Action-Conditioned Contrastive Policy Pretraining

Qihang Zhang\textsuperscript{1} Zhenghao Peng\textsuperscript{1} Bolei Zhou\textsuperscript{2}
\textsuperscript{1}The Chinese University of Hong Kong \textsuperscript{2}University of California, Los Angeles
\{zq021, pengzh\}@ie.cuhk.edu.hk bolei@cs.ucla.edu

Abstract

Deep visuomotor policy learning achieves promising results in control tasks such as robotic manipulation and autonomous driving, where the action is generated from the visual input by the neural policy. However, it requires a huge number of online interactions with the training environment, which limits its real-world application. Compared to the popular unsupervised feature learning for visual recognition, feature pretraining for visuomotor control tasks is much less explored. In this work, we aim to pretrain policy representations for driving tasks using hours-long uncurated YouTube videos. A new contrastive policy pretraining method is developed to learn action-conditioned features from video frames with action pseudo labels. Experiments show that the resulting action-conditioned features bring substantial improvements to the downstream reinforcement learning and imitation learning tasks, outperforming the weights pretrained from previous unsupervised learning methods. Code and models will be made publicly available.

1. Introduction

Deep policy learning makes promising progress to many visuomotor control tasks ranging from robotic manipulation \cite{20, 22, 25, 39} to autonomous driving \cite{4, 47}. By learning to map visual observation directly to control action through a deep neural network, it mitigates the manual design of controller, lowers the system complexity, and improves generalization ability. However, the sample efficiency of the underlying algorithms such as reinforcement learning or imitation learning remains low. It requires a significant amount of online interactions or expert demonstrations in the training environment thus limits its real-world applications.

Many recent works use unsupervised learning and data augmentation to improve the sample efficiency by pretraining the neural representations before policy learning. However, the augmented data in pretraining such as frames with random background videos \cite{16, 17, 43} shifts drastically from the original data distribution, which degrades the overall performance of the model. Also, it remains challenging to generalize the learned weights to the real-world environment as it is hard to design augmentations that reflect the real-world diversity. In this work, we explore pretraining the neural representation on a massive amount of real-world data directly. Figure 1 shows some uncurated YouTube videos, which contain driving scenes all over the world with diverse conditions such as different weathers, urban and rural environments, and various traffic densities. We show that exploiting such real-world data in deep policy learning can substantially improve the generalization ability of the pretrained models and benefit downstream tasks across various domains.

Learning representations from unlabeled data is a popular topic in visual recognition. The learned representations are shown to be generalizable across visual tasks ranging from image classification, semantic segmentation, to object detection \cite{5, 13, 18, 40, 42}. However, these methods are mainly designed for learning features for recognition tasks, which are different from control tasks where an agent takes actions in uncertain environment. Some visual information such as the abstraction of appearance and texture may not be useful in decision making. For example, visual elements like lighting and weather are usually irrelevant to the driving task and might even become confounders in policy learning, lowering the overall performance \cite{46}. On the contrary, it is crucial to learn the features that matter to the output action. For example, at the driving junction, the traffic light occupies only a few pixels in the visual observation but has a significant impact on the driver’s actions.

In this paper, we propose a novel action-conditioned policy pretraining paradigm, which captures important features in the neural representation relevant to the decision-making and benefits downstream tasks. As shown in Figure 1, we first collect a large corpus of driving videos with a wide range of weather conditions, from wet to sunny, from all across the world without labeling. We then generate action pseudo labels for each frame using a pretrained inverse dynamics model. We finally develop a novel policy pretraining method called Action-conditioned
CONtrastive Learning (ACO) that incorporates the action information in the representation learning. Specifically, instead of contrasting images based on different augmented views [18], we define a new contrastive pair conditioned on action similarity. By learning with those action-conditioned contrastive pairs, the representation captures policy-related elements that are highly correlated to the actions.

We evaluate the effectiveness of the action-conditioned pretraining for a variety of tasks, such as policy learning through Imitation Learning (IL) and Reinforcement Learning (RL) in end-to-end autonomous driving, and Lane Detection (LD). The experimental results show that ACO successfully learns generalizable features for the downstream tasks.

Our contributions are summarized below:

1. We propose a new paradigm of action-conditioned policy pretraining on a massive amount of real-world driving videos.
2. We develop a novel action-conditioned contrastive learning approach ACO to learn action-related features.
3. Comparison of various pretraining methods on downstream policy learning tasks shows the feature resulting from the proposed method achieves sufficient performance gain in the driving tasks.

2. Related work

2.1. Image-based Policy Learning

Learning with auxiliary task. Reconstruction tasks [15, 24, 44], self-supervised objectives [23, 33], and future prediction objectives [1, 12, 15, 28, 41] are proposed to mitigate the gap between state-based and image-based Reinforcement Learning (RL). However, unsupervised learning based on visual information may not lead to optimal policy learning performance since the irrelevant visual features might become confounders in the representation and distract policy learning [46].

