Expecting the Unexpected:
Training Detectors for Unusual Pedestrians with Adversarial Imposters

Shiyu Huang
Tsinghua University
Beijing, China
huangsy13@mails.tsinghua.edu.cn

Deva Ramanan
Carnegie Mellon University
Pittsburgh, USA
deva@cs.cmu.edu

Abstract

As autonomous vehicles become an every-day reality, high-accuracy pedestrian detection is of paramount practical importance. Pedestrian detection is a highly researched topic with mature methods, but most datasets focus on common scenes of people engaged in typical walking poses on sidewalks. But performance is most crucial for dangerous scenarios, such as children playing in the street or people using bicycles/skateboards in unexpected ways. Such “in-the-tail” data is notoriously hard to observe, making both training and testing difficult. To analyze this problem, we have collected a novel annotated dataset of dangerous scenarios called the Precarious Pedestrian dataset.

Even given a dedicated collection effort, it is relatively small by contemporary standards (≈ 1000 images). To allow for large-scale data-driven learning, we explore the use of synthetic data generated by a game engine. A significant challenge is selected the right “priors” or parameters for synthesis: we would like realistic data with poses and object configurations that mimic true Precarious Pedestrians. Inspired by Generative Adversarial Networks (GANs), we generate a massive amount of synthetic data and train a discriminative classifier to select a realistic subset, which we deem the Adversarial Imposters. We demonstrate that this simple pipeline allows one to synthesize realistic training data by making use of rendering/animation engines within a GAN framework. Interestingly, we also demonstrate that such data can be used to rank algorithms, suggesting that Adversarial Imposters can also be used for “in-the-tail” validation at test-time, a notoriously difficult challenge for real-world deployment.

1. Introduction

There’s no software designer in the world that’s ever going to be smart enough to anticipate all the potential circumstances an autonomous car is going to encounter. The dog that runs out into the street, the person who runs up the street, the bicyclist, the policeman or the construction worker.

C. Hart, Chairman of National Transport. Safety Board

As autonomous vehicles become an every-day reality, high-accuracy pedestrian detection is of paramount practical importance. Pedestrian detection is a highly researched...
topic with mature methods, but most datasets focus on “everyday” scenes of people engaged in typical walking poses on sidewalks [9, 6, 13, 12, 48]. However, perhaps the most important operating point for a deployable system is its behaviour in dangerous, unexpected scenarios, such as children playing in the street or people using bicycles/skateboards in unexpected ways.

Precarious Pedestrian Dataset: Such “in-the-tail” data is notoriously hard to observe, making both training and evaluation of existing systems difficult. To analyze this problem, we have collected a novel annotated dataset of dangerous scenarios called the Precarious Pedestrian Dataset. Even given a dedicated collection effort, it is relatively small by contemporary standards (≈ 1000 images). To explore large-scale data-driven learning, we explore the use of synthetic data generated by a game engine. Synthetic training data is an actively explored topic because it provides a potentially infinite well of annotated data for training data-hungry architectures [24, 30, 14, 19, 40, 42]. Particularly attractive are approaches that combine a large amount of synthetic training data with a small amount of real data (that may have been difficult to acquire and/or label).

Challenges in Synthesis: We see two primary difficulties with the use of synthetic training data. The first is that not all data is created “equal”: when combining synthetic data with real data, synthesizing common scenes may not be particularly useful since they will likely already appear in the training set. Hence we argue that the real power of synthetic data is generating examples “in-the-tail”, which would otherwise have been hard to collect. The second difficulty arises in building good generative models of images, a notoriously difficult problem. Rather than building generative pixel-level models, we make use of state-of-the-art rendering/animation engines that contain an immense amount of knowledge (about physics, light transfer, etc.). The challenge of generative synthesis then lies in constructing the right “priors”, or scene-parameters, to render/animate. In our case, these correspond to body poses and spatial configurations of people and other objects in the scene.

Adversarial Imposters: We address both concerns with a novel variant of Generative Adversarial Networks (GANs) [17], a method for synthesizing data from latent noise vectors. Traditional GANs learn generative feedforward models that process latent noise vectors, typically from a fixed known prior distribution. Instead, we fix the feedforward model to be a rendering engine, but use an adversarial framework to learn the latent priors. To do so, we define a rendering pipeline that takes as input a vector of scene parameters capturing object attributes and spatial layout. We use rejection sampling to construct a set of scene parameters (and their associated rendered images) that maximally confuse the discriminator. We call such examples Adversarial Imposters, and use them within a simple pipeline for adapting detectors from synthetic data to the world of real images.

