On Improving Management of Duplicate Video-Based Bug Reports

Yanfu Yan
yyan09@wm.edu
William & Mary
Williamsburg, Virginia, USA

ABSTRACT

Video-based bug reports have become a promising alternative to text-based reports for programs centered around a graphical user interface (GUI), as they allow for seamless documentation of software faults by visually capturing buggy behavior on app screens. However, developing automated techniques to manage video-based reports is challenging as it requires identifying and understanding often nuanced visual patterns that capture key information about a reported bug. Therefore, my research endeavors to overcome these challenges by advancing the bug report management task of *duplicate detection* for video-based reports. The objectives of my research are fourfold: (i) investigate the benefits of tailoring recent advancements in the computer vision domain for learning both visual and textual patterns from video frames depicting GUI screens to detect duplicate reports; (ii) adapt the scene-learning capabilities of vision transformers to capture subtle visual and textual patterns that manifest on app UI screens; (iii) construct a more comprehensive and realistic benchmark which contains video-based bug reports derived from real bugs; (iv) conduct an empirical evaluation to potentially demonstrate state-of-the-art improvements achieved by the proposed approach.

CCS CONCEPTS

• **Software and its engineering → Software evolution**.

KEYWORDS

Bug Reporting, GUI Learning, Duplicate Video Retrieval

ACM Reference Format:
Yanfu Yan. 2024. On Improving Management of Duplicate Video-Based Bug Reports. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3639478.3639786

1 INTRODUCTION

Due to the graphical nature of mobile applications, video-based bug reports are a natural fit for capturing buggy behavior as they depict how a given fault manifests through the graphical user interface (GUI). Additionally, they are simple to create since recording the screen information is now a part of the Android operating system, no longer requiring a third-party application[1]. This coupled with popular issue tracking software such as GitHub Issues supporting the attachment of videos[2], video-based reports are quickly becoming a major modality of the information in the mobile application domain. In fact, a recent study [15] performed on open source applications listed on FDroid [1] spanning the years of 2012-2020 illustrated that over 13k visual recordings were present in issue trackers of mobile apps, with the vast majority of these being uploaded between 2018-2020, indicating a growing popularity.

The increasing prevalence of video-based reports brings with it a growing set of challenges related to their management. Specifically, because of the rich, pixel-based data captured in videos, automatically *capturing* the nuanced patterns depicted is a challenging proposition. This difficulty in the automated analysis of video contents drastically complicates the creation of techniques for automated bug report management, such as triaging, duplicate detection, and fault localization. However, given the size and complexity overhead associated with video files as compared to text files, *duplicate detection* may be one of the more important tasks for managing video-based bug reports. For instance, detecting duplicate videos can allow for new reports to be archived or deleted to save space in software repositories and can reduce the amount of valuable developer time spent (re)watching/comparing videos of bugs to identify, group, and manage duplicates [12].

However, only recently have approaches been explored for detecting duplicate video-based bug reports [12] – due in part to the scarcity of data for training and evaluating such approaches. My research aims to build a new approach to improve the state of the art for duplicate video-based bug report detection [36]. Specifically, I will leverage recent advancements in the computer vision domain, namely, the introduction of the Vision Transformer (ViT) and new training schemes [5, 13], for scaling visual deep learning models. Our hypothesis is that rich hierarchical features of self-supervised ViT models contain explicit scene layout information that helps to distinguish subtle visual patterns in video frames depicting GUI screens. Additionally, the proposed approach will leverage similar advancements in image representation learning and Optical Character Recognition (OCR) by adapting various models, such as the Efficient and Accurate Scene Text Detector (EAST) [40], the CRAFT [2], and the TrOCR [25], for improved localization and recognition of text in video frames, compared to the prior technique that only combine learning-based methods and heuristics [12]. To evaluate the proposed approach, a new benchmark will be created

ICSE-Companion '24, April 14–20, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0502-1/24/04.
https://doi.org/10.1145/3639478.3639786

[1] https://support.google.com/android/answer/9075928?hl=en
[2] https://github.blog/2021-05-13-video-uploads-available-github/

for duplicate detection of video-based bug reports that contains duplicate detection tasks constructed from more real bugs, in order to extend the evaluation dataset used in prior work that relied mostly on synthetic bugs [12]. An empirical evaluation will be conducted on this comprehensive benchmark to potentially demonstrate state-of-the-art performance achieved by the proposed approach when compared to the prior baseline [12].

