AN INFORMATION-THEORETIC APPROACH TO UNSUPERVISED KEYPOINT REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Extracting informative representations from videos is fundamental for the effective learning of various downstream tasks. Inspired by classical works on saliency, we present a novel information-theoretic approach to discover meaningful representations from videos in an unsupervised fashion. We argue that *local entropy* of pixel neighborhoods and its evolution in a video stream is a valuable intrinsic supervisory signal for learning to attend to salient features. We, thus, abstract visual features into a concise representation of keypoints that serve as *dynamic information transporters*. We discover in an unsupervised fashion spatio-temporally consistent keypoint representations that carry the prominent information across video frames, thanks to two original information-theoretic losses. First, a loss that maximizes the information covered by the keypoints in a frame. Second, a loss that encourages optimized keypoint transportation over time, thus, imposing consistency of the information flow. We evaluate our keypoint-based representation compared to state-of-the-art baselines in different downstream tasks such as learning object dynamics. To evaluate the expressivity and consistency of the keypoints, we propose a new set of metrics. Our empirical results showcase the superior performance of our information-driven keypoints that resolve challenges like attendance to both static and dynamic objects, and to objects abruptly entering and leaving the scene.

1 INTRODUCTION

Humans are remarkable for their ability to form representations of essential entities in visual cues and store information to effectively learn downstream tasks from experience (Cooper, 1990; Radulescu et al., 2021). Much research evidence shows that the human visual system processes visual information in two stages; first, it extracts sparse features of salient objects (Bruce & Tsotsos, 2005); second, it discovers the interrelations of local features for grouping them to find correspondences (Marr, 2010; Kadir & Brady, 2001). When it comes to videos with dynamic entities, humans not only focus on dynamic objects, but also on a significant part of the structure of the background scene if this plays a key role in the information flow (Riche et al., 2012; Borji et al., 2012). Ideally, we want a learning algorithm to extract similar abstractions that can be useful to various downstream tasks. Notable research works in Computer Vision (CV) and Machine Learning (ML) have proposed different feature representations from pixels for challenging downstream tasks (Szeliski, 2010; Harris et al., 1988; Lowe, 2004; Rublee et al., 2011; Rosten & Drummond, 2006; Mur-Artal et al., 2015). Recent efforts focus on deep learning representations of Points of Interest (PoI) for tasks like localization and pose estimation (DeTone et al., 2018; Florence et al., 2018; Sarlin et al., 2020; Ono et al., 2018; Sarlin et al., 2019; Dusmanu et al., 2019). However, these methods usually extract an unstructured cloud of salient points that lack interpretability but help establish correspondences.
Keypoints stand out as PoI with semantic interpretation (Jiang et al., 2009; Alexe et al., 2010), e.g., representing objects (Xiongwei et al., 2020) or the joints of a human pose (Kreiss et al., 2019) and can represent structure useful for learning control (Xiong et al., 2021). Many keypoint detection methods are trained in a supervised fashion, relying on annotations (Cao et al., 2017). Unsupervised and self-supervised learning methods can compensate for the need for expensive human annotations (Wang et al., 2019; 2020; Minaee et al., 2021; Kim et al., 2019; Yang et al., 2020; Gopalakrishnan et al., 2020; Chen et al., 2019). Current state-of-the-art methods for unsupervised keypoint discovery mainly focus on dynamic entities in a video (Kulkarni et al., 2019; Minderer et al., 2019), not

effectively representing the scene's static and dynamic entities. Namely, these methods are trained to reconstruct differences between frames and cannot easily disambiguate occlusions or consistently represent randomly appearing-then-disappearing objects in a video stream.

This work introduces Maximum Information keypoiNTs (MINT), an information-theoretic treatment of keypoint-based representation learning by considering keypoints as the "transporters" of prominent information in a frame and subsequently through a video stream. Our proposed method relies on *local entropy* computed in neighborhoods (patches) around candidate keypoints. We argue that image entropy, and its changes over time, provide a strong *inductive bias* for training keypoints to represent salient objects, as early works in saliency detection pointed out (Kadir & Brady, 2001; Bruce & Tsotsos, 2005). To compute the entropy, we introduce a novel, efficient *entropy layer* that can operate locally on image patches. MINT maximizes both the image entropy coverage by the keypoints and the conditional entropy coverage across frames. To do so, MINT relies on an original formulation of unsupervised keypoint representation learning with loss functions that seek to *maximize the represented image entropy* and the *information transportation across frames* by the keypoints, imposing spatio-temporal consistency of the represented entities.

We provide qualitative and quantitative empirical results on four different video datasets that allow us to unveil the representation power of MINT against strong baselines in the unsupervised keypoint discovery literature. Unsupervised keypoint representation learning is not easy to benchmark due to the absence of designated metrics and datasets. We, therefore, provide a new set of metrics with a downstream task in the domain of multiple object detection and tracking, based on CLEVRER (Yi et al., 2019). Our keypoint-based representation is suitable for learning also dynamics, as it preserves the structure of the scene. Moreover, we provide results on two challenging datasets (Sharma et al., 2018; Memmesheimer et al., 2019) that contain interesting scenes of various difficulties (close-up frames with dynamic interactions vs. high-res wide frames with clutter). We show that MINT economizes the use of keypoints, deactivating excessive ones when the information is well contained, but it dynamically activates them to represent new entities that enter the scene temporarily and then leave. Finally, to demonstrate the applicability of MINT as a suitable representation for control, we devise an imitation learning downstream task based on the toy environments of the MAGICAL benchmark (Toyer et al., 2020).

To summarize, our contributions are: *(1)* an original information-theoretic approach to unsupervised keypoint representation learning that uses local image entropy as a training inductive bias, seeking to maximize the represented information in videos; *(2)* an entropy layer that can compute local image entropy in patches; *(3)* an unsupervised way for learning to represent a variable number of entities in video streams by activating/deactivating keypoints to cover the necessary information; *(4)* a new set of evaluation metrics in a simple and intuitive downstream task for benchmarking the performance of unsupervised keypoint discovery methods.

2 Maximum Information Keypoints

We propose an unsupervised method for keypoint discovery in videos based on information-theoretic principles. Our goal is to extract keypoints for salient entities in a video stream, and, thus, to detect keypoints for both static and dynamic objects in the scene, and to consistently represent them over time. We achieve this by maximizing the information covered by the keypoint representation in a video through the introduction of two novel losses: *(1)* An information maximization loss, which encourages the keypoints to cover areas with high entropy in a single frame. *(2)* An information transportation loss, that encourages the keypoints to represent the same entity over subsequent frames. In the following, we present the principles of our method, the model architecture and the novel losses.

2.1 Pixel information & Local image entropy

Our information-theoretic approach for unsupervised keypoint discovery requires quantifying the amount of information each pixel location in a single frame carries. We measure the information of a pixel via Shannon's entropy definition (Shannon, 2001), based on the probability of each pixel. We compute the discrete probability density of a pixel using the statistics of the color intensities in its neighborhood, represented by a normalized histogram of the neighboring pixel values (Sabuncu, 2006). In order to compute these histograms efficiently and to derive the final entropy image, we developed a computationally optimized entropy layer as detailed in Appendix B.

Our entropy layer estimates the pixel-wise entropy image $E(I)$ for an RGB input image $I \in$

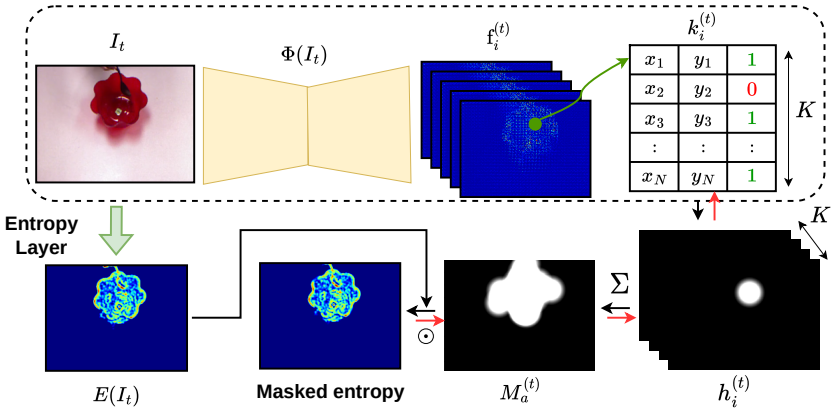

Figure 1: The architecture of our keypoint model $\Phi(I_t)$ (Sec. 2.2) and the masked entropy (Sec. 2.2.1). For an input image I_t our model $\Phi(I_t)$ outputs K feature maps $\mathrm{f}_i^{(t)}$ for each keypoint $\mathrm{k}_i^{(t)}$, $i \in \{1, \dots, K\}$. A heatmap $h_i^{(t)}$ is generated for each keypoint, while the active keypoints are aggregated to form the mask $M_a^{(t)}$. The entropy layer computes the entropy of the image $E(I_t)$. Our masked entropy (ME) loss aims to maximize the percentage of the entropy in the masked entropy image. The red arrows show the backward gradient flow. Only the part encircled by the dashed line is used during inference.

