Structured Prediction with Adversarial Constraint Learning

Hongyu Ren
Peking University
rhy@pku.edu.cn

Russell Stewart
Stanford University
stewartr@cs.stanford.edu

Jiaming Song
Stanford University
tsong@cs.stanford.edu

Volodymyr Kuleshov
Stanford University
kuleshov@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

Constraint learning is a recently proposed form of weak supervision which attempts to reduce the labeling burden by having users specify general properties that hold over the output space (e.g. physical laws). However, specifying constraints can be difficult and may require extensive domain expertise. In this paper, we introduce an adversarial constraint learning framework in which invariants are automatically extracted from data. In this framework, users only need to provide a black-box simulator that generates valid system outputs; at training time, we constrain the model to produce outputs that cannot be distinguished from simulated samples by a learned discriminator. Further providing our framework with a small number of labeled examples gives rise to a new semi-supervised structured prediction method; we evaluate this method on multiple tasks — object tracking and pose estimation, and we find that our framework achieves high accuracy with only a small amount of labels, and no labels at all in some cases.

1 Introduction

Large labeled datasets are key component in many machine learning applications [1–3], but collecting them can be expensive. Constraint learning is a recently proposed form of weak supervision which aims to reduce cost of collecting labels by supervising algorithms through general properties that hold over the output space [4, 5]. Examples of such properties include logical rules [6–8], physical laws [5], or anatomical properties of the human body. Unlike labels, which only apply to their corresponding inputs, properties used in a constraint learning approach are specified once for the entire dataset, providing an opportunity for more cost-effective supervision [9, 5].

However, describing the high level invariants of a dataset may also require a non-trivial amount of effort. First, designing explicit constraints requires strong domain expertise. Second, in the case of high dimensional outputs, encoding the constraints using simple formulas is hard. For example, it is difficult to constrain a pedestrian detector with formulas that describe the shape of walking person. Third, constraints may change across tasks; designing new constraints for new tasks may not scale in many practical applications.

In this paper, we propose an *implicit* approach to constraint learning, in which invariants are automatically learned from a small set of representative output samples (see Figure 1). These samples do *not* need to be tied to corresponding inputs (as in traditional supervised and semi-supervised learning) and may come from a black-box simulator that abstracts away physics-based formulas, examples of outputs collected by humans or from standard datasets used in supervised learning.

$$\begin{tikzcd}\mathbf{x} \arrow[r, "p_\theta"] & \mathbf{y} \arrow[r, "h"] & \mathbb{R}\end{tikzcd} \qquad \begin{tikzcd}\mathbf{x} \arrow[r, "p_\theta"] & \mathbf{y} \arrow[dr] & \\ & & \mathbb{R} \\ & \hat{\mathbf{y}} \arrow[ur, "D"] & \end{tikzcd}$$

Figure 1: Constraint learning allows us to learn a probabilistic model $p_\theta(\mathbf{y}|\mathbf{x})$ without direct labels by specifying properties h that holds over the output space. In prior work (left), h is defined as a formula describing known invariants. In this paper (right), we propose to instead learn h through a discriminator network D that discriminates $\mathbf{y}$ (provided by $p_\theta(\mathbf{y}|\mathbf{x})$) from $\hat{\mathbf{y}}$ (provided by an additional source unrelated to $\mathbf{x}$, such as a simulator).

Inspired by recent advances in generative models, we capture the distribution of outputs using an approach based on adversarial learning [10]. Specifically, we train two distinct learners: a primary model for the task at hand and an auxiliary classification algorithm called a discriminator. During training, we constrain the main model such that its outputs cannot be distinguished by the discriminator from true output samples, thus forcing it to capture the structure of the output space. This approach forms a novel adversarial framework for performing weak supervision with learned constraints. Our framework turns into semi-supervised learning, when given some labeled data. Experimental results demonstrate that this method performs well on a variety of structured prediction problems, outperforming natural baselines with very few labeled inputs.

