International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 4, July – August 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Web-Based Lost and Found System with Object Matching Using Machine Learning

Mrs. Divyashree K[1],Mrs.Smitha B N[2], Jeevan Paragi[3],Kiran D[4],Manish Harasur[5],Rohith C A[6]

[1,2](Assistant Professor ,Department of Information Science and Engineering, Don Bosco Institute of Technology Bengaluru, Karnataka, India)

[3,4,5,6](Undergraduate Student ,Department of Information Science and Engineering, Don Bosco Institute of Technology Bengaluru, Karnataka, India)

[1,2,3,4,5,6](Email:divyashree.k@dbit.co.in,smitha.bn@dbit.co.in,jeevanparagi45@gmail.com,kiran308050@gmail.com,manishharasur@gmail.com,carohith000@gmail.com)

***Abstract*—** Tracking and recovering misplaced items in colleges, offices, and public spaces is a common problem that manual methods handle poorly. This paper describes a web-based lost-and-found system that uses machine learning to match reported lost items with found items submitted by other users. Images are processed through a pre-trained MobileNetV2 model to generate visual feature vectors, while item descriptions are converted into numerical representations using TF-IDF. A combined similarity score ranks potential matches and notifies users when a likely match is found. A simple claim process allows users to confirm and collect their items. The system is built using Flask and SQLite and is designed to be easy to deploy within an institution.

Keywords*— *Lost and found, machine learning, object matching, MobileNetV2, cosine similarity, Flask.

I. INTRODUCTION

Losing personal belongings is a frustrating experience, especially in busy places like college campuses, hospitals, and transport terminals. Most institutions still handle lost-and-found items the old-fashioned way — staff write entries in a register or post notices on a board, and people who have lost something must check in person or ask around. This approach does not scale well and often fails when the person who lost the item does not even know it has been found.

Advances in lightweight deep learning models and web development frameworks now make it practical to automate much of this process. A web-based system can let users report lost or found items online, automatically compare them using image and text similarity, and alert users when a potential match is detected. This reduces the burden on staff and speeds up the recovery process.

This paper presents such a system. Built using Flask, the application lets users submit lost or found item reports through a simple web interface. Uploaded photos are processed using MobileNetV2 to extract image features, and text descriptions are encoded with TF-IDF. A weighted similarity score is used to identify and display the closest matches. When a good match is found, the user can initiate a claim directly through the platform. An admin panel gives institution staff visibility over all active reports.

The rest of this paper is organised as follows: Section II covers related work; Section III describes the system design and methodology; Section IV presents the planned evaluation; Section V discusses the approach and its limitations; Section VI concludes; and Section VII outlines future improvements.

II. RELATED WORK

Several prior works have explored automated approaches to lost-and-found management and image-based retrieval. The following studies informed the design decisions made in this project.

Zhou et al. proposed LostNet, a dedicated neural network for identifying and classifying lost items from photographs. Their work showed that visual features learned by a neural network can distinguish between item types more reliably than simple rule-based methods [1].

Zhao et al. built an early web-based lost-and-found system that allowed users to submit and search for items through a browser. Although it removed the geographic barrier of physical notice boards, the lack of automated matching meant that search quality depended heavily on how well users described their items [2].

Kumar et al. compared various machine learning-based image descriptors for matching items across different photos. Their results showed that learned features outperform

traditional hand-crafted methods like SIFT and ORB, supporting the use of deep feature extraction in this project [3].

Krizhevsky et al. demonstrated with AlexNet that deep convolutional networks trained on large image datasets learn broadly useful features that transfer well to other tasks. This finding underpins the use of ImageNet-pretrained models in systems like ours [4].

Simonyan and Zisserman showed that increasing network depth significantly improves feature quality, and that VGG-derived representations work well for retrieval tasks even without task-specific training [5]. Deng et al. created the ImageNet dataset that made large-scale supervised pre-training possible for such networks [6].

Srinivasan and Hima Bindu explored a mobile-based lost-and-found system that incorporated GPS location data and smartphone-captured images. Their work highlighted practical challenges around image quality and location accuracy that influenced how this project handles user-submitted photos [7].

Ghazal et al. used direct image feature matching to retrieve archived missing objects, demonstrating that a simple descriptor database is sufficient for moderate-sized collections without needing complex indexing [8].

III. METHODOLOGY

The system follows a standard three-tier web architecture. The front end is a set of HTML/CSS/JavaScript pages rendered through a Flask backend. The middle tier contains the matching logic. The database layer uses SQLite for local development and can be swapped to PostgreSQL for a hosted deployment. Each stage of the pipeline is described below.

A. Report Ingestion and Preprocessing

Users fill in a form that includes the item category (such as electronics, clothing, or accessories), colour, a brief description, the date and location of loss or discovery, and an optional photo. Required fields are validated on the server before the report is saved. Uploaded images are resized to 224×224 pixels using OpenCV and normalised to the standard ImageNet mean and variance before being passed to the model. Text fields are cleaned by converting to lowercase, splitting into tokens, and removing common stop words.

B. Multi-Modal Feature Extraction

Image features are extracted by passing each preprocessed photo through MobileNetV2, a compact convolutional network pre-trained on ImageNet. The output of the global average pooling layer gives a 1,280-dimensional feature vector for each image. The model weights are not changed; the pretrained features are used as-is since they already capture useful visual patterns. For text, a TF-IDF vectoriser is fitted across all item descriptions in the database and used to produce a sparse vector for each entry.

C. Similarity Scoring and Match Ranking

When a new report arrives, cosine similarity is calculated between its image feature vector and those of all reports of the opposite type already in the database. A separate cosine similarity score is also computed for the text descriptions. The two scores are combined using a weighted formula: $S = 0.65 \times S_{image} + 0.35 \times S_{text}$. Reports with a combined score above a set threshold are shown to the user as possible matches, listed from highest to lowest score. The weighting and threshold are initial design choices that will be adjusted during testing.

