* measures the fraction of the distance traveled towards a goal location, and *avg. speed* is the average speed at which the agent drives. All metrics are measured on 100 predefined benchmark test episodes. More details in Supp.

Note that CARLA100 is a particularly challenging testbed because it applies the best known techniques for alleviating distributional shift issues in offline imitation, namely, noise injection (Laskey et al., 2017a) which is an offline counterpart of DAGGER (Ross et al., 2011), and multi-camera data augmentation (Bojarski et al., 2016b; Giusti et al., 2015). Further, all approaches use the speed prediction regularization scheme introduced in Codevilla et al. (2019) to partially address the copycat problem (coined there as the “inertia problem”). Finally, we train all approaches with Imagenet-pretrained Resnet-34 backbones (Codevilla et al., 2019) and weighted control losses (Codevilla et al., 2018) to reflect the state of the art. See Supp. More broadly, autonomous driving is the setting in which prior works have most often reported severe copycat issues (Muller et al., 2006; Bansal et al., 2019; Codevilla et al., 2019; Wang et al., 2019). Any persistent copycat issues in CARLA thus represent a key open problem in imitation learning.

MuJoCo-Image (Hopper, HalfCheetah, Walker2D). Following previous work that had identified environments where the copycat problem arises (de Haan et al., 2019; Wen et al., 2020), we evaluate our method in three standard OpenAI Gym MuJoCo continuous control environments: Hopper, HalfCheetah and Walker2D. We set the observation o_t to be the 128x128 RGB image of the environment, naturally excluding velocity and force information and making the environments partial observed. We set the history size $H = 1$, so that $\tilde{o}_t = [o_{t-1}, o_t]$. These tasks vary in their state and action spaces, environmental dynamics, and reward structure. We generate expert data from a TRPO policy (Schulman et al., 2015) with access to true states (1k samples for HalfCheetah, and 20k for Hopper and Walker2D). For each imitation method, we train three policies from random initializations and report the reward mean and standard deviation. See Supp for hyperparameters and training details.

5.1. Baselines and Ablations

We compare our method against the following baselines:

Behavioral Cloning (BC-SO and BC-OH). As introduced in Sec 3.2, BC-SO and BC-OH are BC with a single observation and observation histories respectively.

HistoryDropout. Bansal et al. (2019) proposed to randomly drop out the historical part of the observations to tackle copycat problems in imitation for driving. We implement this baseline by adding a dropout layer to the past observations, i.e. o_{t-1}, o_{t-2}, \dots .

Fighting-Copycat-Agents (FCA). Wen et al. (2020) proposed to remove all information about the last action a_{t-1} from an embedding of the observation history, using adversarial learning. They report promising results in low-dimensional state-based environments, and we extend their publicly available code to our image-based settings, with upgraded backbone networks and re-tuned hyperparameters. See Supp for details.

DAGGER. This is a widely used method to mitigate distributional shift issues in imitation learning (Ross et al., 2011). While our method operates completely offline, DAGGER requires online environmental interaction with a queryable expert. Nevertheless, it provides a useful comparison point. We set the number of environment queries to 100 and 1k for the MuJoCo environments and 120K for CARLA.

We also attempted to compare against de Haan et al. (2019), which, like DAGGER, proposes an online approach that targets “causal confusion”, a more general version of the copycat problem. However, their causal graph learning method, demonstrated with up to 30 observation dimensions at most, does not scale to our image-based settings.

Aside from these standard and published baselines for imitation learning, we also study three ablations of our approach, replacing our APE-based sample reweighting with alternatives: (1) **BCPD** (Xuan and Murphy, 2007) represents the widely used family of Bayesian changepoint detection techniques (Adams and MacKay, 2007; Fearnhead, 2006) for general multivariate time series, (2) **ActFreq** clusters expert actions in the training data to form action “categories” before applying category frequency-based sample reweighting, a standard approach for handling imbalanced data (Bowyer et al., 2011; Dong et al., 2017; Cui et al., 2019; Cao et al., 2019; Kang et al., 2020; Zhou et al., 2020), and (3) **Boosting** uses the standard Adaboost (Freund and Schapire, 1997) scheme for iteratively training BC-OH policies and upweighting high error samples. See Supp for more details about these ablations.

6. Results and Analysis

We now report the results of experiments performed to answer: (1) Does our method improve visual behavior cloning from observation histories? (2) Does it handle changepoints well? (3) To what extent does it reduce distributional shift issues in the learned policies? (4) Do our policies behave less like copycat policies?, and (5) When do our policies perform worse than the baselines?