2 Background

2.1 Structured Prediction

In this paper, we focus on structured prediction problems, in which the outputs $\mathbf{y} \in \mathcal{Y}$ can be a complex object such as a vector, a tree, or a graph [11]. We capture the distribution of $\mathbf{y}$ using a conditional probabilistic model $p_\theta(\mathbf{y}|\mathbf{x})$ parameterized by $\theta \in \Theta$. A model $p_\theta(\mathbf{y}|\mathbf{x})$ maps each input $\mathbf{x} \in \mathcal{X}$ to the corresponding output distribution $p(\mathbf{y}) \in \mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$ denotes all the probability distributions over $\mathcal{Y}$. For example, we may take $p_\theta(\mathbf{y}|\mathbf{x})$ to be a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}), \boldsymbol{\Sigma}_\theta(\mathbf{x}))$ with mean $\boldsymbol{\mu}_\theta(\mathbf{x})$ and variance $\boldsymbol{\Sigma}_\theta(\mathbf{x})$.

A standard approach to learning $p_\theta(\mathbf{y}|\mathbf{x})$ (or p_θ) is to solve an optimization problem of the form

$$\theta^* = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} l(p_\theta(\mathbf{y}|\mathbf{x}_i), \mathbf{y}_i) + R(p_\theta) \tag{1}$$

over a labeled dataset $\mathcal{D}_L = \{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_n, \mathbf{y}_n)\}$. A typical supervised learning objective is comprised of a loss function $l : \mathcal{P}(\mathcal{Y}) \times \mathcal{Y} \to \mathbb{R}$ and a regularization term $R : \mathcal{P}(\mathcal{Y}) \to \mathbb{R}$ that encourages non-degenerate solutions or solutions that incorporate prior knowledge [5].

2.2 Constraint-Based Learning

Let $\mathcal{D}_U = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ be an unlabeled dataset of inputs, without their corresponding label. Formally, constraints can be specified via a function $h : \mathcal{P}(\mathcal{Y}) \to \mathbb{R}$, which penalizes conditional probabilistic models $p_\theta(\mathbf{y}|\mathbf{x})$ that are not consistent with known high-level structure of the output space. Learning from constraints proceeds by optimizing the following objective.

$$\hat{\theta}^* = \arg\min_{\theta \in \Theta} \sum_{i=1}^{m} h(p_\theta(\mathbf{y}|\mathbf{x}_i)) + R(p_\theta) \tag{2}$$

By solving this optimization problem, we look for a probabilistic model parameterized by $\hat{\theta}^*$ that is likely a priori (through the $R(p_\theta)$ term), and satisfies known constraints when applied to the unlabeled dataset $\mathcal{D}_U$, such as physical laws. Note that the constraint h is data-dependent, although it does not require explicit labels. For example, in free fall object tracking, we could ask that the predictions on $\mathcal{D}_U$ over time form a parabola [5] and h measures how the output distribution from p_θ deviates from the equations. The regularization term is used to avoid overly complex and/or degenerate solutions, and may include $L1$, $L2$, and entropy regularization.

2.3 Adversarial Training and Implicit Probabilistic Models

Recent learning methods based on adversarial training and implicit probabilistic models play a key role in our approach [12]. Implicit probabilistic models are defined as the result of a stochastic sampling procedure, rather than through an explicitly defined likelihood function. Prominent examples are generative adversarial networks (GAN), where samples $G(\mathbf{z})$ are obtained by transforming some Gaussian noise $\mathbf{z} \sim \mathcal{N}(0, I)$ through a neural network G, called the generator.

In this work, we are interested in placing constraints on a probability distribution over the *output space* Y. We define this distribution implicitly by the following sampling procedure,

$$\mathbf{x} \sim p_d(\mathbf{x}) \quad , \quad \mathbf{y} \sim p_\theta(\mathbf{y}|\mathbf{x}) \tag{3}$$

where $p_d(\mathbf{x})$ is the data distribution over the input space $\mathcal{X}$, and $p_\theta(\mathbf{y}|\mathbf{x})$ is a conditional distribution of outputs given inputs. The above procedure corresponds to sampling from the marginal distribution over $\mathcal{Y}$, $p_\theta(\mathbf{y}) = \int p_\theta(\mathbf{y}|\mathbf{x}) p_d(\mathbf{x}) d\mathbf{x}$. However, evaluating the marginal likelihood $p_\theta(\mathbf{y})$ exactly is typically intractable due to the integration over $p_d(\mathbf{x})$.