$\mathbb{R}^{H \times W \times 3}$, with H being the height and W the width of an image frame with 3 color channels. $E(I)$ consists of the local entropies $E(I(x,y))$ computed at each pixel location (x,y) by estimating the entropy of the neighborhood region $R(x,y)$ centered at (x,y), using a normalized histogram-based discrete probability function $p(b, R(x,y))$ for each color value b in the region $R(x,y)$ summed and normalized over the color channels (details in Appendix B). The final per-pixel local entropy is

$$E(I(x,y)) = -\sum_{b \in [0,255]} p(b, R(x,y)) \log(p(b, R(x,y))) \,. \tag{1}$$

Computing the entropy for all pixels results in the pixel-wise entropy image $E(I)$.

2.2 Information-theoretic keypoint discovery

We consider the keypoints as a compact representation of images, which attend to salient entities in a scene. In the literature, keypoint features are described by the appearance of pixel patches surrounding the point location (Szeliski, 2010). Correspondingly, in our method, keypoints represent distinctive information patterns overlaid on a set of neighboring pixels in an image frame. We explicitly treat the keypoint as a transporter of information in its surrounding patch, and we develop an end-to-end approach for unsupervised keypoint discovery in videos.
We define a keypoint discovery model $\Phi(I_t)$ (cf. Fig. 1), which is a deep neural network that discovers K keypoints $\mathrm{k}_i^{(t)}$, $i \in \{0, ..., K\}$, in an input color image I at time t. It outputs K feature maps $\mathrm{f}_i^{(t)}$, each corresponding to one keypoint. The coordinates $(x_i, y_i)_t$ of the respective keypoint $k_i^{(t)}$ are obtained with a spatial soft-argmax (Levine et al., 2016). Besides predicting the coordinates, the model also assigns an activation status $s_i^{(t)} = \{0, 1\}$ per keypoint. The activation status determines whether a keypoint is active ($s_i^{(t)} = 1$) or not ($s_i^{(t)} = 0$) in a specific frame, allowing the network to learn to decide on the ideal number of active keypoints. Overall, a keypoint is defined by its coordinates and the activation score $\mathrm{k}_i^{(t)} = (x_i, y_i, s_i)_t$. Additionally, we define a differentiable heatmap $h_i \in \mathbb{R}^{H \times W}$ for each ith keypoint by thresholding a distance-based Gaussian G_i with mean at the coordinates of the keypoint. The detailed procedure is presented in Appendix A.3. The heatmaps localize the information coverage of each keypoint. The heatmaps are part of the backward pass during training, and their differentiability is essential for our method. Next, we explain our novel losses that allow us to learn information-driven keypoint representations.

2.2.1 Maximization of Keypoint Information

With information maximization (IM) we encourage keypoints to seek regions rich in information, typically corresponding to salient entities (Bruce & Tsotsos, 2005). We leverage local image entropy

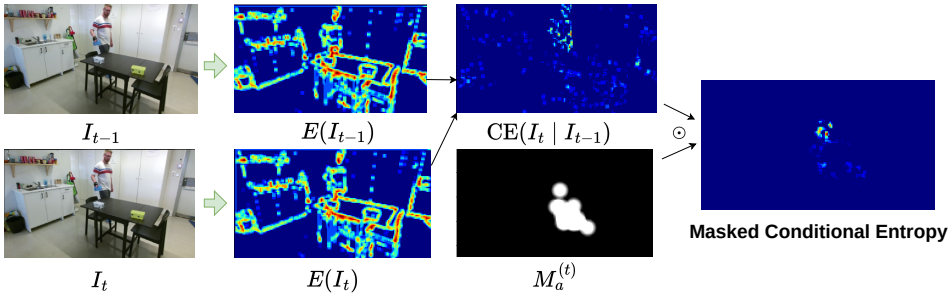

Figure 2: MCE computation. Given two consecutive images I_{t-1} and I_t, we extract the entropy image of each of them $E(I_{t-1})$ and $E(I_t)$. The conditional entropy $CE(I_t|I_{t-1})$ depends on the entropy image of the two images. Multiplying the conditional entropy image by the aggregated mask $M_a^{(t)}$ gives the masked conditional entropy. The MCE loss maximizes the percentage of the masked conditional entropy.

(Sec. 2.1) to define losses for keypoint discovery. The IM loss targets maximum collective information coverage by the keypoints in a frame, for representing all entities in a scene. Our IM loss is effectively composed of two parts; the masked entropy loss and the masked conditional entropy loss.

The **ME loss** encourages the information coverage by the keypoints in a single frame. We use the heatmap $h_i^{(t)}$ of each keypoint $k_i^{(t)}$ to pool the information at time t, serving as a weight for the entropy image $E(I_t)$. The heatmap masks out far away regions, focusing only on the information around the keypoint. By multiplying the heatmap by the activation status flag $s_i^{(t)}$, we filter out inactive keypoints. Aggregating the heatmaps while accounting for the activation status constructs the aggregated mask $M_a^{(t)} = \min(\sum_i^K h_i^{(t)}(x_i, y_i) \odot s_i^{(t)}, 1)$. Summing the pixel-wise product of the aggregated mask $M_a^{(t)}$ and the entropy image $E(I_t)$ over all pixel locations (x, y) consolidates the information transferred by all keypoints. The ME loss $\mathcal{L}_{ME}(I_t)$ optimizes the percentage of the represented information $\sum_{x,y} E(I_t) \odot M_a^{(t)}$ to the available information $\sum_{x,y} E(I_t)$, giving

$$\mathcal{L}_{ME}(I_t) = 1 - \frac{\sum_{x,y} E(I_t) \odot M_a^{(t)}}{\sum_{x,y} E(I_t)} = 1 - \frac{\sum_{x,y} E(I_t) \odot \min(\sum_{i=1}^K h_i^{(t)}(x_i, y_i) \odot s_i^{(t)}, 1)}{\sum_{x,y} E(I_t)} , \quad (2)$$

where $\odot$ denotes the Hadamard (i.e., element-wise) product.

The **masked conditional entropy (MCE) loss** encourages the keypoints to pay special attention to dynamic entities when the available keypoints are insufficient to cover all the information. In information theory, the conditional entropy measures the new information available in a random variable w.r.t. a given random variable. Accordingly, the conditional entropy of an image I_t at time step t given a reference image I_{t-1} at time step $t-1$ contains information about the pixels that have changed, indicating objects which have moved. Optimizing the conditional entropy $CE(I_t|I_{t-1})$ in a sequence of images encourages the keypoint detector to attend to moving objects (cf. Fig. 2). The pixel-wise conditional entropy of an image can be computed by the joint entropy of two images subtracting the reference image entropy $CE(I_t|I_{t-1}) = E(I_t, I_{t-1}) - E(I_{t-1})$. The joint pixel-wise entropy can be approximated as the pixel-wise maximum of the entropy of two consecutive image frames $E(I_t, I_{t-1}) = \max(E(I_t), E(I_{t-1}))$. Similar to the ME loss, the MCE loss $\mathcal{L}_{MCE}(I_t, I_{t-1})$ maximizes the percentage of the masked conditional entropy $\sum_{x,y} CE(I_t|I_{t-1}) \odot M_a^{(t)}$ to the total conditional entropy $\sum_{x,y} CE(I_t|I_{t-1})$

$$\mathcal{L}_{MCE}(I_t, I_{t-1}) = 1 - \frac{\sum_{x,y} CE(I_t|I_{t-1}) \odot M_a^{(t)}}{\sum_{x,y} CE(I_t|I_{t-1})} . \quad (3)$$

2.2.2 Maximization of Keypoint Information Transportation

The **information transportation (IT)** ensures the spatio-temporal consistency of keypoint discovery by solving a keypoint transportation problem. Keypoints should carry information about the same entity over time. Hence, temporal consistency correlates with aligning each keypoint to the same information pattern over its occurrences. Building on this idea and drawing inspiration from the feature transportation in (Kulkarni et al., 2019), we propose the IT loss. Differently to (Kulkarni et al., 2019), we deploy the entropy to assess the transportation, waiving the need for image reconstruction.

In a temporal sequence of frames, optimized keypoint information transport means that we can reconstruct the information of the current frame $E(I_t)$ using information from previous frames $E(I_{t-1})$ and the keypoint information from both frames (cf. Fig. 3). As the heatmaps represent the information coverage of keypoints, the reconstruction process uses them from both time frames ($h_i^{(t)}$ and $h_i^{(t-1)}$), and removes them from the previous frame information to form the source information $E(S_t) = E(I_{t-1}) \odot (1 - h_i^{(t-1)}) \odot (1 - h_i^{(t)})$. Meanwhile, implanting the keypoint information from the current frame into the conditional entropy forms the target information[1] $E(T_t) = E(I_t) \odot (h_i^{(t)}) + \kappa E(I_t|I_{t-1}) \odot (1 - h_i^{(t)})$. The reconstruction result is the sum of the source and target information $E(R_t) = E(S_t) + E(T_t)$. The IT loss seeks to optimize the keypoint transportation by maximizing the mutual information (MI) $\mathcal{I}(R_t, I_t)$ between the reconstructed information $E(R_t)$ and the entropy of the current frame $E(I_t)$. The MI equals $\mathcal{I}(R_t, I_t) = E(R_t) + E(I_t) - E(R_t, I_t)$, with the joint entropy approximated by $E(R_t, I_t) = max(E(R_t), E(I_t))$. The entropy of the current frame is an upper bound for the MI; minimizing the difference between the two values $E(I_t) - \mathcal{I}(R_t, I_t)$ normalized by the area of the heatmap A_h (which is the same for all keypoints) is equivalent to maximizing the mutual entropy. The optimized keypoint transportation requires regularizing excessive keypoints' movement, hence, we minimize the norm of the distance traveled by each keypoint $d_i^{(t)} = ||(x_i, y_i)_t - (x_i, y_i)_{t-1}||_2^2$. The final IT loss becomes

$$\mathcal{L}_{\text{IT}}(I_t, I_{t-1}) = \sum_{i=1}^{K} \frac{E(I_t) - \mathcal{I}(R_t, I_t)}{A_h} + d_i^{(t)} \ . \quad (4)$$

