PicNix: Nixing Image Duplicates

Pandre Vamshi
115077003

Prajjawal Agarwal
115134849

Shreejay V. Jahagirdar
115071683

Abstract

In the era of expanding digital content, the proliferation of Intellectual Property (IP) theft on Non-Fungible Token (NFT) websites and the redundant storage of duplicate images pose significant challenges. Hence, the demand for efficient image processing and storage solutions is more pronounced than ever. We address the escalating challenge of increasing memory consumption by images and IP theft by proposing an innovative methodology. The approach involves converting uploaded images into vectors and extracting moment features, subsequently comparing these features with existing cluster centers. Our refined search algorithm efficiently identifies potential matches and allows the system to update references rather than redundantly storing duplicate images. We aim to efficiently curb increasing memory consumption and combat IP theft in the digital landscape.

1 Background and Motivation

In the contemporary digital age marked by the explosive growth of user-generated content, the Non-Fungible Token (NFT) space has emerged as a dynamic marketplace for digital assets, particularly art and collectibles. This surge in popularity, however, has brought about a concerning trend of Intellectual Property (IP) theft, where unauthorized replicas of unique digital creations proliferate on NFT platforms. Simultaneously, the ubiquitous nature of social media platforms compounds the issue, as users continually contribute to the ever-expanding pool of redundant images.

Recognizing the critical need for effective solutions to these challenges, this project focuses on addressing the dual problem of escalating memory consumption due to the proliferation of images and the rampant IP theft within the NFT ecosystem. The conventional approach to image storage often leads to redundant storage of similar images, exacerbating the challenge of efficient data management. To tackle this, our project proposes an innovative methodology that leverages advanced image processing techniques, including the conversion of images into vectors and the extraction of moment features. By adopting this approach, we aim to streamline image storage and combat IP theft by identifying and referencing duplicate or similar images, rather than redundantly storing them.

Through the integration of cutting-edge technologies such as React, Django, Celery, Redis, and OpenCV, our solution not only provides a robust system for users to create, share, and explore NFT-like posts but also offers a pioneering approach to image deduplication. This introduction sets the stage for a detailed exploration of our project's methodologies, results, and the broader implications of our contributions in mitigating the challenges posed by the contemporary digital landscape.

2 Related Work

[1] The clustering technique employed in this project is inspired by the work of Layek et al. in their paper titled "Fast near-duplicate detection from image streams on online social media during disaster events" (2016). Layek and colleagues addressed the critical need for swift near-duplicate detection within the context of online social media during disaster events. Their research focused on developing a method to efficiently identify similar images in rapidly evolving image streams, a scenario with parallels to the dynamic environment of user-generated content on Non-Fungible Token (NFT) platforms and social media.

In Layek et al.'s study, a clustering technique played a pivotal role in the identification of near-duplicate images. The authors leveraged advanced algorithms to rapidly group visually similar images into clusters, thereby facilitating effective duplicate identification. While Layek et al.'s work formed the foundation for our clustering approach, one notable enhancement in our project is the incorporation of image-to-vector techniques for checking two images. Unlike the static clustering approach presented by Layek et al., our project involves converting uploaded images into vectors and extracting moment features, enabling a more refined comparison of images with existing cluster centers.

Moreover, our implementation introduces a dynamic adaptation of the number of clusters as entirely different images appear. Unlike Layek et al.'s use of a static number of clusters, our project offers a more responsive approach to the evolving nature of user-generated content. The system dynamically adjusts the number of clusters to accommodate new and distinct images, contributing to a more adaptive and scalable system that efficiently organizes and retrieves visually similar content in real-time.

Building upon Layek et al.'s foundational work, our project extends the application of clustering techniques to the specific context of NFT platforms, introducing innovations in image processing, moment feature extraction, and the dynamic adjustment of cluster numbers. This comprehensive approach aims to address the challenges of image processing in dynamic environments, contributing to the ongoing discourse on optimizing content management within digital platforms.

3 Methodologies (Tools)

In the development of our project, we harnessed a robust combination of tools and methodologies to ensure the efficiency, scalability, and seamless functionality of our system. Each chosen tool played a distinctive role in addressing specific challenges, collectively contributing to the overall success of our endeavor.