Question 1. *Does our method improve visual behavior cloning from observation histories?*

CARLA. See Table 4 for %success results, and Supp for other metrics. The single-frame imitator BC-SO performs significantly better than BC-OH, illustrating the copycat problem. Our method easily outperforms all history-based baselines, including, surprisingly, even DAGGER which has the advantage of 120k expert queries! As we show in Supp, DAGGER does drive at higher average speed (18.5 km/h vs. 14.9 km/h), but at the cost of many more collisions (60 vs. 43) than our method. On other metrics (#collision and %progress), our method is comfortably best. Of the three sample reweighting ablations, BCPD performs the best, but still produces worse results compared to BC-OH without any sample reweighting, and falls far short of our approach.

However, even with these large gains over history-based baselines, our approach only recovers the performance of the single-frame imitator BC-SO — we do not significantly surpass it. Specifically, we get comparable %success, %progress, and avg. speed with higher consistency (lower variance) and fewer #collisions. We believe the limited extent of these gains may be because this setting does not emphasize information integration over time. CARLA-100 data (Codevilla et al., 2019) is collected largely in low traffic settings where the ego-agent’s own speed might very well be the main historical information missing in the current image observation. However, CARLA-100 provides the velocity as part of the observation, i.e., $o_t = [\text{image}_t, \text{velocity}_t]$. Thus, BC-SO already has access to agent velocity, meaning that the environment is nearly fully observed.

CARLA-w/o-speed. To understand why the CARLA setting does not reward agents that condition on multiple frames, we report results in a modified setting, CARLA-w/o-speed, where we withhold the ego-agent velocity from the observation for all methods. See Table 4 (right). The main differences from above are: (1) BC-SO is dramatically worse than before, (2) BC-OH improves significantly over BC-SO, and (3) our method improves by a large margin over BC-OH and therefore over BC-SO. These findings suggest that the ego-agent velocity does indeed encapsulate most of the driving-relevant information contained in \tilde{o}_t , and our method makes significant progress towards recovering this information from the frame history. Our approach continues to comprehensively outperform all history-based baselines. All three alternative sample reweighting schemes all continue to perform poorly in this setting. See Supp for other metrics, which are consistent with the above results.

Figure 5 shows an example test sequence from CARLA where BC-OH speeds straight into a slow-moving car in front of it, while our policy correctly slows down as the car nears, to avoid crashing.

Table 1. CARLA %success (\uparrow). More metrics in Supp.

| METHOD | CARLA | CARLA-W/O-SPEED |
|----------------|--------------------------------------|--------------------------------------|
| BC-SO | 42.667 \pm 8.668 | 9.222 \pm 2.380 |
| BC-OH | 33.000 \pm 4.190 | 25.667 \pm 0.981 |
| OURS (STEP) | 43.444 \pm 0.786 | 36.778 \pm 5.808 |
| FCA | 35.667 \pm 3.559 | 27.444 \pm 4.113 |
| HISTORYDROPOUT | 34.000 \pm 2.625 | 25.333 \pm 5.375 |
| DAGGER (120K) | 35.222 \pm 3.067 | 28.333 \pm 3.496 |
| BCPD | 28.667 \pm 2.494 | 20.000 \pm 1.414 |
| ACTREQ | 20.333 \pm 5.825 | 14.667 \pm 1.764 |
| BOOSTING | 3.000 \pm 1.414 | 10.0 \pm 2.160 |

Table 2. MuJoCo-Image environment rewards (\uparrow).

| METHOD | HOPPER | HALFCHEETAH | WALKER2D |
|----------------|-------------------------------|------------------------------|------------------------------|
| BC-SO | 601 \pm 168 | 4 \pm 5 | 481 \pm 40 |
| BC-OH | 740 \pm 35 | 615 \pm 41 | 614 \pm 107 |
| OURS (STEP) | 905 \pm 135 | 470 \pm 205 | 654 \pm 53 |
| OURS (SOFTMAX) | 951\pm117 | 819\pm96 | 769\pm97 |
| FCA | 735 \pm 106 | 270 \pm 168 | 534 \pm 99 |
| HISTORYDROPOUT | 617 \pm 111 | 96 \pm 40 | 594 \pm 61 |
| DAGGER (100) | 745 \pm 157 | 936 \pm 86 | 598 \pm 26 |
| DAGGER (1K) | 1034 \pm 45 | 822 \pm 186 | 699 \pm 111 |