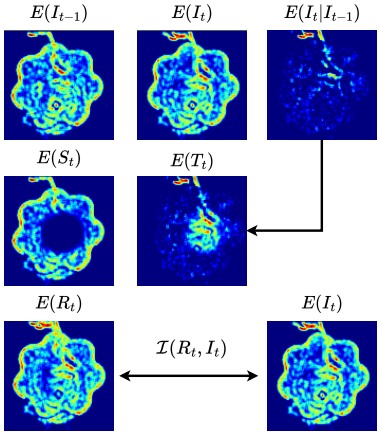

Figure 3: IT. The source is the entropy of the previous image $E(I_{t-1})$ and the target is the entropy of the current frame $E(I_t)$. Removing the patch around the keypoint position at time $t-1$ and t gives the source information $E(S_t)$. The sum of the conditional entropy $E(I_t|I_{t-1})$ and the patch around the keypoint at time $t-1$ gives the target information $E(T_t)$. The reconstructed information after transportation is the sum of target and source information $E(R_t)$. The objective is to maximize the mutual information between the reconstructed entropy and the entropy of the current frame $\mathcal{I}(R_t, I_t)$.

2.2.3 THE MINT LOSS & AUXILIARY LOSSES

The **overlapping loss** provides an auxiliary supervisory signal that spreads the keypoint over the image, encouraging them to cover distinctive regions. The sum of the Gaussians $G_i^{(t)}$ (cf. Appendix A.3) around the keypoints $k_i^{(t)}$ help estimate their overlap, $\mathcal{L}_o = min(max(\sum_i^K G_i^{(t)}) - \beta, 0)/K$. The overlapping loss minimizes the maximum of the aggregation of the Gaussian normalized by the number of keypoints K with a lower bound β to allow occlusion and avoid over-penalization.
The **active status loss** encourages the model to deactivate unnecessary keypoints, by minimizing the normalized sum of active keypoints while maximizing the ME. The interplay of the losses allows the method to eventually reach a trade-off between the number of active keypoints and the covered entropy. The active status loss optimizes $\mathcal{L}_s = (\sum_i^K s_i^{(t)})/K$.
The overall **MINT loss** $\mathcal{L}_{MINT}$ is a weighted combination of all losses (with dedicated weight λ per loss), with the weight of the status loss reversed to schedule it according to the percentage of ME

$$\mathcal{L}_{MINT} = \lambda_{ME}\mathcal{L}_{ME} + \lambda_{CME}\mathcal{L}_{CME} + \lambda_{IT}\mathcal{L}_{IT} + \lambda_o\mathcal{L}_o + (1 - \mathcal{L}_{ME})\lambda_s\mathcal{L}_s \ . \quad (5)$$

Further information about the hyperparameters are available in Appendix D.

3 EXPERIMENTS

We evaluate our information-theoretic approach on four datasets ranging from videos of synthetic objects – CLEVRER (Yi et al., 2019) and MAGICAL (Toyer et al., 2020) – to realistic human video demonstrations – MIME (Sharma et al., 2018) and SIMITATE (Memmesheimer et al., 2019).

[1]The factor $\kappa \leq 1$ encourages the network to concentrate on transportation more than on reconstruction.

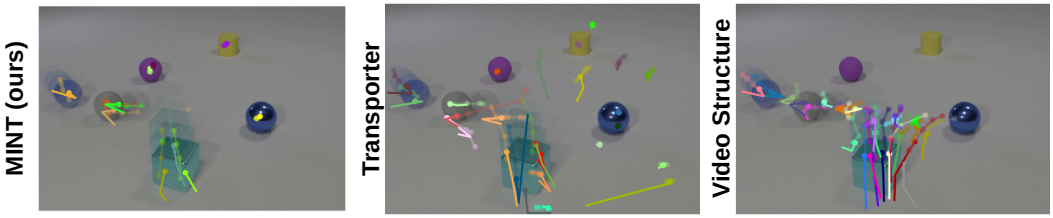

Figure 4: Qualitative results on CLEVRER dataset for Task I (object detection and tracking) and Task II (learning dynamics). Our method is able to assign keypoints to all objects, independent of whether they move or not, and follows their trajectory. The number of keypoints is dynamically adjusted to the number of objects. Future states are transparent, as well as the predicted keypoint and trajectories.

Table 1: Quantitative evaluation of keypoint detection and tracking on CLEVRER (Yi et al., 2019).

Method	**DOP** ⇑	**TOP** ⇑	**UAK** ⇓	**DAK** ⇓
MINT (ours)	**0.855** $\pm$ **0.115**	**0.836** $\pm$**0.118**	**0.798**$\pm$**0.566**	**0.171** $\pm$ **0.151**
Transporter	0.780 $\pm$ 0.110	0.735 $\pm$ 0.115	18.441 $\pm$ 1.643	0.210 $\pm$ 0.148
Video Structure	0.563 $\pm$ 0.256	0.537 $\pm$ 0.249	18.116 $\pm$ 3.502	0.623 $\pm$ 0.360

Table 2: Prediction success rate on CLEVRER (Yi et al., 2019).

Method	1-step prediction	2-steps prediction	3-steps prediction
MINT (ours)	**0.844** $\pm$ **0.116**	**0.827** $\pm$**0.126**	**0.811**$\pm$**0.132**
Transporter	0.764 $\pm$ 0.123	0.743 $\pm$ 0.129	0.723 $\pm$ 0.134
Video Structure	0.734 $\pm$ 0.124	0.719 $\pm$ 0.125	0.699 $\pm$ 0.127

Our experiments aim to show the efficacy of our method as representation for different tasks, and we provide quantitative results w.r.t. evaluation metrics (for object detection and tracking on CLEVRER) and several downstream tasks (learning the dynamics on CLEVRER, imitation learning on MAGICAL). Additionally, we show qualitative results on the challenging datasets of MIME and SIMITATE, demonstrating the superior suitability of our method for real-world deployment. We compare against two recent baselines for unsupervised end-to-end keypoint detection, Transporter (Kulkarni et al., 2019), and Video Structure (Minderer et al., 2019), and against end-to-end CNN-based feature extraction. We report the statistics for all the quantitative results over 5 seeds to show the consistency of our keypoint representation[2].

DOWNSTREAM TASK I: ***OBJECT DETECTION AND TRACKING***

Successful keypoint discovery corresponds to the efficacy of a model to predict keypoints that describe the scene's structure and dynamics. MINT can successfully train a spatio-temporally consistent keypoint representation on videos, leading to a natural application of our model for object (static/dynamic, appearing/disappearing) detection and tracking. We use CLEVRER (Yi et al., 2019) as a testbed. CLEVRER is a dataset for visual reasoning with complete object annotations, and contains videos with static and dynamic objects, with good variability in the structure and dynamics of the scenes. To quantitatively assess the performance of our method for object detection and tracking, we developed evaluation metrics that use object annotations provided in CLEVRER.

Understanding the scene's structure requires detecting all objects in an image. Similarly, tracking the objects over time is essential to represent the scene's dynamics. Accordingly, we propose the **percentage of the detected object (DOP)** and the **percentage of tracked objects (TOP)** as two metrics, with higher values corresponding to better keypoint detection. A keypoint detects an object if it lies on its mask, and tracks it if it detects the same object in the current and the previous frame. Assigning keypoints to areas already represented by other keypoints or empty spaces signals bad keypoint detection. To evaluate these cases, we define two additional metrics for the **density of keypoints in an object (DAK)** and **unsuccessful keypoint assignment (UAK)**, with lower values corresponding to better detection. We refer to the Appendix C for more details.

We train all keypoint detectors on a subset of 20 videos from the CLEVRER dataset and test them on 100 videos. The train-test split emulates a low-data regime and tests the methods' generalization abilities. As seen in Table 1, MINT detects more objects (DOP) and tracks them better (TOP). Meanwhile, it economizes the number of keypoints (UAK), while reducing redundant keypoint assignment (DAK). Visual results in Fig. 4 demonstrate the efficacy of our method in learning

[2]See videos on `https://sites.google.com/view/mint-iclr` or in the supplemental material.