3.1 Celery for Asynchronous Processing

Celery served as the backbone for our project's asynchronous processing needs. By offloading time-intensive tasks, such as image processing, to Celery, we ensured that our application remains responsive during our create post workflow and provides an optimal user experience. This approach allowed us to maintain a non-blocking workflow, enhancing the overall efficiency of our system.

3.2 Redis as a Message Broker

Redis, our chosen message broker, played a crucial role in facilitating seamless communication among different components, particularly with Celery. By employing Redis as a message broker, we enhanced the coordination of tasks, ensuring effective distribution and processing of asynchronous operations. This integration significantly contributed to the scalability and performance of our Celery-based system.

3.3 Django Framework

Django, a versatile web framework, was instrumental in shaping the structure of our project. Its comprehensive feature set and seamless integration with tools like Celery made it the ideal choice. By leveraging Django's Object-Relational Mapping (ORM) with SQLite, we established a robust foundation for our database-backed web application. The integration of Django and Celery provided a cohesive and powerful framework for our project.

3.4 OpenCV for Image Processing

Image processing, a critical component of our project, was efficiently handled by OpenCV. This powerful library allowed us to convert images into vectors and extract moment features, forming the basis for our innovative approach to image deduplication. OpenCV's versatility in image manipulation contributed significantly to achieving our project's goals and ensuring the integrity of the visual content on our platform.

3.5 React.js

React.js played a pivotal role in crafting a dynamic and responsive user interface, enhancing the overall user experience. Users interact with our system through an intuitive and engaging interface built with React.js, allowing seamless image uploads and post creation. It helped to provide users with a dynamic timeline to explore and interact with NFT-like posts.

4 Deliverables

Our deliverable is a system with the following functionalities:

- User Interface:
  - A fully functional and responsive user interface developed using React.js with functionalities similar to an NFT website except for purchase capabilities.
  - Features include the ability to create posts, view all posts, view posts with similar images, and view posts with duplicate images.
- Backend:
  - Backend developed using Django, serving as the server-side logic for the application.
  - CRUD (Create, Read, Update, Delete) operations implemented to manage user posts and associated data
- Image Processor:
  - Integration of image deduplication functionality, utilizing OpenCV for image processing and clustering techniques.
  - Detection and handling of similar and duplicate images to prevent redundancy and enhance user experience.
  - Image processing tasks are efficiently handled in the background to prevent delays in user interactions.