2 RELATED WORK

GUI Comprehension. GUI understanding can help many software engineering tasks related to mobile applications, such as GUI reverse engineering [3, 7, 27, 38], software testing [4, 26, 29, 37], and GUI search [6, 8]. Most GUI understanding techniques need to detect GUI elements first to understand the information provided by the GUI. Chen *et al.* [9] show that deep learning-based object detection models[14, 30, 31] and scene text detector EAST [40] outperform old-fashioned detection models [28] and OCR tool Tesseract [32] respectively. Fu *et al.* [16] utilize the Transformer architecture for GUI element detection but only based on limited pixel words. The most closely related work to our own [12] uses self-supervised approach SimCLR [10] based on ResNet [18] to understand the visual GUI and use OCR to obtain the textual information in order to detect duplicate video-based bug reports.

Duplicate Video Retrieval. To retrieve similar videos, the traditional techniques in the computer vision domain first extract global and/or local features of video frames, then aggregate extracted features to represent a whole video, and finally calculate similarity scores between videos. The visual features are extracted either by handcrafted image processing methods, such as Local Binary Patterns (LBP) [19, 35], Scale Invariant Feature Transform (SIFT) [34, 39], or by the Convolutional Neural Networks (CNNs) [18, 33]. The features can then be aggregated based on global vectors [34], bag-of-words [11, 22], or deep metric learning [23]. Kordopatis-Zilos *et al.* [21] conducted a comprehensive experimental study comparing feature extraction methods, CNN architectures and aggregation schemes, showing that CNN+BoVW is the best performing combination, which is the reason why the most relevant work [12] chose this strategy to obtain video representations.

3 PROPOSED APPROACH

In this section, a brief overview of the proposed approach is provided. The approach will build upon the success of past techniques [12] and adapts a framework that combines visual and textual information modalities for duplicate video-based bug report detection. Specifically, the approach receives as input two video-based bug reports and outputs a similarity score that indicates how similar they are in depicting the same app bug. Therefore, it can be used to compute scores between a new video-based bug report and a corpus of previously-submitted bug reports. The scores allow for ranking the corpus videos as a list of potential duplicate candidates.

Internally, the proposed approach will begin by taking the two videos and subsampling a number of video frames. Next, it vectorizes each video by discretizing the frames using a ViT-based feature extractor (*e.g.,* [5]) into a Bag of Visual Words (BoVW) representation, for the visual component of the approach, and by extracting the text of each frame [2, 25, 40] and constructing a *video document* of the concatenated text, encoded as a Bag of Words (BoW), for the textual component of the approach. The sequential information can be further added to both visual and textual TF-IDF representations. Each pair of visual or textual representations is then compared via cosine similarity. The visual and textual similarities can be used individually to rank duplicate candidates, or they can be combined into a single similarity score to account for both information modalities to enhance the effectiveness [36]. In addition to this, I will further leverage multimodal Transformers (*eg.* [17, 24]) to fuse visual and textual information by mutually-supervised objectives during training and inference for more accurate video representations.

4 RESEARCH DESIGN

My research aims to investigate the performance of two (visual and textual) components of the proposed approach, as well as the performance of these two components combined together when compared to the baseline technique [12]. To evaluate the performance of different models for duplicate video-based bug report detection, previous work [12] collected 60 distinct bugs across six Android apps. App users further recorded 180 videos to construct 4,860 tasks for evaluation. Given that most of the bugs collected in [12] are injected, synthetic faults, as opposed to real-world faults, we will extend this benchmark by constructing an evaluation dataset containing *only* real bugs. Recently, the AndroR2+ dataset [20] was released which contains 180 manually reproduced bug reports for Android apps. For each bug report, AndroR2+ provides a link to the original bug report, an apk binary of the buggy version of the app, and a reproduction script. Therefore, we would be able to collect more real bugs from the AndroR2+ dataset and record more videos following the pipeline provided by [12] in order to create additional tasks for a comprehensive and realistic benchmark. Additionally, since the duplicate bug report detection is modelled as an information retrieval task, standard information retrieval metrics will be used for the evaluation of the proposed approach's performance, including mean reciprocal rank, mean average precision, *etc.*.