RPN+: We use our dataset of real and imposter images to train a suite of contemporary detectors. We find surprisingly good results with a (to our knowledge) novel variant of region proposal network (RPN) [49] tuned for particular objects (precarious people) rather than a general class of objectness detections. Instead of classifying a sparse set of proposed windows (as nearly all contemporary object detection systems based on RCNN do [38]), this network returns a dense heatmap of pedestrian detections, along with regressed bounding box location for each pixel location in the heatmap. We call this detector RPN+. Our experiments show that our RPN+, trained on real+imposter data, outperforms other detectors trained only on real data.

Validation: Interestingly, we also demonstrate that our Adversarial Imposter Dataset can be used to rank algorithms, suggesting that our pipeline can also be used for “in-the-tail” validation at test-time, a notoriously difficult challenge for real-world deployment.

Contributions: The contribution of our work is as follows: (1) a novel dataset of pedestrians in dangerous situations (Precarious Pedestrians) (2) a general architecture for creating realistic synthetic data “in-the-tail”, for which limited real data can be collected and (3) demonstration of our overall pipeline for the task of pedestrian detection using a novel detector. Our datasets and code can be found here: https://github.com/huangshiyu13/RPNplus.

2. Related work

Synthetic Data: Synthetic datasets have been used to train and evaluate the performance of computer vision algorithms. Some forms of ground truth are hard to obtain from hand-labelling, such as optical flow, but easy to synthesize via simulation [14]. Adam et al. [24] used a 3D game engine to generate synthetic data and learned an intuitive physics model to predict falling towers of blocks. Mayer et al. [30] released a benchmark suite of various tasks using synthetic data, including disparity and optical flow. Richter
et al. [40] used synthetic data to improve image segmentation performance, but notably do not control the scene as to explore targeted arrangements of objects. German Ros et al. [42] used Unity Development Platform to generate a synthetic urban scene dataset.

**3D Models for Detection:** A notable application of 3D computer graphics model in vision has been the modeling of the human body shapes [18, 4, 1, 29, 36, 4, 21]. Moreover, 3D simulation can also be used for car detection [34, 32, 20] and scene understanding [45, 23]. Marin et al. [29] used a game engine to generate synthetic training data. Pishchulin et al. [35] used 8 HD cameras to scan human body and built real 3D human models. Then they used synthetic data and some labelled real data to train pedestrian detectors. Hattori et al. [19] used 3D modelling software to build a special scene and randomly put 3D models on a special background for pedestrian detection. Most of these works use synthetic data “as-is”, while we analyze statistical differences between synthetic and real data, describing a pipeline for reconciling such differences through adversarial domain adaption.

**Domain Adaptation:** Domain Adaptation is a standard strategy to deal with data across different domains, such as synthetic versus real. Large synthetic datasets can be used to bootstrap detectors and then adapted to real data by moving to the target domain distribution. Sun and Saenko [46] used 3D models to train detectors for real objects. Such work typically used shallow detectors defined on fixed feature sets, while we focus on gradient-based adaption of “deep” detection networks (such as RCNN). From this perspective, our work is inspired by approaches for deep domain adaptation [15, 16, 26, 27]. Such work typically assumes that one has access to large amounts of unlabeled data from the target domain. In our case, assembling a large target dataset of unlabelled examples (of real Precarious Pedestrians) is itself challenging, necessitating the need for alternative approaches that make stronger use of the source dataset.

Generative Adversarial Nets: GANs [17] are deep networks that can generate synthetic images from latent noise vectors. They do so by adversarially-training a neural network to discriminate between real versus synthetic images. Recent works have shown impressive performance in generation of synthetic images [31, 7, 37, 44, 5]. However, it appears challenging to synthesize high-resolution images with semantically-valid content. We circumvent these limitations with a rendering-based adversarial approach to image synthesis.

