Methods	TAP			SCROLL			INPUT			Overall
	default	cursor	custom	default	cursor	custom	default	cursor	custom	
V2S [8]	84.19%	69.66%	36.10%	85.19%	63.31%	29.00%	-	-	-	61.24%
GIFdroid [4]	85.78%	88.01%	87.16%	72.84%	71.01%	69.77%	35.13%	32.11%	28.39%	63.35%
CAPdroid	**91.06%**	**90.28%**	**92.67%**	**94.87%**	**94.63%**	**95.12%**	**87.86%**	**88.62%**	**88.11%**	**91.46%**

TABLE III: Performance comparison of action attribute inference.

Fig. 8: Examples of bad cases in action localization.

leveraged the metadata of action attributes as the ground-truth. Since our approach employs a deep-learning-based model (Section II-C1) to infer *TAP* location, we trained and tested our model based on the metadata of *TAP* actions. Note that a simple random split cannot evaluate the model's generalizability, as tapping on the screens in the same app may have very similar visual appearances. To avoid this data leakage problem [50], we split the *TAP* actions in the dataset by apps, with the 8:1:1 app split for the training, validation, and testing sets, respectively. We also ensure a similar number of three types of touch indicators (i.e. default, cursor, custom) in the split dataset. The resulting split has 9k actions in the training dataset, 1.5k in the validation dataset, and 1.5k in the testing dataset. The model was trained in an NVIDIA GeForce RTX 2080Ti GPU (16G memory) with 30 epochs. In total, we obtained 1.5k *TAP* locations, 4k *SCROLL* offsets, and 1k *INPUT* text as the attributes of testing data.

Metrics. We employed accuracy as the evaluation metric to measure the performance of our approach in inferring *TAP*, *SCROLL*, and *INPUT* action attributes, respectively. As one element occupies a certain area, tapping any specific point within that area can successfully trigger the action. So, we measured whether our predictions are within the ground-truth element. For *SCROLL* actions, we measured whether our inferred scroll offset is the same as the ground-truth. For *INPUT* actions, we measured whether our approach can infer the correct input text. The higher the accuracy score, the better the approach to infer action attributes.

Baselines. We set up 2 state-of-the-art methods as our baselines to compare with our `CAPdroid`. *V2S* [8] proposed the first GUI video analysis technique, that utilizes deep-learning models to detect the touch indicator for each frame in a video and then classify them to user actions. As V2S only detects the default touch indicator, we followed their procedure to train corresponding deep-learning models to detect cursor and custom indicators. *GIFdroid* [4] developed a novel lightweight tool to detect the user actions by first extracting the keyframes from the GUI recording and then mapping it to the GUI transition graph (UTG) to extract the execution actions. We also followed the details in their paper to obtain the UTG graph.

Results. Table III shows the overall performance of all methods. Our method outperforms in all actions, e.g., on average 91.33%, 94.87%, 88.19% for *TAP*, *SCROLL*, and *INPUT*, respectively. Our method is on average 30.2% more accurate compared with V2S in action attribute inference, due to three main reasons. First, our method models the features from both the spatial (i.e., touch indicator) and temporal (i.e., GUI animation) across the frames to enhance the performance of the model in inferring *TAP* actions, i.e., on average 91.33% vs 63.32% for `CAPdroid` and V2S respectively. Second, our method achieves better performance in inferring action attributes even for the recordings with different touch indicators. This is because, `CAPdroid` proposes a novel touch indicator-independent method by leveraging the similarity of consecutive frames to identify actions, while V2S leverages the opacity of the indicator, e.g., a fully solid touch indicator represents the user first touches the screen, and it fades to less opaque when a finger is lifted off the screen. The opacity of the indicator works well for the default touch indicator (on average 84.69%), but not for the others (on average 66.48%, 32.55% for cursor and custom). Third, `CAPdroid` can accurately (on average 88.19%) infer the input text from the clips, while V2S cannot detect semantic actions.

`CAPdroid` is on average 28% (91.46% vs 63.35%) more accurate even compared with the best baseline (GIFdroid). This is because, the content in GUIs of some apps (e.g., financial, social, music apps) are dynamic, causing the keyframes wrongly map to the states in the UTG. This issue further exacerbates input text inference, as the input text from the recording is specific but the input text in UTG is randomly generated.

Albeit the good performance of our approach, we still make wrong inferences about some actions. We manually check those wrong cases and find two common causes. First, as shown in Fig. 8, the overlap of similar colors between the touch indicators and icons leads to less distinct features of the indicators, causing false-positive action localization. Second, although the good performance of our OCR method, it still makes wrong text recognition, especially missing spaces. We believe the emergence of advanced OCR methods can further improve the accuracy of our approach.

IV. Usefulness Evaluation

In this section, we conducted a user study to evaluate the usefulness of our generated descriptions (reproduction steps) for replaying bug recordings in real-world development environments.

Procedure: We recruited another 8 participants including 6 graduate students (4 Master, 2 Ph.D) and 2 software developers