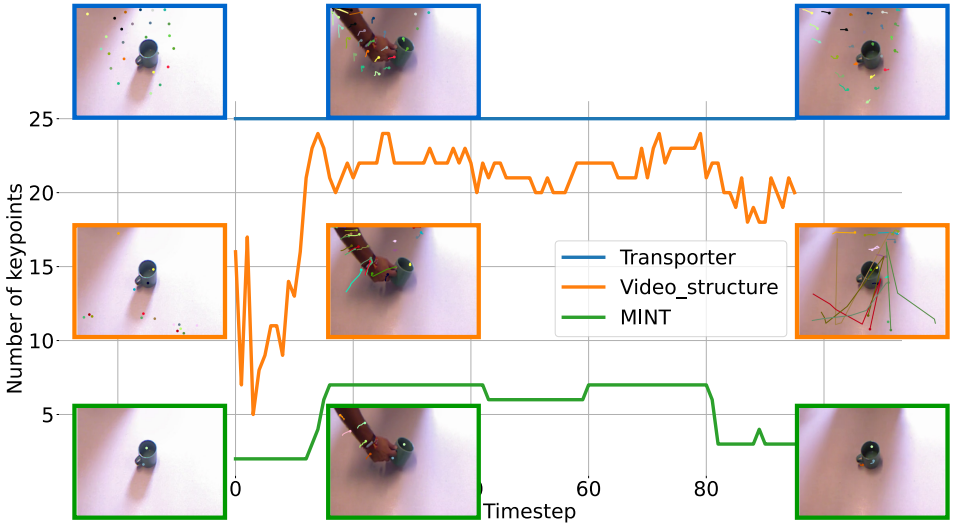

Figure 5: Come-and-go scenario. In a manipulation video from MIME, the hand enters after the start of the video and departs before the end. We plot the number of active keypoints w.r.t. timesteps. Transporter has a fixed number of keypoints. Video structure increases the number of active keypoints when the hand appears, but struggles when it disappears. MINT activates and deactivates the suitable number of keypoints.

superior keypoints apart from our quantitative analysis. MINT detects and tracks the objects in the scene, and its keypoints are more interpretable than the baselines, allowing better scene understanding.

DOWNSTREAM TASK II: ***LEARNING DYNAMICS***

Proper object detection allows us to learn the underlying dynamics that evolve a scene, hence enabling the prediction of the next stage of the scene due to the Physics' law of motion. We test the representation power of the discovered keypoints by training a prediction model (i.e., a model predicting the next state of the objects) using the pre-trained keypoint detectors from Task I (we use the best seed for each method). The prediction model treats the keypoints as graph nodes and uses an Interaction Network (IN) (Battaglia et al., 2016) to model the relational dynamics (cf. Appendix I). We train the prediction model to forecast the future positions of the keypoint given a history window of four-time steps. We compare the prediction against the ground truth position of the object in the predicted frame using again the CLEVRER testbed (Yi et al., 2019). We report in Table 2 the ratio of successfully predicted objects (i.e., a predicted keypoint lying on the same object in the next frame) to the ground truth number of objects in the next time step. The comparison demonstrates that keypoints detected by our method represent the scene better than the baselines and help to predict the next state. Fig. 4 shows the prediction performance using different keypoint detectors.

DOWNSTREAM TASK III: ***OBJECT DISCOVERY IN REALISTIC SCENES***

Our method addresses some challenging aspects in realistic scenes beyond synthetic datasets. We evaluate the keypoint detectors on two additional datasets (1) MIME (Sharma et al., 2018): a collection of close-up videos of human hands manipulating objects, and (2) SIMITATE (Memmesheimer et al., 2019): a dataset of videos of humans performing manipulation tasks in wide-view cluttered scenes. We train the keypoint detectors on a subset of 80 videos from various tasks and test on 100 videos. These datasets do not provide detailed annotations, therefore, we focus mainly on qualitative analysis. In MIME, the human hand enters and leaves the scene abruptly, allowing us to test the performance of our method in come-and-go scenarios. Our method activates only the required number of keypoints, obeying the Occam's razor principle. Meanwhile, the baseline Transporter (Kulkarni et al., 2019) uses a static number of keypoints, and Video structure (Minderer et al., 2019) fails to deactivate the excessive keypoints when the hand disappears. The visual results are depicted in Fig. 5, where we plot the number of active keypoints over time, showing our method's superior performance, both in keypoint economy and in the qualitative representation of the objects in the scene.

Contrarily, SIMITATE provides a good testbed for scenarios with crowded scenes containing static and dynamic objects, with more objects than the number of available keypoints. The qualitative results in Fig. 6 show that only MINT can disambiguate between static and dynamic objects, tracking the human movement while keeping the structure of the keypoints relatively constant over the static objects. The baselines rely on reconstructing the movement, failing to represent the scene's structure.

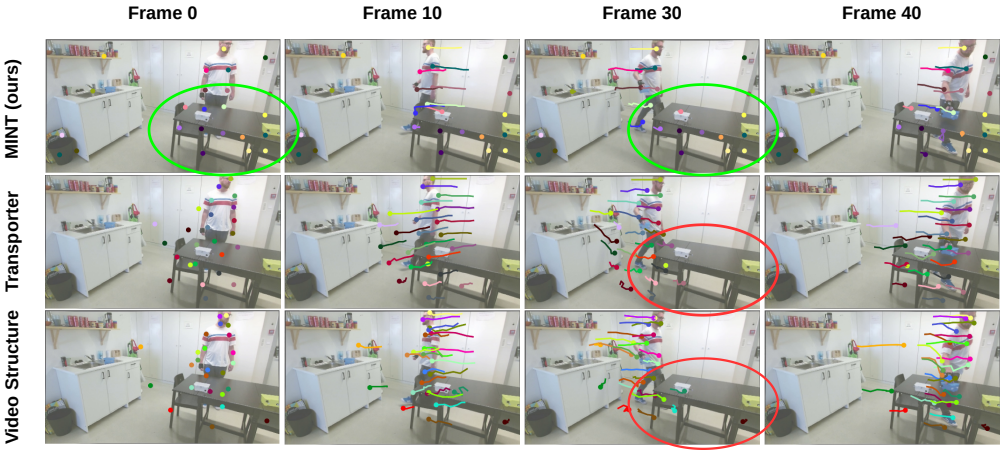

Figure 6: Crowded scenes. A video from SIMITATE dataset with a human moving in a room. All the methods can track the human successfully, but only MINT (ours) can keep the keypoints on the static objects consistently (green ellipses), while the baselines lose track of them (red ellipses).

Table 3: Average score for imitation learning on MAGICAL (Toyer et al., 2020). Higher values are better.

Method	MoveToRegion		MoveToCorner		MakeLine	
	Demo	TestJitter	Demo	TestJitter	Demo	TestJitter
MINT (ours)	**1.00** $\pm$ **0.00**	**0.86** $\pm$ **0.31**	**1.00** $\pm$ **0.00**	**0.80** $\pm$ **0.34**	**0.2** $\pm$ **0.22**	0.06 $\pm$ 0.14
CNN	**1.00** $\pm$ **0.00**	0.84 $\pm$ 0.32	0.74 $\pm$ 0.35	0.30 $\pm$ 0.38	0.00 $\pm$ 0.00	0.01 $\pm$ 0.06

The scenario in SIMITATE proves the qualitative effects of the conditional entropy loss (focusing on moving objects) and the information transportation loss (ensuring the spatio-temporal consistency).

DOWNSTREAM TASK IV: *IMITATION LEARNING*

Imitation learning from video frames is a long-standing challenge for control, where every possible combination of pixel values can define a state. Keypoints can define a low-dimensional representation, that could reduce the computational burden considerably. In this experiment, we investigate whether our keypoint representation is suitable for control tasks, like learning a policy from demonstrations. We consider a toy experiment based on the imitation learning tasks of the MAGICAL benchmark (Toyer et al., 2020). We first pretrain the MINT model on 24 demonstration videos from different tasks of the dataset. Then, we fix the keypoint detector and train an agent (policy) to mimic the demonstrated actions. We use an IN (Battaglia et al., 2016), followed by a fully-connected layer that decodes the actions (cf. Appendix I). The agent uses as input the observed keypoints from four frames. We also found it useful to predict the next state as an auxiliary task. To assess the representation power of the MINT model for policy learning, we compare it against an agent that uses a CNN to extract features directly from the image pixels. The CNN agent is trained from scratch for each environment. Note that the state space for the CNN agent, the image pixels, is one order of magnitude higher than the keypoints' features. For fair comparison, we train the CNN agent longer (twice the epochs used for training the MINT-based agent to counteract for MINT's pretraining). We assess the agents' performances in three environments with different levels of difficulty; **MoveToRegion**: move an agent to a specific region, only the agent is involved - easiest task. **MoveToCorner**: move an object to the top-left corner, one object and the agent are involved - medium difficulty. **MakeLine**: place multiple objects in a line, four objects and the agent are involved - the hardest. We evaluate the learned policy on environment instances from the demonstrations (Demo) and randomly initialized versions (TestJitter). The results in Table 3 reveal that a pretrained keypoint model with MINT is suitable for control, and it achieves comparable or even superior performance to a task-specific CNN-based agent. However, we deem the performance of MINT problematic in MakeLine (Appendix H for details), and we hypothesize that the sparse MINT keypoint representation may require a different network architecture (e.g., a graph attention network) to better encode relations needed for learning this task, or that the available demonstrations are not enough to provide a policy with sufficient support. A viable solution would be to use the imitation learning policy as a boosting for a reinforcement learning agent, however, this is out of the scope of the current work.

LIMITATIONS

Our method relies on local image entropy after filtering high-frequency color changes. As a result,

the method has difficulties in recognizing transparent objects and objects with the same color as the background. We plan to investigate the integration of implicit representation learning to counterpart this issue. Another limitation is the interpretation of the keypoints in the three-dimensional space. The current method operates on images and does not provide 3D information. Adding depth information or extending to a multi-view setting are options for future improvements.