Hopper, HalfCheetah, Walker2D. See Tab 2. BC-OH does manage to yield higher rewards than BC-SO in these settings, but it is further improved by addressing the copycat problem. We experimented with two simple choices of monotonic transformations $f(\cdot)$ applied to APE in Eq (6), namely step and softmax — softmax performs consistently better. While step is arguably a simpler weighting scheme, we find that softmax enjoys the benefits of easier hyperparameter tuning since it requires only a single temperature hyperparameter. Compared to all the other offline baselines, Ours (Softmax) easily yields the highest rewards across all environments. With the advantage of online interaction and expert queries, DAGGER with 100 queries is worse than our method on Hopper and Walker2D but better on HalfCheetah. With 1k queries, DAGGER performs marginally better than our method on all three environments.

Question 2. *Does our method imitate the expert better at the changepoints, as it was designed to do? What about non-changepoints?*

We showed earlier in Figure 3 that high APE samples (i.e., changepoints) do in fact correspond well with imitation errors in BC-OH models. The same figure also plots the error for our approach. At all the changepoints, such as turning right and slowing down in front of a red light, the validation errors of our method are significantly lower than BC-OH. On other frames, it sometimes produces higher errors than BC-OH.

We investigate this phenomenon more quantitatively, report-



Figure 5. An example test scenario from CARLA navigated by a BC-OH policy (top) and ours (bottom). Frames displayed in sequence from left to right. Actions (“Throttle”/“Brake”) are overlaid on the frames.

ing the unweighted imitation MSE (corresponding to Eq (1)) for BC-SO, BC-OH, and ours, on all frames, and then separately for changepoints and other frames. All MSEs are computed on held-out data. See Figure 6. On CARLA, BC-OH performs worse than BC-SO at the APE-based changepoints and better at other frames, once again validating our changepoint-focused approach. In comparison, our approach performs significantly better on changepoints and marginally worse on other samples. Since there are many fewer changepoints, this corresponds to marginally higher overall validation MSE for our approach (despite the higher reward). This is not directly a concern however since, as we have reported above, our method comprehensively outperforms BC-OH in terms of environment reward. Instead, this finding lines up with our intuition that, for optimal reward, it is more important to act correctly at some “keyframes” than at others. Supp has full quantitative results.

Finally, on CARLA-w/o-speed, BC-SO suffers from removing agent velocity information, producing the highest errors on changepoints as well as other samples. Our method yields the lowest errors in both cases (and therefore lowest overall). Note that while we report validation losses here, Supp shows very similar trends for training losses. We find this trend surprising: BC-OH is trained on the unweighted loss over all samples and yet produces a policy that has higher value of this loss on the training set than our approach which optimizes a weighted objective that emphasizes changepoints. We believe this may be a case of data rebalancing avoiding *optimization*-related shortcuts. Similar phenomena have been observed before in ML systems that amplify biases in the training data (Geirhos et al., 2020).

Question 3. *To what extent does our method reduce distributional shift issues in the imitation policies?*

For each policy, we now compute the BC MSE of Eq (1) on the *test data* generated by executing the policy in the environment. The resulting “rollout imitation error” directly measures distributional shift between the expert data and the policy. Note that this measurement is possible in CARLA because the CARLA expert is rule-based rather than learned

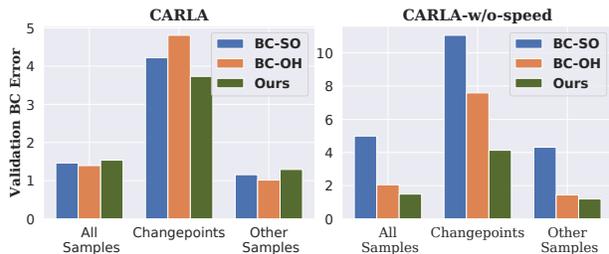


Figure 6. Imitation MSE losses for different sets of validation frames: changepoints, others, and all combined.

from data — therefore, it does not itself suffer from distributional shift, and allows evaluating shift issues with respect to the imitator alone. On both CARLA and CARLA-w/o-speed, our policies (0.07, 0.16) have lower rollout imitation error than BC-OH (0.11, 0.40). This suggests that our approach does in fact suffer less distributional shift.

