The Universal NFT Vector Database: A Scalable Vector Database for NFT Similarity Matching

Samrat Sahoo[1*], **Nitin Paul**[1], **Agam Shah**[1], **Andrew Hornback**[1], **and Sudheer Chava**[1]

[1]Georgia Institute of Technology, Atlanta, Georgia, United States of America
[*]samratsahoo@gatech.edu

ABSTRACT

Non-fungible tokens (NFTs) represent ownership of a particular digital item. Because NFTs are cryptographically unique, their value is not interchangeable. However, the proliferation of new blockchains and increased value in digital data has resulted in entities creating vast amounts of NFTs. Because there is no means to track the similarity across NFTs, this enables the creation of nearly identical NFTs with minimal alterations. To address this issue, we developed a modular, hardware-agnostic, and easily extendable system for processing NFTs by representing them as vectors. We query for NFTs that follow the Ethereum Request for Comment 721 (ERC-721) token standards and construct a vector database from the aggregated vectors from NFTs on the Ethereum blockchain. Additionally, we crafted an NFT visualization dashboard with a user-friendly interface, granting non-technical users access to aggregated NFT data. Our Universal NFT Vector Database establishes a framework for efficiently organizing and storing NFT data based on similarity, potentially aiding blockchain researchers and industry participants in verifying NFT authenticity.

Introduction

NFTs are digital assets on the blockchain that are intertwined with cryptography, containing metadata and a distinct hash for identification[1]. Designed to be cryptographically unique, NFTs originated from the Bitcoin blockchain in 2012 and have become widespread on the Ethereum blockchain[2]. Typically represented with the ERC-721 standard[3], NFTs are ideal for use cases involving collectible items, digital key management, concert tickets, and more because they enable originality between digital tokens. To enforce authenticity between NFTs and improve fairness in NFT ecosystem, especially among image-based NFTs, researchers and blockchain developers are searching for mechanisms to detect and enforce the cryptographic uniqueness enabling blockchain systems and infrastructure to fulfill the purpose of the existence of NFTs[4].

Currently, it is expensive to store raw file data on a blockchain. As a result, NFTs on the blockchain store a reference (i.e., a URL) to the file data, and off-chain storage solutions host the metadata. The NFT is assigned a unique token ID; while the token ID is unique, the off-chain data may not be. With recent blockchain infrastructure development, there is a lack of robust protocols and standards to enforce the authenticity of an NFT. The lack of enforcement technology impedes new digital innovations which rely on NFTs to be unique and not be replicated by another entity, effectively nullifying the original intention of NFT technology.

NFT marketplaces, a prominent one being OpenSea, exemplify the problem our system provides a solution for. Users can create the same image as an NFT across various collections. When users mint NFTs, these marketplaces have no in-built mechanism for detecting counterfeit NFTs and enforcing uniqueness. As a result of this missing mechanism, it is difficult to derive any particular value from the NFT. Creators instead rely heavily on marketing their NFT token ID and contracts through social media or other platforms instead of the actual content of their NFTs.

To address the challenges with NFTs, we propose a modular, easily-extendable, hardware-agnostic, cloud-centered NFT database. Such infrastructure helps enable and facilitate the enforcement of token uniqueness. It standardizes the data to a vector data repository which users can easily access through a web interface for research and utility purposes. The vector database determines similarities between NFT feature vectors to conclude the uniqueness of an NFT.

First, to develop the NFT vector database, we extract the ERC-721 contracts using the Graph Protocol. After receiving the contract information, we collect the individual NFT data through Alchemy (a blockchain infrastructure provider). From the media URL associated with the NFT data, we aggregate the images, embed them, and place the vectors (each containing a distinct ID) in Pinecone, a vector database provider.

Next, we connect the database to an interactive GUI to derive updated metrics and critical information regarding NFT contract data. The web application enables researchers and consumers to interact with the vector database and have a direct access point to the NFT data. Furthermore, this infrastructure for NFT data could be generalized for various applications surrounding the necessity of unique digital assets, such as identity authentication and NFT filtration systems[5].

The main contribution of our work is the following:

- Present a framework for enforcing the uniqueness of NFTs to extract more value from these tokens to be used by research and consumers.
- Produce a web application interfacing with the NFT vector database with a user-friendly graphical user interface to improve access to NFT data and advance future innovations involving NFTs.
- Illustrate the system's extensibility and modular design, enabling developers to replace parts of the system instead of developing new infrastructure for each application area.

Related Work

NFT authenticity has been analyzed in various ways. From a trading perspective, von Wachter et al. investigated NFT markets from January 2018 to November 2021 to look for suspicious trading patterns. Using directed multigraphs, a popular data structure in network analysis, the researchers modeled the transaction history of 3,572,483 NFTs that totaled 21,310,982 transactions stemming from 459,954 addresses. In total, 3.93% of the addressess analyzed were flagged as suspicious, indicating that by using addresses as nodes and transactions as edges, self-directed transactions and rapid sequence transactions that avoid market risk were noticeable in the constructed graphs[6].

Incentives for illicit transactions can vary, but typically involve wash trading, where someone puts a buy and sell order in simultaneously to create an illusion of high demand. Wash trading for market participants with ample capital, commonly referred to as whales, can stem for the desire to increase ranking on OpenSea and other platforms[7]. One particular example involved a trade that occurred on the Ethereum blockchain for over half-a-billion US dollars. The trade involved a user transferring an NFT to another address, which then sold the NFT for 124,457 Ether, equivalent to $532 million, that was borrowed from three sources. Following the sell, the seller who received the money returned it to the buyer, who repaid the loan to the three sources and then offered the NFT for sale for 250,000 Ether, more than $1 billion USD[8].