Augmenting the input distribution. Data augmentation like random shift [17, 43] and inserting random videos at background [16, 17] can improve policy robustness and generalization ability. Visual and physical attributes of the environment and the agent can be randomized through domain randomization [2, 27, 35]. However, it is hard to design augmentation that covers real world’s visual variations. Significant computational resources are also required to train the policy with augmented input [35].

Decoupling visual-level and policy-level learning. Chen et al. [4] first train a policy with full state observation and then uses this cheating policy as a teacher to train an image-based policy. Similar teacher-student architecture is adopted in [6] to learn visuomotor grasping policy. These methods decouple visual knowledge and policy knowledge and learn in two stage fashion. However, they require accessing the underlying proprioceptive state as observation to train the teacher agent.

Pretraining. RRL [32] pretrains ResNet on ImageNet dataset as image encoder and freezes it in following policy learning process. However, the supervised training on visual classification task does not necessarily benefits the policy learning, especially when no incentive is given to encode cues about action into representation. We instead use action-conditioned self-supervised learning to learn task-relevant representation. Several works pretrain policies on offline demonstrations [38, 45]. However, the training data is from the policy learning environment and requires expert policy to collect trajectories, limiting the scalability and efficiency. On the contrary, the proposed ACO conducts pretraining based on YouTube dataset that is more realistic and scalable than the data extracted in simulation environ-
2.2. Contrastive Learning

Contrastive learning [18, 40] is a popular pretext task for self-supervised learning. It creates supervisory labels via considering each image (instance) in the dataset forms a unique category and applies the learning objective of instance discrimination. Previous researches [5, 13, 18, 40] in computer vision use contrastive learning to learn general vision knowledge that transfers well to various downstream tasks. In reinforcement learning, CURL [23] and ATC [33] leverage contrastive learning as an additional signal to learn representation with environmental reward.

Common practice [18] considers two views, i.e. different augmentations, of a single image as a positive pair and views of different images as negative pairs. We call this type of positive/negative pair as Instance Contrastive Pair (ICP). However, ICP only preserves knowledge for visual discrimination without task-relevant information. We introduce Action Contrastive Pair (ACP) that contrasts images based on the underlying actions. Learning together with ACP and ICP will force the representation to be sensitive to policy-relevant elements.

3. Method

We aim to learn generalizable visual feature for visuomotor policy learning, by pretraining on large amounts of uncurated real-world driving videos. We propose a novel contrastive learning algorithm called Action-Conditioned COntрастive Learning (ACO).

The essential of ACO is that we define two types of contrastive pair: Instance Contrastive Pair (ICP) and Action Contrastive Pair (ACP). Two views of a single image form a positive ICP, while two views of different images form a negative ICP. Only pretraining with ICP may make the representation include unnecessary information for downstream policy learning tasks. For example, visual cues like weather and lighting conditions are essential for forming ICP but contribute little to decision-making in autonomous driving. To focus the feature on policy-relevant properties, we introduce another type of contrastive pair called Action Contrastive Pair (ACP). Positive ACP is composed of two different images showing scenarios where the drivers’ actions are similar. As an example, the two real-world snapshots in Figure 2 show the first-view images when the drivers are turning left. These two images thus form a positive ACP. ACO learns representation based on both ICP and ACP, where ICP focuses on learning discriminative general visual feature and ACP focuses on policy-relevant feature.

Figure 2 illustrates the training pipeline of ACO. We first augment the given images twice to create query and key views and form ICPs. The ICPs are used to compute the ICP loss for visual discriminative feature learning, as discussed in Section 3.1. We create another learning flow based on ACP. To create ACP, each frame in the dataset is tagged with the pseudo labels of action as introduced in Section 3.2. In Section 3.3, the action labels are used to form ACPs between different images and compute the ACP loss for policy feature learning.

3.1. Visual Feature Learning with ICP

As shown in Figure 2, our contrastive learning is performed in ICP space and ACP space. To project given images into ICP space and form ICP, our pipeline contains the following parts:

Data Augmentation module $Aug(\cdot)$. For each sample image $x$, we generate two random augmentations, $x^q = Aug(x)$ as query view and $x^k = Aug(x)$ as key view.

Encoder module $f(\cdot)$. Each query view $x^q$ is further mapped to a feature vector $v^q = f(x^q) \in R^{d_E}$, where $d_E$ is the feature dimension. For the key view $x^k$, a momentum-updated Encoder module $f_m(\cdot)$ is used to extract feature vector $v^k = f_m(x^k) \in R^{d_E}$. The update rule of the encoder’s parameters is:

$$\theta_{f_m} \leftarrow \alpha \theta_{f_m} + (1 - \alpha) \theta_f, \quad (1)$$

where $\alpha$ is the momentum coefficient and $\theta_f$, $\theta_{f_m}$ are the parameters of $f$ and $f_m$, respectively. The output of encoder module is the representation passed to policy head in downstream task.