Given label samples $\mathcal{D}_S = \{\mathbf{y}_1, \cdots, \mathbf{y}_k\}$, implicit generative models can be trained using likelihood-free methods that define a distance metric between distributions alternative to KL divergence. GANs [10] are trained using an approximation of the Jensen-Shannon divergence (JSD) with the following minimax objective, which can be optimized by stochastic gradient descent.

$$\min_G \max_D \mathbb{E}_{\mathbf{y} \sim p_s(\mathbf{y})}[\log D(\mathbf{y})] + \mathbb{E}_{\mathbf{y} \sim p_\theta(\mathbf{y}|\mathbf{x}), \mathbf{x} \sim p_d(\mathbf{x})}[\log(1 - D(\mathbf{y}))]$$

Other metrics can also be used as an objective for training implicit probabilistic models, such as maximum mean discrepancy (MMD) [13] or Earth Mover's distance (EMD) [14, 15].

3 Adversarial Constraint Learning

The process of describing high level constraints can be time-consuming and may require significant domain expertise. In the sciences, discovering general invariants is often a data-driven approach, for example, physical laws are often discovered by validating hypotheses with experimental results. Motivated by this idea, we propose to learn constraints from a set of representative output samples.

Learning Constraints from Data. Suppose that we are given a small number of labels $\mathcal{D}_S$ (not necessarily associated with input data), or a black-box mechanism/simulator for generating such labels, i.e., to sample from the empirical label distribution $p_s(\mathbf{y})$. We formulate the task of learning a constraint loss h from these label samples using the framework of generative adversarial learning [10].

Our ultimate goal is to learn a conditional probability distribution $p_\theta(\mathbf{y}|\mathbf{x})$ that assigns high probability to correct outputs $\mathbf{y}$. To enforce this goal, we define an auxiliary classifier D_ϕ (parametrized by ϕ) called a discriminator which scores outputs in the label space, and tries to assign higher scores to a small set of given representative output labels $\mathcal{D}_S$, while assigning lower scores to samples of $p_\theta(\mathbf{y}|\mathbf{x})$. Thus, the discriminator learns to effectively extract latent constraints that hold over the output space and are implicitly encoded in the output samples $\mathcal{D}_S$. The goal of the $p_\theta(\mathbf{y}|\mathbf{x})$ is to produce outputs that score higher under the discriminator, meeting the constraints.

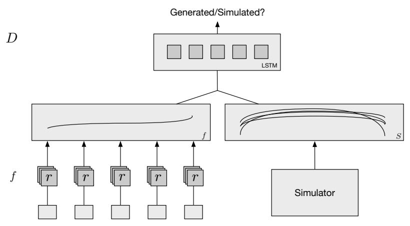
We train D_ϕ and $p_\theta(\mathbf{y}|\mathbf{x})$ jointly through adversarial training, optimizing the Wasserstein GAN objective [14, 15]

$$\min_\theta \max_\phi \mathcal{L}^{\mathcal{A}} = \mathbb{E}_{\mathbf{y} \sim p_s(\mathbf{y})}[D_\phi(\mathbf{y})] - \mathbb{E}_{\mathbf{y} \sim p_\theta(\mathbf{y}|\mathbf{x}), \mathbf{x} \sim p_d(\mathbf{x})}[D_\phi(\mathbf{y})] \tag{4}$$

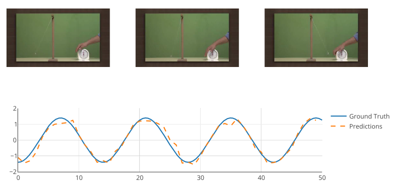
where $p_s(\mathbf{y})$ denotes the distribution of samples in $\mathcal{D}_S$ and $p_d(\mathbf{x})$ denotes the distribution of the input data. At the optimal solution to the objectives in Eq. 4, the discriminator cannot distinguish between the given set of labels and those predicted by the model, suggesting that the latter satisfy the set of constraints identified by the discriminator. Figure 2(a) shows an overview of the adversarial constraint learning framework in the context of an object tracking task.

When given a set of labeled examples, we may formulate our objective as the sum of a constraint learning term (over both labeled and unlabeled data) and a standard regression loss term (over the labeled data), resulting in a semi-supervised framework,

$$\mathcal{L}^{SS} = \mathcal{L}^{\mathcal{A}} + \alpha\, \mathbb{E}_{\mathbf{x}_i, \mathbf{y}_i \sim p_l}[l(p_\theta(\mathbf{y}|\mathbf{x}_i), \mathbf{y}_i)] \tag{5}$$



(a) Our architecture trains f_θ (or in this example, r_θ) by asking it to generate trajectories T_f that cannot be discriminated from sample trajectories T_S. Training D eliminates the needs to hand-engineer constraints.



