LAKEHEAD UNIVERSITY

An Integrated Framework for Art Image Novelty Detection and Ownership Tracing Using Deep Siamese Networks and Blockchain

by

Yutao Zhou

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE FACULTY OF GRADUATE STUDIES
OF LAKEHEAD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Master OF Computer Science

April 2026

Thesis Committee

Committee Member (Internal)

Dr.Moira MacNeil

(Assistant Professor, Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada.)

Committee Member (External)

Dr. Yong Deng

(Assistant Professor, Software Engineering, Lakehead University, Thunder Bay, Ontario, Canada.)

Supervisor

Dr. M. Mazhar Rathore

(Assistant Professor, Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada.)

Abstract

Digital artworks are increasingly distributed and shared across online platforms, raising critical challenges in both ownership verification and similarity detection. On one hand, it is difficult to establish secure, tamper-resistant records of artwork ownership in decentralized environments. On the other hand, identifying whether a newly submitted artwork is visually similar to an existing one remains a non-trivial task, especially under various artistic transformations such as style transfer, inpainting, and compositional editing. Vision Models can effectively compare image content, but they lack mechanisms to securely protect ownership. In contrast, blockchain technologies offer immutability, traceability, and decentralized data storage, yet lack the capability to evaluate visual similarity. These limitations highlight the need for an integrated solution that jointly addresses both visual similarity detection and secure ownership verification.

To resolve those issues, this thesis proposes a blockchain-based artwork verification system that integrates deep visual similarity detection with decentralized ownership registration. In the proposed framework, blockchain is used to register artwork ownership and store compact image feature payloads as immutable on-chain records, while off-chain deep learning models extract visual embeddings and perform similarity matching. Multiple visual models are trained and evaluated under different distance metrics, loss functions, and Siamese architectures. To improve the practicality of on-chain storage, the extracted embeddings are projected into lower dimensions, quantized into compact payloads, and then analyzed for storage feasibility and matching performance. The blockchain component is further evaluated through experiments on payload storage, update cost, retrieval efficiency, and large-scale matching simulation.

Experimental results show that the ResNet50 model trained with NT-Xent loss and Euclidean distance achieves the best overall performance among the tested settings, while DeiT-small performs competitively at higher embedding dimensions. The results further indicate that quantized and compressed embeddings can significantly reduce blockchain storage cost while preserving most retrieval capability, although lower-dimensional embeddings increase the false-positive rate in the final similarity simulation.

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Dr.M. Mazhar Rathore, for his support, encouragement, and guidance in my research pursuits. His valuable suggestions and support were vital in preparing this thesis and in my development as a graduate student.

I am also thankful to my thesis committee members, Dr.Moira MacNeil and Dr.Yong Deng, for their valuable suggestions and comments.

My thanks extend to my colleagues at Lakehead University for their support and valuable discussions. I would also like to thank the Department of Computer Science and the Faculty of Graduate Studies for providing the environment for my research.

I would like to thank my family for their support and encouragement.

Contents

List of Figures

List of Tables

Abbreviations

NT-Xent	**N**ormalized **T**emperature-scaled **C**ross-Entropy **L**oss
BCE	**B**inary **C**ross-Entropy **L**oss
AP	**A**verage **P**recision
AUROC	**A**rea **U**nder the **R**eceiver **O**perating **C**haracteristic Curve
CNN	**C**onvolutional **N**eural **N**etwork
ViT	**V**ison **T**ransformer
FPR	**F**alse **P**ositive **R**ate
DeiT	**D**ata-efficient **I**mage **T**ransformer
ResNet	**R**esidual **N**etwork
ETH	**E**thereum

Chapter 1

Introduction

1.1 Introduction

With the rapid development of digital art creation tools, an increasing number of people are using generative models to create images. While these models have greatly improved the efficiency of digital artwork production, they have also introduced significant challenges related to deepfakes. A large number of generated images now appear on online platforms, including many that imitate, alter, or plagiarize original artworks [2]. In some cases, individuals have copied the work of other artists and falsely claimed it as their own, thereby violating the rights of the original creators. Relying on manual inspection of these images is not only inefficient and subjective, but also difficult to maintain stable evaluation standards in a large-scale data environment. Therefore, building an intelligent identification system that automatically analyzes image features and evaluates the similarity of paintings is particularly important. Compared to traditional image similarity methods, large vision models can capture high-level artistic styles, content, and fine-grained structural details. Unlike rule-based or heuristic approaches, deep learning-based similarity detection is more flexible, stable, and accurate [3, 4]. Furthermore, visual model-based approaches operate end-to-end, require minimal manual feature extraction, and can adapt efficiently to complex, multi-style image datasets [5]. These capabilities offer a promising framework for automated analysis and detection of art forgery and plagiarism.

While visual models provide high efficiency and strong performance in image feature extraction and similarity detection, they cannot by themselves guarantee secure ownership registration, tamper resistance, or trustworthy traceability of digital artworks. In other words, a visual model can determine whether a newly submitted image is similar to existing works, but it cannot independently provide an immutable record of authorship

and ownership. Therefore, relying solely on visual models is insufficient for applications that require both similarity verification and secure ownership protection of digital assets.

In contrast, blockchain offers significant advantages in data integrity, transparency, and digital asset protection. Unlike traditional centralized or semi-decentralized storage systems, blockchain maintains a distributed ledger in which each participating node stores a copy of the validated records. This structure reduces the risk of a single point of failure and limits the possibility of unauthorized modification of ownership records[6]. In addition, through distributed consensus, confirmed on-chain records become highly resistant to tampering and can be independently verified by authorized participants. As a result, blockchain provides a trustworthy foundation for recording artwork ownership, registration history, and related feature data, thereby improving accountability, transparency, and traceability[7].

Based on these complementary advantages, this thesis combines a visual similarity model with blockchain technology. The visual model is responsible for extracting image features and identifying potentially plagiarized or highly similar artworks, while the blockchain securely and immutably stores ownership information and registered feature records. In this way, the system supports both efficient artwork similarity verification and reliable ownership protection.

This work is important because it addresses the practical need for a unified framework that integrates intelligent similarity analysis with secure, trustworthy ownership management. The proposed framework is not only relevant to digital artwork protection but also contributes to a broader research direction integrating machine-learning-based feature representations with decentralized systems.

1.2 Problem Statement

Despite the strong performance of deep learning models in visual similarity detection and the robustness of blockchain in secure data management, there remains a fundamental disconnect between these two technologies when applied to digital artwork protection.

On the one hand, existing visual similarity models, particularly those based on convolutional neural networks and Siamese architectures, can extract high-quality feature representations and accurately identify visually similar or potentially plagiarized images. However, these models typically operate in centralized environments and lack mechanisms for secure ownership registration, tamper resistance, and trustworthy traceability. As a result, they cannot independently ensure the authenticity and ownership of digital artworks.

On the other hand, blockchain technology provides a decentralized and immutable infrastructure for recording digital asset ownership. It ensures data integrity, transparency, and resistance to unauthorized modification. Nevertheless, blockchain systems are not designed to perform visual similarity analysis and cannot directly process high-dimensional feature representations generated by deep learning models. In particular, raw feature embeddings are often stored in floating-point formats (e.g., float32), which are inefficient and costly to store on-chain, limiting their practical deployment.

Therefore, current solutions cannot simultaneously achieve accurate visual similarity detection and secure, scalable ownership verification within a unified framework. This gap prevents the development of an effective system for protecting large-scale digital artwork, where both similarity assessment and trustworthy ownership management are required.

1.3 Contributions, Scope, and Assumptions

The main contributions of this thesis are as follows:

- A comparative evaluation of Siamese visual similarity models using multiple backbones, distance metrics, and training losses for artwork plagiarism detection.
- A compact embedding quantization framework for efficient blockchain-compatible feature storage.
- A unified blockchain-assisted artwork verification framework integrating deep visual similarity detection with decentralized ownership registration.
- A detailed evaluation of storage overhead, gas consumption, and matching performance for on-chain embedding deployment.

The scope and assumptions of this thesis are as follows:

- The study evaluates selected artwork datasets and does not claim universal coverage of all artistic media.
- The system focuses on similarity-based verification rather than legal copyright adjudication.
- Similarity detection is used as decision support for plagiarism screening, not as absolute proof of infringement.

This thesis begins with an introduction to the research background and a review of related work, followed by the presentation of the proposed approach and research gap. The system framework is then outlined, and the experimental setup's implementation details are described in depth. Next, the evaluation methodology, metrics, and experimental results are presented and analyzed. Finally, the thesis concludes with a summary of key findings and discusses potential directions for future work.

Chapter 2

Background and Related Work

This chapter reviews the theoretical background and related research relevant to this study. First, deep learning-based visual similarity models and Siamese network architectures are introduced. Next, feature compression and blockchain-based storage techniques are discussed. Finally, existing research is reviewed to identify current limitations and motivate the proposed framework.

2.1 Deep Learning-Based Visual Similarity Models

Image feature extraction has long been a fundamental task in computer vision. Traditional methods relied on manually designed local features, such as SIFT and SURF, which often lack generalizability across diverse datasets, particularly in artistic domains [8].

With the rapid development of deep learning, data-driven approaches have significantly improved the performance of visual representation learning. Convolutional neural networks (CNNs) have demonstrated strong capabilities in extracting hierarchical features and have achieved remarkable success in applications such as face recognition [9]. More recently, transformer-based architectures, such as Vision Transformer (ViT), have further advanced image representation learning by modeling global dependencies within images [10].

Compared to traditional handcrafted features, deep learning-based methods can learn more discriminative and semantically meaningful representations. These representations capture not only low-level visual patterns but also high-level structural and stylistic information, making them particularly suitable for digital art similarity analysis and verification tasks.

2.2 Model Selection

This study employs four representative backbone models: ResNet50, ResNet101, InceptionResNetV1, and DeiT-Small. These models are selected to encompass both convolutional and transformer-based architectures, enabling a comprehensive comparison of feature-representation capabilities.

ResNet50 is adopted as the primary convolutional baseline due to its strong balance between representation capability and computational efficiency. It is based on residual learning, in which identity shortcut connections are introduced to mitigate the vanishing gradient problem and enable training of deeper networks. The bottleneck design allows the model to learn hierarchical spatial features while maintaining a relatively low computational cost. In the context of artwork similarity detection, ResNet50 is particularly effective at capturing local texture patterns, brush strokes, and structural details that are critical for distinguishing subtle visual differences between artworks.

ResNet101 extends the ResNet architecture by increasing its depth, thereby providing greater representational capacity at the cost of increased computational complexity. It is included to evaluate whether deeper convolutional networks yield meaningful improvements in similarity-detection performance.

DeiT-Small (Data-efficient Image Transformer) is selected as the representative transformer-based model. Unlike CNNs, DeiT processes images as sequences of patches and models global relationships through self-attention mechanisms. This enables the model to capture long-range dependencies and global semantic structures, which are important for understanding high-level artistic styles and compositions. Additionally, DeiT introduces a distillation strategy that allows efficient training on relatively smaller datasets while maintaining competitive performance. This makes it suitable for evaluating whether attention-based architectures offer complementary advantages over convolutional models for visual similarity tasks.

InceptionResNetV1, commonly used in the FaceNet framework, combines Inception modules with residual connections to produce highly discriminative embeddings [11]. In this study, it is used as an alternative backbone to examine the effectiveness of architectures originally designed for metric learning tasks. Unlike the original FaceNet implementation that uses Triplet Loss, this work applies both binary cross-entropy (BCE) loss and NT-Xent contrastive loss to maintain consistency across all models and enable controlled comparison.

Overall, these models are relatively lightweight compared to large-scale vision architectures, making them suitable for practical deployment scenarios. The use of compact backbones helps reduce inference latency, computational cost, and the resource requirements of system nodes, which is particularly important in blockchain-integrated environments.

2.3 Siamese Network

All models in this study are trained using a Siamese network framework. Siamese networks are widely used for image similarity learning and verification tasks. The architecture consists of two identical subnetworks that share parameters, each processing one input image to produce feature embeddings.

The similarity between two images is then measured based on the distance between their embeddings. During training, the model learns to minimize distances between similar image pairs while maximizing distances between dissimilar pairs. This pairwise learning strategy enables the model to capture relative relationships between images more effectively than traditional single-input classification approaches.

Compared to conventional methods, Siamese networks offer improved generalization in similarity-based tasks, such as image retrieval, duplicate detection, and forgery analysis [12]. Given these advantages, the Siamese framework is adopted as the core training architecture in this study.

2.4 Feature Compression and Quantization

Although deep visual embeddings are highly effective for similarity matching, their high dimensionality poses significant challenges for storage and computational efficiency. In large-scale systems, such as image retrieval or similarity verification, storing floating-point embeddings (e.g., FP32) incurs substantial memory overhead, limiting scalability and increasing system latency.

To address this issue, embedding compression techniques have been widely studied, including quantization, dimensionality reduction, and pruning. Among these approaches, quantization is one of the most practical and widely adopted methods due to its simplicity and efficiency. It reduces the precision of floating-point embeddings by mapping them into low-bit integer representations, such as 8-bit (INT8) or 4-bit (INT4), thereby significantly decreasing memory consumption. In addition, lower-precision representations

can accelerate similarity computation by reducing arithmetic complexity and improving hardware utilization.

Recent studies have demonstrated the effectiveness of low-bit quantization in embedding-based retrieval systems. For example, Jeong et al. proposed a 4-bit quantization scheme for vector embeddings in retrieval-augmented generation (RAG) systems, achieving up to an 8× reduction in storage while maintaining acceptable retrieval performance [13]. However, their findings also indicate that aggressive quantization introduces approximation errors, which may degrade similarity accuracy, especially at extremely low bit widths.

More broadly, prior work has shown that embedding compression presents a trade-off between memory efficiency and model performance. Quantization-based methods, in particular, offer a favorable balance by substantially reducing storage requirements while preserving the relative similarity structure of embeddings, which is critical for matching tasks.

In this work, a similar quantization strategy is used to compress visual feature embeddings prior to storage. Specifically, floating-point embeddings are transformed into compact integer-based representations, enabling efficient storage and transmission. This design is particularly suitable for blockchain-based systems, where storage costs and data size are critical constraints. By reducing payload size while preserving sufficient discriminative information, the proposed approach enables scalable, efficient similarity verification in a decentralized environment.

2.5 Blockchain-based Storage for Image Verification

Blockchain technology provides a decentralized and tamper-resistant framework for data storage and verification. Its core properties, including immutability, transparency, and distributed consensus, make it particularly well-suited for applications that require trust and data integrity, such as digital ownership management and copyright protection [14].

In this work, the blockchain component is implemented based on Ethereum, a widely used programmable blockchain platform that supports smart contracts. In Ethereum, data is stored and updated through transactions initiated by user accounts. Each transaction is recorded in blocks and validated by network nodes through consensus mechanisms. Once confirmed, the data becomes effectively immutable. Storage and computation on Ethereum incur costs in the form of gas, which reflects the computational resources required to execute operations. As a result, both data size and execution complexity must be carefully controlled in practical applications.

Ethereum enables the deployment of smart contracts, which are self-executing programs stored on-chain. These contracts define the logic for data storage, access, and verification. In this study, smart contracts are used to register artwork ownership and to store compact feature representations for each image. Each stored record typically includes ownership information and a corresponding feature payload, enabling decentralized, verifiable management of artwork data.

In the context of multimedia systems, blockchain has been explored for applications such as registering ownership of digital artwork, tracking provenance, and verifying copyright. By recording metadata or ownership information on-chain, these systems enable verifiable, transparent tracking of digital assets without relying on centralized authorities [15]. However, directly storing raw multimedia data, such as images, on the blockchain is generally impractical due to high storage costs and limited throughput. Blockchain platforms impose strict constraints on data size, and storage operations often incur monetary costs (e.g., gas fees in Ethereum-based systems). As a result, existing approaches typically store only lightweight representations, such as cryptographic hashes or off-chain references, while keeping the original data in external storage systems.

While hash-based approaches ensure data integrity, they are limited in capturing semantic similarity. Even minor modifications to an image can yield entirely different hash values, rendering such methods unsuitable for similarity-based verification tasks, such as detecting stylistic imitation or partial reuse in digital artworks. To overcome this limitation, recent research has begun exploring the integration of machine learning techniques with blockchain systems [14]. Instead of storing raw data or hashes, compact feature representations generated by deep learning models can be stored on-chain. These feature embeddings preserve semantic information and enable similarity-based comparison, while maintaining relatively low storage overhead.