Projector module $g_{ins}(\cdot)$. We map key and query feature $v^q$ and $v^k$ to instance-contrastive space:

$$z^q_{ins} = g_{ins}(v^q), \quad z^k_{ins} = g_{ins,m}(v^k), \quad (2)$$

where $z^q_{ins}, z^k_{ins} \in R^{d_P}$ and $d_P$ is the dimension of the projected vector. $z^q_{ins}$ and $z^k_{ins}$ are further normalized to the unit hypersphere in $R^{d_P}$. The projector $g_{ins,m}$ for key representation is also updated in a momentum way:

$$\theta_{g_{ins,m}} \leftarrow \alpha \theta_{g_{ins,m}} + (1 - \alpha) \theta_{g_{ins}}, \quad (3)$$

Key Dictionary module $D$. Following [18], we use a key dictionary to store historical encoded key $z^k_{ins}$ to enable larger contrastive batchsize. Concretely, we generate key features $z^k$ of images following the lower branch in Figure 2. These key features not only form positive pairs with query features at this training epoch, but also be stored into $D$ and used to form negative pairs at future training, if sampled. The samples in the dictionary are progressively replaced if $D$ exceeds its maximum size.

At each iteration of training, a batch of images are sampled from dataset. The images are sequentially processed by Data Augmentation, Encoder and Projector modules to get query and key features. The query-key pair forms positive...
steering = 0.35
steering = 0.40

Figure 2. In the training pipeline of ACO, we form both ICP and ACP for the input images and update the model with contrastive losses in ICP space and ACP space. The upper branch shows the calculation flow of query images and the lower shows the flow to generate the features of key images. History key features are stored in a key dictionary and re-used in future.

ICP since they come from the same image. To create negative ICP, we sample a batch (num=N) of key features from D and form negative pairs with current queries. The negative keys are concatenated with current key features to form the key set K. Current key features are inserted into key dictionary for later use. Then we perform contrastive learning on ICP between query and key features.

The ICP loss for visual feature learning is computed as:

\[
L_{ins} = - \log \sum_{z^+ \in P_{ins}(z^+_q)} \exp\left(\frac{\sum_{z^- \in N_{ins}(z^-)} z^+ \cdot z^-}{\tau}\right)
\]

\[
P_{ins}(z^q) = \{z | \text{id}(z) = \text{id}(z^q), z \in K\},
\]

\[
N_{ins}(z^q) = K \setminus P_{ins}(z^q).
\]

The symbol \(\cdot\) denotes inner (dot) product, \(\text{id}(z) = x\) returns the image from which the projected vector is generated, \(P_{ins}(z^q)\) and \(N_{ins}(z^q)\) are the sets of keys of all positive and negative ICP, respectively. Notice that \(P_{ins}(z^q)\) only contains one sample and \(N_{ins}(z^q)\) contains \(N\) samples. This is because only one element in key set comes from the anchor image \(x\) while other key features are sampled from \(D\).

3.2. Generating Pseudo Labels for Video Frames

We propose to conduct contrastive learning based on Action Contrastive Pair (ACP). However, uncurated videos on the web do not contain any ground-truth action. To generate the action label for each frame, we train an inverse prediction model to predict the pseudo label [21, 36]. The inverse prediction model \(\phi\) predicts the action \(\hat{a}_t\) that has happened between two consecutive frames \(I_t\) and \(I_{t+1}\). Nevertheless, many visual cues, such as weather and lighting, are irrelevant to action prediction. Using two RGB frames as input may introduce unnecessary distractions. Instead, we find that employing optical flow, the concise description of interframe movement, improves action prediction accuracy.

As illustrated in Figure 3A, we first extract the optical flow between frames by using an off-the-shelf algorithm RAFT [34]. We then use the optical flow as the input to the inverse prediction network \(\phi\) to predict the action. We choose NuScenes [3] dataset which contains consecutive first-view driving scenes with labeled steering action to perform supervised learning of \(\phi\). L1 loss between ground truth action \(a_t\) from the dataset and predicted \(\hat{a}_t\) is optimized. As shown in Figure 3B, we then use the extracted optical flow together with the learned \(\phi\) to label action for each frame in YouTube videos. The pseudo labels of action are further used to formulate ACPs as discussed in the next section.