Research around other types of fraud, such as image manipulation that our application seeks to address, has also been conducted. NFT authenticity has been analyzed in various ways.[9] identified three types of counterfeit categories: similar collection names, identical image URLs, similar images.

Similar collection names involve using an inherently deceptive name, given its closeness to a legitimate collection, by changing the case of a letter or adding passive punctuation, such as a period to the end of the collection name. Identical image URLs involve scammers using the image URL of a legitimate collection and minting tokens pointing to legitimate URLs. Similar images involve copying the digital asset and minting a modified version. While marketplaces such as OpenSea have restrictions to combat similar collection names, there are currently no checks available for the similar image issue.

Their research analyzed image similarity by examining the image pointed to by the image URL value in each NFT. 9,991,013 images were downloaded and analyzed for similarity using an image hashing tool, and their research found 59,425 hash collision pairs. Images from the same collections were not compared, as similarity would be possible without necessitating counterfeits. To complement support for their automated analysis, the group randomly picked 100 of the resulting similar NFT pairs found and performed a manual analysis that confirmed 90% were identical[9].

Hive, an Enterprise AI company listed among Fast Company's ten most innovative artificial intelligence companies of 2020[10], has developed an NFT Search API to address duplicate NFT creation issues[11]. Hive's models use a deep vision image similarity model optimized for digital art. Hive designed the models to target exact visual matches while simultaneously being resilient to manual manipulations and other mechanisms that could potentially bypass hashing checks[11].

The wealth of NFT (non-fungible token) data created and collected daily provides opportunities for applications in finance and opportunities for individuals and businesses to leverage the scarcity and functionality of NFTs to develop solutions for billions of people worldwide. Although the current research on the field is still highly evolving, the inherent properties of NFTs encompass a similar set of issues that academia and industry are investigating. To create a transparent marketplace where arm's length transactions are absent of deceit and fraud as much as possible, developers and researchers must address the challenges mentioned by those such as[9] with technology being developed by companies such as Hive.

Methods

Blockchain Data Querying and Collection

Ethereum blockchain data is challenging to query due to the blockchain's inherent "linked-list" structure. Traditionally, to query the blockchain for all ERC-721 contracts and tokens, it would be necessary to spin up an Ethereum RPC node and iterate through every block. Our system, instead, leverages the Graph Protocol, a decentralized querying layer built on top of Ethereum[12], allowing for easy and fast querying of subsets of blockchain data (called subgraphs) via a GraphQL API. We

Table 1. Benchmarks for Alchemy and the Graph Protocol for NFT querying speed. Values are average over 20 trials.

Number of NFTs	Alchemy Performance	Graph Protocol Performance	Graph Protocol to Alchemy Speed Ratio
136	1.31 Seconds	10.01 Seconds	7.638 : 1
2139	23.45 Seconds	1201.89 Seconds	51.259 :1
14213	97.71 Seconds	8768.16 Seconds	89.732: 1

utilize an EIP-721 subgraph[13] that monitors all ERC-721 contracts and collects contract addresses through this subgraph. We collect these contract addresses in a paginated manner – processing anywhere from 10 to 100 contracts at a time – to ensure the subgraph infrastructure or our infrastructure is not overwhelmed with excessive data input and output operations.

We test two methods to obtain the individual NFT data for each ERC-721 contract. The first option was to utilize the subgraph used for the contracts to acquire the NFTs and individual details of each NFT. The second option was integrating the system with Alchemy's – a Web3 infrastructure provider – NFT infrastructure. Since speed was a priority for us in retrieving as many NFTs as possible, we ran benchmark tests to measure which method would run faster. We design the benchmark tests to retrieve all NFT media URLs from a single NFT collection and repeat this process 20 times. We then take the average speeds over the length of the 20 trials for the Alchemy NFT infrastructure and Graph Protocol and use them as a metric for evaluation. We also repeat this experiment for three NFT collections of different sizes (less than 1000 NFTs, 1000 to 10000 NFTs, and 10000 or more NFTs) to test the scalability of each infrastructure. We discover that Alchemy's NFT infrastructure has significant time improvements and scales roughly linearly according to the number of NFTs. The linear scaling is because Alchemy caches NFT media information in a database, allowing almost instant retrieval.

Conversely, the Graph Protocol method is significantly slower and suffers scalability issues. The reason for this is that while the Graph Protocol can retrieve all of the NFTs almost instantaneously via a GraphQL API call, this particular subgraph does not retrieve the NFT media information but rather the metadata information. Only by processing the metadata further can the Graph Protocol method retrieve the NFT media information, which therefore causes it to take more time. The increasing Graph Protocol to Alchemy speed ratio demonstrates the issues the Graph Protocol method has with scalability when increasing the number of processed NFTs. Table 1 details the results from the Alchemy vs Graph Protocol benchmarks. The dedicated NFT infrastructure of Alchemy, as opposed to the Graph Protocol, yielded significant net benefits of faster and scalable NFT data querying.