In this work, we adopt this approach by storing quantized visual feature embeddings on the Ethereum blockchain through smart contracts. By combining embedding compression with on-chain storage, the system enables efficient similarity verification while significantly reducing storage costs. This design bridges the gap between secure data management provided by blockchain and semantic understanding enabled by deep learning, making it suitable for scalable, decentralized image-verification applications.

2.6 Related Work

CNN-based models have already achieved good results in judging image similarity in many fields, such as face recognition [9], face matching [16, 17], speaker recognition [18],

instance retrieval [3], and medical images [19–21]. Specifically, in the field of facial recognition, Hangaragi et al.'s research [22] reported a deep learning facial recognition accuracy of 0.9423, and the research by Phillips et al. [23] indicates that deep learning facial recognition models have outperformed humans in recognizing strangers' faces. It's worth noting that, compared to deep learning models in the medical field, where a single model is typically used to identify lesions in only one area [24], or deep learning face recognition models that are trained extensively on faces, models need to handle data with a wider stylistic range when processing similar artwork images. These artwork images include, but are not limited to, abstract and realistic styles, portraits and landscapes, sketches and oil paintings, etc. This places higher demands on the model's generalization ability. In the area of AI Art identification, Angeline et al. [25] used a trained CNN to perform binary classification of AI-generated images, achieving an accuracy of 0.8795. The research by S. Chopra et al. [12] indicates that convolutional networks trained in a Siamese architecture are better suited to handling large volumes of heterogeneous image data. This characteristic enables such models to effectively handle the complex and diverse visual styles commonly found in digital artworks and provides a robust training paradigm for similarity learning. Anas Laamouri et al. [26] reported that a CNN model trained with a Siamese network framework achieved a matching accuracy of at least 0.95 in a product image recommendation task. Furthermore, the research by C. Ranjeeth Kumar et al. [27] abandoned traditional keypoint-based feature extraction methods and instead used the Euclidean distance to measure similarity between faces using the Siamese network framework.

On the other hand, for the blockchain, there is some existing work on model feature quantization and blockchain image data storage. In Teahe Jeong's research, 4-bit quantization is applied to large language models output vectors, reducing the memory storage requirements for high-dimensional vector databases [13]. On the other hand, the research from Moreaux combines visual fingerprinting technology with blockchain, providing transparency and traceability for digital assets [28]. Also, Zhou's research proposed a blockchain-based visual art copyright protection framework, and the image similarity check is on off-chain using conventional image matching [29].

2.7 Research Gap

Several limitations remain in existing blockchain-based artwork verification systems. From the figure 2.1, Moreaux and Zhou's research does not perform direct file storage on-chain; they only store part of the information on the blockchain. Secondly, they used

	Moreaux et.al	Zhou et al.	This Research
Direct on-chain	Offchain	Offchain	Onchain
Similar check	Visual fingerprint	SIFT	Visual model
Feature compress	No	No	Yes

FIGURE 2.1: Work compare

a relatively traditional approach to perform an image similarity check, and, lastly, they do not perform feature compression.

On the other hand, deep learning-based visual embeddings provide significantly richer and more discriminative representations by learning hierarchical semantic features from data. These embeddings enable more accurate similarity comparison in complex visual domains. However, their high dimensionality introduces substantial storage overhead, making them difficult to deploy directly in blockchain environments.

Although embedding compression techniques, such as quantization, have been proposed to reduce storage requirements, existing work primarily focuses on centralized systems and large language models [13], the quantization performance and result for a lightweight visual model are not clear.

Therefore, there remains a gap in developing a unified framework that combines semantically meaningful feature representation, efficient compression, and blockchain-based storage. This work aims to bridge this gap by replacing hand-crafted perceptual hashes with deep visual embeddings and integrating quantization techniques to enable scalable, semantically robust image verification on the blockchain.

2.8 Research Objective

This study aims to explore a method for verifying the similarity of artistic images using deep learning models and blockchain.

First, this study will train and evaluate selected models with different architectures and training configurations using the Siamese Network training framework to identify models that perform well on the artistic image similarity judgment task.

Second, this study will analyze changes in the performance of the model's output features under dimensionality constraints and evaluate the impact of accuracy loss during feature quantization and dequantization on similarity judgment results.

Finally, this study will further explore the feasibility of directly storing quantized image features on the blockchain and evaluate its performance in terms of storage overhead and actual deployment costs.

Chapter 3

Proposed Framework

This chapter presents the overall architecture of the proposed blockchain-assisted artwork verification framework. First, the system components and their interactions are introduced. Next, the image similarity query and ownership verification workflows are described. Finally, the projection-head optimization strategy used for embedding dimension control is discussed.

3.1 System Overview

An overview of the proposed system is illustrated in Figure 3.1. The system integrates deep visual similarity models with blockchain-based storage to enable secure, efficient artwork verification. Each service node consists of both online and offline components, forming a hybrid architecture that balances computational efficiency and data integrity.

The system is composed of four main components: (1) a visual Siamese network model for feature extraction and similarity computation, (2) a feature quantization module for compact representation, (3) a blockchain layer for secure storage, and (4) an off-chain node with a local database for efficient retrieval. These components are described as follows.

3.1.1 Visual Siamese Network Model

The visual model is responsible for extracting feature embeddings from input images and measuring similarity between image pairs. In this study, Siamese architectures are adopted with different backbone networks, including ResNet-based and transformer-based models such as DeiT. Given an input image, the model produces a fixed-dimensional

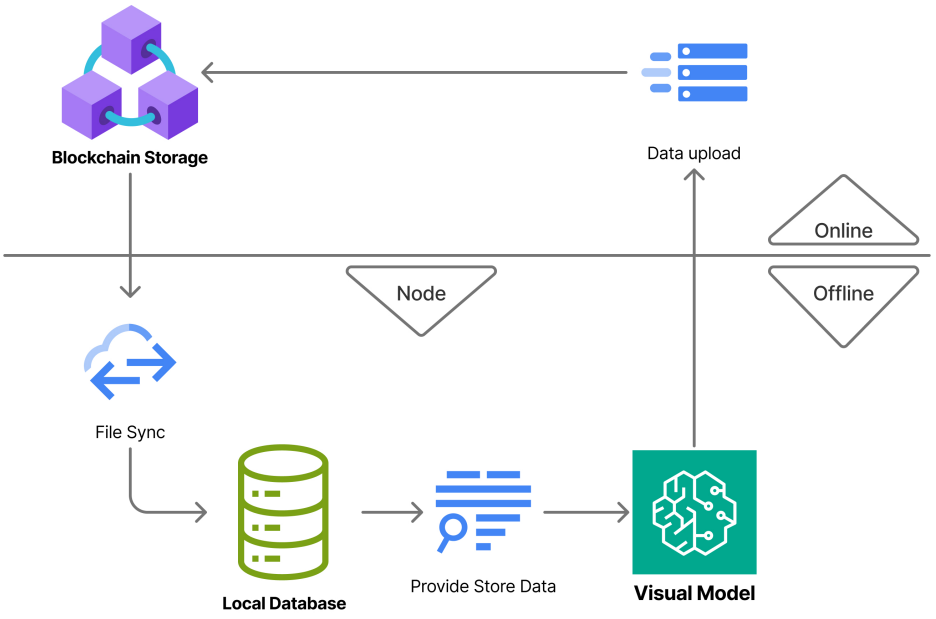

FIGURE 3.1: System Model Overview

embedding vector that captures both low-level visual details and high-level semantic information.

During inference, the model compares the embedding of a query image with stored embeddings using similarity metrics such as cosine similarity or Euclidean distance. This enables the system to automatically identify visually similar or potentially plagiarized artworks.

3.1.2 Feature Quantization

To enable efficient storage on blockchain, the extracted feature embeddings are further compressed through quantization. Since raw embeddings are typically represented in high-dimensional floating-point format, directly storing them on-chain would incur significant storage costs and reduce system scalability.

Therefore, this study adopts a lightweight quantization strategy to convert continuous-valued embeddings into compact integer representations. This process significantly reduces payload size while preserving the relative similarity relationships between embeddings. As a result, the system achieves a balance between storage efficiency and matching performance.

3.1.3 Blockchain Storage Layer

The blockchain serves as a decentralized and immutable storage layer for verified image features and ownership information. In this system, feature embeddings are stored on-chain via smart contracts after validation by the similarity detection module.

Once uploaded, the data becomes part of the distributed ledger and is synchronized across all participating nodes. This ensures that ownership records and feature representations are tamper-resistant, transparent, and verifiable. Compared to traditional centralized storage systems, blockchain provides a trusted environment for managing digital artwork registration and traceability.

3.1.4 Off-chain Node and Local Database

To improve system efficiency, each node maintains an off-chain local database that mirrors the on-chain data. Instead of querying the blockchain for each similarity comparison, the system performs matching locally using the synchronized database.

The off-chain node is responsible for handling computationally intensive tasks such as feature comparison and candidate retrieval. Synchronization between the blockchain and local database is achieved through mechanisms such as blockchain event monitoring, data broadcasting, and periodic updates. This design significantly reduces latency while maintaining consistency with the on-chain records.

3.2 Image Similarity Query Workflow

In each service node, the trained models are uniformly deployed on the off-chain component. These models are relatively lightweight, reducing the node's performance requirements and accelerating processing time. Users can access these service nodes and use their services by submitting images they want verified for similarity or protected for ownership.

After a user uploads an image, the system initiates a matching process to determine whether it is similar to any images already stored on the blockchain. The overall query workflow is illustrated in Figure 3.2. First, the uploaded image is processed by the trained visual model to extract its feature embedding. This embedding is then compared against all stored embeddings in the local database, which mirrors the on-chain data, using a similarity metric such as cosine similarity. Based on the similarity scores, the system generates a set of candidate matches. If no similar images are found (i.e., all

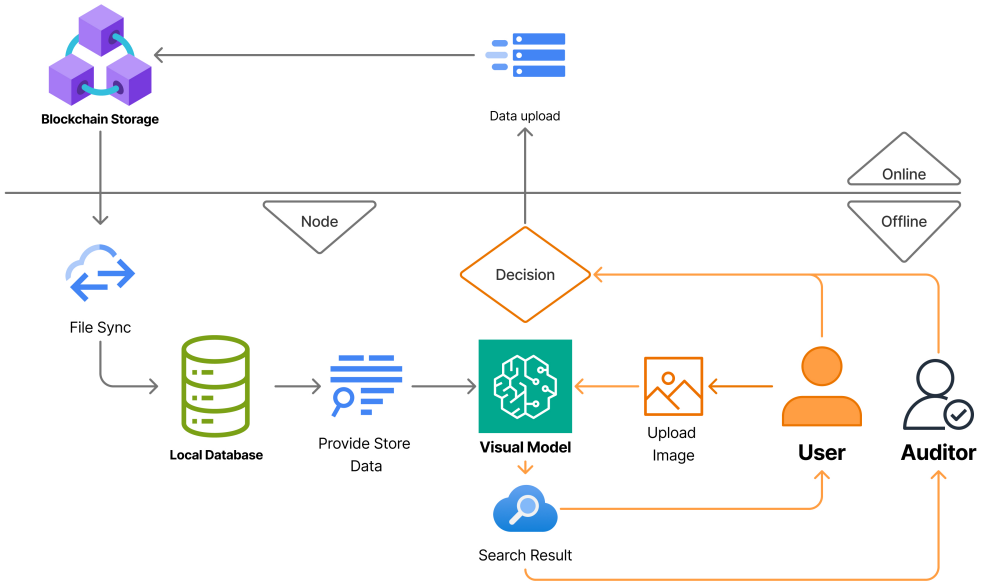

FIGURE 3.2: System Model in Query Process

similarity scores fall below a predefined threshold), the user can upload the new image to the blockchain.

If similar images are detected, the corresponding results are forwarded to an auditor for manual verification. The auditor reviews the model-selected candidates and determines whether the images are indeed similar, ensuring reliability and reducing false positives.

Once a new image is approved and uploaded to the blockchain, its corresponding feature payload is broadcast across the network and synchronized to the local databases of all nodes, maintaining consistency between on-chain and off-chain data. In this system, each node maintains an off-chain local database that is synchronized with blockchain records through conditional triggers, such as the upload of a new image, blockchain broadcast events, or periodic synchronization. The off-chain database primarily supports efficient model inference and similarity matching, while image ownership, image features, and related registration records are maintained on-chain.

Because ownership and feature records are stored directly on the blockchain, user queries about image ownership can be answered solely from on-chain data, without relying on the off-chain database, as illustrated in Figure 3.3. As a result, ownership verification is protected by the immutability of blockchain records and is not exposed to the tampering risks associated with off-chain storage. The off-chain database mainly serves as an

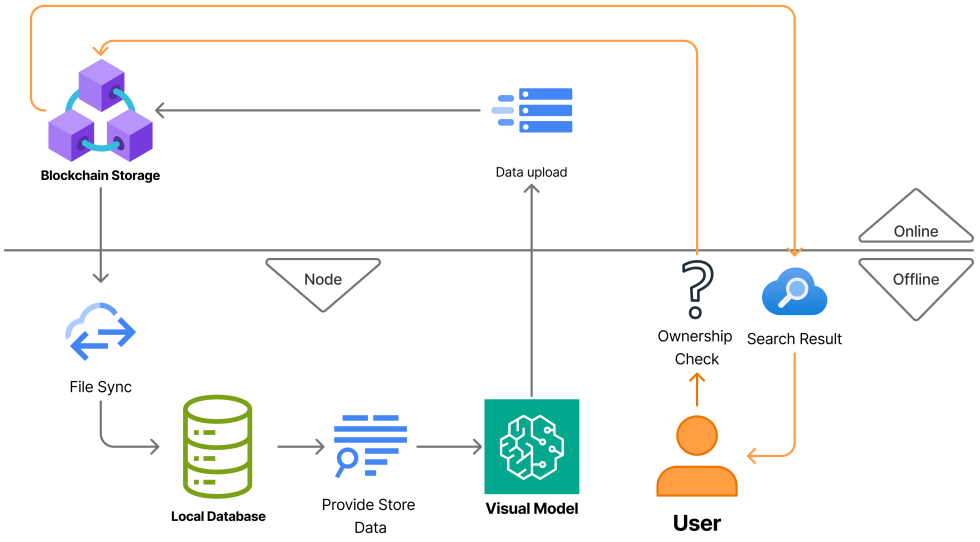

FIGURE 3.3: System Model in Ownership Check

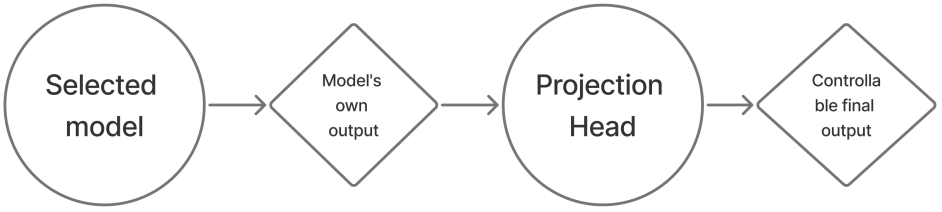

FIGURE 3.4: Projection Head

auxiliary component for computation, while the blockchain remains the trusted source of truth for ownership and registration information.

3.3 Model Optimization

To decouple the on-chain cost of image embedding from model selection in this experiment, a projection head was applied uniformly across all models during training to produce a consistent, controllable number of output vectors. For details, please refer to Figure 3.4. The projection head serves as a learnable transformation module that maps the raw feature representations produced by the backbone network to a target embedding space of specified dimensionality. During training, the backbone network and the

projection head are optimized jointly in an end-to-end manner, as the loss function is applied directly to the projected embeddings. For detailed training methods, please see Chapter 4 and Section 5.1.

Chapter 4

Visual Similarity Learning Framework

This chapter introduces the training methodology for the deep visual similarity models, including data preparation, model architecture, loss functions, and training procedures.

4.1 Dataset Usage and Preprocessing

We used two datasets for this study. The first art-images dataset is taken from WikiArt [30], and we applied additional augmentations to better fit our investigation. The second one is the DeepfakeArt Challenge dataset from Kaggle [31], which contains approximately 33.68 GB of data, comprising 36.1k image files. These images cover multiple types of deepfake artworks, including adversarial, CutMix, inpainting, style transfer, PeopleFaceArt, and self-augmented images.