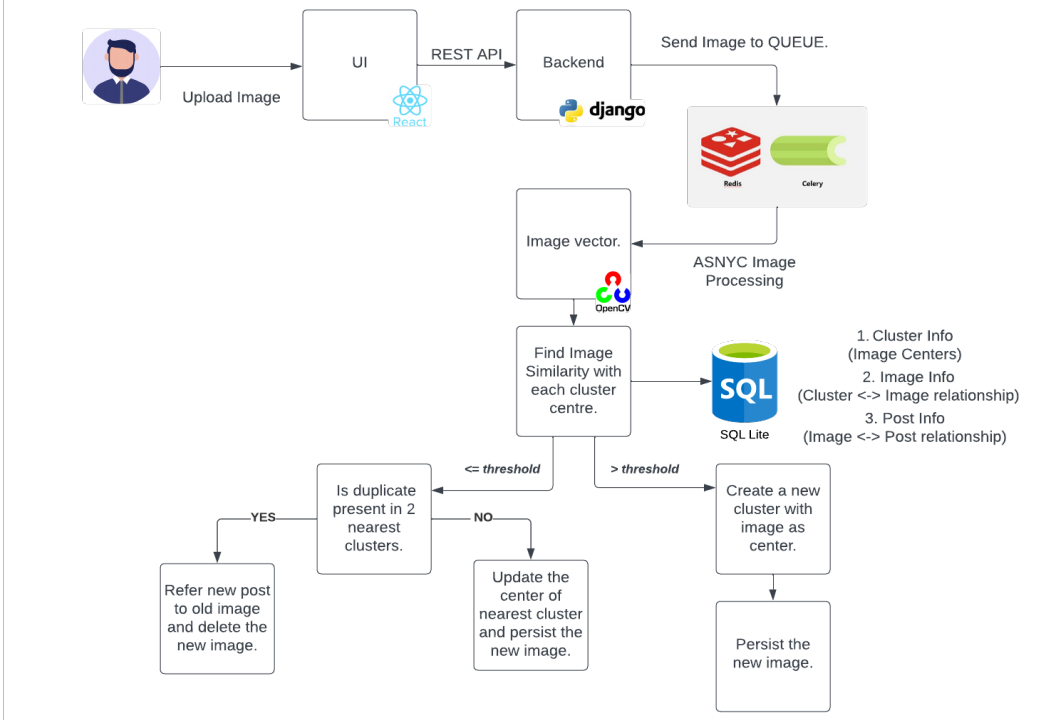

Figure 1: Architecture diagram

5 Approach

Our approach (described in Figure 1) to image deduplication is meticulously designed to address the challenges posed by redundant image storage and intellectual property (IP) theft.

The process seamlessly integrates various technologies, including React.js for the user interface, Django for the backend, Celery for asynchronous processing, and OpenCV for image manipulation. The approach can be delineated into distinct stages:

5.1 User Interaction and REST API Call

The process begins with users interacting with the system through an intuitive React.js-based user interface. Upon uploading an image, a REST API call is initiated to the Django backend, triggering the start of the deduplication workflow.

5.2 Asynchronous Processing with Celery and Queue

Utilizing Celery for asynchronous task execution, the uploaded image is pushed to a message queue. This design ensures a non-blocking workflow, preventing delays in user interaction and providing a scalable solution for handling time-intensive tasks.

5.3 Image Processing and Vectorization

Once picked up from the queue, the image undergoes asynchronous processing. OpenCV is employed to convert the image into a numerical vector, capturing essential features that form the basis for similarity assessments.

5.4 Database Interaction

Interaction with the database is crucial for storing cluster centers, individual images, and user posts. This involves three primary tables: Cluster Center Table, Image Table, and Post