(b) Top: frames from video used in the pendulum experiment. Bottom: the network is trained to detect angles that cannot be distinguished from the simulated dynamics, encouraging it to track the metal ball over time.

Figure 2: Architecture and results of the pendulum detection experiment.

where $\mathcal{L}^{\mathcal{A}}$ is the adversarial constraint learning objective defined in Eq. 4, p_l is the distribution for the labeled dataset, and α is a hyperparameter that balances between fitting to the general (implicit) output distribution (first term) and fitting to the explicit labeled dataset (second term).

Constraint Learning by Matching Distributions. Our approach can also be interpreted as matching the marginal distribution over predicted labels to the label samples.

Assuming we can obtain samples from $p(\mathbf{y})$, another way of formulating the constraint h is to let

$$\mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[h(f_\theta(\mathbf{x}))] = \rho(\mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[p_\theta(\mathbf{y}|\mathbf{x})], p(\mathbf{y})) \tag{6}$$

where ρ is some divergence/metric that can be approximately computed through samples from $p(\mathbf{y})$ (such as KL divergence, MMD, JSD, or EMD). Minimizing ρ ensures that $p_\theta(\mathbf{y}|\mathbf{x})$ provides a reasonable $\mathbf{y}$ that lies in the true manifold of labels.

4 Experimental Results

We evaluate our framework on two structured prediction problems. First, we train a network to track the angle of a pendulum in a video using supervision provided by a physics-based simulator. Next, we extend the output space to higher dimensions and perform human pose estimation.

In both experiments, we consider $p_\theta(\mathbf{y}|\mathbf{x})$ to be a Dirac-delta distribution $\delta(\mathbf{y} - f_\theta(\mathbf{x}))$, thus we refer to the conditional probabilistic model as the mapping $f_\theta(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$, implemented as a neural network parametrized by θ. Please refer to the appendix for detailed network architectures and training details.

4.1 Pendulum Tracking

For this task, we aim to extract the angle of the pendulum on images from a YouTube video [16], i.e., learn a regression mapping $r_\theta : \mathbb{R}^{h\times w\times 3} \to \mathbb{R}$. Since the outputs of r_θ over continuous frames form sine waves and are thus constrained, we concatenate continuous outputs of r_θ and form a high dimensional trajectory $f_\theta([\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]) = [r_\theta(\mathbf{x}_1), r_\theta(\mathbf{x}_2), \cdots, r_\theta(\mathbf{x}_n)]$. We also design a simulator that produces sine waves with frequency based on observations.

We manually label the horizontal position of the ball of the pendulum in each frame in the test set, and measure the correlation of the predicted position with the ground truth label in pixels. As shown in Figure 2, we achieve a correlation of 96.3% while using explicit formulas to supervise training yields 96%. Our model is generally robust against the accuracy of the simulator.

Overall, the real world pendulum experiment shows that our adversarial framework makes it possible to extract object information from real images using only a simulator of physics that the object obeys.

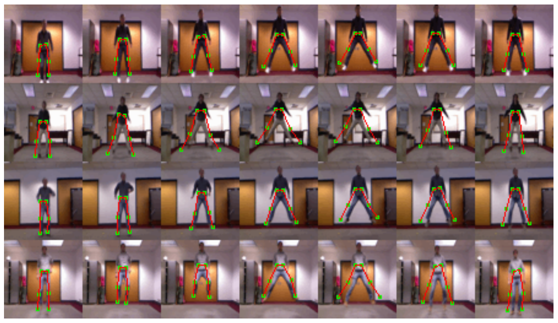


Figure 3: Pose estimation results when 25% of the training data are labeled. The regression network takes in single image and outputs the location of 6 joints (in green).

4.2 Pose Estimation

In this experiment, we benchmark the proposed model on pose estimation, which has a larger output space. We use multi-modal action database (MAD) [17], and aim to detect left/right hip/knee/foot on images. We attempts to learn a regression network $r_\theta : \mathbb{R}^{h\times w\times 3} \to \mathbb{R}^k$, where k denotes the number of joints we detect. As before, r_θ is applied to several frames to produce a trajectory. We train the network based on a sequence of images, and output a sequence of joint locations, which should be indistinguishable from the sample data. Similar to the pendulum experiment, r_θ is applied to each frame independently, *no knowledge of the neighbor frames* is used in this process.