Data Processing

By utilizing Alchemy's NFT Infrastructure, we obtain the media URL of an individual NFT. We feed this media URL through a data processing system that obtains the binary information of the media at the URL. Our system conducts no preprocessing steps on the media contents to retain the inherent attributes of the NFT. The image is then directly converted into a 2016x1 dimensional vector via a RegNetY-080 embedding[14]. Once we obtain the embedding, a unique hex-based Universal Unique Identifier (UUID) v4 is generated for the vector – allowing for the vector to be queried individually by ID. The system divides the data into two database pipelines. The first pipeline stores the generated vector in a vector database provider called Pinecone. In addition to the generated vector, the vector database stores critical information about the NFT in the metadata, including the NFT collection address, NFT token ID, and NFT media information. The second pipeline stores additional information about the NFT, such as chain information, the NFT metadata URL, and NFT standard information on a MongoDB database. The dual database system enables users to obtain NFT details (from the MongoDB database) based on visual aspects of NFTs (using the vector database). Our design differs from traditional methods, which rely solely on querying based on the NFT collection contract ID and token ID.

Vector Operations and Dimensionality Reduction

Raw images have large, arbitrary dimensions when converted to a vector, posing two issues our system attempts to solve. First, large dimensions are intensive storage-wise, with even smaller images requiring tens of thousands of elements per vector. By utilizing image embeddings, we can apply dimensionality reduction to the image while retaining the inherent attributes of the NFT. The RegNetY-080 represents the image as a 2016 dimensional vector, significantly reducing the amount of space each image takes up in the database.

The second problem of arbitrary vector dimensions poses an issue for vector comparison. Current metrics for vector comparison, such as euclidean distance, require consistently sized vectors[15]. However, with different-sized images, it is impossible to compare the images without some standardization. After standardizing the vectors, we were able to add functionality to search for similar NFTs – we chose to use cosine distance as our search metric as it performed the best in our experiments outlined below.

To determine which search and embedding techniques were the most effective, we performed experiments with five state-of-the-art image embeddings in conjunction with three different distance metrics. We create a custom NFT dataset with three different parts for the experiments. Figure 1 illustrates the custom-created dataset in a visual manner:

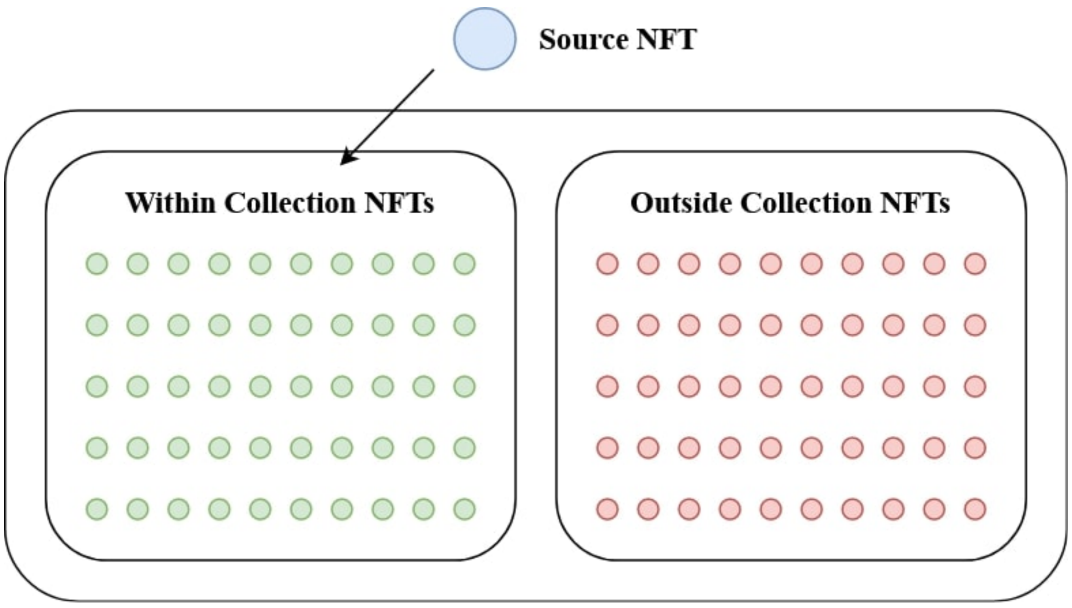
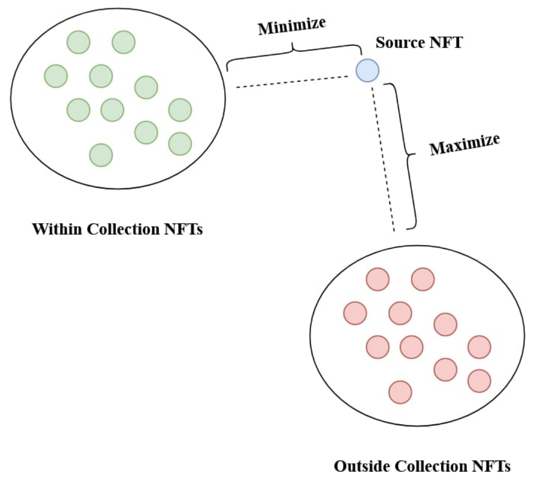


Figure 1. Custom created dataset. We designate one source NFT and 50 within collection NFTs and 50 outside colection NFTs. Within collection NFTs indicate NFTs as part of the same collection as the source NFT. Outside collection NFTs indicate NFTs as part of a different collection as the source NFT.