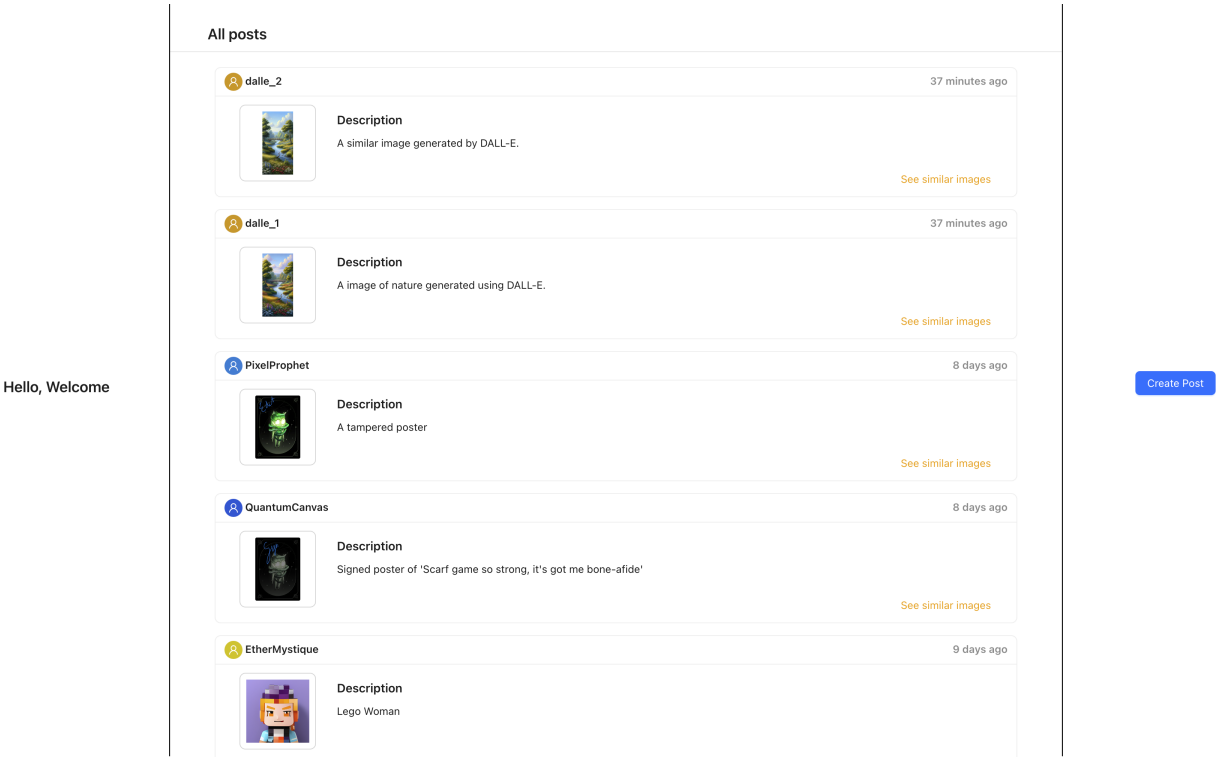

Figure 2: The homepage view

Table. The database serves as a repository for critical information during the image deduplication process.

5.5 Image Similarity Check and Decision Making

The processed image is then compared for similarity with existing cluster centers in the database. A decision-making process follows:

- Threshold Exceeded:
  - If the similarity exceeds a predefined threshold, indicative of dissimilarity to existing clusters, a new cluster is created with the current image as its center.
  - The new image is persisted in the database.
- Threshold Not Exceeded:
  - If the similarity falls below the threshold, the system checks the two nearest clusters to determine duplication presence.
  - **Duplicate Found**: In the presence of a duplicate, the new image is deleted, and the associated post is linked to the existing image.
  - **No Duplicate Found**: If no duplicate is found, the center of the nearest cluster is updated with the new image, ensuring efficient clustering.

5.6 Dynamic Clustering and Storage Optimization

This approach ensures dynamic clustering, adapting to evolving content, and minimizing storage redundancy. The process optimizes storage space by intelligently updating cluster centers and pointing posts to existing images when duplicates are detected, contributing to efficient data management.

This comprehensive approach guarantees a robust image deduplication system, effectively addressing redundancy challenges and safeguarding against IP theft in image-centric platforms.

6 Evaluation (Response to Professor's suggestions)

To rigorously assess the efficacy and efficiency of our system, a comprehensive evaluation was conducted, necessitating the acquisition of relevant image data from platforms targeted by our application. This led to the development of a dedicated web crawler designed to traverse specified websites and