4 Related work

Our work draws inspiration from classical approaches for saliency detection in images, using local information to detect salient objects (Kadir & Brady, 2001; Bruce & Tsotsos, 2005; Fritz et al., 2004; Renninger et al., 2004; Borji & Itti, 2012). The idea of extracting sparse feature representations of high-dimensional visual data is dominant in computer vision and machine learning research (Harris et al., 1988; Lowe, 2004), and connects to the functioning of the human visual system (Marr, 2010). Such sparse representations are generally known as PoI, which are 2D locations that are stable and repeatable under various lighting conditions and viewpoints (DeTone et al., 2018). Traditional geometric computer vision methods relied on the extraction of hand-crafted feature descriptors (Lowe, 2004; Rublee et al., 2011) for tasks like localization (Schmid et al., 2000; Mur-Artal et al., 2015). In the deep learning era, CNN architectures have proven superior to handcrafted features (Yi et al., 2016; DeTone et al., 2018; Song et al., 2020; Zheng et al., 2017). Deep approaches extract clouds of PoI that are useful for correspondence searching in visual place recognition from different viewpoints (Hausler et al., 2021), or pose-estimation for control (Florence et al., 2019).
Keypoints represent a class of PoI that have a semantic entity, e.g., representing objects (Duan et al., 2019), or human joints (Cao et al., 2017; McNally et al., 2021), but most methods rely on explicit annotations of keypoint locations. Related to our work are methods that rely on image reconstruction for detecting keypoints in video streams in an unsupervised fashion (Jakab et al., 2018; Zhang et al., 2018; Minderer et al., 2019; Kulkarni et al., 2019; Gopalakrishnan et al., 2020). Jakab et al. (2018) use an autoencoder architecture with a differentiable keypoint bottleneck trained on the difference between a source and a target image, trying to restrict the information flow. MINT also uses a differentiable keypoint representation, but it operates on the output of an hourglass architecture. Our results suggest that learning to redistribute the information after compression is beneficial for keypoint discovery (Newell et al., 2016). Minderer et al. (2019) use a similar architecture as Jakab et al. (2018) but operate on video sequences for detecting keypoints, using the intensity of the bottleneck heatmap as an indicator of the importance of a keypoint. Setting up a threshold on the intensity is challenging and domain specific. Contrarily, we learn a binary classification of active/inactive keypoints and optimize the number of keypoints used in every frame, as we showed in our results. Kulkarni et al. (2019) propose feature transportation in the keypoint bottleneck of Jakab et al. (2018) before reconstruction. Inspired by this idea, MINT performs information transportation and waives the need for image reconstruction, which would require an additional appearance encoder and a reconstruction decoder from keypoints. Gopalakrishnan et al. (2020) devised a three-stage architecture that first learns a spatial feature embedding, then solves a local spatial prediction task related to object permanence, and finally converts error maps into keypoints. While this method employs local information, it trains three architectural modules separately, unlike (Kulkarni et al., 2019; Minderer et al., 2019) and MINT that use a single end-to-end pipeline for keypoint discovery.

5 Conclusion

We presented MINT, an information-theoretic approach for unsupervised keypoint discovery from videos. Treating keypoints as transporters of information, we defined two losses inspired by information theory; an information maximization loss and an information transportation loss. The losses promote covering areas with high information while ensuring spatio-temporal consistency. Additionally, the method learns to activate keypoints upon necessity and reduces redundant keypoints thanks to auxiliary losses. The experimental evaluation demonstrated the superior performance of our method for various downstream tasks, ranging from object detection to dynamics prediction and imitation learning. Moreover, we showed qualitatively that our method can tackle many challenges for keypoint detection in realistic scenarios, such as attending to static and dynamic objects and handling come-and-go situations in realistic videos. Overall, we demonstrated how to learn reasonable keypoint representations purely unsupervised from videos, with promising results for future applications.

REPRODUCIBILITY STATEMENT

We aim to guarantee the reproducibility of our work by providing additional details of the method and the conducted experiments in the appendix. Moreover, we provide the code in the supplementary material, and we will make it public upon acceptance.

REFERENCES

Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 73–80, 2010.

Mor Avi-Aharon, Assaf Arbelle, and Tammy Riklin Raviv. DeepHist: Differentiable joint and color histogram layers for image-to-image translation. *arXiv preprint arXiv:2005.03995*, 2020.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in Neural Information Processing Systems*, 29, 2016.

Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):185–207, 2012.

Ali Borji, Dicky N Sihite, and Laurent Itti. Quantitative analysis of human-model agreement in visual saliency modeling: A comparative study. *IEEE Transactions on Image Processing*, 22(1):55–69, 2012.

Neil Bruce and John Tsotsos. Saliency based on information maximization. *Advances in Neural Information Processing Systems*, 18, 2005.

Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291–7299, 2017.

Po-Yi Chen, Alexander H Liu, Yen-Cheng Liu, and Yu-Chiang Frank Wang. Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2624–2632, 2019.

Lynn A Cooper. Mental representation of three-dimensional objects in visual problem solving and recognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 16(6):1097, 1990.

Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition workshops*, pp. 224–236, 2018.

Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. CenterNet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6569–6578, 2019.

Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A trainable cnn for joint description and detection of local features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8092–8101, 2019.

Marco Ewerton, Angel Martínez-González, and Jean-Marc Odobez. An efficient image-to-image translation hourglass-based architecture for object pushing policy learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3478–3484. IEEE, 2021.

Peter Florence, Lucas Manuelli, and Russ Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2019.

Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018.

Gerald Fritz, Christin Seifert, Lucas Paletta, and Horst Bischof. Attentive object detection using an information theoretic saliency measure. In *International Workshop on Attention and Performance in Computational Vision*, pp. 29–41. Springer, 2004.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Anand Gopalakrishnan, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Unsupervised object keypoint learning using local spatial predictability. *arXiv preprint arXiv:2011.12930*, 2020.

Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, volume 15, pp. 10–5244. Alvety Vision Club, 1988.

Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. Patch-NetVLAD: Multi-scale fusion of locally-global descriptors for place recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14141–14152, 2021.

Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. *Advances in Neural Information Processing Systems*, 31, 2018.

Yu-Gang Jiang, Jun Yang, Chong-Wah Ngo, and Alexander G Hauptmann. Representations of keypoint-based semantic concept detection: A comprehensive study. *IEEE Transactions on Multimedia*, 12(1):42–53, 2009.

Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

Yunji Kim, Seonghyeon Nam, In Cho, and Seon Joo Kim. Unsupervised keypoint learning for guiding class-conditional video prediction. *Advances in Neural Information Processing Systems*, 32, 2019.

Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. PifPaf: Composite fields for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11977–11986, 2019.

Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *Advances in Neural Information Processing Systems*, 32, 2019.

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.

Yunzhu Li, Antonio Torralba, Anima Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos. *Advances in Neural Information Processing Systems*, 33:9180–9192, 2020.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.

William McNally, Kanav Vats, Alexander Wong, and John McPhee. Rethinking keypoint representations: Modeling keypoints and poses as objects for multi-person human pose estimation. *arXiv preprint arXiv:2111.08557*, 2021.

Raphael Memmesheimer, Ivanna Kramer, Viktor Seib, and Dietrich Paulus. Simitate: A hybrid imitation learning benchmark. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5243–5249. IEEE, 2019.

Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin P Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. *Advances in Neural Information Processing Systems*, 32, 2019.

Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pp. 483–499. Springer, 2016.

Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. LF-Net: Learning local features from images. *Advances in Neural Information Processing Systems*, 31, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

Angela Radulescu, Yeon Soon Shin, and Yael Niv. Human representation learning. *Annual Review of Neuroscience*, 44(1):253–273, 2021.

Laura Renninger, James Coughlan, Preeti Verghese, and Jitendra Malik. An information maximization model of eye movements. *Advances in Neural Information Processing Systems*, 17, 2004.

Nicolas Riche, Matei Mancas, Dubravko Culibrk, Vladimir Crnojevic, Bernard Gosselin, and Thierry Dutoit. Dynamic saliency models and human attention: A comparative study on videos. In *Asian Conference on Computer Vision*, pp. 586–598. Springer, 2012.

Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pp. 430–443. Springer, 2006.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to sift or surf. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564–2571. IEEE, 2011.

Mert Rory Sabuncu. *Entropy-based image registration*. Princeton University, 2006.

Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12716–12725, 2019.

Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4938–4947, 2020.

Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on Robot Learning*, pp. 906–915. PMLR, 2018.

Yafei Song, Ling Cai, Jia Li, Yonghong Tian, and Mingyang Li. SEKD: Self-evolving keypoint detection and description. *arXiv preprint arXiv:2006.05077*, 2020.

Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

Sam Toyer, Rohin Shah, Andrew Critch, and Stuart Russell. The MAGICAL benchmark for robust imitation. *Advances in Neural Information Processing Systems*, 33:18284–18295, 2020.

Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5310–5319, 2019.

Zeyu Wang, Klint Qinami, Ioannis Christos Karakozis, Kyle Genova, Prem Nair, Kenji Hata, and Olga Russakovsky. Towards fairness in visual recognition: Effective strategies for bias mitigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8919–8928, 2020.