3.3. Policy Feature Learning with ACP

Based on the pseudo labels of action, we conduct contrastive learning on ACP space to discover policy-relevant representation. We share the Data Augmentation and Encoder modules with ICP learning in Section 3.1. During learning with ACP, the projector \(g_{act}(\cdot)\) shares the same architecture as ICP’s, but not the weights. A momentum-updated ACP projector \(g_{act,m}(\cdot)\) is updated in a similar way as ICP projector in Equation 3. Thus the key and query vectors in action-contrastive space are computed as:

\[
z_{act}^q = g_{act}(z^q) = g_{act}(f(x^q)),
\]

\[
z_{act}^k = g_{act,m}(z^k) = g_{act,m}(f(x^k)).
\]

We also sample a batch of key features from Key
Dictionary and concatenate them with current key features to form key set $K$. The ACP loss is computed as:

$$
\mathcal{L}_{act} = -\log \sum_{z^a \in P_{act}(z^a_{act})} \exp(z^a_{act} \cdot z^t / \tau) \sum_{z^q \in N_{act}(z^q_{act})} \exp(z^q_{act} \cdot z^t / \tau). \quad (6)
$$

Different from ICP, the positive and negative keys in ACP are determined based on the action similarity. The projected vectors of frames with similar underlying actions form positive pairs. Therefore, when collecting the key set, we additionally store the action labels $a$ with the key features $z^k$. We use L1 distance to measure the similarity of two actions and the positive and negative key sets are determined as:

$$
P_{act}(z^q) = \{ z_i \mid ||a - a^q|| < \epsilon, (z, a) \in K \},$$

$$N_{act}(z^q) \equiv K \setminus P_{act}(z^q), \quad (7)
$$

where $a^q$ is the action of the query image and $\epsilon$ is an action distance threshold.

### 3.4. Loss

The total contrastive loss of ACO is:

$$
\mathcal{L} = \lambda_{ins} \mathcal{L}_{ins} + \lambda_{act} \mathcal{L}_{act}, \quad (8)
$$

where $\lambda_{ins}$ and $\lambda_{act}$ are weighted factors to balance ICP and ACP.

### 4. Experiments

We evaluate the pretrained feature from our method on two main policy learning tasks: Imitation Learning (IL) and Reinforcement Learning (RL) in end-to-end driving. We also evaluate on one related computer vision task: Lane Detection (LD). Section 4.1 lists baselines that we compare with. The main experimental results compared with different pretraining methods are presented in Section 4.2 for IL, in Section 4.3 for RL, and in Section 4.4 for LD. Section 4.5 gives a qualitative analysis of the learned action-conditioned representation. Ablation study and further discussion are included in Section 4.6.

**YouTube driving dataset.** We crawl first-view driving videos from YouTube. 134 videos with a total length of over 120 hours are collected. As shown in Figure 1, these videos cover different driving scenes with various weather conditions (sunny, rainy, snowy, etc.) and regions (rural and urban areas). We sample one frame every one second, resulting to a dataset of 1.30 million frames. We split the YouTube driving dataset into training set with 70% data and test set with 30% data and conduct the training of ACO on the training set. The video list will be released.

**Inverse predictor.** An inverse prediction model with ResNet-34 architecture is used to predict action labels for frames in YouTube driving dataset. We use RAFT [34] as the optical flow extractor and apply Adam optimizer with learning rate of 0.001 and weight decay of 0.001 to learn the model on the training set of NuScenes [3]. We normalize steering value between 0 and 1. In our preliminary experiments, the L1 error of steering prediction averaged over the test set of NuScenes [3] is 0.10 when using consecutive images as input, and 0.04 when using optical flow. Optical flow brings better inverse prediction performance. Figure 4 shows sampled frames. The steering action is predicted precisely across different visual conditions.

**Pretraining.** In the representation learning, we largely follow the hyper-parameters from the official implementation of MoCo-v2 [7]. We optimize the model with synchronized SGD over 8 GPUs with a weight decay of 0.0001, a momentum of 0.9, and a batch size of 32 on each GPU. We perform 100 epochs optimization with an initial learning rate of 0.03 and a cosine learning rate schedule. Two projectors for ICP and ACP are both instantiated as two-layer MLPs. We maintain a memory queue of 40,960 samples as the Key Dictionary $D$. The momentum coefficient in Equation 1 and Equation 3 is set to 0.999 for updating the key encoder and projector.
Data augmentations.} During pretraining, we follow MoCo-v2 [7] and use image augmentations to generate key and query views of an image. Specifically, we apply random gray scaling, color jittering and Gaussian blurring. Random resized cropping and horizontal flipping are excluded because we find that they will largely change the scene’s semantic, thus degrading the overall performance, as shown in the ablation study.

4.1. Baselines

We choose ResNet-34 as the backbone (encoder) architecture and compare the downstream tasks’ performance when using different pretrained models as the initial parameters of the backbone network. Different pretraining methods are as follows:

- **Random.** The backbone’s parameters are initialized randomly. Specifically, we use Kaiming normalization [19] to initialize convolution layers. Constant initialization is applied to batchnorm layer (1 for weight and 0 for bias).