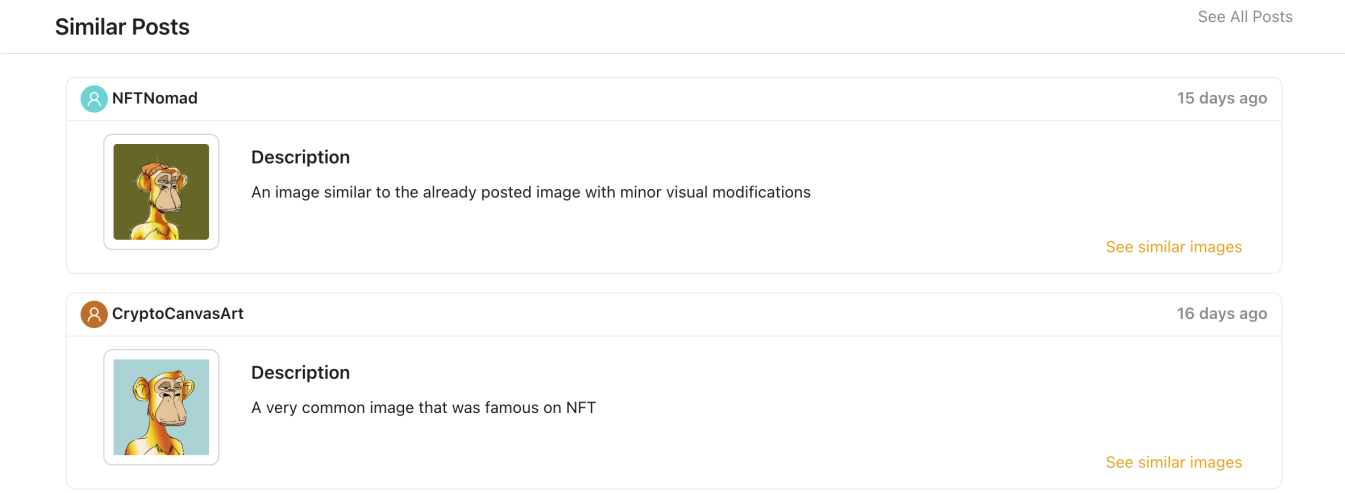

Figure 3: Similar Images clubbed together

procure a dataset reflective of the diverse content within our system's scope.

6.1 Dataset

Our dataset encompasses a diverse collection of over 500 images, meticulously obtained through the assistance of our web crawler. These images were sourced from various platforms, including NFT marketplaces and image-loaded websites.
The dataset intentionally incorporates duplicated images and instances of near-duplication, where minor modifications distinguish otherwise identical images.

6.2 Test

The evaluation procedure involved executing our system's script to systematically upload the acquired images as posts within our platform. A comparative analysis was then undertaken, juxtaposing the performance of our proposed approach against a naive method.

The naive approach, characterized by a simplistic search for image similarity and exact match checks, would entail scanning every image exhaustively on each iteration - a process prone to unnecessary time consumption.

To quantify the effectiveness of our approach, we measured key performance metrics, including processing time and memory consumption. This involved scrutinizing the results obtained through both approaches to discern any notable disparities and advantages offered by our optimized image deduplication methodology.

6.3 Results and Performance Analysis

Our primary objective was to efficiently optimize memory consumption. As outlined in the approach if exact image match found we increment the number of references in the original image rather than storing it again. Although the naive algorithm can also perform this task, its efficiency diminishes as the dataset size increases. Therefore, our system is preferred for its efficiency in handling larger datasets.

Since the naive approach checks each and every image everytime to find the exact match it takes almost equal time initially but as the number of images increases the time to execute for the naive algorithm also increase. Our algorithm however, due to utilization of clustering techniques based on similarity keeps the time lower.

No. of Images	Time Taken in sec	
	Our	**Naive**
10	0.148	0.213
50	0.120	1.070
100	0.163	1.190
150	0.207	1.410
200	0.254	1.630

Table 1: Performance Analysis of Our Approach vs Naive Approach execution time in seconds