The adversarial category includes original artwork images as well as images modified through blurring, noise injection, or a combination of both. The CutMix category, as shown in Figure 4.1 shows, contains pairs of original artwork images along with generated images created by scaling and cropping one image and inserting it into another. The inpainting category consists of original images and images modified through content replacement, such as transforming a village in an artwork into hills, as shown in Figure 4.2. The images in the PeopleFaceArt category come from a customized dataset that includes portrait artwork and multiple derived versions, such as grayscale, mirrored, or emoji- or text-overlaid images, as depicted in Figure 4.3. The self-augmented category contains original artwork images and their mirrored counterparts, exemplified in Figure 4.4. The style transfer category includes original artworks and images that have

FIGURE 4.1: Examples of CutMix-generated artwork images

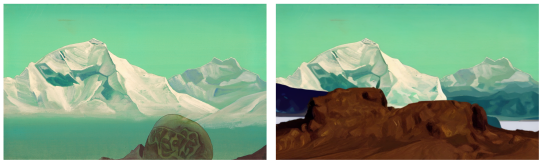

FIGURE 4.2: Examples of inpainting-based artwork modifications

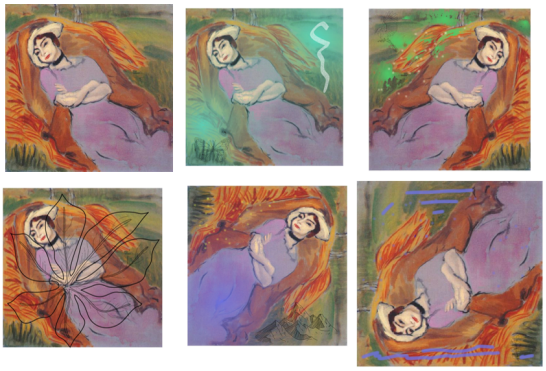

FIGURE 4.3: Examples from the PeopleFaceArt dataset

undergone stylistic transformations, such as color shifting or the application of artistic filters, exemplified in Figure 4.5.

The initial dataset was batch-processed and reorganized using Python. Within each category directory (e.g., Adversarial and CutMix), multiple subfolders are numbered. Each subfolder contains one original artwork image and several visually similar images belonging to the same manipulation category, exemplified in Figure 4.6. During training, images within the same subfolder are paired and labeled as similar, while images from different subfolders are paired and labeled as dissimilar. To avoid redundancy, duplicate pairs are not generated; for example, if the pair {A, B} already exists, the reversed pair {B, A} is excluded.

To prevent class imbalance, the number of dissimilar image pairs is constrained to match

FIGURE 4.4: Examples of self-augmentation transformations

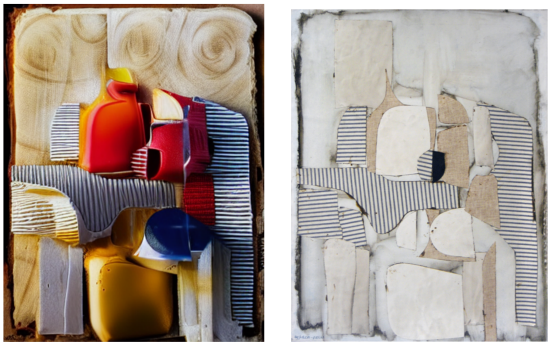

FIGURE 4.5: Examples of style-transfer-based artwork modifications

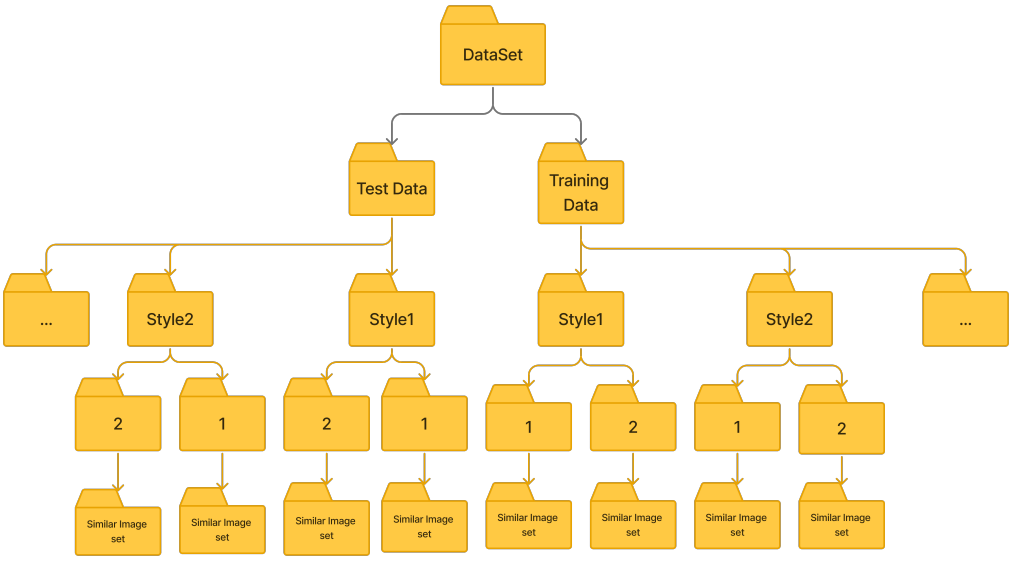

FIGURE 4.6: Organization structure of the artwork similarity dataset

the number of similar image pairs. Furthermore, the dataset was divided into two unequal parts: 25 percent for testing and 75 percent for training. The dataset contains 25,305 images, with 6,319 used for testing and 18,986 for training.

Before being fed into the neural networks, all images are resized and normalized to ensure consistent input distributions across different backbone architectures.

Specifically, all images are resized to a fixed resolution required by each model architecture (e.g., 224×224 for ResNet and DeiT-Small, and 160×160 for InceptionResNetV1). Pixel values are then normalized to the range $[-1, 1]$ using the transformation

$$x' = \frac{x}{255} \times 2 - 1 \tag{4.1}$$

where x represents the original pixel value and x' represents the normalized input.This preprocessing step ensures stable model convergence and consistent feature extraction across different architectures.

4.2 Model Selection

The four models used in this study are ResNet50, ResNet101, InceptionResNetV1, and DeiT-small. ResNet50 and ResNet101 are both residual networks in the CNN architecture, using the same bottleneck building blocks, but differ in the number of layers (50 vs. 101), resulting in differences in model capacity and feature representation. InceptionResNetV1, frequently used as the backbone in the FaceNet framework, combines the Inception architecture with residual connections to form a hybrid model that effectively learns discriminative embedded features [11]. It is worth noting that this paper does not employ the Triplet Loss training method as defined in the original FaceNet model. Instead, InceptionResNetV1 is used as a general feature-extraction backbone, and the BCE loss function and the NT-Xent comparative loss function are used for training, ensuring consistency across experimental settings and enabling controlled comparisons of variables. In addition, this experiment introduced DeiT-Small, a visual Transformer-based architecture, to systematically analyze differences in feature discrimination ability and similarity learning across network architectures.

4.3 Siamese Network

Models were trained using a Siamese network framework. Siamese networks have been widely used for image similarity learning and verification. This framework allows the model to process a pair of input images simultaneously, extract feature representations for the pair, and then measure and learn the similarity between the two images using a loss function. Compared to single-input learning methods, Siamese networks strengthen relative relationships between images during training, thereby offering advantages in

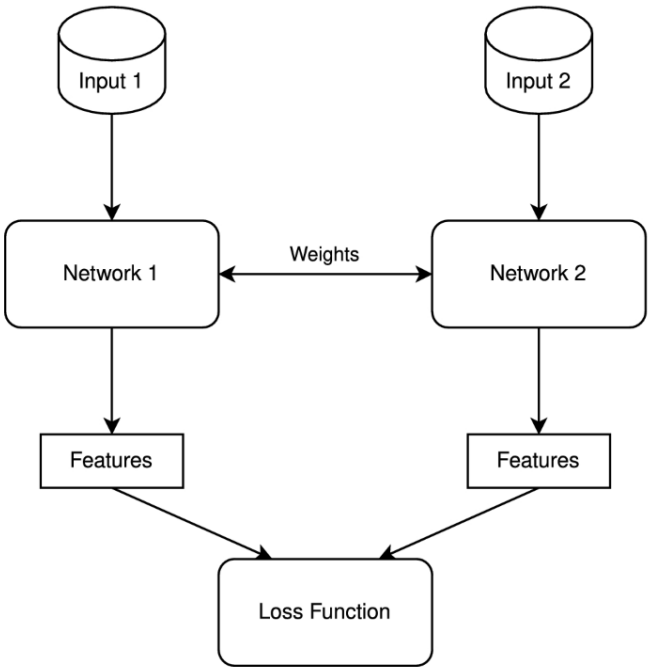

FIGURE 4.7: Structure of Siamese neural network[1]

similarity detection and forgery analysis tasks, while also exhibiting stronger generalization [12]. Based on this, this study will use the Siamese network as the training framework for the models tested in this experiment.

This study employs a unified Siamese network framework to evaluate the effectiveness of different backbone architectures and training objectives for discriminating between digital art images. Specifically, ResNet-50, ResNet-101[32], InceptionResNetV1[33], and DeiT-Small[34] are adopted as backbone feature extractors, covering convolutional, hybrid, and transformer-based architectures.

To ensure fair comparison, all backbone models share the same data construction strategy, parameter initialization scheme, backbone freezing and unfreezing schedule, learning rate scheduler, and optimizer configuration. The only experimental variables are the backbone architecture, the loss function formulation (BCE or NT-Xent), and the distance metric, while the remaining training settings are kept consistent across all experiments.

4.4 Unified Model Output and Projection Head

The models used in this experiment produce feature embeddings with different dimensionalities. ResNet50 and ResNet101 generate 2048-dimensional float32 feature vectors

as their raw outputs after image feature extraction. InceptionResNetV1 outputs a 512-dimensional feature vector, while DeiT-Small outputs a 384-dimensional feature vector.

Since the extracted feature vectors need to be processed and stored on the blockchain, controlling the embedding size is important for reducing on-chain costs. Therefore, projection heads are applied to all models to map the extracted features into a unified embedding dimension for the final output.

Given an input image pair $(\mathbf{x}_a, \mathbf{x}_b)$, both images are processed by a shared-weight backbone feature extraction network $f_\theta(\cdot)$, followed by a projection head $g_\phi(\cdot)$ to obtain low-dimensional embedding representations:

$$\mathbf{z}_a = g_\phi\big(f_\theta(\mathbf{x}_a)\big), \quad \mathbf{z}_b = g_\phi\big(f_\theta(\mathbf{x}_b)\big). \tag{4.2}$$

Here, $f_\theta(\cdot)$ denotes the backbone network, which may be instantiated as a convolutional network (ResNet-50/101), a hybrid architecture (InceptionResNetV1), or a transformer-based model (DeiT-Small). The projection head $g_\phi(\cdot)$ consists of two fully connected layers and maps backbone features into a fixed-dimensional embedding space. To stabilize similarity computation and eliminate feature scale variations, all embeddings are ℓ_2-normalized prior to distance or similarity evaluation. During the model selection and testing phase, the outputs of all models are fixed to 512-dimensional float32 feature vectors through a projection head layer, allowing their performance to be compared under the same embedding size.

After the model selection process is completed, the embedding dimension of the selected model will be further compressed (see Section 5.1) to evaluate model performance under smaller feature sizes and to balance on-chain storage costs.

4.5 Distance Metrics

In image similarity learning tasks based on embedded features, the choice of distance metric significantly impacts the model's final performance. Different distance functions reflect different definitions of similarity in the feature space and may affect the model's discriminative ability and stability [35]. This study trains four models using three different distances and two different loss functions (as described in the following sections), and uses the best epoch for each model under each configuration as the final test result. This results in a total of 24 final model configurations. In this study, three distance metrics are considered for measuring the similarity between two image embeddings x and y.

4.5.1 Euclidean distance

$$d_{\mathrm{E}}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Euclidean distance is one of the most commonly used distance metrics. It is a fundamental metric for measuring the straight-line distance between two feature vectors in a high-dimensional embedding space.

4.5.2 Cosine Distance

$$d_{\cos}(x, y) = 1 - \frac{x^\top y}{\|x\| \, \|y\|}$$

Cosine distance is a metric derived from cosine similarity that measures the direction rather than the magnitude of two feature vectors. It is insensitive to vector magnitude and is often used for similarity tasks where feature distributions exhibit scale variations.

4.5.3 Mahalanobis Distance

$$d_{\mathrm{M}}(x, y) = \sqrt{(x - y)^\top S^{-1}(x - y)}$$

where S is the covariance matrix of the feature vectors, and S^{-1} is its inverse. The inverse covariance matrix reweights the feature differences according to their variance and correlation structure, so that dimensions with larger variance or strong correlation contribute less redundantly to the final distance. Based on the covariance matrix, the Mahalanobis Distance accounts for correlations among feature dimensions, enabling the embedding space to be modeled more flexibly and offering potential advantages in handling complex feature distributions [35].

4.6 Loss Functions

In image similarity learning tasks, the choice of different loss functions during the learning process will affect the training results of the model [36]. In this experiment, in addition to using three different distance metrics, two loss functions were used to train the model separately: BCE and NT-Xent.

In similarity learning scenarios, the BCE loss function treats image-pair similarity as a binary classification problem. This method guides the model to distinguish between similar and dissimilar image pairs by minimizing the cross-entropy error between the

predicted similarity probability and the true label [37]. However, BCE itself does not explicitly optimize the structure of the feature embedding space during model training [38].

In contrast, the NT-Xent loss function is commonly used in contrastive learning. Its core principle is to directly optimize the distance between positive sample pairs in the embedding space [39]. Unlike BCE, NT-Xent does not require negative samples for learning; after NT-Xent receives positive samples, cross-sample pairs are treated as negative samples [40]. This study trains models using both the BCE and NT-Xent loss functions within the same Siamese network framework, aiming to analyze the applicability of different loss functions in digital art image similarity discrimination tasks and their impact on model performance.

It is worth emphasizing that both BCE-based and NT-Xent-based training strategies share identical feature extraction, projection, and normalization procedures. The distinction between the two methods lies solely in the similarity modeling mechanism and the corresponding loss function design applied to the normalized embeddings.

4.6.1 BCE training

Binary cross-entropy (BCE) loss is employed to formulate the image similarity discrimination task as a binary classification problem.

$$p_{ij} = \sigma(W \cdot d(f_i, f_j) + b) \tag{4.3}$$

Given an input image pair (x_i, x_j), both images are first encoded into feature embeddings (f_i, f_j) by a shared-weight Siamese network. A distance function $d(\cdot)$ is then applied to measure the distance between the two embeddings in the feature space.

The resulting distance is transformed into a similarity logit through a linear scaling operation and subsequently mapped to a probability value in the range $[0, 1]$ using a sigmoid function, representing the confidence that the image pair is semantically similar. In the actual implementation, the scale and bias parameters are fixed to $W = -1$ and $b = 0$, such that smaller embedding distances correspond to higher similarity probabilities.

During training, images from the same source group are labeled as similar samples with a label of 1, while dissimilar images from different source groups are labeled with a label of 0. The goal of the training is to make the predicted probability of positive samples close to 1 and the predicted probability of negative samples close to $0(P_{ij})$.

$$\mathcal{L}_{\text{BCE}} = -\big[y_{ij}\log(p_{ij}) + (1-y_{ij})\log(1-p_{ij})\big] \tag{4.4}$$

Here,$\mathcal{L}$ is the loss function, $y \in \{0,1\}$ denotes the label of an image pair, where $y=1$ indicates a similar (positive) sample pair and $y=0$ indicates a dissimilar (negative) sample pair. The function $\sigma(\cdot)$ denotes the sigmoid function. This formulation is equivalent to learning a similarity score $\sigma(s)$, where smaller embedding distances correspond to larger similarity logits. Consequently, more similar image pairs yield predicted probabilities $\sigma(s)$ closer to 1. During learning, the model learns that similar images should have more similar feature vectors, and dissimilar images should maintain a certain distance in the feature space.

4.6.2 NT-Xent Training

In addition to the BCE-based approach, this project also adopts the NT-Xent loss for contrastive learning. Unlike BCE, which treats image similarity discrimination as an independent binary classification task, NT-Xent directly optimizes the embedding space by modeling relative similarities among samples.