- **ImageNet.** We use official ImageNet’s pretrain weights to initialize the backbone. This is considered as the most common approach for current image-based policy learning [32].

- **AutoEncoder.** We train an autoencoder on the collected YouTube videos and use the encoder to initialize the backbone, which has been used in previous policy learning works [14].

- **MoCo.** As a straightforward comparison to our method, contrastive learning is conducted following MoCo-v2 [7] on the collected YouTube videos. As a baseline, we only use ICP to conduct contrastive learning and set $\lambda_{act} = 0$ in Equation 8.

4.2. Imitation Learning

**Setup.** We evaluate the performance of different pretrained weights for Imitation Learning in the open-source CARLA [10] simulator, as shown in 5. We use the original CARLA [10] benchmark (also known as CORL2017) and collect 50 trajectories of expert demonstration, which amount to 40K transitions, in Town01 with train weather. Town02 with test weather is used for evaluation. The expert is a rule-based PID controller. We adopt CILRS [9], a conditional behavior cloning algorithm, to train policies based on different pretrained models. To create diverse scenarios, we follow [8] and inject random noise to expert action. All experiments are repeated three times with different random seeds. Information about other hyper-parameters is given in Appendix.
Figure 5. **Example tasks.** We show example trajectories navigating in CARLA simulator [10].

Table 1. **Success Rate of Imitation Learning.** We evaluate the performance of different pretrained models under different data scales.

| Pretrain Method | 10%   | 20%   | 40%   | 100%  |
|-----------------|-------|-------|-------|-------|
| Random          | 0.0±0.0 | 0.0±0.0 | 37.3±6.6 | 81.3±5.2 |
| AutoEncoder     | 0.0±0.0 | 4.0±3.3  | 6.7±2.5  | 46.0±5.9  |
| ImageNet        | 21.3±7.5 | 52.7±13.1 | 72.0±4.3 | 90.7±7.4  |
| MoCo            | 19.3±3.4 | 60.3±8.2 | 76.0±9.1 | 80.7±2.5  |
| ACO             | 30.7±3.4 | **66.0±5.7** | **82.0±5.0** | **96.0±3.3** |

**Results.** To evaluate different pretrained models under a limited number of expert demonstration, we experiment on the expert datasets with 10%, 20%, 40% and 100% of all 40K samples. As shown in Table 1, ACO outperforms all other baselines in datasets of all scales. Notably, AutoEncoder is less competitive even compared to random initialization. This is because pixel reconstruction target is misaligned with policy learning target and makes the encoder pay more attention to areas with larger size like buildings and the sky, which however are useless for imitation learning. MoCo and ImageNet perform well at full dataset but have degraded performance when dataset size is reduced. ACO-pretrained models consistently outperform baselines at all different dataset sizes.

### 4.3. Reinforcement Learning

**Setup.** Similar to Imitation Learning, we also use CARLA [10] as the simulator for evaluating pretrained models in Reinforcement Learning. We use Town01 in NoCrash benchmark [9] as the environment and evaluate the performance of Proximal Policy Optimization (PPO) [31] with different pretrained weights.

**Results.** As shown in Figure 6A, PPO with ACO initialization introduces a clear improvement of 7% route completion compared to other baselines when the backbone is fine-tuned during policy learning. This suggests that incorporating policy-relevant objects into pretraining stage benefits downstream RL’s performance. ImageNet and MoCo, which only learn visual knowledge, are less competitive compared to ACO. For random initialization, we observe a poor performance: 15% route completion in average means that the agent can only drive in straight road but fails to learn turning.

To further understand the importance of considering policy-relevant information in pretraining, we freeze the backbone and only finetune the policy head during RL. As shown in Figure 6B, compared to backbone fine-tuned case, the performance of PPO declines dramatically when using ImageNet pretraining. This indicates that ImageNet pretrained weight lacks the essential knowledge for policy learning thus leads to a poor RL performance. ACO’s and MoCo’s performance are not affected when freezing the backbone.

### 4.4. Lane Detection

**Setup.** To further prove that the representation learned by ACO is generalizable to other tasks, we conduct lane detection experiments. We train three popular lane detection methods on CULane dataset [26]: UFLD [29], SCNN [26], and RESA [48]. All of these methods employ backbone networks with ResNet architecture. We use ACO and other baselines to initialize the backbone’s parameters before training. We adopt the training settings (e.g. learning rate) of three LD methods fine-tuned on ImageNet pretrained
Samples (K) Samples (K)
Route Completion
B. Backbone Frozen
A. Backbone Fine-tuned

Figure 6. Route completion curves for PPO [31] with different pretrain weights. In the left panel, the backbone is actively updated while policy learning. In the right panel, the backbone is frozen and not updated. The proposed ACO outperforms all baselines.