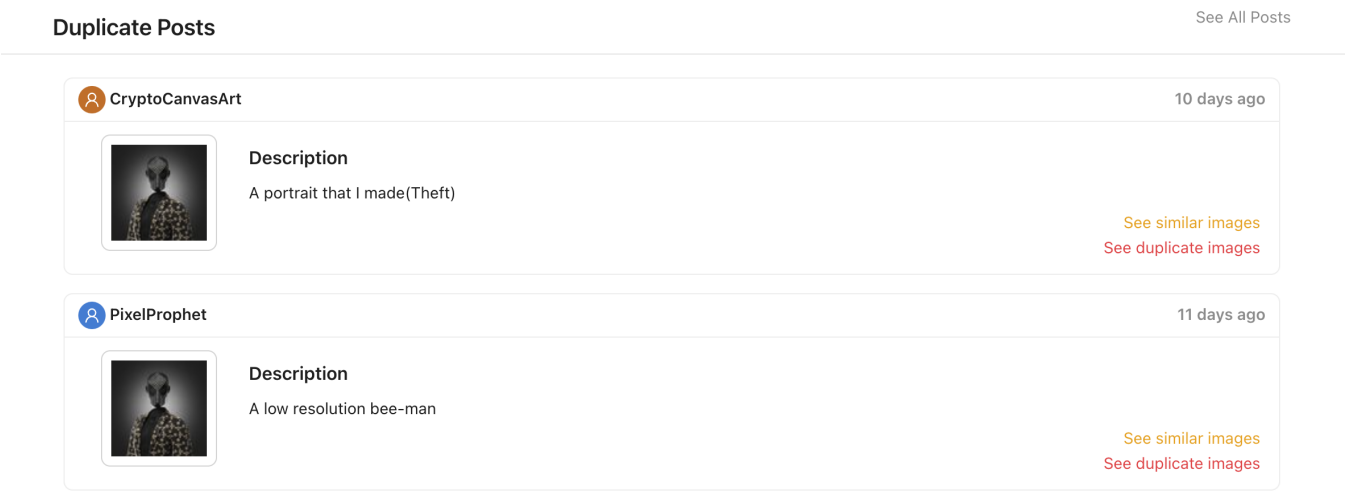

Figure 4: Duplicate Images clubbed together

Table 2 shows our algorithm performance. The correctness of our algorithm can be seen as the images keep going in a cluster the time to search for exact match will increase as expected.

Execution Time in seconds of image going in nth cluster
0.08977
0.08948
0.11196
0.12221
0.11541
0.12743
0.13028
0.15353
0.16966
0.17827

Table 2: Analysis of Execution Time of images that go into the same(any particular cluster but same) in seconds

In Table 3 we compare our approach(where number of clusters are not fixed i.e dynamic) to an approach where we fix the number of clusters(5) and do the same thing. If we fix the number of clusters then definitely the cluster size is going to be very large as compared to that of dynamic clusters. Hence, the time taken there to find exact match increase thereby increasing the overall execution time. Also for a fixed number of clusters as the number of images increases, the performance or effectiveness of the system tends to diminish.

No. of Images	Time Taken in sec	
	Our	**Fixed Number of Cluster Apporach**
10	0.148	0.293
50	0.120	0.807
100	0.163	1.360
150	0.207	1.521
200	0.254	1.814

Table 3: Performance Analysis of Our Approach vs Fixed number of clusters execution time in seconds

Another correctness we can see is that we check the distance of an image with all the current cluster centers present. As the number of clusters increases the time to calculate this distance will increase slightly as seen.
However the time increase is very low compared to the time increase from naive matching if all of them were in the same or smaller number of clusters.

Total Clusters	**Execution time in seconds of image going in Cluster 1**
5	0.031
10	0.072
15	0.119
20	0.157

Table 4: Analysis of Our Approach of an image going into Cluster 1 at any time "t" but the execution time noted for different times as total number of clusters present increases