5 ANTICIPATED CONTRIBUTION

My research is intended to facilitate bug report management, particularly focusing on duplicate video-based bug report detection. The proposed approach will be able to analyze a new video-based bug reports and a corpus of previously-submitted ones, and generate a ranked list of potential duplicates in the corpus videos. By watching the videos in the ranked list rather than randomly, the developers can save their time in identifying duplicate bug reports [12] during bug triaging. The anticipated contributions include a novel approach which includes various components that will identify the visual, textual, and sequential information existing in the video-based bug reports. The visual and textual information is potentially to be mutually-supervised for more accurate video representations. To precisely evaluate the proposed approach, I plan to construct a more comprehensive and realistic benchmark which includes more video-based bug reports derived from real bugs. This dataset will be open to the community to foster future work that aims to advance duplicate video-based bug report detection. Ultimately, my research goal is to implement this proposed approach as an usable tool and release it to the community as open-source software.

REFERENCES

[1] 2023. FDroid https://f-droid.org/en/.

[2] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019. Character region awareness for text detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9365–9374.

[3] Tony Beltramelli. 2018. pix2code: Generating Code from a Graphical User Interface Screenshot. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM. https://doi.org/10.1145/3220134.3220135

[4] Carlos Bernal-Cárdenas, Nathan Cooper, Kevin Moran, Oscar Chaparro, Andrian Marcus, and Denys Poshyvanyk. 2020. Translating video recordings of mobile app usages into replayable scenarios. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. ACM. https://doi.org/10.1145/3377811.3380328

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*. 9650–9660.

[6] Chunyang Chen, Sidong Feng, Zhengyang Liu, Zhenchang Xing, and Shengdong Zhao. 2020. From Lost to Found: Discover Missing UI Design Semantics through Recovering Missing Tags. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW2 (Oct. 2020), 1–22. https://doi.org/10.1145/3415194

[7] Chunyang Chen, Ting Su, Guozhu Meng, Zhenchang Xing, and Yang Liu. 2018. From UI design image to GUI skeleton. In *Proceedings of the 40th International Conference on Software Engineering*. ACM. https://doi.org/10.1145/3180155.3180240

[8] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xin Xia, Liming Zhu, John Grundy, and Jinshui Wang. 2020. Wireframe-based UI Design Search through Image Autoencoder. *ACM Transactions on Software Engineering and Methodology* 29, 3 (June 2020), 1–31. https://doi.org/10.1145/3391613

[9] Jieshan Chen, Mulong Xie, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, and Guoqiang Li. 2020. Object detection for graphical user interface: old fashioned or deep learning or a combination?. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM. https://doi.org/10.1145/3368089.3409691

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

[11] Chien-Li Chou, Hua-Tsung Chen, and Suh-Yin Lee. 2015. Pattern-Based Near-Duplicate Video Retrieval and Localization on Web-Scale Videos. *IEEE Transactions on Multimedia* 17 (2015), 382–395.

[12] Nathan Cooper, Carlos Bernal-Cárdenas, Oscar Chaparro, Kevin Moran, and Denys Poshyvanyk. 2021. It takes two to tango: Combining visual and textual information for detecting duplicate video-based bug reports. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 957–969.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[14] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. 2019. CenterNet: Keypoint Triplets for Object Detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. https://doi.org/10.1109/iccv.2019.00667

[15] Sidong Feng and Chunyang Chen. 2022. GIFdroid: automated replay of visual bug reports for Android apps. In *Proceedings of the 44th International Conference on Software Engineering*. 1045–1057.

[16] Jingwen Fu, Xiaoyi Zhang, Yuwang Wang, Wenjun Zeng, Sam Yang, and Grayson Hilliard. 2021. Understanding Mobile GUI: from Pixel-Words to Screen-Sentences. *arXiv preprint arXiv:2105.11941* (2021).

[17] Tsu-Jui Fu, Linjie Li, Zhe Gan, Kevin Lin, William Yang Wang, Lijuan Wang, and Zicheng Liu. 2021. Violet: End-to-end video-language transformers with masked visual-token modeling. *arXiv preprint arXiv:2111.12681* (2021).

[18] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 770–778.

[19] Weizhen Jing, Xiushan Nie, C. Cui, Xiaoming Xi, Gongping Yang, and Yilong Yin. 2019. Global-view hashing: harnessing global relations in near-duplicate video retrieval. *World Wide Web* 22 (2019), 771–789.