During NT-Xent training, the model encodes each input image into a feature embedding and subsequently applies ℓ_2 normalization to obtain unit-length embedding vectors, such that similarity is determined by vector direction rather than magnitude:

$$z_i = \frac{f_i}{\|f_i\|}. \tag{4.5}$$

where f_i denotes the feature embedding and z_i denotes the normalized embedding vector. The NT-Xent loss encourages embeddings of positive pairs to be close while pushing apart embeddings of negative samples. The strength of this contrastive separation is controlled by a temperature parameter τ. For a given positive pair (i,j), the per-sample NT-Xent loss is defined as:

$$\mathcal{L}_{\text{NT-Xent}} = \frac{1}{2N}\sum_{i=1}^{2N} -\log\frac{\exp\big(\text{sim}(i,p(i))/\tau\big)}{\sum_{j\neq i}\exp\big(\text{sim}(i,j)/\tau\big)}. \tag{4.6}$$

Here, $p(i)$ denotes the index of the positive counterpart of sample i within the mini-batch, and $\text{sim}(\cdot)$ represents the cosine similarity between two normalized embedding vectors. The final training objective is computed by averaging the loss across all positive pairs in a mini-batch. By explicitly enforcing relative similarities among samples, NT-Xent promotes a globally structured, discriminative embedding space, which is particularly beneficial for similarity retrieval and feature-matching tasks. Compared to BCE-based

training, this contrastive formulation often yields improved representation quality and generalization performance.

4.7 Training Strategy

In this project, additional training optimizations were applied during model training. First, a dynamic learning rate was used, gradually increasing from a low initial rate in the early stages of training to a target rate and remaining at that target until the specified training epochs. Then, the learning rate decays using cosine annealing, gradually decreasing in each subsequent training epoch. This dynamic learning rate avoids excessive disruption of the model's backbone weights due to an overly large learning rate in the early stages of training, as well as oscillations caused by an overly large learning rate in the later stages, while accelerating the model's convergence speed in the middle stages. Furthermore, a separate multiplier was used to scale the backbone learning rate. In this experiment, all training used an unscaled learning rate for the projection head, and a scaled learning rate, applied via a learning rate multiplier, for the backbone. Additionally, the model's backbone network was frozen for the first few training epochs, with only the projection head trained. This allowed the model to first adapt to the task objective in the feature projection space, resulting in smoother training and more stable, robust convergence. Finally, an early stopping mechanism based on accuracy judgment was also introduced.

4.8 On-The-Fly Validation and Threshold Selection

This study adopts an on-the-fly validation evaluation strategy during training and conducts a separate test-phase evaluation to assess the final model performance. The evaluation dataset and test dataset are randomly selected from both the DeepfakeArt Challenge and the custom dataset. The evaluation dataset is a 15 percent sample of the training dataset randomly selected before the start of each epoch. This evaluation dataset is not used in the current epoch's training but is randomly sampled again in the next epoch. This means that although the model does not use the evaluation dataset during the current training epoch before entering the evaluation phase, it can still use data from the entire training dataset during multiple epochs. The test dataset is split before training begins to ensure the model never uses those images during training.

During training, the model is evaluated on the validation set at the end of each epoch, and the evaluation results are recorded immediately. The validation evaluation process consists of the following steps:

- Feature embeddings are extracted for image pairs in the validation set using the model parameters from the current training epoch;
- Similarity scores between image pairs are computed according to the distance metric specified in the training configuration;
- An automatic threshold search is performed based on the similarity score distribution of the validation set;
- Multiple evaluation metrics are calculated at the optimal threshold and recorded as the validation results for the corresponding epoch.

Instead of using a manually selected threshold, this study determines the optimal threshold automatically through a threshold scanning procedure.After computing similarity scores for all image pairs in the evaluation dataset, the minimum and maximum similarity scores are obtained. A set of candidate thresholds is then generated by uniformly sampling values within this score range. In this work, 201 candidate thresholds are linearly spaced between the minimum and maximum similarity scores.For each candidate threshold t, predicted labels are obtained using the rule

$$\hat{y}_{ij} = \begin{cases} 1, & s_{ij} \geq t \\ 0, & s_{ij} < t \end{cases} \tag{4.7}$$

where s_{ij} denotes the similarity score between the image pair (i, j) and $\hat{y}_{ij}$ represents the predicted label.For each threshold candidate, classification metrics including accuracy, precision, recall, and F1-score are computed. The threshold that produces the highest classification accuracy on the evaluation dataset is selected as the optimal threshold.

4.9 Training Configuration

During training, the learning rate was fixed at 3e-4, and the backbone network was frozen for 3 epochs at the start, with a learning-rate multiplier of 0.02. Both methods employed early stopping with a patience of 4 epochs, meaning that if the accuracy did not improve significantly after 4 epochs, training would automatically stop, saving the best and last trained models. ResNet50 was trained with a batch size of 768, while other models were trained with a batch size of 600. The temperature parameter was set to 0.05 for models trained using the NT-Xent loss function.

Chapter 5

Blockchain-Based Storage and Matching Framework

This chapter presents the blockchain-related framework and storage methodology used in this study. First, embedding-dimension optimization for blockchain deployment is introduced. Next, the feature quantization and payload transformation strategies are described. Finally, the similarity matching process and large-scale deployment simulation methodology are presented.

5.1 Embedding Dimension Optimization for Blockchain Storage

Since one of the primary objectives of this study is to store image feature embeddings directly on the blockchain, it is necessary to reduce their dimensionality to minimize gas consumption during data upload. Following model training, a performance comparison (see Section 6.4) identified ResNet50 with NT-Xent loss and cosine distance and DeiT-Small with NT-Xent loss and cosine distance as the best-performing models. Therefore, these two models will be used for further training and testing, and their integration with blockchain will be explored. Although ResNet50 and DeiT-Small produce feature vectors with different numbers of dimensions, this study introduced a projection head during model training, allowing direct control over the number of feature dimensions in the model output. To further evaluate the models' reliability across different compression levels and to reduce blockchain storage overhead, both models are retrained using the same training methodology described in Chapter 4. The output dimension of the projection head is systematically reduced to 256, 128, and 64, while also including the

original 512-dimensional setting for comparison. As a result, each backbone model is evaluated under multiple embedding configurations, specifically 512, 256, 128, and 64 dimensions, allowing analysis of the trade-off between model performance and blockchain storage cost. The method for analyzing these training results is consistent with that in Chapter 4.

5.2 Feature Quantization and Payload Transformation

While the number of dimensions in the model's output doesn't significantly impact the model itself, the original vector output data format is float32. Each float32 vector uses 4 bytes of storage. This isn't large for offline storage, but it's substantial for blockchain. Furthermore, blockchain support for floating-point numbers is very limited; Ethereum and Solidity, in particular, don't support them. Therefore, to reduce the storage size of image feature vectors and convert them to blockchain-supported data types, this research uses quantization to shrink the model's output feature vectors, converting float32 to int8 to reduce storage space consumption. Upon on-chain processing, the quantized image features are mapped to the blockchain payload. After converting the feature vectors to int8, each vector value uses 1 byte of storage. For details on the difference in model accuracy before and after quantization, please refer to Section 6.6.

5.2.1 Feature Quantization Methodology

First, from the model's raw output for every image feature embedding:

$$\mathbf{z} = (z_1, z_2, \ldots, z_D) \in \mathbb{R}^D \tag{5.1}$$

where $\mathbf{z} \in \mathbb{R}^D$ denotes a D-dimensional real-valued feature vector. Then, the maximum absolute value of the embedding is used to determine the scaling factor:

$$s = \frac{\max_{1 \le j \le D} |z_j|}{Q} \tag{5.2}$$

After obtaining the scale s, perform the following operation on all feature vectors:

$$q_i = \text{clip}\left(\text{round}\left(\frac{z_i}{s}\right), -Q, Q\right) \tag{5.3}$$

Where

$$\frac{z_i}{s} = z_i \cdot \frac{Q}{\max_{1 \le j \le D} |z_j|} \tag{5.4}$$

Since the representable range of signed 8-bit integers is $[-128, 127]$, a symmetric range of $[-127, 127]$ is adopted in this work. Therefore, Q is set to 127 to fully utilize the available dynamic range while avoiding overflow during quantization. During quantization, the original vector z_i is divided by the scale s, then the quotient of z_i and s is rounded to the nearest integer using the round function. The operation $\frac{z_i}{s}$ can be interpreted as scaling the original feature vector such that its maximum absolute value is mapped to Q, thereby preserving as much numerical precision as possible within the integer representation. A clipping operation is used to ensure that the quantization result is not greater than Q or less than $-Q$.

Since the scale in this quantization method is computed from each image's feature vector, it is not fixed. Therefore, we need to save and upload the scale to the blockchain for inverse quantization. Because the quantized scaling factor is stored as an integer, we can use a uint8 to represent it. uint8 also consumes 1 byte of storage space, but its value range is $[0, 255]$. uint8 provides sufficient representational range for compact storage of the quantized scaling factor.The quantization method for scale is as follows:

$$s_u = \text{clip}\left(\text{round}(s \cdot 2^k), 0, 255\right) \tag{5.5}$$

This study investigates the feasibility of storage and model performance across different embedding dimensions (512, 256, 128, and 64). A larger k value provides a higher scaling factor, which helps preserve more numerical details during quantization. However, the distribution of feature values is affected by the embedding dimensionality. Experimental results show that as the dimensionality decreases, the magnitude of individual feature values tends to increase.

As a result, lower-dimensional embeddings are more likely to exceed the representable range after scaling when using larger k values (see Section A.1). Therefore, a fixed k value is not optimal across different models. Based on empirical observations, the selected k values in this study are: $k = 17$ for 512-dimensional models, $k = 16$ for 256- and 128-dimensional models, and $k = 15$ for 64-dimensional models.

After scale quantization, the scaling factor is appended to the quantized feature vector and encoded into a hexadecimal payload for on-chain storage. Given an N-dimensional quantized feature vector, the resulting payload occupies $(N + 1)$ bytes, where one byte is used to store the quantized scaling factor, and the remaining N bytes correspond to the int8 feature values. An example payload is shown in Section A.2.

5.2.2 De-quantization Methodology

During de-quantization, the payload is decoded to recover the quantized scaling factor and the int8 feature vector. The scaling factor is first reconstructed as:

$$\hat{s} = \frac{s_u}{2^k} \tag{5.6}$$

where s_u denotes the quantized scaling factor stored in the payload. The embedding is then approximately reconstructed as:

$$\hat{z}_i = \hat{s} \cdot q_i \tag{5.7}$$

where q_i represents the quantized feature values. Note that $\hat{z}_i$ is an approximation of the original feature value z_i due to quantization.

5.3 Matching Methodology

To determine whether a query image matches any artwork stored on the blockchain, a feature-based matching strategy is adopted. This process involves reconstructing feature embeddings from stored payloads and computing similarity scores between the query embedding and all stored embeddings. After decoding and de-quantizing the payload, the feature embedding $\mathbf{z}_q$ for a given query image is extracted using the trained model and normalized to unit length. The similarity between the query embedding and each reconstructed embedding is computed using cosine similarity:

$$s_i = \mathbf{z}_q \cdot \hat{\mathbf{z}}_i \tag{5.8}$$

where $\mathbf{z}_q$ denotes the normalized query embedding and $\hat{\mathbf{z}}_i$ denotes the reconstructed stored embedding. Since the query embedding is normalized, cosine similarity reduces to the dot product between vectors. For a database containing N stored embeddings, this results in a set of similarity scores:

$$\{s_1, s_2, \ldots, s_N\} \tag{5.9}$$

A pre-trained threshold t is then applied to determine whether a pair of images is considered similar:

$$\hat{y}_i = \begin{cases} 1, & s_i \geq t \\ 0, & s_i < t \end{cases} \tag{5.10}$$

Where the value of $\hat{y}_i$ is the final binary decision.

5.3.1 Vectorized Similarity Computation

Although similarity can be computed individually as:

$$s_i = \mathbf{z}_q \cdot \hat{\mathbf{z}}_i \tag{5.11}$$

For all stored embeddings, these computations can be vectorized by stacking all reconstructed embeddings into a matrix:

$$\mathbf{Z} = \begin{bmatrix} \hat{\mathbf{z}}_1 \\ \hat{\mathbf{z}}_2 \\ \vdots \\ \hat{\mathbf{z}}_N \end{bmatrix} \in \mathbb{R}^{N \times D} \tag{5.12}$$

This allows all similarity scores to be computed simultaneously as:

$$\mathbf{s} = \mathbf{z}_q \cdot \mathbf{Z}^T \tag{5.13}$$

Where $\mathbf{Z}^T$ is the transpose of Z.

5.3.2 Matching Simulation Methodology

Two implementations are considered for similarity computation: single-query matching and batch-query matching. Although both approaches are mathematically equivalent, they differ in how payload reconstruction and similarity computation are executed. For a single query image, similarity can be expressed as:

$$\mathbf{s} = \mathbf{z}_q \cdot \mathbf{Z}^T \tag{5.14}$$

where $\mathbf{Z}$ is the matrix formed by stacking all reconstructed embeddings from blockchain payloads. In the single-query matching mode, each query image is processed independently. For each query, the gallery matrix $\mathbf{Z}$ is retrieved from the offline data, and the similarity is computed with respect to all stored embeddings. This process is repeated for each query image.

In contrast, in the batch-query matching mode, payloads are retrieved and reconstructed only once to form a shared gallery matrix $\mathbf{Z}$. Multiple query embeddings are then stacked into a matrix:

$$\mathbf{Q} \in \mathbb{R}^{M \times D} \tag{5.15}$$

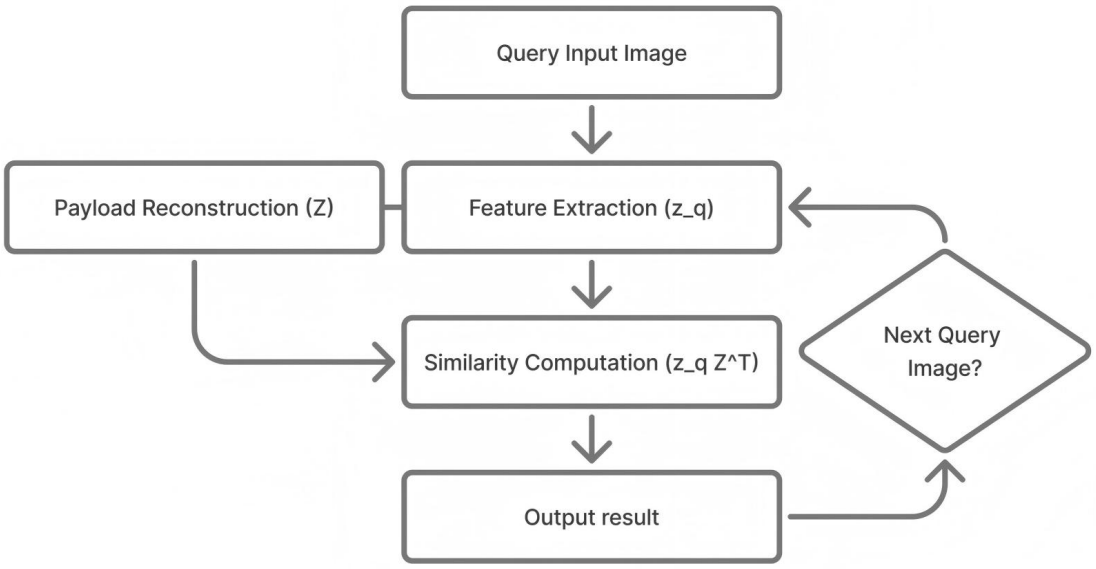

FIGURE 5.1: Single Match workflow

where M is the number of query images. The similarity matrix is computed as:

$$\mathbf{S} = \mathbf{Q}\mathbf{Z}^T \tag{5.16}$$

where $\mathbf{S} \in \mathbb{R}^{M \times N}$ contains similarity scores between all query images and stored embeddings.

Although both approaches use the same cosine-similarity formulation, batch-query matching improves computational efficiency by avoiding repeated payload reconstruction and by leveraging parallel computation on hardware.

5.3.3 Matching Simulation Deploy

To further evaluate the practical performance of the proposed system, a simulation of a real-world deployment scenario is conducted using the final trained model and its quantized feature representations stored on the blockchain.