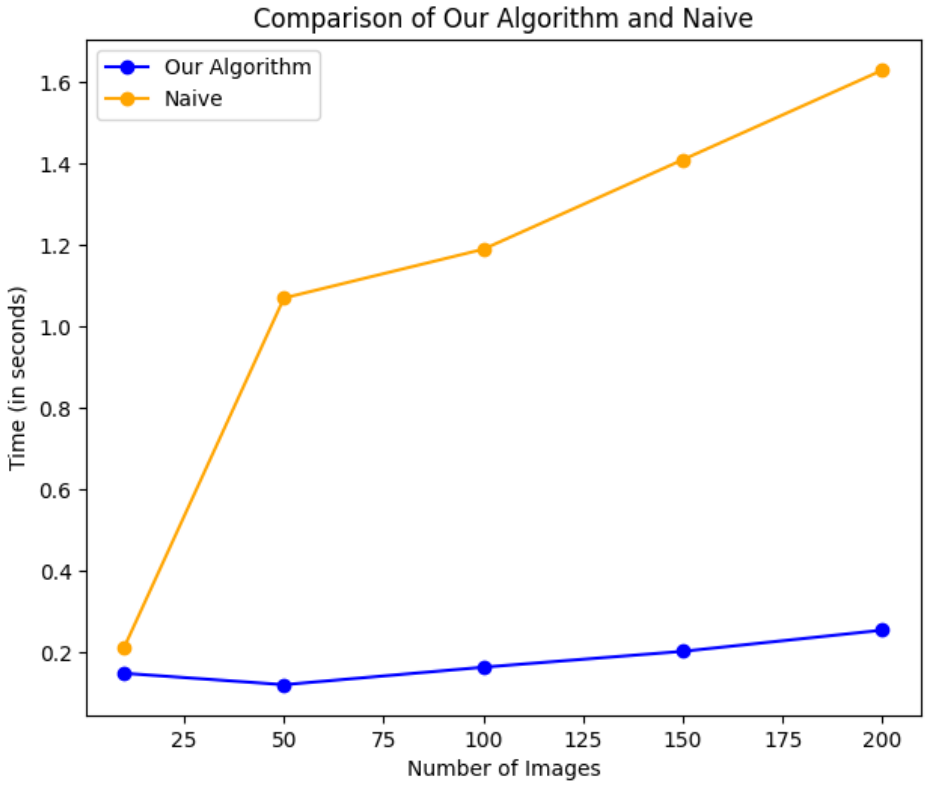

Figure 5: Comparison of our algorithm with the naive way

6.4 Bottleneck

From the results and analysis we noticed that if we find a similar cluster match we have no option but to check the images one by one which will take time as more and more similar images are being grouped together.
This can be mitigated in the future if a database can efficiently query this data from database rather than us linearly comparing. This would also scale the system.

7 Limitation

While our clustering techniques contribute significantly to accurate image deduplication, it is crucial to acknowledge the inherent limitation that perfection in clustering, reaching 100% accuracy, may be an elusive goal. The diversity in image content, variations in user preferences, and the dynamic nature of evolving datasets pose challenges that make achieving absolute precision an intricate task.

The system, while highly effective, operates within the constraints of inherent complexities in content clustering and user-generated metadata, suggesting that a nuanced approach to accuracy is essential in the realm of image deduplication.

The system's performance is sensitive to threshold values used in distance calculations and similarity checks. Striking a balance between sensitivity and specificity is crucial, and future work could explore adaptive thresholding mechanisms.

8 Conclusion

In summary, our evaluation substantiates the efficacy of the proposed image deduplication system. The results underscore its efficiency, and scalability, positioning it as a formidable solution in mitigating redundancy challenges in image-centric platforms. The integration of React.js, Celery, Redis, Django, and OpenCV forms a cohesive and powerful technological stack that seamlessly addresses the dynamic demands of our project. We have listed down a few concluding items.

- **Impact on Intellectual Property (IP) Theft**: Our image deduplication system serves as a robust deterrent to Intellectual Property (IP) theft, a prevalent concern in platforms hosting Non-Fungible Tokens (NFTs) and social media alike. By identifying and preventing the re-

posting of identical or substantially similar images, our system contributes significantly to safeguarding the intellectual rights of content creators. This, in turn, fosters an environment where original creators are rightfully acknowledged and protected from unauthorized use of their creations.

- **Optimizing Storage Space**: Beyond mitigating IP theft, our system plays a pivotal role in optimizing storage space. The redundant storage of duplicate images is a common challenge in image-centric platforms, leading to increased memory consumption. The dynamic clustering and deduplication approach employed in our system ensures that storage resources are utilized judiciously. By minimizing the redundancy of identical images, we not only enhance system performance but also contribute to more sustainable and resource-efficient data management practices.

In conclusion, our project represents a significant stride in the domain of image deduplication, offering a sophisticated, adaptive, and scalable solution. By addressing issues of IP theft and optimizing storage space, we contribute to creating a more secure, ethical, and resource-efficient digital ecosystem. The success of our system highlights the potential for advanced technological solutions to reshape the landscape of content management, providing a glimpse into a future where redundancy is minimized, and digital creativity is protected and celebrated.

9 Future Work

Looking ahead, the success of our image deduplication system lays the groundwork for broader applications across diverse digital platforms. Some of the future works that can be planned are:

- Extend the image deduplication system's compatibility to various digital platforms and formats (like PDFs, videos, etc.). Tailoring the system to integrate seamlessly with emerging technologies and diverse content-sharing platforms will enhance its versatility and impact.
- Explore the integration of deep learning models for image recognition and similarity assessment. Leveraging neural networks or other deep learning architectures may provide more nuanced insights into image content, leading to improved clustering accuracy.
- The current implementation relies on SQLite for managing certain aspects of the system, including handling multiple joins in the database queries. As the project evolves, future work could explore the integration of more robust and scalable database solutions. Transitioning to a database management system with advanced join optimization capabilities could significantly enhance query performance, particularly when dealing with large datasets and intricate relational structures.