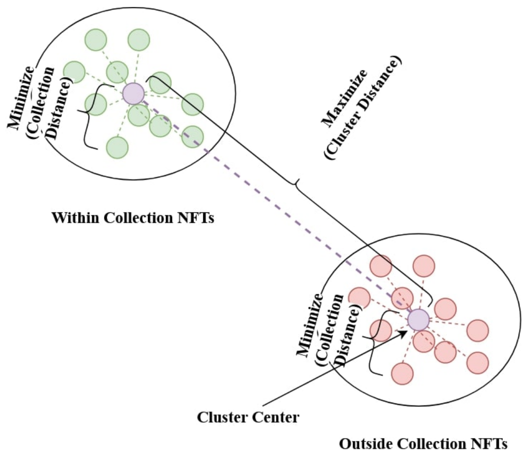
- **Source NFT:** The source NFT is an NFT used as a basis for comparing all other NFTs in the dataset.
- **Within Collection NFTs:** Within collection NFTs are NFTs that share the same general aesthetics as the source NFT but are still distinct from the source. This entails looking at key attributes the NFTs shared with the source NFT and seeing if they are aesthetically similar. As the name suggests, within collection NFTs are found in the same collection as the source NFT.
- **Outside Collection NFTs:** Outside collection NFTs would consist of NFTs that did not share any similar attributes to the source NFT. All outside collection NFTs however, do share similar characteristics. As the name suggests, outside collection NFTs are not found in the same collection as the source NFT.

We take every image embedding and distance metric combination for the experiment and compare distances between a source NFT and 50 within collection NFTs, and 50 outside collection NFTs. This experiment aimed to find the embedding and distance metric combination that would minimize the average distance between the source NFT and the within collection NFTs while maximizing the average distance between the source NFT and outside collection NFTs. Since the distance measurements do not measure on the same scale between different distance-embedding combinations, we took the ratio between the average distance of outside collection NFTs and the average distance of within collection NFTs to determine the best distance-embedding metric combination. The dot product ratios are reciprocals because a higher dot product indicates a greater similarity. Figure 2a illustrates the objective of the experiment in a visual manner. Our experiment results are detailed in Tables 2 and 3.

For the visualization capabilities, we need an effective dimensionality reduction technique to reduce the embedding vectors into a two-dimensional surface. To determine which dimensionality reduction technique is most effective, we run experiments to compare different methods including multidimensional scaling, t-distributed stochastic neighbor embedding, principal component analysis, truncated singular value decomposition, and isometric mapping. For this experiment, we define and calculate a metric called the cluster ratio to determine which low-dimensionality reduction technique is most effective – a



(a) Visualization of Distance Embedding Experiment. The goal here is to designate a source NFT and minimize the distance between the source NFT and the within collection NFTs and maximize the distance between the source NFT and the outside collection NFTs


Minimize (Collection Distance)
Maximize (Cluster Distance)
Within Collection NFTs
Minimize (Collection Distance)
Cluster Center
Outside Collection NFTs

(b) Visualization of Dimensionality Reduction Experiment. The goal here is to minimize the distance between the individual collection NFTs and the center of their respective cluster while maximizing the distance between the centers of both collections

Figure 2. Visualizations of embedding and distance metric experiments.

Table 2. Raw Data of Average Distances

	Average Euclidean Distance (Within Collection)	Average Cosine Distance (Within Collection)	Average Dot Product (Within Collection)	Average Euclidean Distance (Outside Collection)	Average Cosine Distance (Outside Collection)	Average Dot Product (Outside Collection)
EfficientNet-B5	11.417	0.466	73.979	13.918	1.240	27.507
ViT Large	66.644	0.305	5088.192	90.932	*0.859*	*3338.553*
RestNet-50	5.187	0.243	41.152	6.783	0.724	22.859
RegNetY-080	17.205	0.134	928.943	24.012	0.401	783.930
ConvNeXT Large	52.831	0.168	7205.935	61.579	0.397	6533.807

Table 3. Ratios Between All Combinations of Embeddings and Metrics

	Euclidean Distance	Cosine Distance	Dot Product
EfficientNet-B5	1.219	2.661	2.689
ViT Large	1.364	2.816	1.524
RestNet-50	1.307	2.979	1.800
RegNetY-080	1.396	2.992	1.185
ConvNeXT Large	1.166	2.363	1.103

Table 4. Cluster and Collection Distance Metrics

	Multidimensional Scaling	T-Distributed Stochastic Neighbor Embedding	Principal Component Analysis	Truncated Singular Value Decomposition	Isometric Mapping
Cluster Distance	19.376	63.543	14.613	14.616	77.584
Collection Distance	8.828	15.784	3.306	2.206	13.287
Cluster Ratio	2.195	4.026	4.420	6.626	5.839

higher cluster ratio indicates a more effective dimensionality reduction technique. We define effectiveness as the algorithm that maximizes the distance between NFT collections clusters centers while minimizing the distance between the individual NFTs and the NFT collection cluster centers.

We define cluster ratio, cluster distance, and collection distance as follows:

- **Cluster Distance:** This is the Euclidean distance between the centers of two clusters.
- **Collection Distance:** This is the average Euclidean distance between the individual elements of each cluster and the center of the cluster.
- **Cluster Ratio:** This is the ratio of the distance between two cluster centers and the average distance between the individual elements of each cluster and the respective cluster center of the elements. The cluster ratio is better defined as the ratio of cluster distance to collection distance.