Question 4. *Do our policies behave less like copycats?*

We have thus far measured APE by training copycat policies on expert action sequences and measuring their errors. We now define a similar notion called $\text{APE}(\pi)$. To measure $\text{APE}(\pi)$ for some policy π , we generate data \mathcal{D}_π by executing π , then train a new optimal copycat policy ψ_π^* on \mathcal{D}_π , and measure its average error on held-out data (generated from π again). This “avgAPE(π)” measures how temporally correlated actions from π tend to be — lower error corresponds to less interesting policies that generate smooth, predictable action sequences. Wen et al. (2020) used a similar metric and showed that approaches that suffer from the copycat problem commonly have lower avgAPE(π) than the expert policy. This is related to our comment above about bias amplification and shortcuts: if the expert policy has low avgAPE(π), the imitator trained to mimic it ends up with even lower avgAPE(π).

Our results, shown in Table 3, are consistent with this. BC-OH has much lower avgAPE(π) than the expert in all environments. Our method consistently improves upon BC-OH,

Table 3. avgAPE for various approaches. All values are ($\times 10^{-2}$)

| METHOD | CARLA | CARLANS | HOPPER | HALFCHEETAH | WALKER2D |
|--------|--------------|--------------|-------------|-------------|-------------|
| EXPERT | 1.602 | 1.602 | 0.86 | 9.81 | 2.47 |
| BC-OH | 0.966 | 0.741 | 0.61 | 5.86 | 0.74 |
| OURS | 1.187 | 1.305 | 0.75 | 9.00 | 0.85 |

but continues to produce lower avgAPE(π) than the expert. While this is not a direct metric, it indicates that our method makes progress towards resolving copycat policy learning, and that there may still be further room for improvement.

Question 5. *When does keyframe-focused imitation perform systematically worse than simple behavior cloning baselines?*

Our approach specifies that weights for frames in training data should be set as monotonic functions f of the action prediction error of a copycat policy, as specified in Eq 6. In practice, the choice of the weighting function f is important. While experimenting with various options for f , we observed that overly flat functions f would not sufficiently alter the behavior of BC-OH, but overly steep functions would sometimes assign inordinately large weights to changepoint keyframes, causing the model to underfit to ordinary frames which constitute the majority of the data. For example, when f is set to the step function, with a high value W assigned to high-APE frames, that trained policy fails even to follow its lane sometimes. In our experimental setups, we found that the sweet spot of functions f that produced our desired behavior was easy to find through a search over the parameters of simple function families (step and softmax). Further, the same parameters worked well across many setups. We report hyperparameter sensitivity results in Supp.

Another potential failure case is in imitation datasets where some samples have noisy action labels. Upweighting changepoints using our approach might assign high weights to such samples, since the copycat policy would fit the noise poorly. Eventually, this might produce bad policies. While we haven’t encountered this in our experiments, it might warrant systematic study in future work.

7. Conclusion

We have proposed a sample weighting strategy to learn effective imitation policies that can integrate information over time without succumbing to learning copycat shortcut policies, while also being very easy to implement and tune. Stepping back to take a broader view, our results show that minimizing the standard empirical risk as in Eq (1) is not optimal in offline imitation learning because of distributional shift issues. Instead, minimizing a carefully reweighted empirical risk produces better-performing policies.

Across four image-based environments spanning simulated locomoting robots and photorealistic urban driving, our approach trivially scales well and yields better results than all prior approaches tackling similar issues. On the long-standing problem of behavioral cloning for driving, we demonstrate that the widely used current standard benchmark CARLA100 might not be challenging enough to effectively benefit from information integration across time, and show large gains in a modified variant that does require such information integration. A future benchmark with more unpredictable vehicles, pedestrians, congested roads, and obstructions would offer a more realistic evaluation of current approaches.

8. Acknowledgement

This work is supported by an Amazon Research Award and gift funding from NEC Laboratories America to DJ, and funding from the Ministry of Science and Technology of the People’s Republic of China, the 2030 Innovation Megaprojects ”Program on New Generation Artificial Intelligence” (Grant No. 2021AAA0150000) to YG.

References

- Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.
- Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.
- Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst. *Robotics: Science & Systems (RSS)*, art. arXiv:1812.03079, 2019.
- Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473, 2018.
- Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoona Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016a.
- Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoona Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016b.

- Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *CoRR*, abs/1106.1813, 2011.
- Kiante Brantley, Wen Sun, and Mikael Henaff. Disagreement-Regularized Imitation Learning. *International Conference in Learning Representations*, pages 1–19, 2020.
- Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.
- Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.
- Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9329–9338, 2019.
- Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019.
- Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems*, pages 11693–11704, 2019.
- Qi Dong, Shaogang Gong, and Xiatian Zhu. Class rectification hard mining for imbalanced deep learning. *CoRR*, abs/1712.03162, 2017.
- Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.
- Paul Fearnhead. Exact and efficient bayesian inference for multiple changepoint problems. *Statistics and computing*, 16(2):203–213, 2006.
- Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
- Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.
- Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.
- Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2015.
- Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4565–4573. Curran Associates, Inc., 2016.
- Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *Eighth International Conference on Learning Representations (ICLR)*, 2020.
- Michael Laskey, Anca Dragan, Jonathan Lee, Ken Goldberg, and Roy Fox. Dart: Optimizing noise injection in imitation learning. In *Conference on Robot Learning (CoRL)*, volume 2, page 12, 2017a.
- Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on Robot Learning*, pages 143–156, 2017b.
- Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L Cun. Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746. Citeseer, 2006.
- Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.
- Kevin P Murphy. A survey of pomdp solution techniques. *environment*, 2:X3, 2000.
- T Osa, J Pajarinen, G Neumann, JA Bagnell, P Abbeel, and J Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, 2018.

- Dean A Pomerleau. Alvin: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.
- Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research*, 15:627–635, 2011. ISSN 15324435.
- Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.
- John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Lu Sheng, Dan Xu, Wanli Ouyang, and Xiaogang Wang. Unsupervised collaborative learning of keyframe detection and visual odometry towards monocular deep slam. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4302–4311, 2019.
- Wen Sun, Arun Venkatraman, Geoffrey J. Gordon, Byron Boots, and J. Andrew Bagnell. Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction. *34th International Conference on Machine Learning, ICML 2017*, 7:5090–5108, 2017.
- Wen Sun, J. Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Deep combination of reinforcement and imitation. In *International Conference on Learning Representations*, 2018.
- Gerrit J. J. van den Burg and Christopher K. I. Williams. An evaluation of change point detection algorithms, 2020.
- Dequan Wang, Coline Devin, Qi-Zhi Cai, Philipp Krähenbühl, and Trevor Darrell. Monocular plan view networks for autonomous driving. In *IROS*, 2019.
- Qing Wang, Jiechao Xiong, Lei Han, Peng Sun, Han Liu, and Tong Zhang. Exponentially weighted imitation learning for batched historical data. In *NeurIPS*, pages 6291–6300, 2018.
- Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. In *Advances in Neural Information Processing Systems*, volume 33, 2020.
- Bernard Widrow and Fred W Smith. Pattern-recognizing control systems, 1964.
- Xiang Xuan and Kevin Murphy. Modeling changing dependency structure in multivariate time series. In *Proceedings of the 24th international conference on Machine learning*, pages 1055–1062, 2007.
- Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9719–9728, 2020.

9. Appendix

9.1. Additional Details on CARLA Experiments

In Sec 5, we specified the experimental setup for our experiments in the CARLA driving environment, which closely follows standard protocol from Codevilla et al. (2018; 2019); Chen et al. (2020). We provide additional details here.

Data Collection. The CARLA100 dataset (Codevilla et al., 2019) in our experiments is collected with noise injection (Laskey et al., 2017a) to perturb 10% of expert actions. It also uses three cameras: a forward-facing one and two lateral cameras facing 30 degrees away towards left or right (Bojarski et al., 2016b), for data augmentation to guard against distributional shift.

Architectures and Training Details. All our baseline models use ImageNet-pretrained Resnet34 as our perception model (Codevilla et al., 2019). We use the state-of-the-art conditional imitation learning model *CILRS* (Codevilla et al., 2019) as our backbone, with a weighted control loss (Codevilla et al., 2018) that assigns weights of 0.5 to steer, 0.45 to throttle and 0.05 to brake. We train all models for 10^5 training iterations with minibatch size 120. We use Adam optimizer, set the initial learning rate to 1×10^{-4} and decay the learning rate by 0.1 whenever the loss value no longer decreases for 5000 iterations. We use L_1 loss as our loss function.

Defining the Metrics. We now more thoroughly define all the metrics we used to measure the performance of the CARLA experiments, following standard protocol (Codevilla et al., 2018; 2019; Chen et al., 2020):

- The %success is the number of episodes fully completed by the ego vehicle among all the 100 predefined test episodes. Higher is better.
- #collision reports the total times the ego car crashes into the pedestrians, vehicles and other obstructions (over 100 test episodes). Lower is better.
- For %progress, we calculate the euclidean distance from the start point to the target point at the beginning of each episode (initial distance) and then compute the distance from the start point to the final point after the episode is over (final distance); and the %progress is 1 minus the ratio between final distance and initial distance. Higher is better.
- The avg. speed is calculated by dividing the path length that the ego car traveled by the travel time. Higher is better in general, but approaches that drive the vehicle straight into pedestrians at high speed will still get avg. speed, so

this metric is not very informative without the context of the other metrics.