10 Acknowledgement

We would like to express our sincere gratitude to Professor Zhenhua Liu for their invaluable guidance and unwavering support throughout the development of this project. Their insightful feedback, constructive critiques, and wealth of knowledge have played a pivotal role in shaping our ideas and refining the methodologies employed.The project has benefited immensely from their expertise, and their encouragement has been instrumental in overcoming challenges and achieving our goals. We are truly thankful for the mentorship provided, which has significantly enriched our learning experience and contributed to the success of this endeavor.

11 Timeline

- Phase 1 (September 11 - September 27)
  - Group Formation.
  - Idea Creation
- Phase 2 (September 28 - October 11)
  - Define project scope, objectives, and deliverables.
  - Set up project repositories on version control (e.g., GitHub).
  - Create an initial project plan, outlining milestones and deadlines.
- Phase 3 (October 12 - October 24)
  - Built a web crawler for downloading images from the internet.
  - Developed a PoC for checking images via hashes.
  - Gave midterm presentation.
- Phase 4 (October 25 - November 1)
  - Got the midterm feedback from professor.
  - Started thinking about a different use case and a different way than hash to compare images.
- Phase 5 (November 2- November 6)
  - Finalized on using IP theft on NFT as our primary use case.
  - Pivoted to using Image to vector for image comparison instead of hashes.

- Phase 6 (November 6 - November 12)
  - Start developing the React.js-based frontend and implement UI components for image upload, post creation, and viewing posts.
  - Researching related work and clustering using image to vector.
  - Researched the feasibility of referring to the original image when a cropped image is uploaded.
- Phase 7 (November 12 - November 19)
  - Implement the RESTful API for creating, reading, updating, and deleting posts and set up SQLite and integrate it with the backend.
  - Develop the image processing logic using OpenCV for vectorization and clustering.
  - Researched the feasibility of referring to the original image when a filtered image is uploaded.
- Phase 8 (November 20 - November 26)
  - Connect the frontend and backend, enabling basic data flow.
  - Integrate Celery and a message queue (e.g., Redis) for asynchronous processing.
  - Implement image deduplication functionality, including the decision-making process.
- Phase 9 (November 28 - December 3)
  - Conducted testing to ensure a seamless user experience.
  - Working on completing things and preparing for final presentation.
  - Started creating the report.
- Phase 10 (December 4- December 11)
  - Continued and finalized the work on the report.
  - Gave the final presentation and demo.
  - Included responses to professor's suggestions in the report (Fetched number and made graph for performance metrics).

12 Contributions

- Pandre Vamshi *(115077003)*
  - Researched feasibility to refer to the original image when a cropped image is uploaded.
  - Researched feasibility to refer to the original image when a filtered image is uploaded.
  - Connected Frontend with the Backend.
- Prajjawal Agarwal *(115134849)*
  - Designed and developed frontend.
  - Created REST APIs for post creation and fetching filtered posts in Django.
  - Researched and utilized Celery and Redis for Asynchronous processing.
- Shreejay V. Jahagirdar *(115071683)*
  - Researched means for clustering, studied image to vector & feature extraction.
  - Built the processor to process images in the pipeline and perform necessary clustering and table actions.
  - Developed a Webcrawler to create image dataset and tested it to find a threshold.

13 Links

- Code Links
  - *NFT UI*: [https://github.com/prajjawal05/nft-ui]
  - *Backend API and Processing*: [https://github.com/prajjawal05/picnix]
- *Demo Link*: [https://drive.google.com/file/d/1bKgn8XZgWDR3B0O-RzF2SuK2z9QnNokO/view?usp=sharing]

References

[1] Layek, A. K., Gupta, A., Ghosh, S., and Mandal, S. Fast near-duplicate detection from image streams on online social media during disaster events. In *2016 IEEE Annual India Conference (INDICON)* (2016), pp. 1–6.