First, 2000 unique images from the test dataset are processed by the model to extract normalized feature embeddings. These embeddings are then quantized and converted into payloads, which are stored locally to simulate a synchronized on-chain database of artworks. Next, 100 query images are selected from datasets that are not used during either the training or testing phases. These images are assumed to be dissimilar to the stored images. Each query image is compared against all stored embeddings using

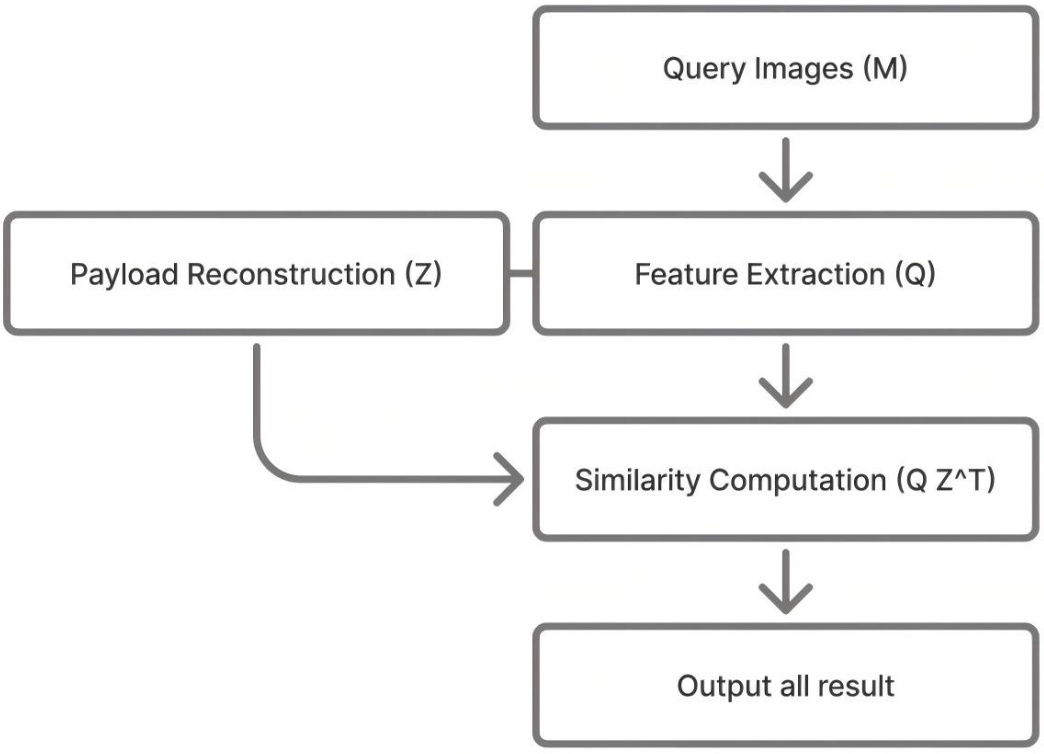

FIGURE 5.2: Batch Match workflow

cosine similarity. The simulated matching will use both single-match and batch-match on these 100 images. A single match performs an image query one by one, as shown in Figure 5.1, while a batch match queries all 100 images at once, as shown in Figure 5.2. Since no true matches are expected in this setup, any detected matches are considered false positives. The false positive rate (FPR) is calculated based on the proportion of incorrectly identified similar pairs. The average FPR across all query images is used as the final evaluation metric. In addition, to better approximate real-world conditions, the 2,000 data points used in the FPR evaluation are duplicated to construct an expanded dataset of up to 100,000 entries. The embedding dequantization time and similarity computation time are then measured again to evaluate the scalability of the matching process on large datasets. For detailed results, please see Section 6.8.

Chapter 6

Experiments and Results

This chapter presents the experimental evaluation results of the proposed framework. First, the system environment and evaluation metrics are introduced. Next, the performance of different visual similarity models is analyzed. Finally, blockchain storage performance, embedding quantization, and large-scale matching simulations are evaluated to assess the practicality and scalability of the proposed system.

6.1 System Environment

All training in this experiment was performed on a Canadian computing cluster using an NVIDIA H100 GPU. After saving all training results, the final testing phase uses an NVIDIA RTX 4080 Super in the WSL(Windows Subsystem Linux) environment to perform inference, compute performance metrics on the test dataset, and record the total inference time for each model.

All blockchain evaluation processes were conducted on a personal workstation equipped with an Intel Core i9-13900HK CPU and an NVIDIA GeForce RTX 4080 Super GPU. The system is configured with 32 GB of RAM and runs on Windows 11. Model inference was performed in a Linux-based environment using Windows Subsystem for Linux (WSL), while blockchain-related operations were executed on the Windows host system. The deep learning models were implemented using PyTorch. Blockchain experiments were conducted on Ganache, a local Ethereum test network, with smart contracts developed and tested using Remix.

6.2 Model Evaluation Metrics

This study uses Accuracy, Area Under the Receiver Operating Characteristic Curve (AUROC), F1-score, and average precision (AP) as the final performance metrics for the model.

The Accuracy is calculated as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{6.1}$$

Here, TP, TN, FP, and FN denote the numbers of true positives, true negatives, false positives, and false negatives, respectively. Accuracy measures the proportion of correctly classified image pairs among all evaluated samples.

AUROC can be interpreted as:

$$\text{AUROC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR}), \tag{6.2}$$

where $\text{TPR} = \frac{TP}{TP+FN}$ and $\text{FPR} = \frac{FP}{FP+TN}$ denote the true positive rate and false positive rate, respectively. AUROC is a threshold-independent metric that evaluates the ranking quality of similarity scores. It represents the probability that the model assigns a higher similarity score to a randomly selected positive image pair than to a randomly selected negative image pair.

F1-score is calculated as follows:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.3}$$

where $\text{Precision} = \frac{TP}{TP+FP}$ and $\text{Recall} = \frac{TP}{TP+FN}$.The F1-score provides a balanced measure of a model's classification performance by jointly considering false positives and false negatives under a fixed decision threshold. In the image similarity setting, a high F1-score indicates that the model can accurately identify similar image pairs while effectively suppressing false matches.

The Average Precision (AP) is calculated as follows.

$$\text{AP} = \frac{1}{P} \sum_{k=1}^{N} \text{Precision}(k) \cdot y_{(k)} \tag{6.4}$$

where $y_{(k)} \in \{0, 1\}$ denotes the ground-truth label of the k-th ranked sample, $\text{Precision}(k)$ denotes the precision computed over the top-k ranked samples, and P is the total number of positive samples.In the image similarity setting, AP reflects how well the model

ranks truly similar image pairs ahead of dissimilar ones across the entire similarity score list.

This chapter shows the results of all the experiments performed.

6.3 Blockchain Implementation and Evaluation

This study used Ganache as a blockchain simulation tool and Remix to develop smart contracts.

6.3.1 Smart Contract Design

The blockchain storage component of the proposed system is implemented through a Solidity smart contract. The purpose of this contract is to store compact image feature payloads together with ownership information, so that each artwork can be associated with both a creator-controlled blockchain address and a feature representation for later verification. The contract maintains a monotonically increasing counter that serves as the unique identifier for each stored artwork. A fixed payload length is specified during contract deployment and stored as an immutable variable. This design ensures that all stored feature vectors have a consistent byte length, which simplifies validation and prevents malformed data from being written to the blockchain.

Each artwork is represented by a structure containing the owner's address and the corresponding payload. The owner field records the blockchain address of the artwork's submitter, while the payload field stores the quantized feature embedding, encoded as a byte array. To support storage operations, the contract provides both single-item and batch interfaces. The `store()` function stores one payload at a time, while `storeMany()` allows multiple payloads to be inserted within a single transaction. Batch insertion is introduced to reduce the overhead of repeated transactions and improve storage efficiency when multiple embeddings must be uploaded. The contract also supports post-deployment modification through `update()` and `updateMany()`. These functions allow an existing payload to be replaced only if the caller is the artwork's recorded owner. This ownership check prevents unauthorized modification of previously stored records or a feature vector change due to a model update. Several validation mechanisms are included in the contract. The payload length must exactly match the predefined contract parameter; empty batch submissions are rejected, and invalid artwork identifiers are disallowed during update operations.

6.3.2 Blockchain Storage Evaluation

Blockchain performance and feasibility are evaluated from multiple perspectives, including storage cost and retrieval efficiency. First, the cost of uploading feature embeddings to the blockchain is analyzed in terms of gas consumption. Both single-item operations and batch operations are considered. Specifically, the total gas consumption and the average gas cost per embedding are measured for the following scenarios: single embedding storage (`store`), batch embedding storage (`storeMany`), single embedding update (`update`), and batch embedding update (`updateMany`). This allows for a comprehensive comparison of different interaction patterns with the smart contract. Second, the efficiency of retrieving stored embeddings is evaluated. The query performance is measured by the time required to retrieve multiple payloads from the blockchain and process them for similarity comparison. The retrieval time is analyzed across different dataset sizes to assess the system's scalability. For detailed results, please see Section 6.7.

6.4 Model Training Result

The experimental methods used in this section are derived from Chapter 4. The purpose of the experiments is to select the best-performing model for further development. Based on the experimental results, ResNet50 and DeiT-Small were ultimately selected as the models for this study.

6.4.1 Accuracy Performance Results

The Accuracy results, shown in Figure 6.1, indicate that most models trained with NT-Xent loss outperformed those trained with BCE loss on the test set. Among these, ResNet50 and DeiT-Small achieved the highest Accuracy when trained with NT-Xent loss using Euclidean or cosine distances. Specifically, ResNet50 trained with NT-Xent loss and Euclidean distance achieved 0.9300, DeiT-Small with NT-Xent and Euclidean distance achieved 0.9297, and DeiT-Small with NT-Xent and cosine distance achieved 0.9229. In contrast, InceptionResNetV1 performed relatively poorly even with NT-Xent loss. The lowest Accuracy scores were observed for models trained with BCE loss; the worst-performing model was InceptionResNetV1 trained with BCE loss and Euclidean distance, achieving 0.7535 on the test dataset.

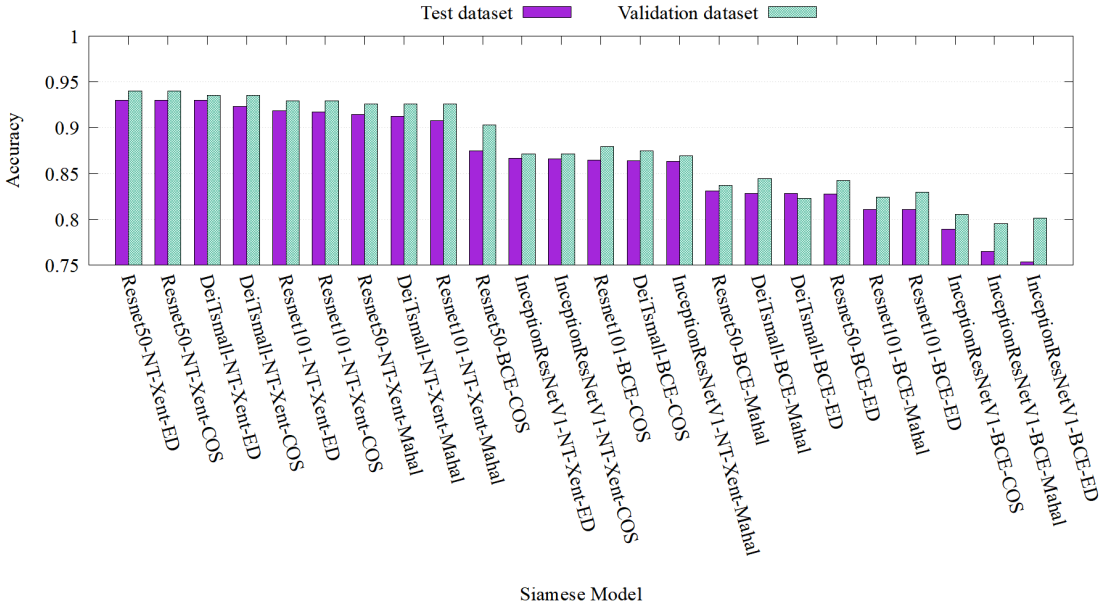

FIGURE 6.1: Accuracy performance of each model

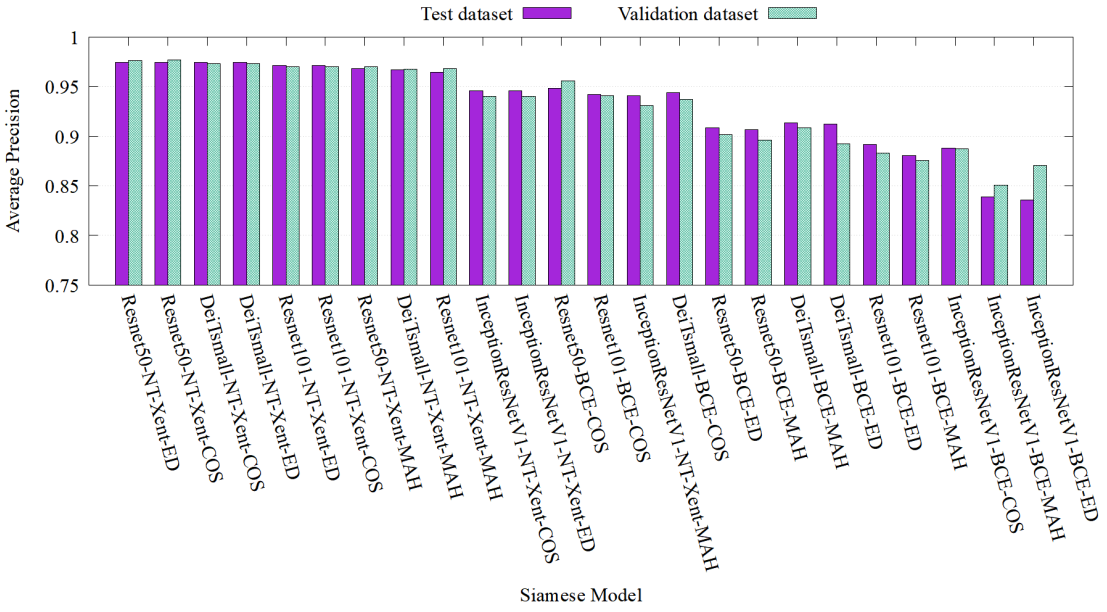

FIGURE 6.2: Average precision of each model

6.4.2 Average Precision Performance Results

Similar patterns are observed in AP results (Figure 6.2), where ResNet50 trained with NT-Xent loss and Euclidean distance achieved the highest score of 0.9747, closely followed by ResNet50 with cosine distance. DeiT-Small and ResNet101, trained with the NT-Xent loss using both Euclidean and cosine distances, also performed similarly, with AP values ranging from 0.9747 to 0.9713. In contrast, models trained with BCE loss consistently ranked lower. These results demonstrate that training with the NT-Xent

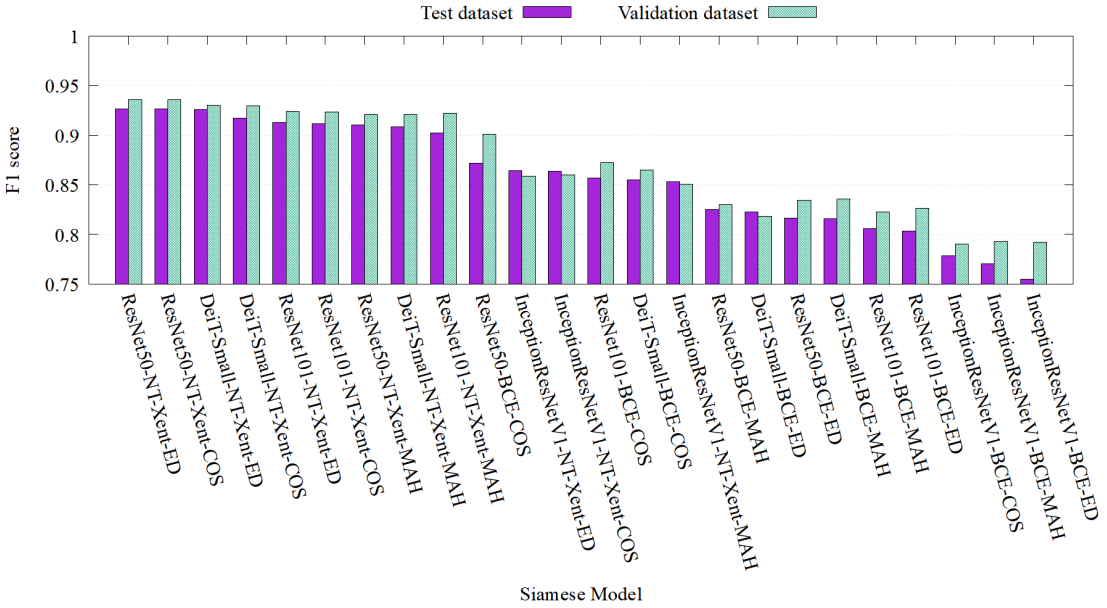

FIGURE 6.3: F1-score of each model

loss maintains high recall while effectively suppressing mismatched samples, thereby improving stability and robustness in similarity detection tasks.