In this experiment, the output samples are labels $\{\mathbf{y}_1, \cdots, \mathbf{y}_n\}$, but we assume we have no knowledge of the corresponding input vectors. The results and notations are shown in Table 1 listed in appendix. When only trained adversarially ("0%+adv"; i.e., optimizing just $\mathcal{L}^{\mathcal{A}}$), the network is able to find the correct "shape" of the joints for each frame, but the predictions are biased with a minor shift $(\Delta x, \Delta y)$. It shows that mere adversarial training is insufficient for this task. To mitigate the problem, we provide a small amount of labeled training data and consider the semi-supervised objective $\mathcal{L}^{\mathcal{SS}}$. The label loss helps adjust f_θ to output the precise locations. Given just 25% of the available labeled data, f_θ converges to detecting the joints with high accuracy, as shown in Figure 3. "50%+adv" achieves same performance as "100%" (fully supervised on all training data) on the detection of feet.

We also evaluate the following baselines. First, we test the result of pure guessing, using a randomly picked label from the simulator for each test data as its prediction. Furthermore, we run ablative experiments "t%", where only t% of the labeled data is used for supervised learning. In this case, some data points are neither trained nor tested. "t%+adv" generally shows much better results compared to "t%". Lastly, we test "t%+rand", where we randomly assign labels from the simulator to unlabeled data points, and then use supervised learning. Although this random label assignment could be incorrect, it could still provide some signal. However, the results demonstrate that if the remaining data are used in this random manner, the detection accuracy rarely increases. This emphasizes the importance of our adversarial training loss.

This experiment shows that our model is robust when the output space is large; hand-crafting formula-based constraints is tedious and error-prone. Our model avoids the problematic complexity by extracting constraints implicitly from the output samples, which are capable of describing the feature of output space. In cases where limited input-output pairs are given, our model can be trained in a semi-supervised way by minimizing the label loss and the adversarial loss simultaneously.

5 Conclusion

We propose a new framework for semi-supervised structured prediction using adversarial constraint learning. Instead of using hand-designed constraints, which are difficult to obtain and application-specific, we instead learn the constraints from data using an implicit generative model with adversarial loss. Experimental results on structured prediction tasks show that our method is robust against high dimensional outputs. As future work, we plan to explore adversarial constraint learning applied to scenarios where the output space is characterized by multiple constraints (the output distribution is factored).

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[3] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649, IEEE, 2013.

[4] I. Shcherbatyi and B. Andres, "Convexification of learning from constraints," in *German Conference on Pattern Recognition*, pp. 79–90, Springer, 2016.

[5] R. Stewart and S. Ermon, "Label-free supervision of neural networks with physics and domain knowledge.," in *AAAI*, pp. 2576–2582, 2017.

[6] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1, pp. 107–136, 2006.

[7] M.-W. Chang, L. Ratinov, and D. Roth, "Guiding semi-supervision with constraint-driven learning," in *ACL*, pp. 280–287, 2007.

[8] A. Choi, G. Van den Broeck, and A. Darwiche, "Tractable learning for structured probability spaces: A case study in learning preference distributions," in *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[11] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[12] S. Mohamed and B. Lakshminarayanan, "Learning in implicit generative models," *arXiv preprint arXiv:1610.03483*, 2016.

[13] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1718–1727, 2015.

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[15] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[16] KClassScienceChannel, "Time period of a pendulum depends on its length." `https://www.youtube.com/watch?v=02w9lSii_Hs`, 2013.

[17] D. Huang, S. Yao, Y. Wang, and F. De La Torre, "Sequential max-margin event detectors," in *European conference on computer vision*, pp. 410–424, Springer, 2014.

[18] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[19] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2878–2890, 2013.

A Network Layout

We use the gradient penalty Wasserstein GAN [14] throughout the experiments since it reduces training time and is more stable in general. In both experiments, the discriminator D_ϕ is a recurrent neural network (RNN) with Long Short-Term Memory (LSTM) units. The input of D_ϕ is first fed through 3 FC/ReLU layers with 64 hidden dimensions and terminating in n LSTM inputs of dimension 128, where n is the group size, and equals 5 in both experiments. After processing the entire sequence, we pass the final LSTM hidden state through a single FC layer to score the input sequence. For the pendulum tracking, we use a convolutional neural network (CNN) for the regression network r_θ, with 2 Conv/ReLU/Pool modules followed by a FC layer with 1 output. In the pose estimation experiment, we adopt a VGG-based network, instead of pooling, we follow the advice of [18] and use strided convolution layers to downsample. After the extraction of 512 feature maps, we use two FC layers to output a 12-dimensional vector.