Haoyu Xiong, Quanzhou Li, Yun-Chun Chen, Homanga Bharadhwaj, Samarth Sinha, and Animesh Garg. Learning by watching: Physical imitation of manipulation skills from human videos. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 7827–7834. IEEE, 2021.

Wu Xiongwei, Doyen Sahoo, and HOI Steven. PolarNet: Learning to optimize polar keypoints for keypoint based object detection. In *International Conference on Learning Representations*, 2020.

Tianhan Xu and Wataru Takano. Graph stacked hourglass networks for 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16105–16114, 2021.

Ge Yang, Amy Zhang, Ari Morcos, Joelle Pineau, Pieter Abbeel, and Roberto Calandra. Plan2Vec: Unsupervised representation learning by latent plans. In *Learning for Dynamics and Control*, pp. 935–946. PMLR, 2020.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. CLEVRER: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019.

Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned invariant feature transform. In *European Conference on Computer Vision*, pp. 467–483. Springer, 2016.

Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2694–2703, 2018.

Liang Zheng, Yi Yang, and Qi Tian. SIFT meets CNN: A decade survey of instance retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(5):1224–1244, 2017.

APPENDIX

The appendix provides additional details on the architecture, the entropy layer, the evaluation metrics, experimental details with an ablation study, additional results, and discussions, as well as comments on the used code. Moreover, we provide video results for better visualization and the code in the supplemental material.

A ARCHITECTURE

This section contains additional information about the model architecture in Sec. 2.2, and implementation details on how to get the keypoint coordinates from the feature map, and the heatmaps for keypoints to ensure reproducibility.

A.1 KEYPOINT MODEL

Keypoints provide a low-dimensional representation of high-dimensional RGB images. Therefore, many keypoint detection techniques are inspired by the autoencoder architecture (Goodfellow et al., 2016) which uses the bottleneck to consolidate the information into a reduced dimensionality for keypoint extraction (Minderer et al., 2019; Kulkarni et al., 2019). Instead, we suggest taking an hourglass architecture (Newell et al., 2016) which upscales the compressed information again and outputs several feature maps with high activation in places with eminent information (Ewerton et al., 2021; Xu & Takano, 2021; Newell et al., 2016). This allows the network to predict the information at the original image size yielding finer resolution and correspondence to the original pixels.

Our keypoint detector consists of an hourglass convolutional neural network with three convolutional layers, with kernel sizes of $5, 3, 3$ and strides of $3, 2, 2$, respectively. The upsampling part of the model consists of three transposed convolutional layers, with kernel sizes of $3, 3, 3$ and strides of $1, 2, 2$. The number of input and output channels for each layer depends on the number of keypoints K and the number of input image channels C, see Table 4. The result is passed through a softplus layer to ensure the positivity of the feature maps. Lastly, we append a spatial softargmax layer (see Appendix A.2) to get the coordinates of the keypoints from the feature maps f_i. We initialize all the convolutional layers with Xavier's normal initialization (Glorot & Bengio, 2010) and add a leaky ReLU activation and a batch normalization layer after each of them. The total number of the parameters is $58,725$ for the input of size 320×420. We normalize the input to the range $[-0.5, 0.5]$.

Table 4: Architecture details for an RGB-image of 320×480 and $K = 25$ keypoints, there is a leaky ReLU layer and a BatchNorm2d layer (50 parameters) after each convolutional layer.

layer (type)	Input channels	Output channels	kernel size	stride	Output shape	num_params
Normalize-1	C	C	-	-	[3, 320, 480]	0
Conv2d-1	C	K	5	3	[25, 79, 106]	1,900
Conv2d-2	K	K	3	2	[50, 26, 35]	5,650
Conv2d-3	K	2K	3	2	[50, 26, 35]	11,300
ConvTranspose2d-1	2K	2K	3	1	[50, 53, 71]	22,550
ConvTranspose2d-2	2K	K	3	2	[25, 107, 143]	11,275
ConvTranspose2d-3	K	K	3	2	[25, 107, 143]	5,650
Softplus-14	K	K	-	-	=	0
SpatialSoftargmaxLayer-15	K	K	-	-	[25, 2]	0
Final output					[25,3]	total = 58,725

A.2 FEATURE MAP TO KEYPOINT

The argmax operator is not differentiable, so we opted to use a differentiable spatial soft argmax as an alternative to extract the keypoints coordinates from the feature maps. The spatial soft argmax (Levine et al., 2016) takes K 2D feature maps f_i, flattens the feature maps and compute for each of them the weights ω_i

$$w_i = \frac{e^{f_i - max(f_i)}}{\sum e^{f_i - max(f_i)}} \ . \tag{6}$$

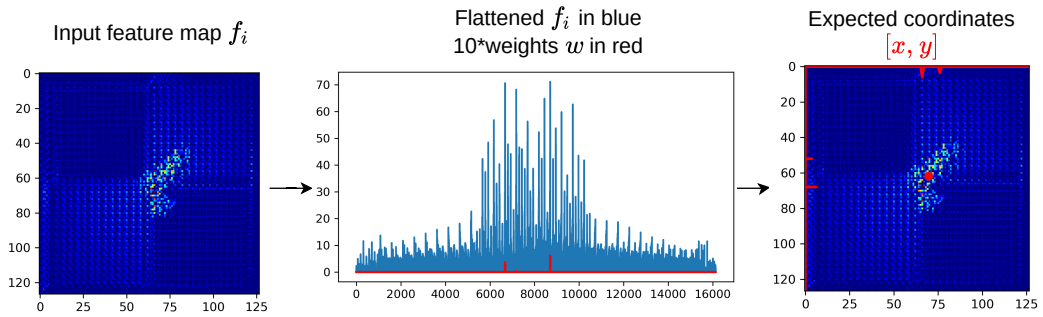

Figure 7: Feature map to keypoint (spatial-soft argmax)

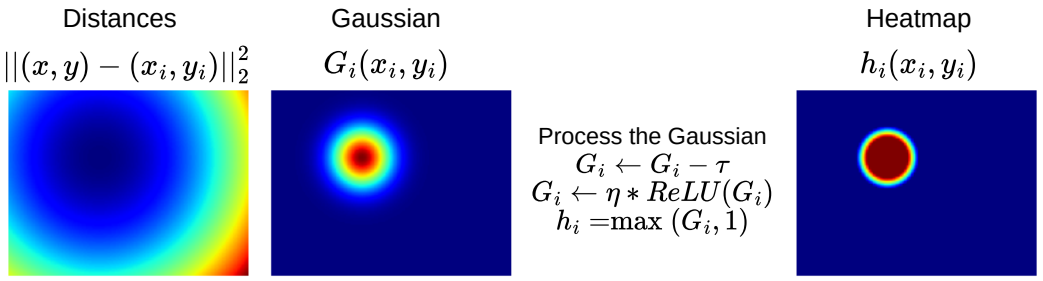

Figure 8: Keypoints to heatmap (Gaussian around keypoints)

Before applying the softmax, we subtract the maximum value from the input, which does not change the output of the softmax but helps for numerical stability. In order to map the weights to coordinates, we generate a mesh grid (x_{grid}, y_{grid}) of x and y coordinates, with the same size as the input image. We flatten the mesh grid and compute the expected coordinates $[\hat{x}_i, \hat{y}_i]$ as the weighted sum of the coordinate grid with ω_i. This process is visualized in Fig. 7.

A.3 Keypoints to heatmaps

The heatmap h_i generation for a keypoint takes coordinates as a pair of real numbers $(x_i, y_i) \in \mathbb{R}^2$. We start by generating a pixel-coordinates array with the same width and height as the original image $H \times W \times 2$, where 2 denotes the coordinates of each pixel $(x, y) \in \mathbb{N}^{H \times W \times 2}$. Then we compute the squared distance between the input and all the pixels $||(x, y) - (x_i, y_i)||_2^2$. We use the squared distance to generate a Gaussian distribution around the input coordinates $G_i(x_i, y_i)$ with a standard deviation σ_{G_i}.

The heatmap defines the area and weighting of information belonging to each keypoint. The heatmap should be 1 around the center of the keypoint, as the keypoint covers the information in this point completely, and descend gradually to 0 representing information out of reach of the keypoint. We achieve that by thresholding and clamping the Gaussian. We use a threshold τ and a scaling factor η for the thresholded Gaussian to get the final heatmap h_i. Table 5 for more details on the scale of these hyperparameters. The process is visualized in Fig. 8.

B Entropy layer

The entropy layer (Sec. 2.1) is the one of the main modules of our method. For an input image, the entropy layer outputs the entropy image. In this section, we provide additional implementation details. We split the explanation into two subsections: *(1)* the entropy module definition in PyTorch (Paszke et al., 2019), and, *(2)* the CUDA extension for the parallel execution.