**3. Datasets**

**3.1. Precarious Pedestrian Dataset**

We begin by describing our Precarious Pedestrian Dataset. We perform a dedicated search for targeted keywords (such as “pedestrian fall”, “traffic violation” and “dangerous bike rider”) on Google Images, Baidu Images, and some selected images from MPII Dataset [2], producing a total of 951 reasonable images. We then label bounding boxes for each image manually. Precarious Pedestrians contains various kinds scenes, such as children running on the road, people tripping, motorcyclists performing dangerous movements, people interacting with objects (such as bicycles or umbrellas). One important dangerous but increasingly common scenario consists of people watching their phones or texting while crossing the street, which is potentially dangerous as the person may not be aware of their surroundings (Figure 1). To quantify the (dis)similarity of Precarious Pedestrians to standard pedestrian benchmarks such as Caltech [10], we tabulate the percentage of images with more than one people, as well as the number of irregular “pedestrians” such as bicyclists or motorcyclists. Compared to Caltech, Precarious Pedestrians contains images with many more overall people as well as many more cyclists and motorbikes (Figure 3). We split the Precarious dataset equally for training and testing.

**3.2. Synthetic Dataset**

To help both train and evaluate algorithms for detecting precarious pedestrians, we make use of a synthetic data. In this section, we describe our rendering pipeline for generating synthetic data. We use the Unity 3D game engine as our basic platform for simulation and rendering, due to the large availability of both commercial and user-generated assets, in the form of 3D models and character animations.
Within the field of vision (z), also synthesize labels L. Scene parameters: urban city [29].

Our approach can generate a diverse set of background images from INRIA dataset [6] and the 1726 background images from INRIA dataset [6] and a custom set of outdoor scenes downloaded from Internet. We assume a fixed camera viewpoint. Note that the root position and orientation (rotation in the ground plane). the number of 3D models, and for each model, an index the set of background images, the set of 3D models, we first define a set of parameters and parameter ranges. We decide the gestures of 3D models. 

Figure 2 shows the commercial 3D human models that we use for data generation, consisting of 20 models spanning different women, men, cyclists and skateboarder avatars. Because these are designed for game engine play, each 3D model is associated with characteristic animations such as jumping, talking, running, cheering and applauding. We animate these models in a 3D scene with animations such as jumping, talking, running, cheering and applauding. We animate these models in a 3D scene with a 2D billboard to capture the scene background [11], as shown in Figure 2. Billboards are randomly sampled from the 1726 background images from INRIA dataset [6] and a custom set of outdoor scenes downloaded from Internet. Our approach can generate a diverse set of background scenes, unlike approaches that are limited to a single virtual urban city [29].

Scene parameters: To build a large library of synthetic images that will potentially be used for training and evaluation, we first define a set of parameters and parameter ranges. We index the set of background images, the set of 3D models, and the animation frame number for each model. In brief, the scene parameters include directional light intensity and direction (capturing sunlight), the background image index, the number of 3D models, and for each model, an index specifying the avatar ID and animation frame, as well as a root position and orientation (rotation in the ground plane). We assume a fixed camera viewpoint. Note that the root position affects both the location and scale of the 3D model in the rendered image. All these parameters can be summarized as a variable-length vector z ∈ Z, where each vector corresponds to a particular scene instantiation.

Synthesis: Our generator G(z), or rendering engine, synthesizes an image corresponding to z. Importantly, we can also synthesize labels L(z) for each rendered image, specifying object type, 3D location, pixel segmentation masks, etc. In practice, we make use of only 2D object bounding boxes. Table 1 shows the viable ranges of each parameter. In addition, we found the following heuristic to simulate reasonable object layouts: we enforce the maximum overlap between any two 3D models to be 20% (to avoid congestion) and the projected location of the 3D models should lie within the camera’s field-of-view. These conditions are straightforward to verify for a given vector z without rendering any pixels, and so can be efficiently enforced though rejection sampling (i.e., generate a random vector and only render those that pass these conditions). Unlike Hironori et al. [19], who generate training data by manually tuning z to match specific scenes, our approach is not scene specific and does not require any manual intervention.