Reporting Results In Terms of All Four Metrics. In Sec 5 in the paper, we only had space to report the quantitative results in terms of %success, but we remarked on broad trends in terms of the other metrics above too. We now report those complete results in Table 4 (for unmodified CARLA, including the agent velocity inputs, corresponding to Table 1 in the paper) and Table 5 (for the CARLA-w/o-speed setting with increased partial observability).

Expanded Fig 6. Figure 6 in the paper compares the imitation errors of various models on changepoint and non-changepoint samples. We now report those same results in tabular form for increased precision, in Table 6.

9.2. Additional Details on MuJoCo Experiments

Architectures and Training Details. We use ResNet18 as our backbone and a two-layer MLP with 300 hidden units on top of it to get the predicted action. We use Adam optimizer with initial learning rate 2×10^{-4} and reduce the learning rate by a factor of 10 every 100k iterations during training. We use a batchsize of 128. We train the imitation agent for 300k iterations until convergence. We use L_2 as our loss function.

9.3. Hyperparameter Choices

Since there are only few hyperparameters in our method, we can perform grid search to find the best set of hyperparameters based on evaluation reward. For softmax weighting function, we select its temperature τ from $\{0.1, 0.2, 0.5, 1, 5, 10\}$ and for step weighting function, we select its threshold THR from $\{10\%, 20\%\}$ and its weight W from $\{3, 5, 10\}$.

Our method We use step function for CARLA environment. In both of *CARLA* and *CARLA-w/o-speed*, we set THR to 10% and W to 5, i.e., upweight the top 10% of samples by a factor of 5.

In *MuJoCo-Image*, we experiment with both step and softmax function. For the softmax function, we use a temperature of 0.2 for Hopper and Walker2D and 0.1 for HalfCheetah. For the step function, we set the THR to 10% and W to 5.

Baselines In HistoryDropout, we set the dropout rate to 0.5 for all environments. In FCA, we set different adversarial loss weight for different environment to balance the adversarial loss and the imitation loss. More specifically, the adversarial loss weights for the different environments are *CARLA* and *CARLA-w/o-speed*: 0.2, Hopper: 0.1, HalfCheetah: 0.1.

Table 4. CARLA test results in Nocrash-Dense benchmark. BC-SO is significantly better than BC-OH, verifying the causal confusion phenomenon. Our method outperforms both BC-SO and BC-OH with higher success rate. Moreover, our method significantly reduces the #collision, which is very important for urban driving tasks. Comparing with the baselines, we do better in all the four metrics than the two offline algorithms and even beats the Dagger which require online query.

| METHOD | %SUCCESS (/100) (\uparrow) | #COLLISION (\downarrow) | %PROGRESS (\uparrow) | AVG. SPEED (\uparrow) |
|--------------------------------------|--------------------------------------|--------------------------------------|-------------------------------------|--------------------------------------|
| BC-SO (CODEVILLA ET AL., 2019) | 42.667 \pm 8.668 | 48.444 \pm 9.044 | 0.580 \pm 0.055 | 15.559 \pm 3.035 |
| BC-OH | 33.000 \pm 4.190 | 52.111 \pm 5.878 | 0.497 \pm 0.042 | 11.775 \pm 3.225 |
| OURS (STEP) | 43.444 \pm 0.786 | 42.615 \pm 2.228 | 0.580 \pm 0.040 | 14.938 \pm 2.759 |
| FCA (WEN ET AL., 2020) | 35.667 \pm 3.559 | 50.333 \pm 4.643 | 0.551 \pm 0.030 | 13.927 \pm 2.905 |
| HISTORYDROPOUT (BANSAL ET AL., 2019) | 34.000 \pm 2.625 | 60.222 \pm 3.119 | 0.506 \pm 0.029 | 17.847 \pm 2.120 |
| DAGGER 120K (ROSS ET AL., 2011) | 35.222 \pm 3.067 | 60.000 \pm 2.625 | 0.512 \pm 0.026 | 18.515 \pm 1.586 |
| BCPD | 28.667 \pm 2.494 | 36.667 \pm 8.260 | 0.440 \pm 0.057 | 10.071 \pm 6.668 |
| ACTFREQ | 20.333 \pm 5.825 | 26.000 \pm 5.354 | 0.410 \pm 0.089 | 9.229 \pm 3.353 |
| BOOSTING | 3.000 \pm 1.414 | 19.333 \pm 4.110 | 0.101 \pm 0.026 | 1.933 \pm 2.174 |