Table 2. F1 metric of lane detection.

| Pretrain Method | LD Method | ImageNet | SCNN [26] | RESA [48] |
|-----------------|-----------|----------|-----------|-----------|
| UFLD [29]       | 70.9      | 74.1     | 76.1      |
| Random          | 68.4      | 70.4     | 73.9      |
| AutoEncoder     | 68.6      | 70.1     | 74.0      |
| MoCo            | 69.5      | 72.0     | 74.1      |
| MoCo-bn         | 69.8      | 71.9     | 74.1      |
| ACO             | 69.5      | 71.6     | 74.5      |
| ACO-bn          | **70.4**  | 71.8     | **75.0**  |

backbone. Better performance is expected after tuning the hyper-parameters for different pretrained methods respectively.

MoCo [18] suggests that a batch normalization layer after backbone can alleviate the gap between contrastive learning and supervised learning caused by different parameters’ distribution. We thus follow this procedure on ACO and MoCo baseline and present the results with “-bn” suffix.

Results. As shown in Table 2, ImageNet pretraining achieves the best performance among all candidates. Although trained in an object-centric dataset, ImageNet still provides high quality feature for lane detection learning. Within the self-supervised baselines, ACO outperforms others in UFLD [29] and RESA [48]. In SCNN [26], ACO is on par with MoCo. Introducing batch normalization improves the performance of ACO.

4.5. Visualization

We perform t-SNE [37] analysis on features extracted by different layers of ACO and compare them with that of MoCo. We randomly select frames from the test set and divide them into three categories: left turn, right turn, and go straight scenarios respectively.

As shown in Figure 7A, features extracted by ACO’s encoder, which will be passed to policy head in downstream task, form separable clusters and have clear semantics related to the possible actions of drivers. On the contrary, the action-conditioned clustering does not emerge in MoCo in Figure 7D. Instead, we find that MoCo feature demonstrates the instance clustering phenomenon, where the features of three temporal neighboring frames lay closely and form a clique in the t-SNE visualization.

Diving into the contrastive learning modules, we find the ICP projector demonstrates instance clustering as in Figure 7B. The phenomenon does not exist in ACP projected features in Figure 7C, where features cluster more tightly according to action information instead of temporal relationship. t-SNE visualization shows that ICP and ACP in ACO both achieve their respective goals: ICP endows features with the capacity to discriminate across instances, whereas ACP creates features that tightly couple with action information.
Table 3. **Ablation study of pretraining dataset.** We use ACO to pretrain models on YouTube dataset and NuScenes dataset and conduct IL training.

| Pretraining Dataset | Frame num (M) | IL Demonstration Size ($\times 40K$) |
|---------------------|---------------|-------------------------------------|
| NuScenes [3]        | 0.19@10Hz     | 13.3±3.4 27.3±10.4 67.3±9.0 82.7±8.4 |
| **YouTube**         | 1.30@1Hz      | **30.7±3.4 66.0±5.7 82.0±5.0 96.0±3.3** |

Table 4. **Ablation study of augmentations.** CF stands for Clipping and Flipping.

| Methods           | IL Demonstration Size ($\times 40K$) |
|-------------------|-------------------------------------|
|                   | 10% 20% 40% 100%                   |
| MoCo w/ CF        | 19.3±3.4 60.3±8.2 76.0±9.1 80.7±2.5 |
| MoCo w/o CF       | 13.3±3.4 47.3±10.7 72.7±9.0 84.0±4.9 |
| ACO w/ CF         | 27.3±0.9 36.0±5.9 74.0±7.5 89.3±5.0 |
| ACO w/o CF        | **30.7±3.4 66.0±5.7 82.0±5.0 96.0±3.3** |

4.6. Ablation Study

We conduct ablation studies that examine the effectiveness of collecting and labeling YouTube driving videos rather than using NuScenes dataset [3], the impact of geometric-aware augmentations (cropping and flipping), and the effects of ICP loss and ACP loss in ACO.

**YouTube Dataset.** To highlight the importance of collecting large dataset from YouTube, we train IL agents based on models pretrained on different datasets and compare the downstream task performance, namely the success rate on IL benchmark. As shown in Table 3, the size of our collected YouTube dataset is six times larger than NuScenes dataset [3] even though we sample frames at 1Hz from the video, compared to 10Hz of NuScenes. The imitation learning performance of ACO trained on YouTube dataset outperforms that on NuScenes dataset [3] at all IL training settings. The result shows that YouTube dataset improves the generalization ability of the pretrained models and leads to better performance in the downstream task.

**Augmentations.** We conduct imitation learning experiments to discover the impact of two augmentations, cropping and clipping frames (CF in the following for lucidity). As shown in Table 4, the performance of MoCo without CF is significantly lower than MoCo with these two augmentations, showing that CF is important for instance discriminative learning. However, the result suggests ACO is incompatible with CF. Excluding CF in ACO will affect the feature’s instance discriminative ability but improve the overall performance in downstream driving tasks with action-conditioned contrastive learning.