Table 1: Constraints of parameters for synthesizing images. The index and time(normalized) of animations will jointly decide the gestures of 3D models.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of 3D models</td>
<td>4, 8</td>
</tr>
<tr>
<td>Index of background images</td>
<td>[0, 1726]</td>
</tr>
<tr>
<td>Index of 3D models</td>
<td>[0, 20]</td>
</tr>
<tr>
<td>Position of 3D models</td>
<td>Within the field of vision</td>
</tr>
<tr>
<td>Index of Animations</td>
<td>(0, maxnumber)</td>
</tr>
<tr>
<td>Time of animation</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>Model’s angle on the x axis</td>
<td>[−90°, 90°]</td>
</tr>
<tr>
<td>Model’s angle on the y axis</td>
<td>[-180°, 180°]</td>
</tr>
<tr>
<td>Model’s angle on the z axis</td>
<td>[−90°, 90°]</td>
</tr>
<tr>
<td>Light intensity</td>
<td>[0.5, 2]</td>
</tr>
<tr>
<td>Light’s angle on the x axis</td>
<td>[−45°, 45°]</td>
</tr>
<tr>
<td>Light’s angle on the y axis</td>
<td>[-45°, 45°]</td>
</tr>
</tbody>
</table>

4. Proposed Method

Domain adaptation: In this section, we introduce a novel framework for adversarially adapting detectors from synthetic training data to real training data. We use x ∈ X' to denote an image and y ∈ Y' to denote its label vector (a set of bounding box labels). Let p_s(x, y) to refer to the distribution of image-label pairs from the source domain (of synthetic images), and p_t(x, y) to refer to the target domain (of real Precarious Pedestrians). In our problem, we expect large amounts of source samples, but a limited amount of target ones. We factorize the joint into a marginal over
image appearance and conditional on label given the appearance - e.g., $p_s(x)p_r(y|x)$. Importantly, we discriminatively train a feedforward function $f_s(x) = p_s(y|x)$ to match the conditional distribution. Our central question is how to transfer feedforward predictors trained from source samples $f_s(x)$ to the target domain $f_t(x)$.

**Fine-tuning:** The most natural approach to domain adaptation may simply be to fine-tune a predictor $f_s(x)$, originally trained on the source, with samples from the target $p_t(x, y)$. Indeed, virtually all contemporary methods for visual recognition make use of fine-tuned models that were pre-trained on Imagenet [43]. We compare to such a strategy in our experiments, but find that fine-tuning works best when source and target distributions are similar. As we argue, while rendering engines can produce photorealistic scenes, it is difficult to specify a prior over scene parameters that mimic real (Precarious) scenes. We describe a solution that adversarially learns a prior.

**Generators:** As introduced in Sec. 3.2 let $z \in \mathcal{Z}$ be a vector of scene parameters, $G(z) \in \mathcal{X}$ be a feedforward generator function that renders a synthetic image given the scene parameters, and $L(z) \in \mathcal{Y}$ be a function that generates labels from the scene parameters. We can then reparameterize the distribution over synthetic images as a distribution over scene parameters $p_z(z)$. We now describe a procedure for learning a prior $p_z(z)$ that allows for easier transfer. Specifically, we learn a prior that fools an adversary that is trying to distinguish samples from the source and target.

**Adversarial generators:** To describe our approach, we first recall a traditional generative adversarial network (GAN):

$$\min_{D} \max_{G} V(D, G) = \quad \text{[Gen. Adversarial Net]} \quad (1)$$

$$\mathbb{E}_{x \sim p_l(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

where the minimax optimization jointly tries to estimate a discriminator $D$ that can distinguish real versus synthesized data examples, and the generator $G$ tries to synthesize realistic examples that fool the discriminator. Typically, the discriminator $D(x)$ is trained to output the probability that $x$ is real (e.g., a real Precarious Pedestrian), while $p_z(z)$ is fixed to be a zero-mean, unit-variance Gaussian. This optimization can be performed with stochastic gradient updates, that converge (in the limit) to a fixed point of the minimax problem. We refer the reader to the excellent introduction in [17]. Importantly, the generator must encode complex constraints about the manifold of natural images, that capture amongst other knowledge the physical properties of light transport and material appearance.