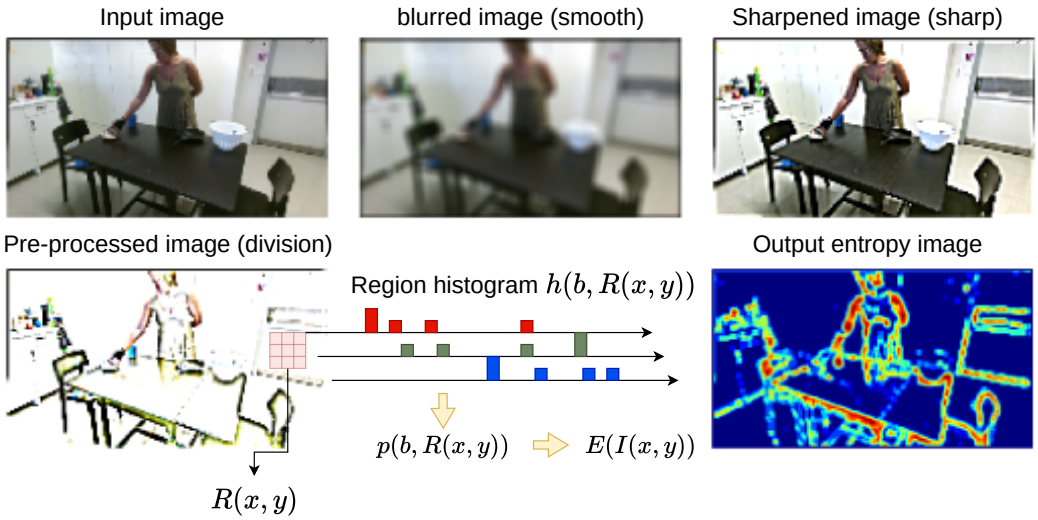

Figure 9: Entropy layer with the prepossessing

B.1 ENTROPY MODULE

The input of the entropy module is an RGB image $I \in \mathbb{R}^{H \times W \times 3}$. The input image is processed to remove high-frequency color changes. The processing is completely vectorized to allow efficient execution using PyTorch during training. The processing of the input image starts by blurring the image using an average blur layer, followed by sharpening the result, and finally dividing the sharp image by the smooth image. Fig. 9 reflects an example of the processing inside the entropy layer.

Before forwarding the processed image into the entropy function, we generate the neighborhood region $R(x, y)$ for each pixel location (x, y). These regions have square shape with the corresponding pixel (x, y) being the center. Instead of iterating over all the pixels with for loops, we use the stridden operation to factorize the extraction of the regions. Using strides to extract the regions aligns with how the image is stored in the memory and does not create overhead. These tricks are essential for the efficiency of our entropy layer.

B.2 CUDA EXTENSION

We need the histogram of the color values to compute the entropy. However, the naive implementation via vectoring the code requires computing a pairwise distance-matrix between each pixel with every histogram bin, which means multiplying the number of the possible regions by 256. This causes exploding GPU memory requirements (more than 50GB). Motivated by this observation, we present in this work an efficient entropy layer based on kernel density estimation.

To estimate the value for each histogram bin b inside a region $R(x, y)$ centered on pixel at location (x, y), we use the kernel density estimator

$$\hat{f}(b, x, y) = \sum_{(x_n, y_n) \in R(x,y)} \mathcal{K}(\frac{I(x_n, y_n) - b}{B}) \ , \tag{7}$$

where $I(x_n, y_n)$ is the pixel value at location (x_n, y_n) in the region $R(x, y)$, and B is the bandwidth, used as a smoothing parameter.
We follow (Avi-Aharon et al., 2020) and use the derivative of the logistic regression function, the Sigmoid function $\sigma(.)$, as a kernel $\mathcal{K}(.)$, that is for a variable v

$$\mathcal{K}(v) = \frac{d}{dv}\sigma(v) = \sigma(v)\sigma(-v) \ . \tag{8}$$

The integral of the function $\hat{f}(b, x, y)$ defined in Eq. (7) over the region gives the histogram value of the bin b in a color channel c:

$$hist_c(b, R(x, y)) = \sum_{(x_n, y_n) \in R(x,y)} \left[\sigma(\frac{I_c(x_n, y_n) - b - L/2}{B}) - \sigma(\frac{I_c(x_n, y_n) - b + L/2}{B}) \right] , \tag{9}$$

where $L = 1/256$ is the bin size, so that each bin represents a color value. We get the probability of each color value by dividing the sum of the histogram values by the size of the region $|R|$ and the number of channels C

$$p(I(x, y)) = p(b, R(x, y)) = \frac{1}{C \cdot |R|} \sum_{c \in \{r,g,b\}} hist_c(b, R(x, y)) . \tag{10}$$

The entropy of the pixel in the center of the patch is

$$E(I(x, y)) = - \sum_{(x_n, y_n) \in R(x,y)} p(I(x_n, y_n)) \log(p(I(x_n, y_n))) \tag{11}$$

$$= - \sum_{b \in [0,255]} p(b, R(x, y)) log(p(b, R(x, y))) . \tag{12}$$

The entropy module uses our entropy function, implemented as an autograd function in PyTorch, to realize the CUDA extension of the entropy computation. The input to the entropy function is the regions of the processed images $R(x, y)$. The CUDA extension allocates a GPU block for each region, hence, the grid size equals to the number of all possible regions for all images in the batch. The block size is 256 threads, i.e., a thread for each bin b. Each thread iterates over the whole region and computes the histogram of its corresponding bin value b according to Eq. (9). Then, it normalizes the result by the region size and the number of channels to get the probability Eq. (10). And finally, each thread computes the entropy of the pixel Eq. (11), which is equivalent to the sum of the entropy of the histogram bins Eq. (12).[3]

C EVALUATION METRICS FOR OBJECT DETECTION AND TRACKING

Each keypoint should provide a representation of a feature in an object, and keypoints should be distinctive and distributed over the shallow scene. Keypoints assigned to empty spaces are considered unsuccessfully assigned. To judge the performance of our method, we developed metrics that use the object masks provided by CLEVRER (Yi et al., 2019) over a set of test videos V, each of which of length T.

C.1 PERCENTAGE OF THE DETECTED OBJECT (DOP).

We consider an object detected if there is at least one keypoint on its mask M_{obj}. At each time frame, we count the percentage of detected objects with respect to the ground truth (GT) number of objects and average these values over the whole video. We get the final result by averaging the value over all the videos in the test dataset. Better detection corresponds to a higher percentage of detected objects.

$$M_{DOP} = \frac{1}{V \cdot T} \sum_{v=1}^{V} \sum_{t=1}^{T} \frac{N_{detected}}{N_{GT}} , \tag{13}$$

where $N_{detected}$ is the number of detected objects (at least one keypoint lies in the object mask)

$$N_{detected} = \sum_{obj \in O} \left[\sum_{i=1}^{K} \mathbb{I}((x_i, y_i) \in M_{obj}) \right] > 0, \tag{14}$$

and N_{GT} is the ground truth number of objects and O is the set of all objects in the scene.

[3]The entropy layer is part of the submitted code, that will be open-sourced upon acceptance.

C.2 PERCENTAGE OF TRACKED OBJECTS (TOP).

We consider an object tracked if there is at least one keypoint on its mask in the current and the previous timeframe. At each time frame, we count the percentage of tracked objects with respect to the ground truth (GT) number of objects and average these values over the whole video. We get the final result by averaging the value over all the videos in the test dataset. Better detection corresponds to a higher percentage of tracked objects.

$$M_{TOP} = \frac{1}{V \cdot T} \sum_{v=1}^{V} \sum_{t=1}^{T} \frac{N_{tracked}}{N_{GT}} \tag{15}$$

Where $N_{tracked}$ is the number of tracked objects (at least one keypoint lies in the object mask in time frames t and t-1)

$$N_{tracked} = \sum_{obj \in O} \left[\sum_{i=1}^{K} [\mathbb{I}((x_i, y_i)^{(t)} \in M_{obj}^{(t)}] \cdot [\mathbb{I}((x_i, y_i)^{(t-1)} \in M_{obj}^{(t-1)})] \right] > 0. \tag{16}$$

C.3 UNSUCCESSFUL KEYPOINT ASSIGNMENT (UAK).

A keypoint is unsuccessfully assigned in a time frame if it does not belong to any object. We average the number of unsuccessful keypoint over the whole video, and then over test videos to get a global value over the testset

$$M_{UKA} = \frac{1}{V \cdot T} \sum_{v=1}^{V} \sum_{t=1}^{T} N_{uk} \ , \tag{17}$$

where N_{uk} is the number of unsuccessful keypoints (does not belong to the sum of the masks)

$$N_{uk} = \sum_{i=1}^{K} \sim \mathbb{I}((x_i, y_i) \notin \sum_{obj \in O} M_{obj}) \ . \tag{18}$$

A lower unsuccessful keypoint assignment metric M_{UKA} corresponds to better keypoints activation.

C.4 DENSITY OF KEYPOINTS IN AN OBJECT (DAK).

Assigning keypoints to areas already represented by other keypoints signals bad, because unnecessary, keypoint detection. The DAK metric accounts for the number of keypoints over the area of the object. The number of keypoints on an object mask should be proportional to its area A_{obj}. We assume a keypoint can represent some area of pixels A_k. To get the density, we divide the product of A_k with the number of keypoints on an object mask by the total area of the object

$$M_{DAK} = \frac{1}{V \cdot T} \sum_{v=1}^{V} \sum_{t=1}^{T} \prod_{obj \in O} \frac{A_k n_{obj}}{A_{obj}} \ , \tag{19}$$

where A_k is the representation area of a keypoint (e.g. average object areas in the dataset) A_{obj} is the area of the object's mask and n_{obj} is the number of keypoints assigned to the object

$$n_{obj} = \sum_{i=1}^{K} \mathbb{I}((x_i, y_i) \in M_{obj}) \qquad \text{for each } \ obj \in O \ . \tag{20}$$

The lower the value of M_{DAK}, the better, because more efficient, is the distribution of the keypoints.