Table 5. CARLA-w/o-speed results in Nocrash-Dense benchmark. In this case, BC-SO’s performance is much worse than BC-OH for lack of historical information and it is pretty difficult to infer speed from a single frame. Our method still performs well in such an information-constrained situation.

| METHOD | %SUCCESS (/100) (\uparrow) | #COLLISION (\downarrow) | %PROGRESS (\uparrow) | AVG. SPEED (\uparrow) |
|--------------------------------------|--------------------------------------|--------------------------------------|-------------------------------------|--------------------------------------|
| BC-SO (CODEVILLA ET AL., 2019) | 9.222 \pm 2.380 | 84.667 \pm 4.769 | 0.204 \pm 0.033 | 35.182 \pm 1.594 |
| BC-OH | 25.667 \pm 0.981 | 60.889 \pm 1.911 | 0.467 \pm 0.049 | 15.543 \pm 3.714 |
| OURS (STEP) | 36.778 \pm 5.808 | 45.333 \pm 4.690 | 0.540 \pm 0.050 | 14.416 \pm 3.145 |
| FCA (WEN ET AL., 2020) | 27.444 \pm 4.113 | 59.222 \pm 5.996 | 0.453 \pm 0.052 | 13.856 \pm 2.729 |
| HISTORYDROPOUT (BANSAL ET AL., 2019) | 25.333 \pm 5.375 | 66.778 \pm 5.865 | 0.449 \pm 0.047 | 16.086 \pm 3.889 |
| DAGGER 120K (ROSS ET AL., 2011) | 28.333 \pm 3.496 | 62.667 \pm 3.771 | 0.457 \pm 0.014 | 16.988 \pm 2.757 |
| BCPD | 20.000 \pm 1.414 | 44.000 \pm 9.899 | 0.400 \pm 0.0626 | 5.379 \pm 2.259 |
| ACTFREQ | 14.667 \pm 1.764 | 24.667 \pm 3.972 | 0.313 \pm 0.031 | 3.649 \pm 3.073 |
| BOOSTING | 10.000 \pm 2.160 | 49.667 \pm 4.028 | 0.223 \pm 0.026 | 12.502 \pm 3.242 |

tah and Walker2D: 0.5.

9.4. Sensitivity Study

To study the sensitivity of our method to the hyperparameter choices, we ran some additional experiments by perturbing the hyperparameters. For Walker 2D, we set the softmax temperature to 0.1, 0.2, and 0.5, and the test rewards are 765 ± 103 , 769 ± 97 , and 724 ± 61 . And for CARLA, the weight W of step function is set to 4, 5 and 6, producing success rate 41.89, 43.44, and 41.33%. These results suggest that our method is pretty robust to the hyperparameter choices.

9.5. Additional Sandbox Environment: ToyCar

In addition to the standard environments in the paper, we constructed a simple sandbox environment for fast experimentation, that we will refer to as ToyCar. The environment consists of a car on a straight road with a traffic light that stays on red or green for random lengths of time as shown

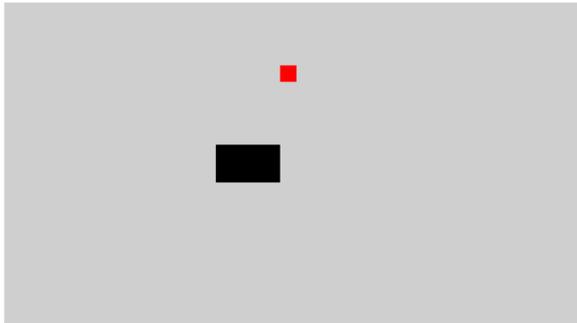


Figure 7. Snapshot of the ToyCar environment.

in Fig 7. The car must successfully drive through the road from left to right without breaking red lights. The car follows point mass double integrator dynamics, and has two actions, throttle, and brake, that apply fixed positive and negative accelerations. There is a fixed upper limit on the velocity. The expert is a rule-based agent and has access to the full observation, i.e. the car position, the velocity,