**ACP and ICP.** We conduct ablation experiments in imitation learning under three settings: optimizing the pretrained model (1) with solely ACP loss, (2) with solely ICP loss, or (3) with both ACP and ICP losses. As shown in Table 5, optimizing both ACP and ICP losses achieves the best performance. Although we do not use cropping and flipping, limiting the potential benefit of ICP, including ICP together with ACP can still improve the performance.

5. Conclusion and Discussion

In this paper, we propose a novel contrastive policy pretraining method ACO using hours-long uncurated YouTube videos. By learning action-conditioned features from unlabeled video frames with pseudo labels of action, our methods greatly improve the generalizability of the learned representation and brings substantial improvements to the downstream tasks.

**Limitations.** Despite the generalizable representation provided by pretraining with action information, we incorporate a strong assumption: each driving scene corresponds to only one action. This does not hold true as drivers may have various driving intentions at complex scenarios. Future study will concentrate on how to label frames more accurately, e.g. labeling each frame with an action distribution, to produce more precise action contrastive pairs. Also, action-conditioned contrastive learning has conflict with widely used geometric-aware augmentations like cropping and flipping, suppressing feature’s instance discrimination ability. We leave this problem open for future study.

**References**

[1] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki,
Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. The International Journal of Robotics Research, 39(1):3–20, 2020. 2

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 11621–11631, 2020. 4, 5, 9

[4] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In Conference on Robot Learning, pages 66–75. PMLR, 2020. 1, 2

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In International conference on machine learning, pages 1597–1607. PMLR, 2020. 1, 3

[6] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In Conference on Robot Learning, pages 297–307. PMLR, 2022. 2

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297, 2020. 5, 6

[8] 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 4693–4700. IEEE, 2018. 6

[9] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 9329–9338, 2019. 6, 7, 12

[10] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. CARLA: an open urban driving simulator. CoRR, abs/1711.03938, 2017. 6, 7

[11] DI drive Contributors. DI-drive: OpenDIlab decision intelligence platform for autonomous driving simulation. https://github.com/opendilab/DI-drive, 2021. 12

[12] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. arXiv preprint arXiv:1509.06113, 25, 2015. 2

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Ghaslaghi Azar, and et al. Bootstrap your own latent—a new approach to self-supervised learning. Advances in Neural Information Processing Systems, 33:21271–21284, 2020. 1, 3

[14] David Ha and Jürgen Schmidhuber. World models. arXiv preprint arXiv:1803.10122, 2018. 6

[15] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In International conference on machine learning, pages 2555–2565. PMLR, 2019. 2

[16] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. arXiv preprint arXiv:2007.04309, 2020. 1, 2

[17] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 13611–13617. IEEE, 2021. 1, 2

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 9729–9738, 2020. 1, 2, 3, 8, 12

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision, pages 1026–1034, 2015. 6

[20] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In Conference on Robot Learning, pages 651–673. PMLR, 2018. 1

[21] A Kumar, S Gupta, and J Malik. Learning navigation subroutines by watching videos. corr abs/1905.12612 (2019), 1905. 4

[22] Sascha Lange, Martin Riedmiller, and Arne Voigtlander. Autonomous reinforcement learning on raw visual input data in a real world application. In The 2012 international joint conference on neural networks (IJCNN), pages 1–8. IEEE, 2012. 1

[23] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In International Conference on Machine Learning, pages 5639–5650. PMLR, 2020. 2, 3

[24] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. Advances in Neural Information Processing Systems, 33:741–752, 2020. 2

[25] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research, 17(1):1334–1373, 2016. 1

[26] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018. 7, 8

[27] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In 2018 IEEE international conference on robotics and automation (ICRA), pages 3803–3810. IEEE, 2018. 2

[28] Lerrel Pinto, Dhijraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. In European Conference on Computer Vision, pages 3–18. Springer, 2016. 2

[29] Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. In European Conference on Computer Vision, pages 276–291. Springer, 2020. 7, 8
A. YouTube Driving Dataset

A.1. Dataset overview

Our YouTube Driving Dataset contains a massive amount of real-world driving frames with various conditions, from different weather, different regions, to diverse scene types. We collect 134 videos with a total length of over 120 hours. Figure 8 illustrates the videos’ lengths distribution. For pretraining, we sample one frame every one second, resulting to a dataset of 1.30 million frames in total.

Following YouTube’s license, we will not directly release the frames we collect. We will instead release a list of video links that we use for training. Predicted action labels and models will also be made publicly available.

A.2. Sampled frames

We provide sampled frames of five cities in Figure 9, 10, 11, 12, 13. Note that in captions we describe corresponding weather with blue text, region with red text.
and scene style with green text. The columns are organized by different predicted steering values.