6.4.3 F1-score Results

The F1-score Results on the test dataset, shown in Figure 6.3, follow a ranking pattern similar to that of Accuracy. ResNet50 with NT-Xent loss and Euclidean distance leads with a score of 0.9266, closely followed by ResNet50 with NT-Xent loss and cosine distance at 0.9265. DeiT-Small with NT-Xent loss, using Euclidean and cosine distances, performs slightly worse with scores of 0.9260 and 0.9173, respectively. The worst-performing model is InceptionResNetV1 with BCE loss and Euclidean distance, scoring 0.7546. These results further confirm the effectiveness of NT-Xent loss in maintaining balanced classification performance while minimizing false positives and false negatives.

6.4.4 AUROC Performance

Performance in terms of AUROC, as depicted in Figure 6.4, further confirms these observations. ResNet50 trained with NT-Xent loss and Euclidean distance again ranked first, with an AUROC of 0.9652, essentially identical to its cosine distance variant. DeiT-Small with NT-Xent loss achieved AUROC scores of 0.9632 and 0.9631 for Euclidean and cosine distances, respectively. As with AP, top-ranked results correspond almost entirely to NT-Xent loss, while bottom-ranked results mostly involve BCE loss. The worst

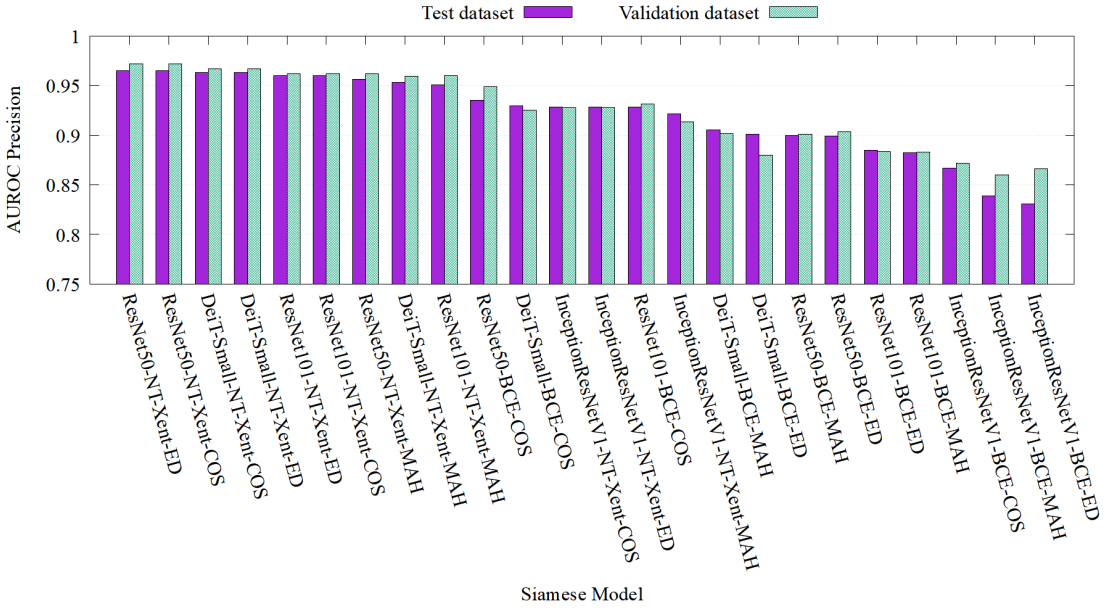

FIGURE 6.4: AUROC of each model

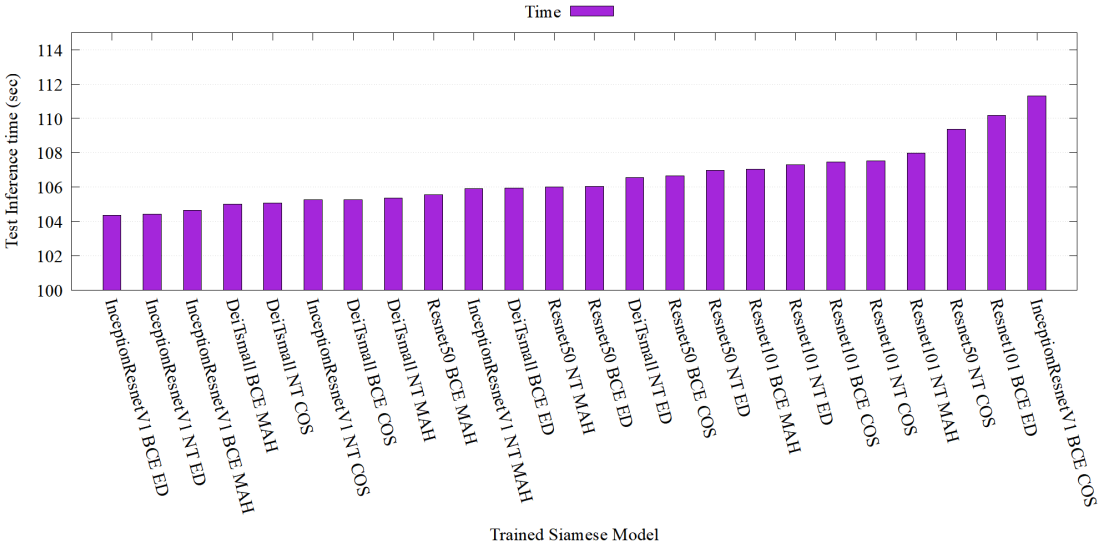

FIGURE 6.5: Inference time on test dataset

performer was InceptionResNetV1 with BCE loss and Euclidean distance at 0.8307. AUROC, being threshold-independent, indicates that NT-Xent loss not only achieves high Accuracy at specific thresholds but also provides enhanced stability and robustness in overall similarity ranking and discrimination between similar and different art-image pairs.

6.4.5 Inference Time

The model inference time recorded in this experiment includes end-to-end inference time, including I/O and image preprocessing, to compare the model's performance overhead. The results are depicted in Figure 6.5. Results show that none of the models exhibited significant differences in speed. On the inference test dataset, the fastest model was InceptionResNetV1, trained with BCE loss and Euclidean distance, which took 104.3 seconds. The slowest model was also InceptionResNetV1 trained with BCE loss and cosine distance, which took 111.3 seconds. The efficiency experiment indicates that all the models could be used in practical environments, as they are highly efficient.

Across all four evaluation metrics, models trained with NT-Xent loss consistently outperformed those trained with BCE loss under the same distance metrics, highlighting the crucial role of the loss function in model training. NT-Xent-based models are more effective at reducing both false positives and false negatives while maintaining high recall on unseen data and suppressing mismatched samples, demonstrating greater stability and robustness in similarity ranking and retrieval tasks.

Distance metrics also influenced performance, though not to a large extent. Training with Euclidean and cosine distances generally yielded similar results, whereas Mahalanobis distance resulted in comparatively lower performance. When using NT-Xent loss, the difference between Euclidean and cosine distances was minimal. This indicates that the choice of loss function is more critical than the choice of distance metric in achieving optimal performance.

Metric-specific observations further support these trends. F1-score Results (Figure 6.3) mirrored the Accuracy ranking, with ResNet50 trained using NT-Xent loss and Euclidean distance leading at 0.9266, closely followed by ResNet50 with cosine distance (0.9265). DeiT-Small with NT-Xent loss scored 0.9260 and 0.9173 for Euclidean and cosine distances, respectively, while the worst-performing model was InceptionResNetV1 with BCE loss and Euclidean distance at 0.7546. Validation-test discrepancies in AP and AUROC were observed for some models, as expected, because these metrics evaluate overall ranking quality rather than performance at a fixed threshold. Challenging positive-negative pairs near the decision boundary in the validation set can slightly reduce AP and AUROC, whereas the test set often exhibits clearer separation, yielding higher scores. Similar patterns were observed for F1-score, while overall Accuracy remained stable due to its sensitivity to the total number of true negatives.

Overall, ResNet50 achieved the best performance, followed by DeiT-Small and ResNet101, with relatively small differences when trained using NT-Xent loss, suggesting that contrastive learning reduces performance gaps across backbone architectures. In contrast,

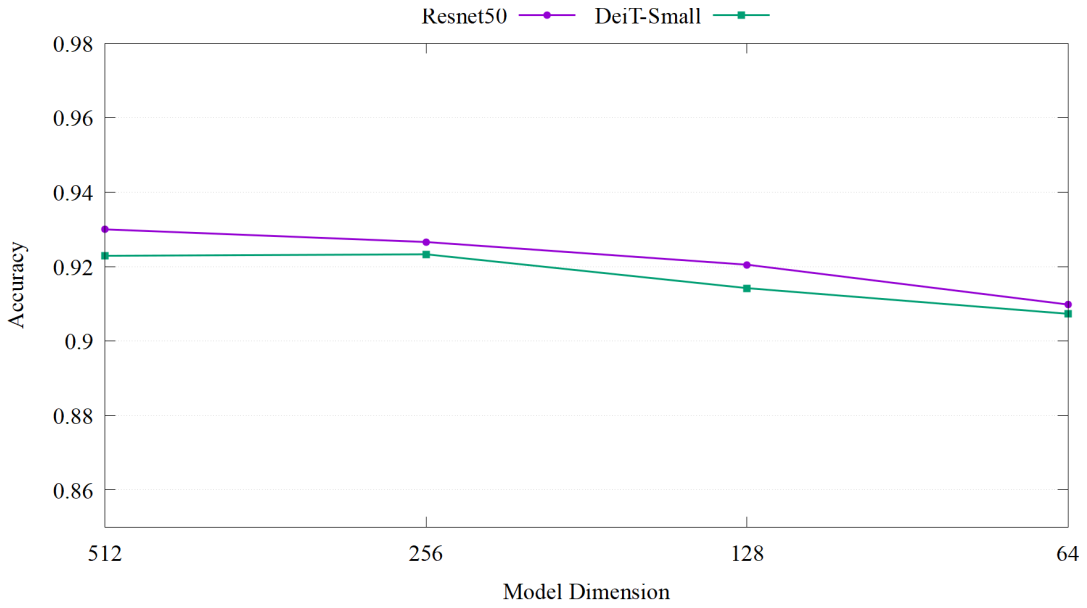

FIGURE 6.6: Accuracy performance of embedding dimension

InceptionResNetV1 exhibited slightly inferior performance, likely because it was originally designed for triplet loss in the FaceNet framework, whereas this study used BCE and NT-Xent losses for a controlled comparison. These results indicate that the dataset characteristics and the similarity discrimination task considered in this study are better aligned with ResNet50 and DeiT-Small architectures, which provide more robust and stable performance across evaluation metrics.

6.5 Embedding Dimension Optimization Result Compare

This section shows the performance of ResNet50 and DeiT-smal models trained under different Embedding Dimension constraints. The models in the following results were all trained using NT-Xent loss and cosine distance.

6.5.1 Accuracy Performance Results

Figure 6.6 shows the Accuracy performance of ResNet-50 and DeiT-Small across different output dimension settings.As the allowed output dimension decreased, the model's Accuracy also declined. Although the output dimension constraint gradually decreased from a maximum of 512 to a minimum of 64, the model's Accuracy did not drop precipitously; the decrease was minor relative to the extent of the dimensionality reduction.

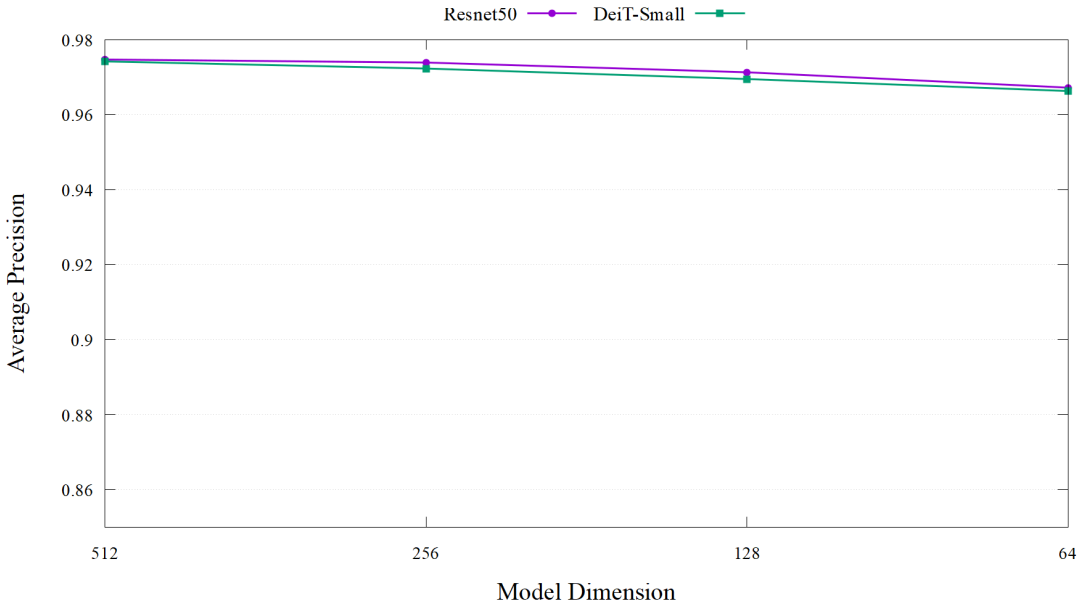

FIGURE 6.7: Average Precision of embedding dimension

The Accuracy of ResNet50 decreased from 0.93 in 512 dimensions to 0.9098 in 64 dimensions, and the Accuracy of DeiT-Small decreased from 0.923 in 512 dimensions to 0.907 in 64 dimensions.

6.5.2 Average Precision Performance Results

Figure 6.7 shows the Average Precision performance of ResNet-50 and DeiT-Small across different output dimension settings. As with Accuracy, significantly reducing the number of dimensions did not cause a precipitous drop in the model's average precision. However, the model's performance decreased very slowly after reducing the number of dimensions. The average precision of ResNet50 gradually decreased from 0.974 for 512 dimensions to 0.967 for 64 dimensions, and the average precision of DeiT-Small gradually decreased from 0.974 for 512 dimensions to 0.966 for 64 dimensions.