The metrics judge the efficacy of methods for keypoint detection and tracking collectively, where only detected objects can be tracked, so the DOP metric is an upper bound to the TOP metric. Additionally, the value of the metric DAK will go to zero in the case of not detecting any object. It serves its purpose only if the DOP value is higher than 75% and the UKA value is less than 1.

Table 5: Hyperparameters

Parameter name	value	Parameter name	value	Parameter name	value
learning rate	0.001	clip value	10.0	weight decay	0.00001
epochs	100	num keypoints K	25	number of stacked frames	3
activation threshold γ	15	entropy region size $\sqrt{\lvert R \rvert}$	3	std for heatmap σ_{G_i}	9.0
Threshold for heatmap τ	0.1	Thresholded heatmap scale η	3.5	CE contribution (IT) κ	0.5
ME weight λ_{ME}	100	MCE weight λ_{MCE}	100	IT weight λ_{IT}	20
active status weight λ_{s}	10	overlapping weight λ_{o}	10		

Table 6: Ablation study on MINT losses.

Method	**DOP** $\Uparrow$	**TOP** $\Uparrow$	**UAK** $\Downarrow$	**DAK** $\Downarrow$
MINT (IM+IT+S+O)	0.855 $\pm$ 0.115	0.836 $\pm$0.118	**0.798**$\pm$**0.566**	**0.171** $\pm$ **0.151**
IM	**0.951** $\pm$ **0.042**	**0.927** $\pm$**0.05**	7.046$\pm$1.675	2.612 $\pm$ 2.361
IM+IT	0.879 $\pm$ 0.104	0.861$\pm$0.107	1.689$\pm$0.880	0.433 $\pm$ 0.452
IM+IT+S	0.883 $\pm$ 0.097	0.864$\pm$0.102	1.578$\pm$0.769	0.429 $\pm$ 0.525

D HYPERPARAMETERS

Table 5 provides the hyperparameters used for CLEVRER dataset (Yi et al., 2019) in our experiments. We use the same values for all other datasets, *i.e.*also for MIME (Sharma et al., 2018), SIMITATE (Memmesheimer et al., 2019), and MAGICAL (Toyer et al., 2020). The only exceptions are the activation threshold γ, the std for heatmap σ_{G_i} and the threshold of the heatmap τ where we use values depending on the size of the input image (*i.e.*, $\gamma = 15$,$\sigma_{G_i} = 9.0$, $\tau = 0.1$ for MIME, $\gamma = 10$,$\sigma_{G_i} = 9.0$, $\tau = 0.7$ for SIMITATE, $\gamma = 15$,$\sigma_{G_i} = 7.0$, $\tau = 0.3$ for MAGICAL). Our method requires a sequence of 2 frames for the loss computation, and we found that the batch size does not affect the training and can be chosen based on the available GPU resources.

E ABLATION STUDY

We investigated the contribution of each loss in MINT for the CLEVRER dataset using the proposed evaluation metrics. We report the results in Table 6. We compared the final loss (MINT) Eq. (5) to the information maximization loss alone (IM), using the information maximization and the information transportation losses (IM+IT) and adding the status loss (IM+IT+S). The results show that the IM loss can outperform all the losses in detecting and tracking objects (highest **DOP** and **TOP** values), but assigns a lot of redundant and unsuccessful keypoints (worst **UAK** and **DAK**). Adding the IT loss improves the tracking (**TOP** compared to **DOP**) and reduces the unsuccessful keypoint assignment (lower **UAK**). Introducing the specialized status loss S does not help by itself, but its interplay with the overlapping loss allows the overall MINT loss to achieve the desired behavior, *i.e.*, an overall trade-off in good detection and tracking performance and resource assignment.

F ADDITIONAL VIDEO RESULTS

We provide additional video results both as supplemental material and on the website of our project: https://sites.google.com/view/mint-iclr

G CODE

We include our code in the supplemental material, and we plan to publish our code under an open-source license to accompany the paper upon acceptance. We provide instructions to run the code, with sample datasets to reproduce the results in the paper.

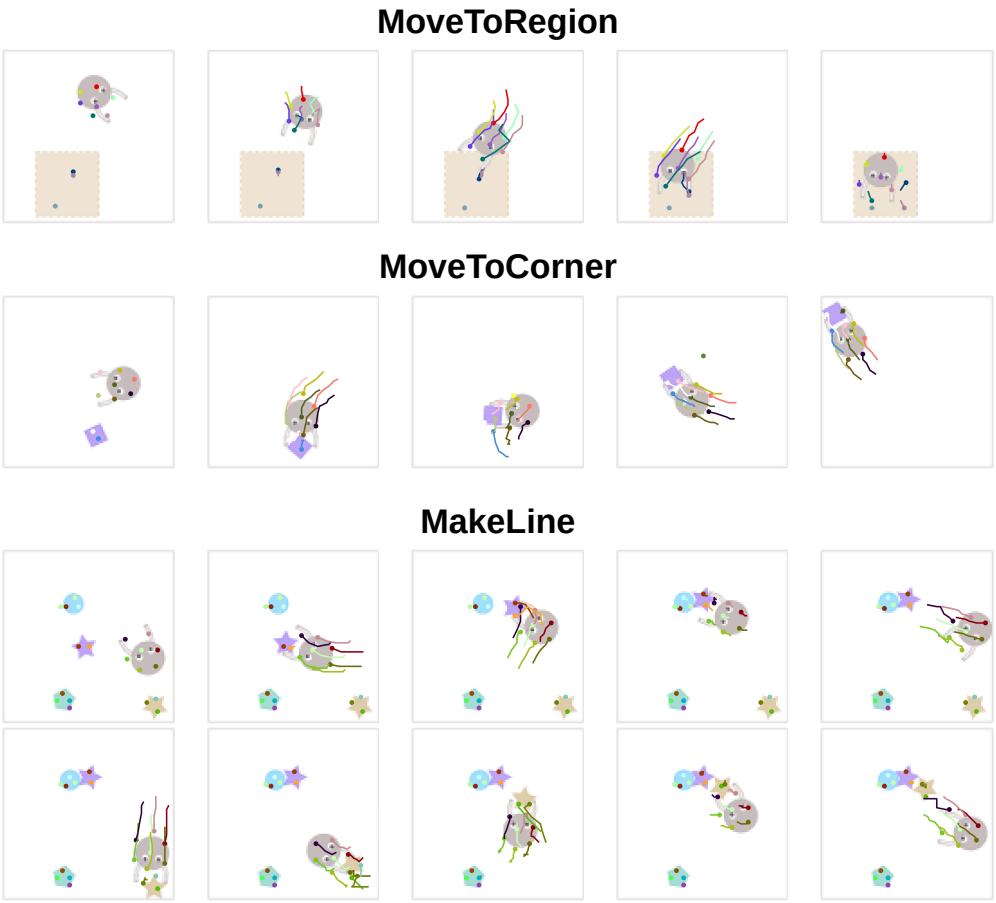

Figure 10: Rollouts from MINT agent in MAGICAL (Toyer et al., 2020) dataset.

The implementation of video structure (Minderer et al., 2019) [4] uses outdated libraries. Due to compatibility reasons, we reimplemented their code in PyTorch with our best effort.

We adapted the implementation of Transporter from Li et al. (2020) [5] into the codebase of MINT [6].

H IMITATION LEARNING RESULTS

We provide visualizations of the learned policies of the MINT-based agent on the three environments of MAGICAL in Fig. 10. The visualization shows that MINT can assign reasonable keypoints for the agent and all the objects in the environment. The imitation agent can solve the first two tasks **MoveToRegion** and **MoveToCorner**, but it struggles with the last task **MakeLine**. The agent receives a score of 1.0 when it sorts all 4 objects in one line, while it gets a score of 0.5 for putting 3 out of 4 in one line. Our imitation agent could sort only 3 out of 4 in the depicted environment (which led to an overall 0.2 mean score over 5 seeds – Table 3), despite being able to assign keypoints to all objects. The results suggest that there is a problem in encoding the relational features between keypoints, hindering the agent to reason upon getting the right locations. We argue that further investigation of the appropriate model to pool information from the keypoints is necessary to solve this harder task, but this is out of scope of the current work.

[4] https://github.com/google-research/google-research/tree/master/video_structure
[5] https://github.com/pairlab/v-cdn
[6] The baselines are part of the submitted code, that will be open-sourced upon acceptance.

I INTERACTION NETWORK ARCHITECTURE

The interaction network (Battaglia et al., 2016) is a model developed for learning the interaction relations between physical objects to infer the physics of the environment. The interaction network treats the objects as nodes of a graph, with the relations as edges. In our case, we use the keypoints as object nodes, with the coordinates, status, and positional encoding as features. We form a fully connected graph of the keypoints, with no relational features for edges.

The interaction network used in our experiments has two sub-models; a relational model and an object model. The relational model uses the relational information and object attributes to predict the effects of all interactions. The object model uses the effects to update the features of the object. We encode node features before passing them to the interaction network. After one pass through the interaction network, we decode the features into coordinates for the prediction task, and we add another prediction head for the action decoding in the imitation learning task.