This experiment leverages the NFT dataset created from the distance and embedding experiment. The initial step to determining the best algorithm was to create embeddings for all of the images in the dataset. We then apply the dimensionality reduction technique we are testing and create two clusters on the lower dimensional data using a K-means clustering algorithm. We use the K-means results to determine these clusters' centers. To calculate the collection distance, we take each lower dimensional NFT, calculate the euclidean difference between the NFT and the center, and average all the distances together. To calculate the cluster distance, we took the euclidean distance between the two cluster centers. The cluster ratio was obtained by dividing the cluster distance by the collection distance. Figure 2b illustrates the objective of the experiment in a visual manner. Our results are detailed in Table 4. The truncated singular value decomposition had the higher cluster ratio and was the chosen dimensionality reduction technique for the visualization capabilities.

Software Infrastructure

Control Panel, Dashboard, and Backend Server

Our GUI grants nontechnical users complete access to the system's capabilities. To create this GUI, we leveraged a standard software engineering model for building applications called the Model-View-Controller (MVC) pattern to ensure organized, secure, and easy access to data. We were able to utilize MVC architecture by splitting our application into three main components:

- **Model:** The model portion of the application primarily entails the database schemas, which enable a better organization of data – we write and enforce using the Mongoose client for MongoDB when inserting data into our database. Additionally, using a strongly-typed language like TypeScript allows us to declare and enforce types strictly.
- **View:** The view portion of the application enables users to view current data and perform create, read, update, and delete (CRUD) operations, all abstractedly. In the case of the GUI, users leverage CRUD operations through actions such as viewing analytics (such as the total number of NFTs in the database, NFT and NFT collection processing success rate), controlling the task queue (adding contracts into the queue to be processed), and updating analytics. We built this portion of the system entirely with the Vue.js frontend framework.
- **Controller:** The system's controller is a backend server that manages all CRUD operations' logic. The CRUD operations include actions like retrieving analytics and updating the task queue. Additionally, the controller enables greater security in the application's admin panel, where users are authenticated via JavaScript Object Notation (JSON) web tokens, allowing them to control the task queue. The controller was written in TypeScript using a full-stack web framework known as Nuxt.js, enabling client-side and server-side under a mono repository structure, greatly simplifying the organization and infrastructure of the project.

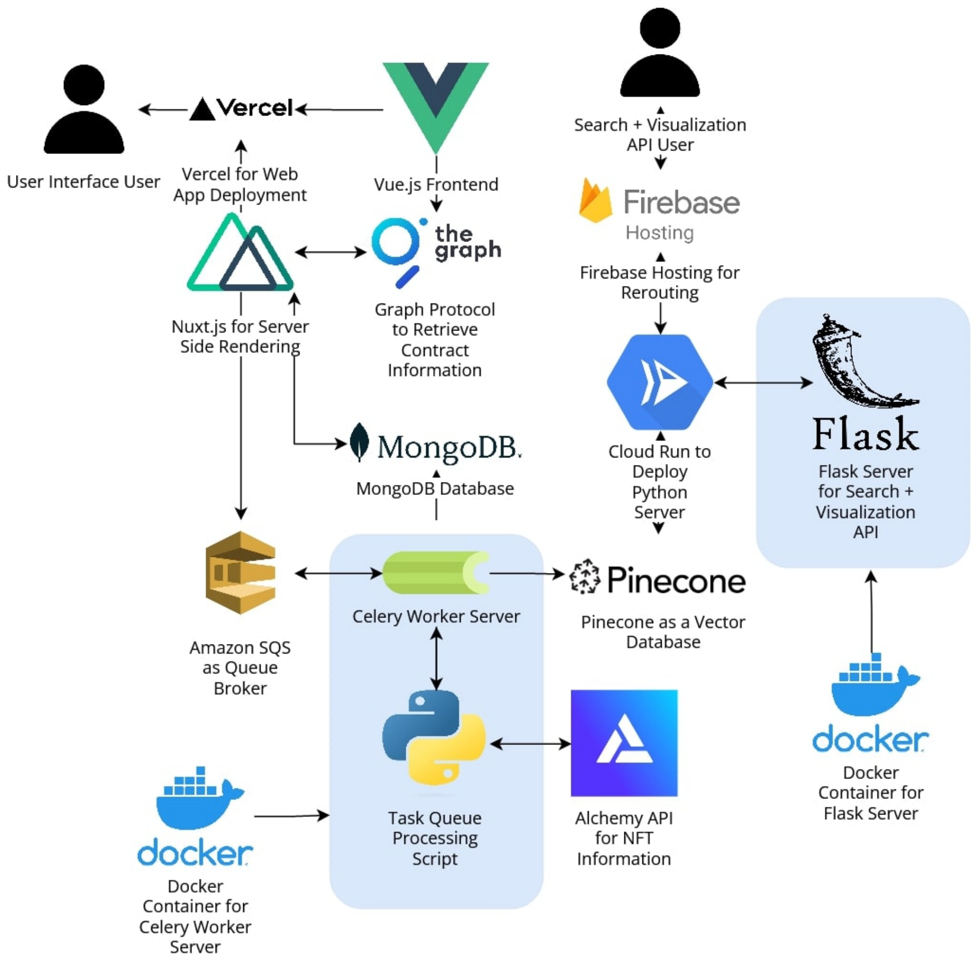

Vercel
User Interface User
Vercel for Web App Deployment
Vue.js Frontend
the graph
Nuxt.js for Server Side Rendering
Graph Protocol to Retrieve Contract Information
MongoDB.
MongoDB Database
Amazon SQS as Queue Broker
Celery Worker Server
Pinecone
Pinecone as a Vector Database
Task Queue Processing Script
Alchemy API for NFT Information
docker
Docker Container for Celery Worker Server
Search + Visualization API User
Firebase Hosting
Firebase Hosting for Rerouting
Cloud Run to Deploy Python Server
Flask
Flask Server for Search + Visualization API
docker
Docker Container for Flask Server

Figure 3. This figure illustrates the infrastructure of the system we have designed. The upper left illustrates our client and server-side logic and blockchain data collection tools. The bottom represents the task queue processing system which processes NFT collections and NFTs. The upper right shows the infrastructure for our search and visualization APIs.