6.5.3 F1-score and AUROC Result

Figure 6.8 shows the F1-score, and 6.9 shows the AUROC performance of ResNet-50 and DeiT-Small across different output dimension settings. The performance of the F1-score and AUROC is similar to that of Accuracy and average precision. They were not significantly affected by the substantial reduction in dimensionality. The F1-score of ResNet50 gradually decreased from 0.927 in 512 dimensions to 0.905 in 64 dimensions, and the AUROC performance decreased from 0.965 in 512 dimensions to 0.956 in 64 dimensions. Similarly, the F1-score of DeiT-Small gradually decreased from 0.917 in

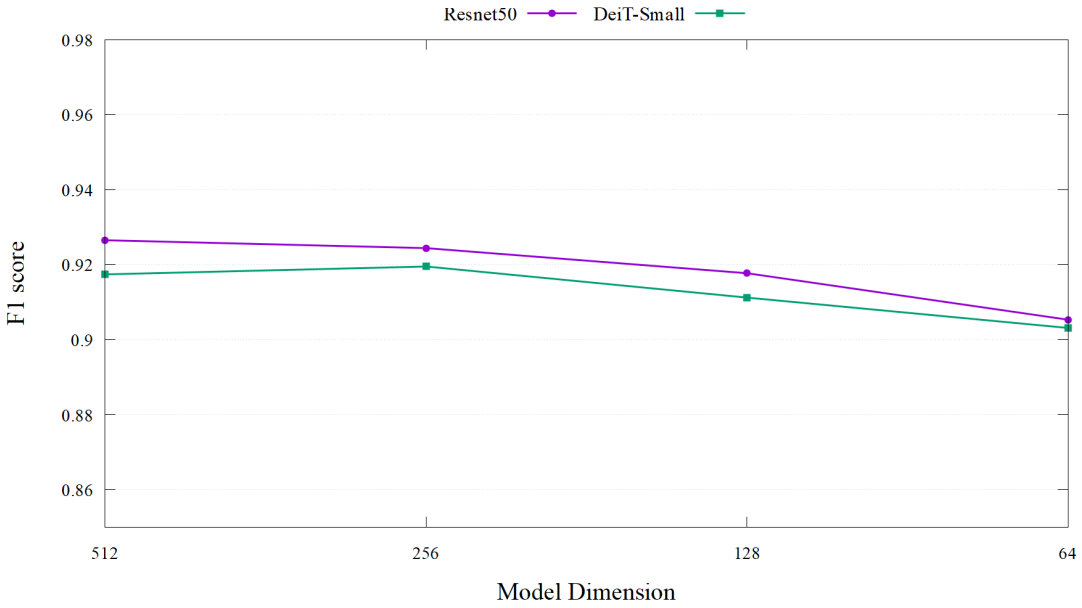

FIGURE 6.8: F1-score of embedding dimension

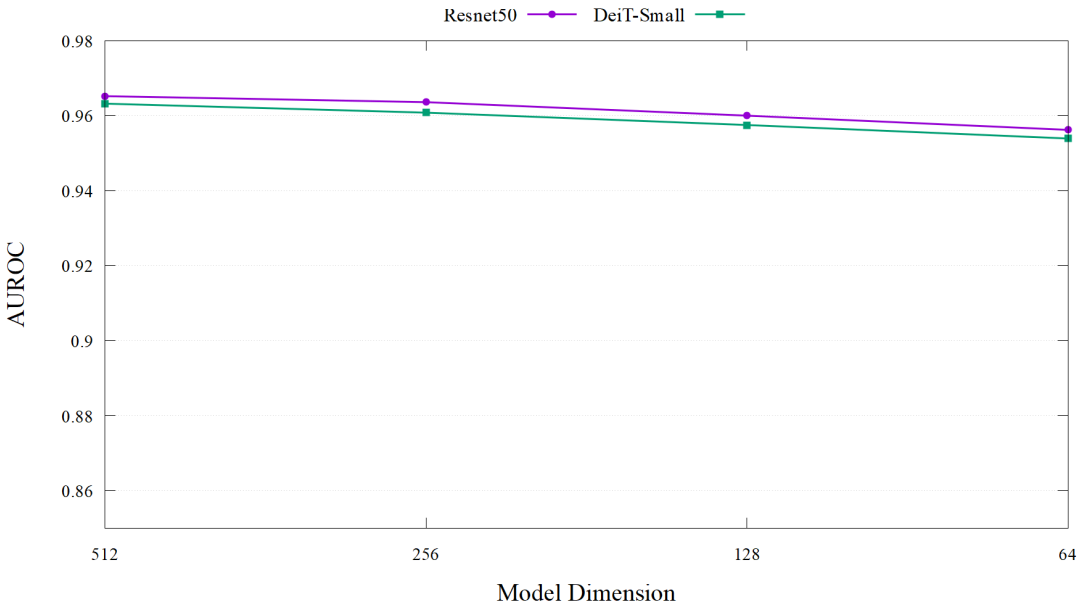

FIGURE 6.9: AUROC of embedding dimension

512 dimensions to 0.903 in 64 dimensions, and the AUROC performance decreased from 0.963 in 512 dimensions to 0.953 in 64 dimensions.

6.5.4 Inference Time

Figure 6.10 shows the inference time of ResNet-50 and DeiT-Small across different output dimension settings. Reducing the number of output dimensions in the model did not reduce inference time. The inference times of the models were similar to those in

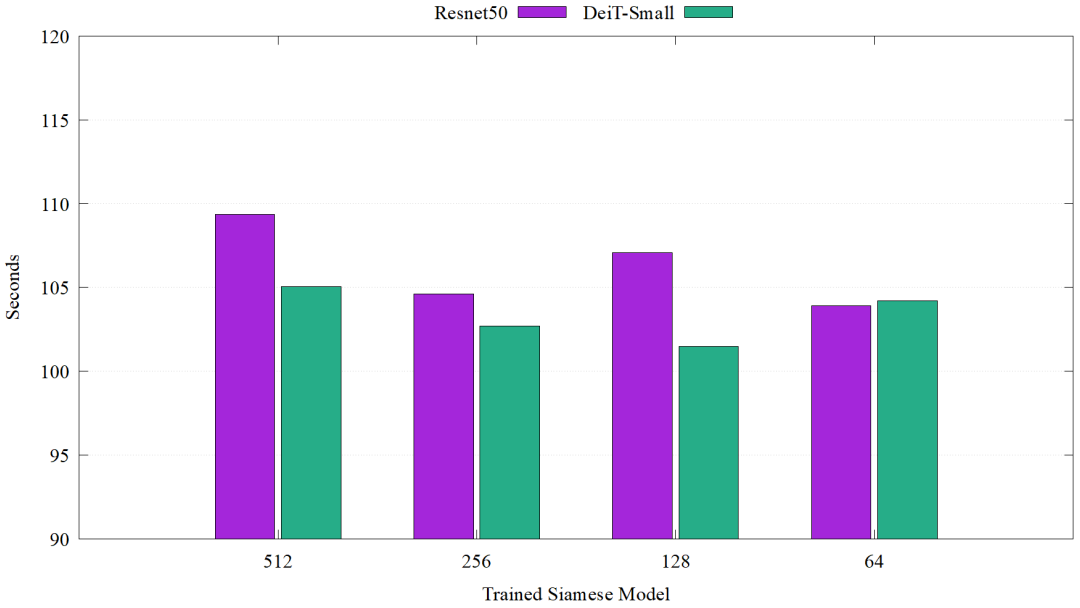

FIGURE 6.10: Inference time of different embedding dimensions

Section 6.4.5, with the longest at 109 seconds for the 512-dimensional-output ResNet50 and the shortest at 101 seconds for the 128-dimensional-output DeiT-Small.

6.6 Embedding Quantization Result Compare

All quantization testing procedures involve quantizing and dequantizing vectors from ResNet50 and DeiT-Small(both with NT-loss and cosine distance) to different output dimensions, then comparing the model's native performance.

6.6.1 Accuracy Performance Results

Figure 6.11 shows the Accuracy performance difference between the ResNet50 native image embedding output and the quantized image output. The performance difference is 0.001. Figure 6.12 shows a similar performance result on DeiT-Small; the quantization did not have a significant impact on the model's Accuracy.

6.6.2 Average Precision Performance Results

The model's AP and Accuracy were similar, and quantization had no significant impact on performance. Figure 6.13 shows ResNet50 performance before and after embedding quantization. The native output performance is only slightly higher in 128 dimensions;

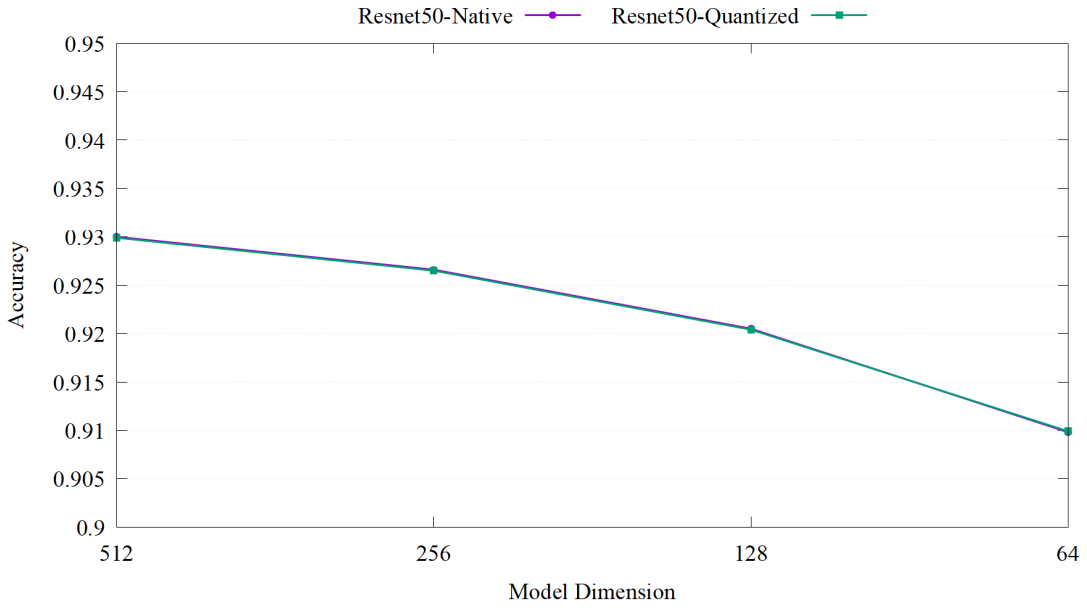

FIGURE 6.11: Accuracy Difference Between ResNet50 Native and Quantized

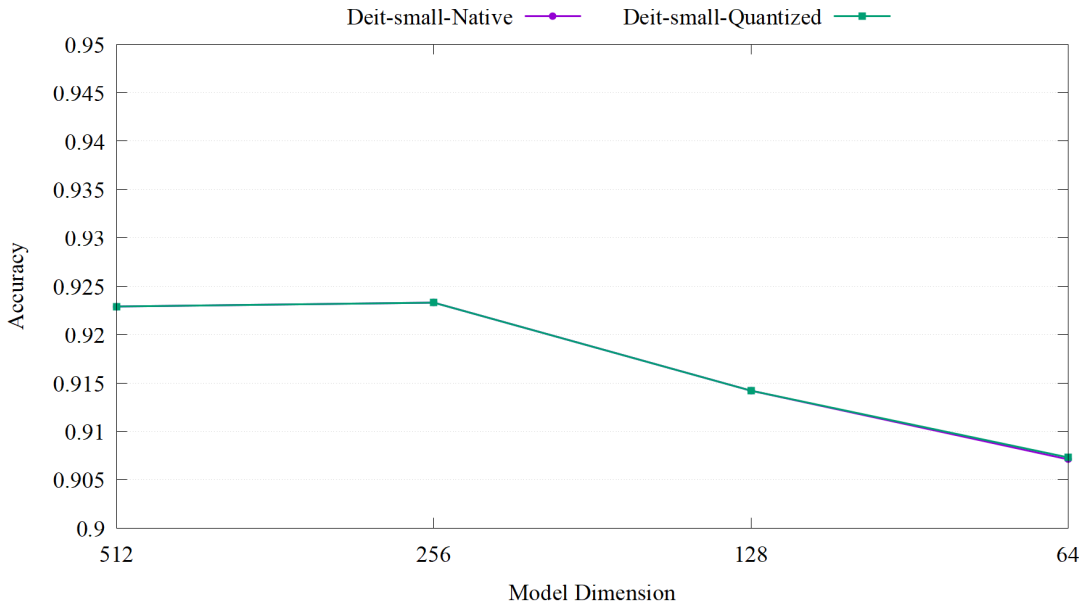

FIGURE 6.12: Accuracy Difference Between DeiT-Small Native and Quantized

in other dimensions, performance does not differ significantly after rounding the performance statistics to four decimal places. Figure 6.14 shows the performance of DeiT-Small. After retaining four decimal places in the performance statistics, the model's AP remains almost unchanged before and after quantization.

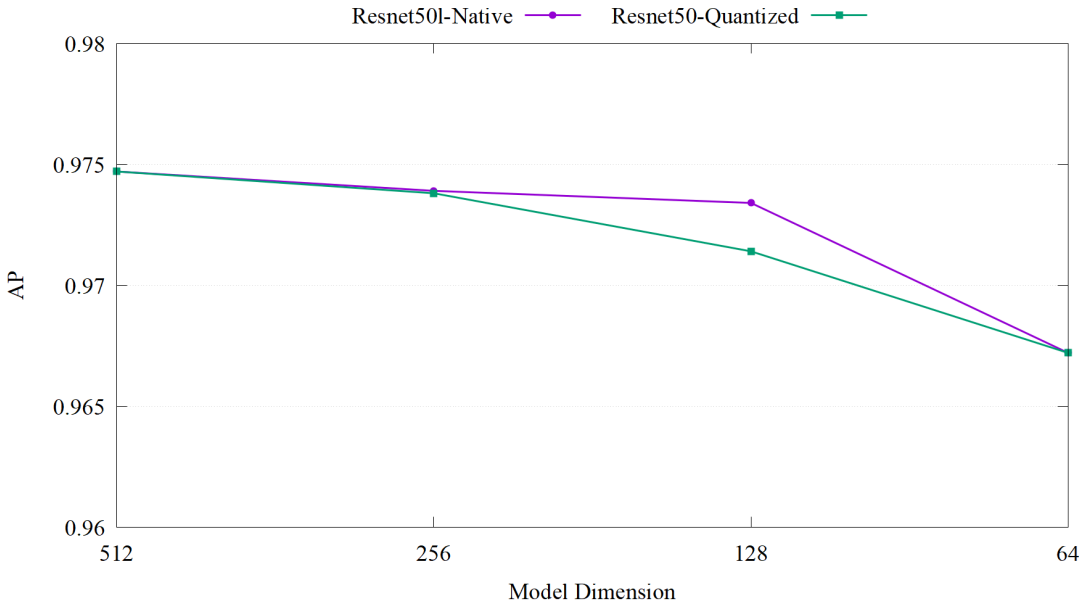

FIGURE 6.13: AP Difference Between ResNet50 Native and Quantized

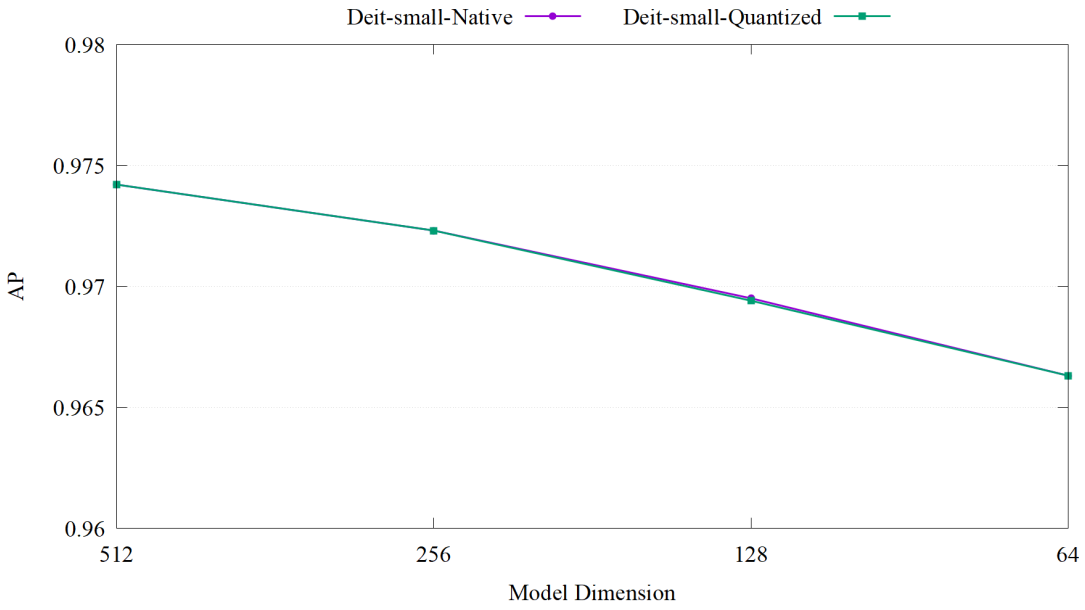

FIGURE 6.14: AP Difference Between DeiT-Small Native and Quantized

6.6.3 F1-score and AUROC Result

F1-score and AUROC results for both models also show that quantization had no significant impact on model performance. Figure 6.15 and 6.16 show the AUROC and F1-score of ResNet50, and the results are close. Figure 6.17 and 6.18 show the AUROC and F1-score of DeiT-Small, and the results are also close.