**Adversarial priors:** We note that rendering engines can be viewed as generators that already contain much of this knowledge, and so we fix $G$ to be a production-quality rendering platform (Unity 3D). Instead, we learn the prior over parameter vectors in a adversarial manner:

$$\min_{D} \max_{I} V(D, I) = \quad \text{[Adversarial Priors]} \quad (2)$$

$$\mathbb{E}_{x \sim p_l(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

If the generator $G$ is differentiable with respect to $z$, it is possible to use backprop to compute gradient updates for simple prior distributions $p_z(z)$, such as Gaussians [22, 39]. This implies that the above formulation of adversarial priors is amenable to gradient-based learning.

**Imposter search:** We see two difficulties with directly applying (2) to our problem: (1) It seems unlikely that the optimal prior for precarious scene parameters will be a simple unimodal distribution with a single mean parameter vector (and associated covariance matrix). (2) Rendering, while readily expressed as a feed-forward function, is not naturally differentiable at object boundaries (where small changes in parameters can generate large changes in the rendered image). While approximate differentiable renderers do exist [28], we wish to make use of highly-optimized commercial packages for animation and image synthesis (such as Unity 3D). As such, we adopt a simple sampling-based approach that addresses both limitations:

$$\min_{D} \max_{I} V(D, I) = \quad \text{[Imposter Selection]} \quad (3)$$

$$\mathbb{E}_{x \sim p_l(x)}[\log D(x)] + \mathbb{E}_{z \sim \text{Unif}(z)}[\log(1 - D(G(z)))]$$

where $Z_I \subseteq Z$. That is, we search for a subset of parameter vectors (the “imposters”) that fool the discriminator. One could employ various sequential sampling strategies for optimizing the above: start with a random sample of parameter vectors, update the discriminator (with gradient based updates using a batch of real and synthesized data), generate additional samples close to those imposters that fool the discriminator, and repeat. We found a single iteration to work quite well. Our algorithm for synthesizing a realistic set of precarious scenes is given in Alg. 1 and the overall approach for adversarial domain adaption is given in Alg. 2.

---

**Algorithm 1 Imposter Selection**

**Input:** Set of examples from source domain $S$ and target domain $T$.

**Output:** Subset of imposters $I \subseteq S$.

1. **Train** a binary discriminator network $D(x)$ that distinguishes examples $x \in S$ from $x \in T$.
2. **Return** the subset of $k$ samples from $S$ that best fool the discriminator.

Here, the set $S$ consists of synthetic image-label pairs rendered from an exhaustive set of scene parameters $\{z\}$ and the set $T$ consists of real (Precarious) image-label pairs. Without Step 2, Alg. 2 reduces to standard fine-tuning from
Algorithm 2 Domain Adaption with Imposters

Input: Set of examples from source domain $S$ and target domain $T$.
Output: Predictor $f(x)$ for target set $T$.
1. Pre-Train a predictor $f(x)$ on source set $S$.
2. Adapt the predictor on $T \cup I$, where $I$ is the set of imposters found with Alg. 1.
3. Fine-tune the predictor on only target set $T$.

![Figure 5: Architecture of RPN+.

We concatenate several layers on different stages in order to improve the ability of locating people in different resolutions. We use 9 anchors (reference boxes with 3 scales and aspect ratios) at each sliding position. During training, a candidate bounding box will be treated as a positive if its intersection-over-union overlap with a ground-truth box exceeds 50%, and will be a negative for overlaps less than 20%. To accelerate training time, we initialize with a pre-trained VGG-16 model where the first two convolutional layers are frozen.

5. Experiments

5.1. Evaluation

We follow the evaluation protocol of the Caltech pedestrian dataset [10], which use ROC curves for 2D bounding box detection at 50% and 70% overlap thresholds.

Testsets: We use three different datasets for evaluation: our novel Precarious Pedestrian testset of real images, our novel Adversarial Imposter Testset, and for diagnostics, a standard pedestrian benchmark dataset (Caltech).

Baselines: We compare our approach with the following baselines:

ACF: An aggregate channel features detector [8].
LDCF: A LDCF detector [33].
HOG+Cascade: A cascade of boosted classifiers working with HOG features [50].
HARR+Cascade: A cascade of boosted classifiers working with haar-like features [47, 25].
RPN/BF: A RPN detection model trained with boosted forest [49], which appears to be the state-of-the-art pedestrian detection system at the time of publication.