Database Design

As aforementioned, we utilize the Mongoose library and TypeScript to enforce strict types of data inserted. Our three primary database collections include contracts, NFTs, and task queue items.

- **Contract Collection:** The contracts collection indicates a successfully processed contract where the system has added all NFTs of that contract to the task queue.

- **NFT Collection:** Successfully processed NFTs in the vector database are included in The NFTs collection – at that point, that NFT has been completely processed and is now ready to be used for analysis.

- **Task Queue Items Collection:** The task queue items collection includes items currently in the queue to be processed. The data field in the task queue item includes a contract or NFT that the system is waiting to process.

- **Miscellaneous Collections:** In addition to those collections, we also include a collection for analytics which contains one document of the system's comprehensive analytics – the system updates this document each time a contract or NFT gets processed. Finally, our database includes a checkpoints collection which also hosts one document to keep track of the last contract that the system processed – this contract is stored and updated when processing the next group of contracts in a paginated manner.

Task Queue, Workers, and Horizontal Scalability

Converting an image into a vector using image embeddings is significantly computationally intensive. As a result, loading this onto an API would cause an overload of requests, requiring vertical scalability (increasing computational resources) and resulting in a less robust image processing system. To solve this, we implement a task queue system in which NFT data can be saved and processed on a separate server, allowing horizontal scalability. The flow of this system starts at the admin dashboard, where authenticated users can add NFT collections to the task queue. When the admin adds these collections, the system stores supplementary information regarding the processed item along with an ID and status in a MongoDB task queue collection. The system sends the ID to a messaging queue. We use Amazon Web Services (AWS) Simple Queue Service (SQS) as the messaging queue. These items are then individually processed on a separate server running a Celery worker service with two different tasks:

- **Contract Task:** The first task is a contract task where the workers will get all the individual NFTs for a certain contract using Alchemy's NFT Infrastructure and add task items into the task queue for each NFT.
- **NFT Task:** NFT Task: The second task is an NFT task where the workers will get the data at the media URLs and convert this to a 2016 dimensional vector via the RegNetY-080 embedding. Taking this approach to processing contracts and NFTs means we no longer need to scale our infrastructure's hardware to process an increasingly greater number of NFTs. Instead, we can store the different items that need to be processed, and depending on the hardware of our infrastructure, we can spin up as many or as few worker processes as required.

This architecture allows the system to be agnostic to a machine's physical hardware and focuses on reusability across devices.

Search and Visualization API

We offer a public search and visualization API to enable users to interact with the vector database. Both APIs also have interactive user interfaces in our web application.

- **Search API:** The search API will take any base64 encoded image and find the closest NFTs to the image by taking the cosine distance between the source image vector – derived from the RegNetY-080 image embedding – and the vectors in the database. We utilize Pinecone's vector database search API to search for the appropriate vectors.
- **Visualization API:** The goal of the visualization API is to apply dimensionality reduction on a set of vectors via a t-distributed stochastic neighbor embedding. The visualization API will take in a list of vectors and reduce them to two dimensions. Each vector in the newly produced vectors corresponds to a point on a cartesian coordinate system.

System Deployment

Deploying with a combination of cloud and on-premise servers enables us to scale and process more NFTs per second on dedicated, robust hardware. When deploying the system, the objective was to keep costs low and the processing efficiency high. We had to deploy five systems: the vector database, MongoDB database, task queue, public APIs, and web application. Figure 3 details the deployment infrastructure.

- **Vector Database:** For the vector database, we decided to leverage a managed vector database service known as Pinecone.io. In doing so, we greatly simplify the architecture by avoiding manually configuring scaling, storage, sharding, and other infrastructure details. Additionally, Pinecone enables many automatic vector operations, such as searching and filtering, allowing us to build out many public APIs quickly.
- **Visualization API:** For the MongoDB database, we use a serverless MongoDB server that we host on MongoDB Atlas. Because our application relies on storing large amounts of data, the serverless option assisted in keeping the system cost-efficient by only paying for the number of database operations while having access to high storage infrastructure.
- **Task Queue:** For the task queue processing system, we deploy two different systems. The first system was the task queue, which stored the other task IDs. For this, we utilized Amazon Simple Queue Service, which would act as the broker for our Celery worker server. The second is the Celery worker server which is set up as a system service in a Docker container and deployed on an on-premise server where it runs 24/7 and is always waiting to process task queue items. Because of the necessity to run this server constantly, using an on-premise server cuts costs significantly while giving us access to powerful computational machines to process NFTs faster.
- **Public APIs:** For the public APIs, we dockerize and deploy them to Google Cloud Run as serverless functions. We use Firebase hosting to direct HTTP requests to trigger functions in our containerized application. Following a serverless architecture, the container horizontally scales up or down depending on API traffic while ensuring that the costs match the usage.