B Training Details

B.1 Pendulum Tracking

We sample the 17 second video at 10 frames per second, resulting in a total of 170 images. We hold out 34 images for evaluation. We manually observe that the pendulum completes one full oscillation approximately every 12 frames. Based on this observation, we write a simulator of these dynamics with a simple harmonic oscillator having a fixed amplitude and random sample period of 10 to 14 frames. r_θ takes in one image in 56×56 size and outputs a scalar, representing the angle of the pendulum in the image. D_ϕ is trained to distinguish between the output of r_θ across 5 continuous images and a random trajectory sampled from the simulator. We train the network with the Adam optimizer for $5,000$ iterations at a learning rate of 10^{-4} [19].

B.2 Pose Estimation

CMU multi-modal action database (MAD) [17] contains 40 videos of 20 subjects (2 for each subject) performing a sequence of 35 actions in each video. We edit the 40 videos, extract the frames when the subjects perform the "Jump and Side-Kick" action. The processed dataset contains 620 valid frames (40 groups). The 40 groups of motion data are divided into 32 groups and 8 groups for training and testing respectively. Each group contains 14 to 17 images. In our experiment, we train on randomly selected intervals of 5 contiguous frames. We use PCK@0.1 [20] for evaluation. The prediction is considered correct if and only if it lies within $\alpha \max(h, w)$ pixels from the correct location, where h and w denote the height and width of the tightest bounding box that covers the whole body, and we use $\alpha = 0.1$, which is a fairly strict criterion.

Images are resized to 64×64 pixels before r_θ is applied. r_θ takes a single image as input and outputs a 12 dimensional vector, representing the location of 6 joints. We concatenate the outputs of the regression network for each group and pass it to the discriminator. The discriminator is LSTM-based and tries to tell the generated locations and sample joint locations apart. We use the Adam optimizer with a learning rate of 10^{-4}. The network is trained for $20,000$ iterations. In each iteration, D_ϕ is updated 5 times and r_θ once. We split the dataset randomly and repeat training and testing 50 times. The results are averaged over these 50 trials.

C Pose Estimation Results

PCK@0.1(%)	Left Hip	Left Knee	Left Foot	Right Hip	Right Knee	Right Foot
RSS	0.517	0.414	0.300	0.520	0.412	0.299
0%+rand	0.743	0.620	0.493	0.750	0.604	0.442
0%+adv	0.846	0.578	0.414	0.824	0.636	0.514
12.5%	0.820	0.794	0.717	0.813	0.729	0.625
12.5%+rand	0.789	0.623	0.498	0.819	0.598	0.464
12.5%+adv	0.857	0.831	0.783	0.939	0.823	0.668
25%	0.766	0.869	0.768	0.769	0.885	0.737
25%+rand	0.852	0.804	0.560	0.864	0.763	0.510
25%+adv	0.923	0.842	0.829	0.914	0.850	0.802
37.5%	0.912	0.884	0.813	0.913	0.897	0.796
37.5%+rand	0.896	0.714	0.591	0.899	0.743	0.579
37.5%+adv	0.944	0.916	0.858	0.951	0.898	0.867
50%	0.943	0.903	0.809	0.958	0.904	0.773
50%+rand	0.841	0.725	0.606	0.847	0.815	0.733
50%+adv	0.965	0.895	0.861	0.968	0.922	0.872
100%	0.994	0.950	0.876	0.994	0.977	0.858

Table 1: PCK@0.1 results on MAD. “Random simulator sample” (RSS) makes a prediction using a random label from the simulator (baseline). “$t\%$” means that we train on $t\%$ of the labeled data (standard supervised learning). “$t\%$+rand” means that we additionally randomly assign labels from the simulator to the remaining $(1 - t\%)$ of the training data (baseline). “$t\%$+adv” means that we use $t\%$ of the labeled training data (supervised loss), with the additional adversarial loss (our approach). Our approach consistently outperforms the baselines.