[20] Jack Johnson, Junayed Mahmud, Tyler Wendland, Kevin Moran, Julia Rubin, and Mattia Fazzini. 2022. An Empirical Investigation into the Reproduction of Bug Reports for Android Apps. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 321–322. https://doi.org/10.1109/SANER53432.2022.00048

[21] Giorgos Kordopatis-Zilos, Symeon Papadopoulos, Ioannis Patras, and Ioannis Kompatsiaris. 2019. FIVR: Fine-grained incident video retrieval. *IEEE Transactions on Multimedia* 21, 10 (2019), 2638–2652.

[22] Giorgos Kordopatis-Zilos, Symeon Papadopoulos, I. Patras, and Yiannis Kompatsiaris. 2017. Near-Duplicate Video Retrieval by Aggregating Intermediate CNN Layers. In *International Conference on Multimedia Modeling*.

[23] Giorgos Kordopatis-Zilos, Symeon Papadopoulos, I. Patras, and Yiannis Kompatsiaris. 2017. Near-Duplicate Video Retrieval with Deep Metric Learning. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)* (2017), 347–356.

[24] Jie Lei, Linjie Li, Luowei Zhou, Zhe Gan, Tamara L Berg, Mohit Bansal, and Jingjing Liu. 2021. Less is more: Clipbert for video-and-language learning via sparse sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 7331–7341.

[25] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2021. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282* (2021).

[26] Zhe Liu, Chunyang Chen, Junjie Wang, Yuekai Huang, Jun Hu, and Qing Wang. 2020. Owl eyes. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. ACM. https://doi.org/10.1145/3324884.3416547

[27] Kevin Moran, Carlos Bernal-Cardenas, Michael Curcio, Richard Bonett, and Denys Poshyvanyk. 2020. Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps. *IEEE Transactions on Software Engineering* 46, 2 (Feb. 2020), 196–221. https://doi.org/10.1109/tse.2018.2844788

[28] Tuan Anh Nguyen and Christoph Csallner. 2015. Reverse Engineering Mobile Application User Interfaces with REMAUI (T). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. https://doi.org/10.1109/ase.2015.32

[29] Ju Qian, Zhengyu Shang, Shuoyan Yan, Yan Wang, and Lin Chen. 2020. RoScript. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. ACM. https://doi.org/10.1145/3377811.3380431

[30] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 6 (June 2017), 1137–1149. https://doi.org/10.1109/tpami.2016.2577031

[32] R. Smith. 2007. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*. IEEE. https://doi.org/10.1109/icdar.2007.4376991

[33] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *ArXiv* abs/1602.07261 (2016).

[34] Xiao Wu, Alexander Hauptmann, and Chong-Wah Ngo. 2007. Practical elimination of near-duplicates from web video search. *Proceedings of the 15th ACM international conference on Multimedia* (2007).

[35] Zhipeng Wu and Kiyoharu Aizawa. 2014. Self-similarity-based partial near-duplicate video retrieval and alignment. *International Journal of Multimedia Information Retrieval* 3 (2014), 1–14.

[36] Yanfu Yan, Nathan Cooper, Oscar Chaparro, Kevin Moran, and Denys Poshyvanyk. 2024. Semantic GUI Scene Learning and Video Alignment for Detecting Duplicate Video-based Bug Reports. In *2024 IEEE/ACM 46rd International Conference on Software Engineering (ICSE)*. IEEE.

[37] Shengcheng Yu, Chunrong Fang, Yulei Liu, Ziqian Zhang, Yexiao Yun, Xin Li, and Zhenyu Chen. 2022. Universally Adaptive Cross-Platform Reinforcement Learning Testing via GUI Image Understanding. *arXiv preprint arXiv:2208.09116* (2022).

[38] Tianming Zhao, Chunyang Chen, Yuanning Liu, and Xiaodong Zhu. 2021. GUIGAN: Learning to Generate GUI Designs Using Generative Adversarial Networks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE. https://doi.org/10.1109/icse43902.2021.00074

[39] Wanlei Zhao and Chong-Wah Ngo. 2009. Scale-Rotation Invariant Pattern Entropy for Keypoint-Based Near-Duplicate Detection. *IEEE Transactions on Image Processing* 18 (2009), 412–423.

[40] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. 2017. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 5551–5560.