Precarious Pedestrians: Results on Precarious Pedestrians are presented in Figure 6. Our detector significantly outperforms alternative approaches, including the state-of-the-art RPN/BF model. At $10^{-1}$ false positive per image, our miss rate of 42.47% significantly outperforms all baselines, including the state-of-the-art RPN/BF model (with a miss rate of 54.5%). Note that all baseline detectors are trained on Caltech. Comparing to baselines is complicated by the fact that both the detection system and training dataset have changed. However, in some sense, our fundamental contribution is method for generating more accurate training datasets through adversarial imposters. To isolate the impact of our underlying detection network RPN+, we also train a variant solely on the Caltech training set (denoted as RPN+Caltech), making it directly comparable to all baselines because they use the same training set. RPN+Caltech performs slightly worse than RPN/BF (with miss-rate of 58.82%), though it outperforms RPN/BF at higher false positive rates. This suggests that our underlying network is close to state-of-the-art, and moreover validates the signif-
Figure 6: (a) and (b) are ROC curves for different detectors under different overlap ratio criteria on the Precarious Pedestrian testset. In the legend, we denote the miss rate at 10^{-1} false positives per image. **RPN+Caltech** refers to our RPN+ network architecture trained only on Caltech, while **Ours** refers to our detector (RPN+) trained on synthetic, imposter, and real images (Alg 2). Note all detectors besides **Ours** are trained on the Caltech Dataset.

Table 2: Miss rate of different fine-tuning strategies at a false positive rate of 10^{-1}, where \( S, T, \) and \( I \) refer to source datasets (of synthetic images), target dataset (of precarious real images), and Imposter dataset.

<table>
<thead>
<tr>
<th>Fine-tuning method</th>
<th>50% overlap</th>
<th>70% overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S )</td>
<td>83.49%</td>
<td>95.18%</td>
</tr>
<tr>
<td>( T )</td>
<td>72.39%</td>
<td>93.70%</td>
</tr>
<tr>
<td>( S \Rightarrow T )</td>
<td>48.45%</td>
<td>77.14%</td>
</tr>
<tr>
<td>( S \Rightarrow (T \cup I) )</td>
<td>45.97%</td>
<td>74.94%</td>
</tr>
<tr>
<td>( S \Rightarrow (T \cup I) \Rightarrow T )</td>
<td>42.47%</td>
<td>73.70%</td>
</tr>
</tbody>
</table>

**6. Conclusion**

We have explored methods for analyzing “in-the-tail” urban scenes, which represent important modes of operations for autonomous vehicles. Motivated by the fact that rare but dangerous scenes are exactly the scenarios on which visual recognition should excel, we first analyze existing datasets and illustrate that they do not contain sufficient rare scenarios (because they naturally focus on common or typical urban scenes). To address this gap, we have collected our own dataset of Precarious Pedestrians, which we will release to spur further research on this important (but under explored) problem. Precarious scenes are challenging because little data is available for both evaluation and training. To address this challenge, we propose the use of synthetic data generated with a game engine. However, it is challenging to ensure that the synthesized data matches the statistics of real precarious scenes. Inspired by generative adversarial networks, we introduce the use of a discriminative classifier (trained to discriminate real vs synthetic data) to implicitly specify this distribution. We then use the synthesized data that fooled the discriminator (the “synthetic imposters”) to both train and evaluate state-of-the-art, robust pedestrian detection systems.

**Acknowledgements.** This work was supported by NSF.
Figure 7: **Results on Precarious Dataset.** The left shows results of RPN+ trained on Caltech, while the right shows RPN+ trained with adversarial imposters.

Figure 8: **Algorithm rankings on real datasets and synthetic dataset.** ROC curves for detectors on Precarious Dataset and Synthetic Dataset. **Ours (RPN+) is trained with selection model.** The results suggest that performances of algorithms on real dataset and synthetic dataset have the same rankings.

Figure 9: (a) and (b) are ROC curves for different detectors under different overlap ratio criteria on Caltech Dataset under the default “reasonable” test protocol.

Figure 10: (a) **Results of different amounts of Imposters.** Optimal performance is obtained with an imposter set that is roughly equal in size to the set of 500 target samples of Precarious training images. (b) **Results of different selectors.** We choose three selectors from different training periods and use them to select imposters separately. As the discriminator gets better, so does the final fine-tuned detector.

References