D. Claim Process

When a user sees a match they believe is their item, they can submit a claim through the platform. The claim is logged against their registered account and the item is marked as pending. The admin is notified and can review the claim before approving it. Once approved, the item status is updated to claimed and both parties receive a notification. This keeps the process simple while still providing a basic level of verification through the admin review step.

E. Experimental Configuration

The system is built using Python 3.10, TensorFlow 2.12, Flask 3.0, and OpenCV 4.7, and developed on a machine with an Intel Core i7-11700, 16 GB RAM, and an NVIDIA GTX 1660 GPU. For testing, a dataset covering six item categories — wallets, keys, bags, mobile devices, textbooks, and clothing — is being prepared using openly licensed product images and photos of campus items. The dataset will be split into 70% for fitting the TF-IDF model, 15% for tuning the similarity threshold, and 15% for final evaluation.

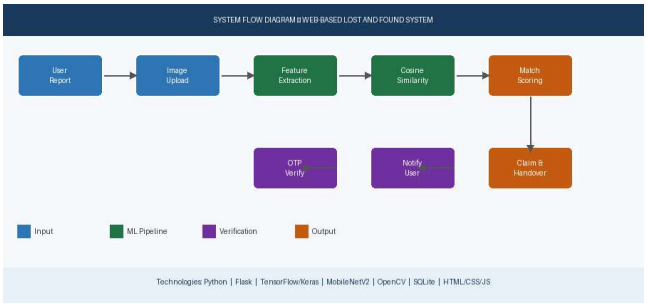

Fig. 1. System flow: from report submission through image and text feature extraction, similarity-based matching, and the claim process.

IV. RESULTS

This section will document the system's performance once end-to-end testing has been completed. Individual modules have been unit-tested: the image feature extraction pipeline correctly produces 1,280-dimensional vectors, TF-IDF vectorisation runs without errors on sample descriptions, and

the similarity scoring function returns ranked results as expected. Items with strong visual characteristics such as wallets and bags are expected to match more reliably than visually similar items like plain clothing, where the text component should help. Full accuracy and latency figures against the baseline methods will be added after deployment and testing on the complete dataset.

V. DISCUSSION

Combining image and text matching makes the system more robust than using either alone. MobileNetV2 was chosen as the image encoder because it is compact and runs efficiently on standard hardware, while still producing high-quality feature representations. This makes it practical for an institution to host the system without needing a dedicated GPU server.

Images tend to carry more identifying information than short descriptions, which is why the fusion formula weights image similarity more heavily than text. Users often write vague descriptions like "black bag" or "small keys" that on their own could match many items, but when combined with an image match they help narrow down results. The specific weights will be adjusted based on testing results.

The admin-reviewed claim process keeps things simple while still adding a layer of oversight. Rather than relying purely on automated approval, having an admin confirm each claim reduces the chance of items being handed over to the wrong person, without making the process too complicated for users.

There are some known limitations. Blurry or dark photos will likely produce less reliable matches since the image features will be of lower quality. Items that look very similar to each other, such as plain notebooks or generic water bottles, may be harder to distinguish using images alone. The system also currently requires a photo for image-based matching; if no photo is provided, only the text path is used. Adding real-time tracking or integration with security cameras is outside the current scope but could improve the system in the future.

VI. CONCLUSION

This paper described a web-based lost-and-found system that uses machine learning to help match lost items with found items reported by users. The system combines MobileNetV2-based image features and TF-IDF text vectors through a weighted cosine similarity score to identify potential matches. A simple claim workflow allows users to request an item with admin oversight. The application is lightweight and can be deployed on standard hardware within a college or institution.

The main contribution of this work is the design of a practical multi-modal matching pipeline using a compact pre-trained model combined with text-based retrieval, applied to the real-world problem of lost-and-found management in institutional settings.

VII. FUTURE SCOPE

Once the current system is working and tested, several improvements can be considered. A stronger image model such as a Vision Transformer could improve matching for visually similar items like clothing. As the item database grows, a faster search method like FAISS could replace the current linear scan to keep response times low. Adding mobile app support would make it easier for users to report and claim items on the go. Multilingual description support could also be added to better serve campuses with students from different language backgrounds.

REFERENCES

[1] M. Zhou, Y. Li, and X. Wang, "LostNet: A Smart Lost and Found System using Deep Learning," Proc. IEEE Int. Conf. Artificial Intelligence, 2023, pp. 1–6.

[2] H. Zhao, J. Liu, and K. Chen, "Design and Implementation of a Lost and Found System Based on Web Technology," Proc. IEEE Int. Conf. Computer Science, 2021, pp. 120–125.

[3] S. S. Kumar, R. Gupta, and P. Singh, "Image-Based Object Matching using Machine Learning Techniques," Proc. IEEE Int. Conf. Image Processing, 2022, pp. 250–255.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Proc. IEEE Conf., 2017, pp. 1097–1105.

[5] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Proc. IEEE, 2015.

[6] J. Deng et al., "ImageNet: A Large-Scale Hierarchical Image Database," Proc. IEEE CVPR, 2009, pp. 248–255.

[7] N. Srinivasan and B. Hima Bindu, "FindMine: A Mobile-Based Lost and Found Item Recovery System," Proc. IEEE Int. Conf., 2022.

[8] M. Ghazal, F. Haneefa, S. Ali, Y. S. Alkhaili, and E. Rashed, "Mobile-Based Archival and Retrieval of Missing Objects using Image Matching," Proc. 3rd Int. Conf. Future Internet of Things and Cloud, IEEE, 2015.