### B. Effects of Batch Normalization on ACO

As suggested in [18], Batch Normalization could help alleviate the problem caused by different distributions of contrastive-learning-based parameters and supervised-learning-based parameters. In this section, we compare the imitation learning performance on ACO and MoCo with and without batch normalization in downstream fine-tuning.

As shown in Table 6, Batch Normalization is essential for both MoCo and ACO. Without adjusting parameters’ distribution by batch normalization, these two contrastive learning methods show declining performance.

### C. Visualization of Auto-Encoder Baseline

To understand the poor performance of auto-encoder in all downstream tasks, we visualize the input and reconstructed image in both YouTube (training) and Carla (testing) domain in Figure 14.

It is obvious that much attention is paid to elements that are not relevant to driving decision-making in reconstructed images, such as buildings and the sky. On the contrary, the policy-relevant features such as the lane lines and road edges are oversimplified. This means that useless information is learned in auto-encoder’s feature and thus damages the performance of downstream tasks.

### D. Implementation Details and Hyper Parameters

We list the hyper-parameters used in ACO pretraining in Table 7, inverse predictor training in Table 8, imitation learning in Table 9, and reinforcement learning in Table 10.

For imitation learning, following CILRS [9], the training target is to optimize L1loss between predicted value and ground truth on steering, throttle, brake, and velocity:

$$L_{il} = \lambda_s L_s + \lambda_t L_t + \lambda_b L_b + \lambda_v L_v,$$  \hspace{1em} (9)

where $L_s$, $L_t$, $L_b$, and $L_v$ stands for L1loss on steering, throttle, brake, and velocity respectively. The value of $\lambda_s$, $\lambda_t$, $\lambda_b$, and $\lambda_v$ we used are listed in Table 9.

For reinforcement learning, we adopt the training pipeline in DI-drive [11].

---

**Table 6. Effects of Batch Normalization in Imitation Learning.** “bn” stands for Batch Normalization.

| Methods       | IL Demonstration Size (×40K) |
|---------------|-------------------------------|
|               | 10%  | 20%  | 40%  | 100% |
| MoCo w/o bn   | 17.3±1.9 | 55.3±16.8 | 59.3±5.7 | 80.0±1.6 |
| MoCo w/ bn    | 19.3±3.4 | 60.3±8.2  | 76.0±9.1  | 80.7±2.5  |
| ACO w/o bn    | 27.3±3.8 | 34.0±4.3  | 72.0±4.3  | 78.0±4.3  |
| ACO w/ bn     | 30.7±3.4 | 66.0±5.7  | 82.0±5.0  | 96.0±3.3  |

**Table 7. Hyper parameters in ACO.**

| Hyper-parameter | Value   |
|-----------------|---------|
| Initial learning rate | 0.03    |
| Weight decay    | 0.0001  |
| Train epoch number | 100     |
| Train batch size | 256     |
| Key dictionary size | 40960  |
| $\lambda_{ins}$ | 1       |
| $\lambda_{act}$ | 1       |
| $\alpha$ for momentum update | 0.999   |
| Hidden size of first layer in Projector | 128     |
| Hidden size of second layer in Projector | 128     |
| $\epsilon$ for action threshold | 0.05    |

**Table 8. Hyper parameters in Inverse predictor.**

| Hyper-parameter | Value   |
|-----------------|---------|
| Learning rate   | 0.0001  |
| Weight decay    | 0.0001  |
| Train epoch number | 50      |
| Train batch size | 256     |

**Table 9. Hyper parameters in Imitation learning (CILRS [9]).**

1DI-drive [11] is an open-source auto-driving platform, supporting reinforcement learning in Carla. The reward scheme can be found at: [https://github.com/opendilab/DI-drive/blob/main/core/envs/simple_carla_env.py](https://github.com/opendilab/DI-drive/blob/main/core/envs/simple_carla_env.py).
Figure 9. Quiet early morning drive at the center of Amsterdam, empty traffic on straight roads, with urban buildings surrounded.
Figure 10. Afternoon drive along the Italian Riviera, along narrow roads with frequent turnings.
Figure 11. Afternoon drive around Asakusa, through traditional streets. Left-hand riding with middle density traffic.
Figure 12. Night drive in Madrid Spain, through Straight roads with middle density traffic.
Figure 13. Sunset drive in Lan Kwai Fong, Hong Kong. Left-hand riding with middle traffic and some one-way streets.
Figure 14. Reconstructed scenes by AutoEncoder in YouTube and Carla domain.

| Hyper-parameter         | Value       |
|-------------------------|-------------|
| Learning rate           | 0.0003      |
| Weight decay            | 0.000001    |
| SGD epoch number        | 6           |
| λ for GAE [30]          | 0.95        |
| Train batch size        | 1024        |
| SGD mini batch size     | 256         |
| Clip ratio              | 0.2         |

Table 10. Hyper parameters in Reinforcement learning (PPO [31]).