- **Web Application:** To deploy the web application, we utilize a full-stack deployment provider known as Vercel. Vercel integrates with the NuxtJS framework, allowing for simplified deployments. Using Vercel, we deploy both our client-side and server-side logic, allowing for automatic scaling with no orchestration required – cutting down on server and client infrastructure.

System Constraints and Limitations

- **Cross Chain and Cross Standard Support:** NFTs come in various standards (i.e., ERC-721 or ERC-1155) across many chains (i.e. Ethereum or Polygon). However, supporting these comes with its inherent challenges. To support other chains and standards, we must extend our data querying methods by including subgraphs that index other chains and contracts.

- **Invalid NFTs:** Because off-chain centralized and decentralized storage solutions store NFT metadata, NFTs are not guaranteed to be accessible. Centralized storage providers may remove NFTs, or decentralized storage systems may unpin an NFT, permanently causing our system to lose access to the NFT. Furthermore, our system handles only image-based NFTs; however, our system's modularity allows for a straightforward extension of processing non-image-based NFTs by replacing the embedding type.

Applications and Future Work

Infrastructure Interoperability with Other Blockchain Data

While our primary use case is with vectorizing image-based NFTs, our system can cater to any form of data on the blockchain that can be represented as feature vectors for further analysis:

- **Time Series Data:** Time series data refers to data collected through a sequence of time intervals. This type of data is especially prevalent in financial systems, including cryptocurrencies. Traditional financial systems use vector-embedded time series information in conjunction with machine learning systems like support vector machines and artificial neural networks to create financial forecasts[16]. Our infrastructure interoperability enables blockchain analysts to organize and collect vectorized time series data. Some examples of blockchain-based time series datasets may include price information of a certain NFT collection or the trading volume of a cryptocurrency. Developers or analysts can use the data collected in the vector database with big data operations to create robust models for cryptocurrency forecasts.

- **Natural Language Data:** With the increased usage of blockchains as a backing for interoperable applications, human-language-based data is becoming increasingly prevalent on public blockchains. For example, the Lens Protocol, a social media protocol built on the Polygon network, stores profile data (i.e., followers and comments) as an NFT[17]. Apps built on such protocols can leverage this infrastructure to vectorize text-based data using language embeddings like Word2Vec and create powerful models to accomplish tasks such as content recommendation, comment filtration, and more.

Applications and Analyses with the Database

- **Filtering Unwanted Content:** With the surge in the development of decentralized applications (DApps), there will be an increased interaction with NFTs. Most decentralized apps that deal with NFTs will incorporate some functionality attributed specifically to NFTs, such as minting the tokens and visualizing them as an entity. Many platforms will need a mechanism to filter NFTs to adhere to specific criteria set by developers – such as filtering counterfeit NFTs or illicit content. This database empowers developers to interact with NFTs at the token and contract level and through the media that is representative of the NFT. Whenever a use case necessitates filtering the unwanted NFTs that are unwarranted and not desired to exist as part of the application, this NFT vector database architecture would serve well on that front. Developers can use the NFT Vector database as a basis for comparison where if the queried NFT is similar to flagged NFTs in the database, the DApp can filter the NFT from the user interface. The database would ensure that an NFT that appears on a DApp adheres to all filtering criteria set by the developers.

- **NFT Recommendations:** The NFT vector database could prove beneficial for NFT recommendations. To do this, a DApp could analyze a user's current NFTs and previously browsed NFTs. It could then calculate a score and recommend similar NFTs that users may be interested in purchasing.

- **Fraudulent Behavior:** Fraudulent behavior is related to blockchain-based systems. Current approaches to detecting such behavior entail keeping a separate manually maintained database with information about counterfeit contract addresses and NFTs. Our system would be extremely helpful in helping cut down the maintenance of such a database. This is because the database allows for fast lookups of the similarity scores representing the NFTs within the database. The enforcement of the uniqueness of the NFTs as part of the database directly combats the fraudulent behavior that people do in selling counterfeit NFTs (selling NFTs that are trying to replicate the original NFT to inflate their value) to make as much profit as possible.

- **NFT Price Trends:** Deriving price trends on NFTs based on their aesthetic attributes isn't currently clear-cut and standardized. However, with the power of the NFT vector database and infrastructure, getting price trends and correlating them with visual features and attributes of the NFTs becomes easier. This is because the data stored in the database contains visual information regarding an NFT. NFTs that perform well can be used as a basis for comparison with similar NFTs, which may also potentially perform well.
- **Integration with Blockchain-Based Operating Systems:** With new developments in operating systems and creating a crypto-native environment[18], NFTs have started playing a more significant role. For example, the new ethOS (the world's first operating system based around the Ethereum blockchain) has built-in capabilities to deal with smart contracts and NFTs. The native NFT minting feature included as part of the operating system can be better optimized with the application and the NFT vector database. With people taking pictures through the ethOS camera and looking to mint the images accordingly as NFTs, we can empower smartphone photographers to take unique photos while exploring.
- **Identity Verification:** NFTs were created with the intention of verifying digital ownership. Extending upon this use case, companies can apply NFTs for identity verification. When looking for straightforward proof of the employee's identity and managing their access concerning their identity role, NFTs can be used to digitally represent a person's identity and the kind of access they have within the company. The NFT vector database would help ensure that duplicate NFTs will not be allowed to exist. If such an NFT exists, the company can reissue a new NFT.