Table 6. The loss of different types of samples. Our method significantly reduces the loss of hard samples, thus reducing the empirical risks among the whole dataset. The standard deviation values are smaller than 1×10^{-3} so we don't put them here.

| EXPERIMENT | METHOD | UNWEIGHTED LOSS $\times 10^{-2}$ | | CHANGEPOINT LOSS $\times 10^{-2}$ | | COPYCAT SAMPLE LOSS $\times 10^{-2}$ | |
|------------|-------------|----------------------------------|-------|-----------------------------------|--------|--------------------------------------|-------|
| | | TRAIN | VAL | TRAIN | VAL | TRAIN | VAL |
| W/-SPEED | BC-SO | 1.769 | 1.464 | 4.995 | 4.222 | 1.410 | 1.157 |
| | BC-OH | 1.549 | 1.395 | 5.721 | 4.808 | 1.086 | 1.016 |
| | OURS (STEP) | 1.742 | 1.537 | 4.545 | 3.731 | 1.431 | 1.294 |
| W/O-SPEED | BC-SO | 5.350 | 4.989 | 11.213 | 11.059 | 4.710 | 4.316 |
| | BC-OH | 2.172 | 2.055 | 8.194 | 7.588 | 1.504 | 1.441 |
| | OURS (STEP) | 1.671 | 1.491 | 4.985 | 4.132 | 1.303 | 1.198 |

the traffic light position, the traffic light status and the time remaining for the current status. All the imitator policies act on $3 \times 128 \times 128$ images.

Table 7. ToyCar results.

| | BC-SO | BC-OH | OURS(STEP) |
|---------------|----------------|----------------|----------------|
| %SUCCESS RATE | 97.2 ± 0.7 | 95.1 ± 4.5 | 97.8 ± 0.5 |

Results We train the BC-SO, BC-OH and our method with 1K expert samples, and test them in the environment. The test success rates are shown in the Table 7. All results are reported over 5 trials. Quantitative results mirror those reported in the paper, but gains are relatively small due to the fact that the environment is very simple. Qualitatively, we observed instances of similar problems with BC-OH to the inertia problem reported before in Codevilla et al. (2019), namely, the car sometimes stops at a seemingly random location and refuses to start again. Our method successfully removes those failure cases.

9.6. Low-Dimensional Environments: MuJoCo-State

Finally, while in the paper, we showed results for image-based MuJoCo settings, we now report results in low-dimensional partially observed MuJoCo settings. Following Wen et al. (2020), we remove the velocity and external force information from the full state to make the environment partially observed.

The environmental rewards are shown in Tab 8. As in the image-based settings, our method successfully outperforms BC-OH easily in all these settings. Further, our results in these environments are comparable to FCA (Wen et al., 2020). We clearly outperform FCA on Walker2D, perform on par with FCA on HalfCheetah, and perform worse on Hopper. However, the FCA method, reliant on adversarial training, scales poorly to higher-dimensional image-based environments as mentioned in Wen et al. (2020), and also verified throughout our other results.

Table 8. MuJoCo-State results.

| | HOPPER | HALFCHEETAH | WALKER2D |
|-------|----------------------------------|---------------------------------|----------------------------------|
| BC-SO | 275 ± 40 | -38 ± 36 | 363 ± 86 |
| BC-OH | 293 ± 83 | 820 ± 60 | 592 ± 124 |
| FCA | 1086 ± 262 | 1250 ± 42 | 1296 ± 288 |
| OURS | 641 ± 7 | 1023 ± 75 | 1460 ± 169 |

9.7. Additional Baseline: Category Frequency Weighting with Discovered Categories

As we mentioned in Sec 2 in the paper, prior approaches (Cui et al., 2019; Cao et al., 2019) have proposed rebalancing data in supervised learning settings based on the frequencies of categories, so that low-frequency categories are upsampled to correct data imbalance issues. We now report the results of a baseline that discovers action categories in an unsupervised manner and apply a similar strategy. This evaluates a different reweighting strategy to the APE-based strategy proposed in the paper.

We assume that there are 6 typical scenarios in urban autonomous driving tasks, i.e. going straight, turning right, turning left, accelerating, slowing down and parking, so we use the k-means algorithm to cluster the actions in the training dataset into $k = 6$ clusters as categories. And we use category frequency based weighting function $w_i = \frac{\sum_{j=1}^6 n_j}{n_i}$ to reweight each sample in the cluster i , where n_j is the number of samples in cluster j .

This naive reweighting strategy-based baseline does not perform very well. The results are shown in the last rows in Table 4 and Table 5.