Conclusion

In this study, we designed and developed a framework for enforcing the uniqueness of NFTs with a system with an NFT vector database as its central component to improve the value that users can extract inherently from NFTs. We developed an NFT vector database by extracting contract information and then querying the individual NFT data to find the media URL from which we aggregated the NFT images and placed them accordingly as vectors in the database. We also improved access to NFT data and associated metrics by allowing easy access through an intuitive web application interface. Our data access and standardized framework solution for NFT uniqueness based on similarity helps make NFT ecosystem fairer for blockchain developers, consumers, and researchers and leverage its potential for future innovations and research. We demonstrated the generalizability of the NFT architectural design across various potential applications. The value of our work is that it provides a solution for NFTs to be unique and irreplaceable so that researchers and consumers can unlock the potential of NFTs for future innovations and research.

Data Availability

All data collected is publicly available and free to use at https://universal-nft-vector-database.vercel.app/search. Data can also directly be provided upon request to the authors.

Author Contributions Statement

S.S. was primarily responsible for the conceptual design, implementation efforts. He also conducted the experiments and wrote a large portion of the main portions of the research paper. N.P. assisted with the system implementation and design and wrote many parts of the research paper. A.H. advised the system design, wrote portions of the paper, and proofreaded the paper. A.S. also advised the experimental portions of the paper and proofreaded the paper. S.C. oversaw the progress of the project and proofreaded the paper.

Additional Information

Competing Interests: The authors declare no competing interests.

References

1. Wang, Q., Li, R., Wang, Q. & Chen, S. Non-fungible token (NFT): overview, evaluation, opportunities and challenges. *CoRR* **abs/2105.07447** (2021). URL `https://arxiv.org/abs/2105.07447`. `2105.07447`.
2. Assia, Y. *et al.* Colored coins whitepaper. Tech. Rep., Colored Coins (2015).

3. Entriken, W., Shirley, D., Evans, J. & Sachs, N. (2018). URL https://eips.ethereum.org/EIPS/eip-721. [Online; modified 24-January-2018].

4. Birch, D. G. W. Nfts: New fraud targets (2022). URL https://www.forbes.com/sites/davidbirch/2022/02/20/nfts-new-fraud-targets/?sh=1a9bbac34f37. [Online; posted 20-February-2022].

5. Mecozzi, R. *et al.* Blockchain-related identity and access management challenges: (de)centralized digital identities regulation. In *2022 IEEE International Conference on Blockchain (Blockchain)*, 443–448 (2022). DOI 10.1109/Blockchain55522.2022.00068.

6. von Wachter, V., Jensen, J. R., Regner, F. & Ross, O. NFT wash trading: Quantifying suspicious behaviour in NFT markets. *CoRR* **abs/2202.03866** (2022). URL https://arxiv.org/abs/2202.03866. 2202.03866.

7. Cuen, L. The Fast-Growing NFT Market Is Problematic Yet Promising — coindesk.com. https://www.coindesk.com/business/2020/09/21/the-fast-growing-nft-market-is-problematic-yet-promising/. [Accessed 08-03-2024].

8. Baker, N. An NFT Just Sold for $532 Million, But Didn't Really Sell at All — bloomberg.com. https://www.bloomberg.com/news/articles/2021-10-29/here-s-a-532-million-nft-trade-that-wasn-t-what-it-appeared. [Accessed 08-03-2024].

9. Das, D., Bose, P., Ruaro, N., Kruegel, C. & Vigna, G. Understanding security issues in the NFT ecosystem. *CoRR* **abs/2111.08893** (2021). URL https://arxiv.org/abs/2111.08893. 2111.08893.

10. Company, F. The 10 most innovative artificial intelligence companies of 2020 (2020). URL https://www.fastcompany.com/90457804/artificial-intelligence-most-innovative-companies-2020. [Online; posted 10-March-2020].

11. AI, H. Find duplicated and modified nft images with new nft search apis (2022). URL https://thehive.ai/blog/find-duplicated-and-modified-nft-images-with-new-nft-search-apis. [Online; posted 11-May-2022].

12. Tal, Y., Ramirez, B. & Pohlmann, J. The graph: A decentralized query protocol for blockchains, 2nd ed. Tech. Rep., The Graph Foundation (2019).

13. Sandford, R. Eip-721 subgraph (2019). URL https://thegraph.com/hosted-service/subgraph/wighawag/eip721-subgraph. [Online; posted 2019].

14. Radosavovic, I., Kosaraju, R. P., Girshick, R. B., He, K. & Dollár, P. Designing network design spaces. *CoRR* **abs/2003.13678** (2020). URL https://arxiv.org/abs/2003.13678. 2003.13678.

15. Dokmanic, I., Parhizkar, R., Ranieri, J. & Vetterli, M. Euclidean distance matrices: A short walk through theory, algorithms and applications. *CoRR* **abs/1502.07541** (2015). URL http://arxiv.org/abs/1502.07541. 1502.07541.

16. jae Kim, K. Financial time series forecasting using support vector machines. *Neurocomputing* **55**, 307–319 (2003). URL https://www.sciencedirect.com/science/article/pii/S0925231203003722. DOI https://doi.org/10.1016/S0925-2312(03)00372-2. Support Vector Machines.

17. Protocol, T. L. What is lens? (2022). URL https://docs.lens.xyz. [Online].

18. Phone, E. Ethos. Tech. Rep., Ethereum Phone (2022). URL https://www.ethosmobile.org/.