In conclusion, after applying the quantification method used in this study, no significant

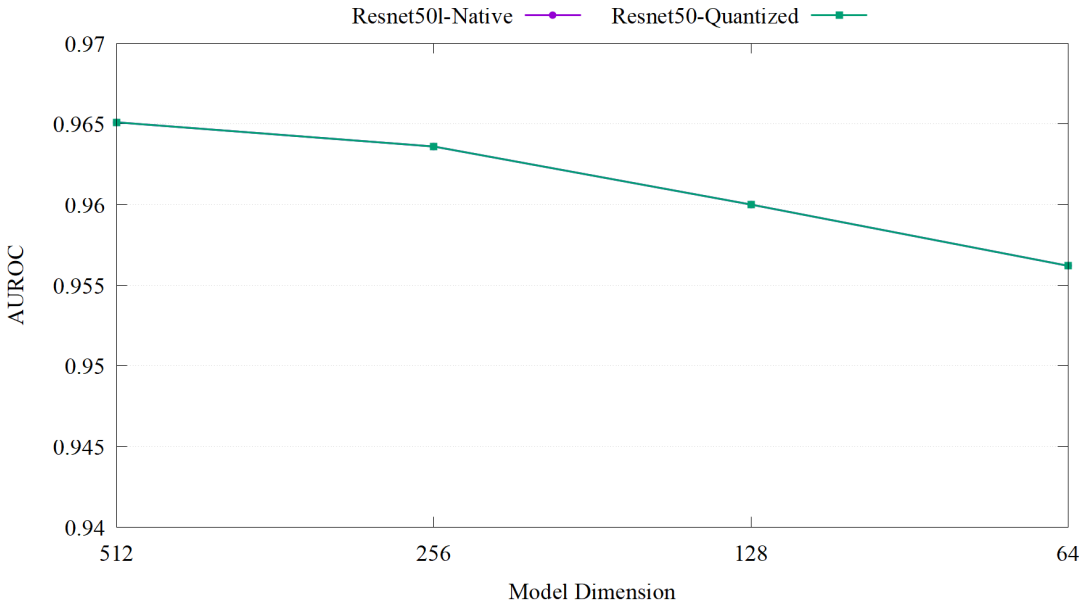

FIGURE 6.15: AUROC Difference Between ResNet50 Native and Quantized

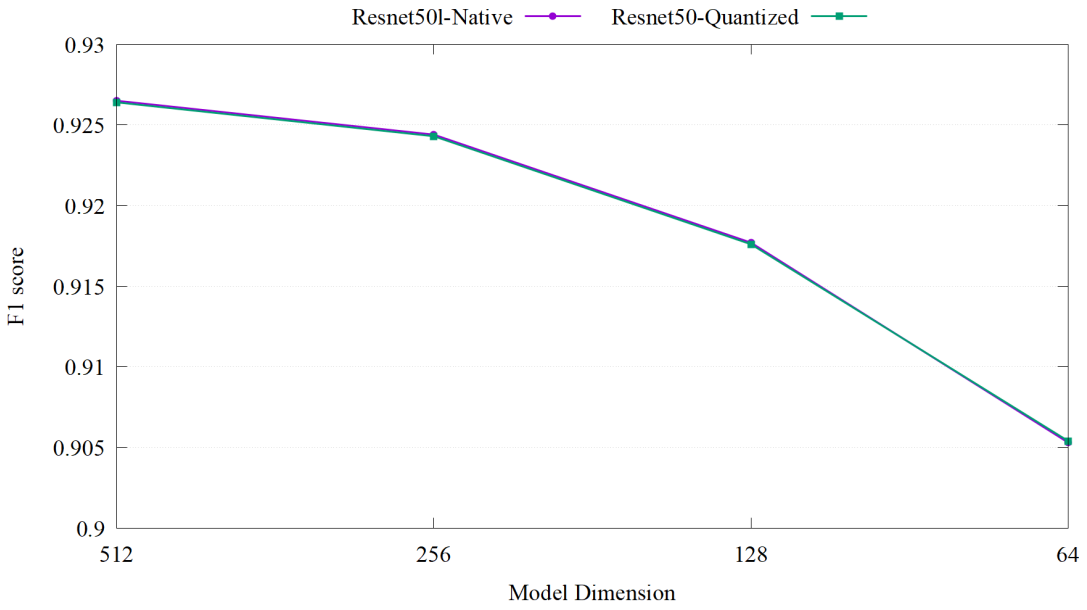

FIGURE 6.16: F1-score Difference Between ResNet50 Native and Quantized

impact on model performance was observed. After further reducing the output embedding dimension and applying quantization–dequantization prior to similarity computation, the same evaluation protocol was applied to the same dataset. The results indicate that neither dimensionality reduction nor quantization introduces significant degradation in overall performance for pairwise image comparison.

As shown in the corresponding figures, reducing the embedding dimension has a noticeable impact on Accuracy and F1-score (6.8, 6.6), while its effect on AP and AUROC is relatively minor (6.7, 6.9). This suggests that the model's ranking capability—i.e., its

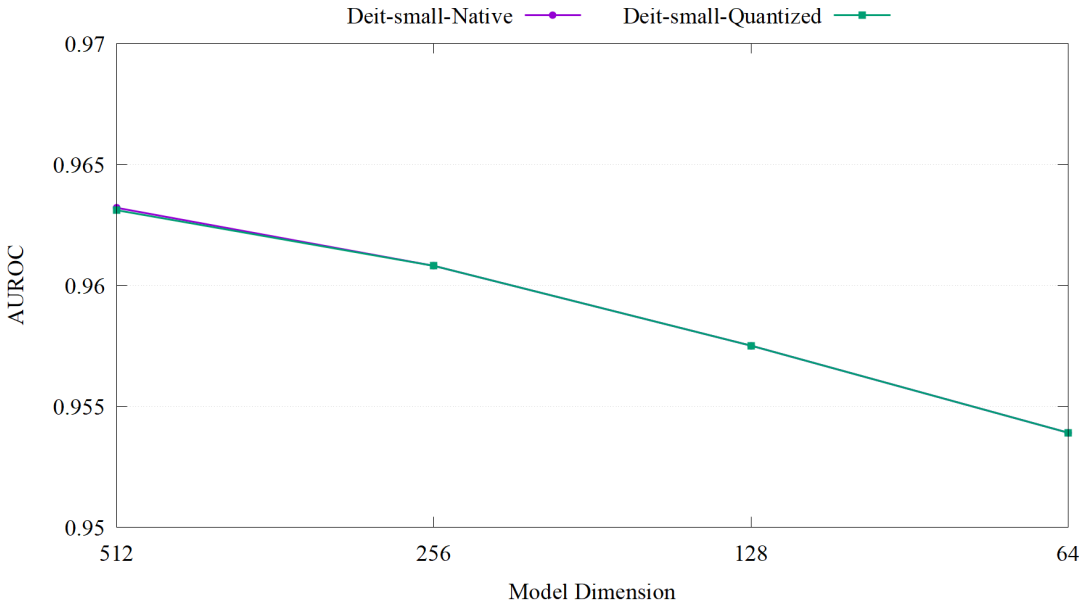

FIGURE 6.17: AUROC Difference Between DeiT-Small Native and Quantized

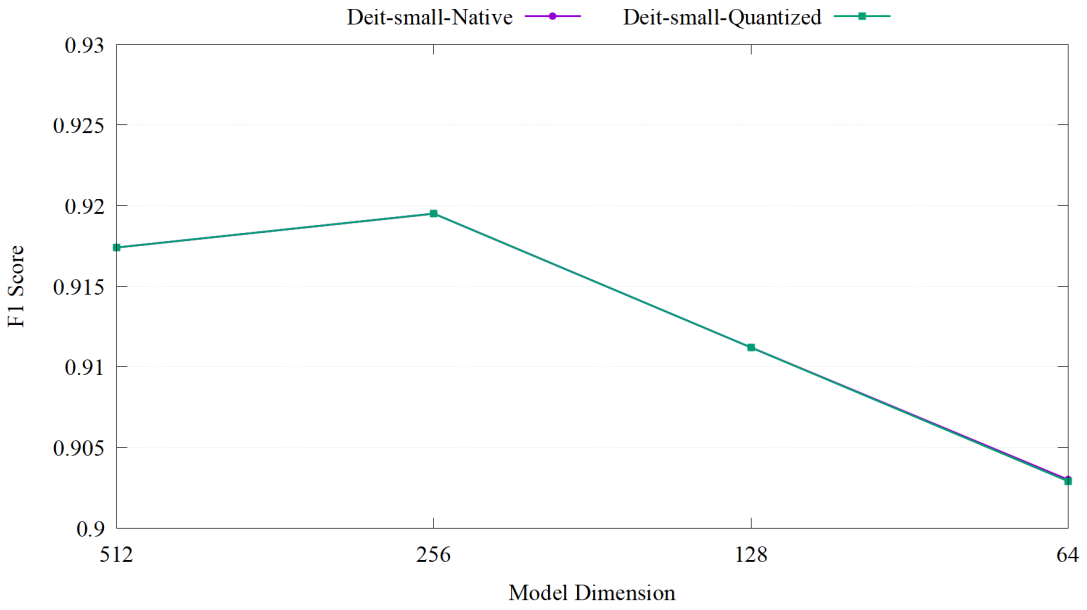

FIGURE 6.18: F1-score Difference Between DeiT-Small Native and Quantized

ability to correctly order samples from most similar to least similar—is largely preserved under dimensionality reduction. However, the separability between positive and negative samples decreases, resulting in greater overlap in their similarity score distributions. Consequently, threshold-based classification performance (Accuracy and F1-score) is degraded.

In the comparison of quantization results, the quantization–dequantization process shows a negligible impact on most evaluation metrics. For Accuracy (6.11, 6.12), F1-score (6.16, 6.18), and AUROC (6.15, 6.17), the performance before and after quantization

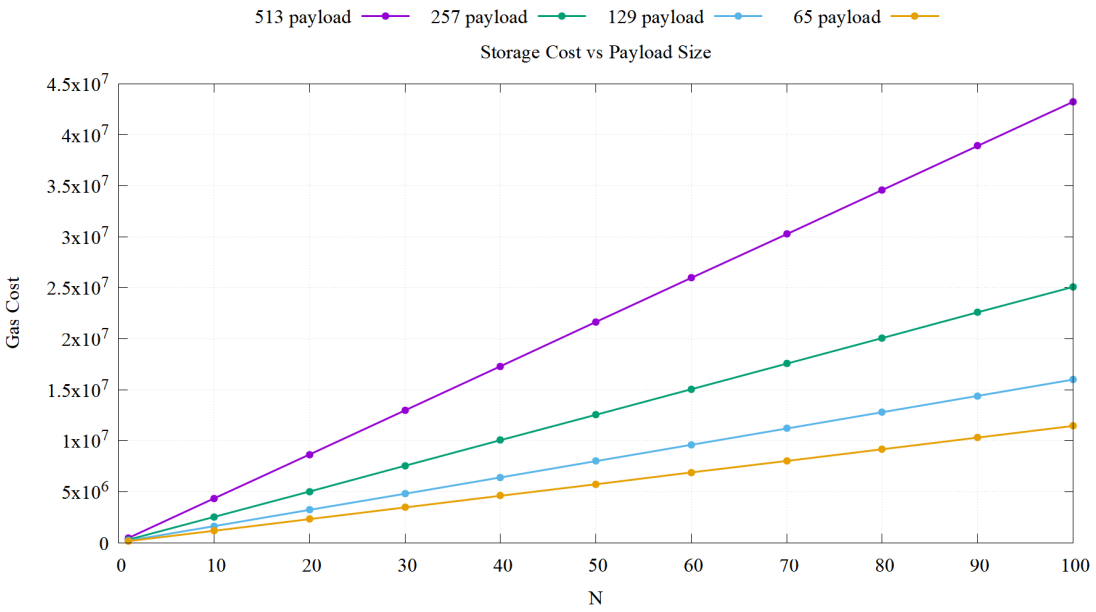

FIGURE 6.19: Storage Cost

is nearly identical, even when reported to four decimal places. A minor deviation is observed in the AP metric for the 128-dimensional ResNet50 model (6.13), where a small gap exists between native and quantized outputs; however, this difference is negligible. In contrast, DeiT-Small does not exhibit any noticeable degradation in AP (6.14).

These results indicate that the proposed quantization method introduces only a controllable, minimal loss of precision on the current dataset. Meanwhile, it significantly reduces the storage requirements of the embedding representations without compromising model performance, making it well-suited for resource-constrained environments such as on-chain storage.

6.7 Blockchain Storage/Update Cost and Read Performance

This study uses Ganache to simulate a blockchain. The storage gas consumption method was tested by storing different amounts of payload in a single batch, and the update method was tested by updating different amounts of payload in a single batch.

6.7.1 Storage Cost

Figure 6.19 shows the relationship between storage cost and different lengths of payload, where N is the number of payloads stored at one time. The results show that storage costs increase linearly with the amount of data stored. Longer payloads have higher

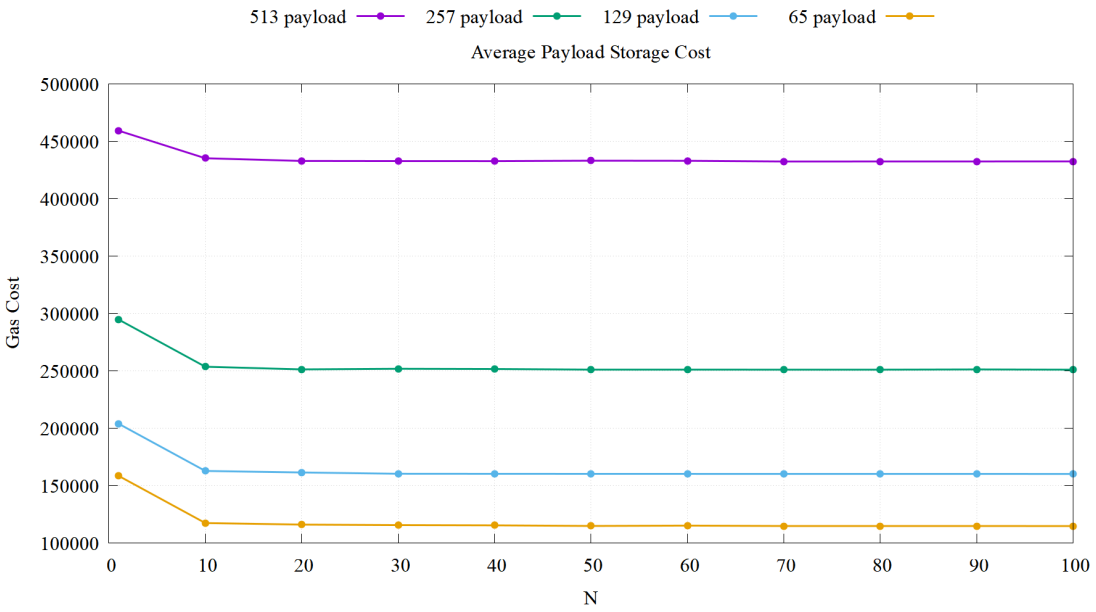

FIGURE 6.20: Average Storage Cost

storage costs. When storing 100 images at a time, a 513-byte payload will consume 43,233,242 gas, a 257-byte payload will consume 25,081,082 gas, a 129-byte payload will consume 15,995,588 gas, and a 65-byte payload will consume 11,452,805 gas. Figure 6.20 shows the average store gas cost per payload. The results show that the storage cost of a single payload is highest at any length. When storing 10 payloads at once, the storage cost decreases and stabilizes. However, as N increases, the average cost decreases slightly. Ultimately, when storing 100 payloads at once, the average cost is 432,332 gas for a 513-byte payload, 250,810 gas for a 257-byte payload, 159,955 gas for a 129-byte payload, and 114,528 gas for a 65-byte payload.

6.7.2 Update Gas Consumption

Figure 6.21 shows the relationship between update cost and different lengths of payload, when N is the number of update payloads at one time. The results show that larger payloads not only cost more to store but also consume more gas for updates. However, the updated gas consumption of all payloads is significantly lower than their storage gas consumption. When N is 100, the total update cost is 10,133,748 gas for a 513-byte payload, 5,669,264 gas for a 257-byte payload, 3,454,700 gas for a 129-byte payload, and 2,327,816 gas for a 65-byte payload. Figure 6.22 shows the average update gas cost per payload. The results show that the upload cost of a single payload is the highest. When updating 10 payloads at once, the storage cost decreases and stabilizes. As N increases, the average cost also decreases slightly. When N is 100, the average costs are 101,337

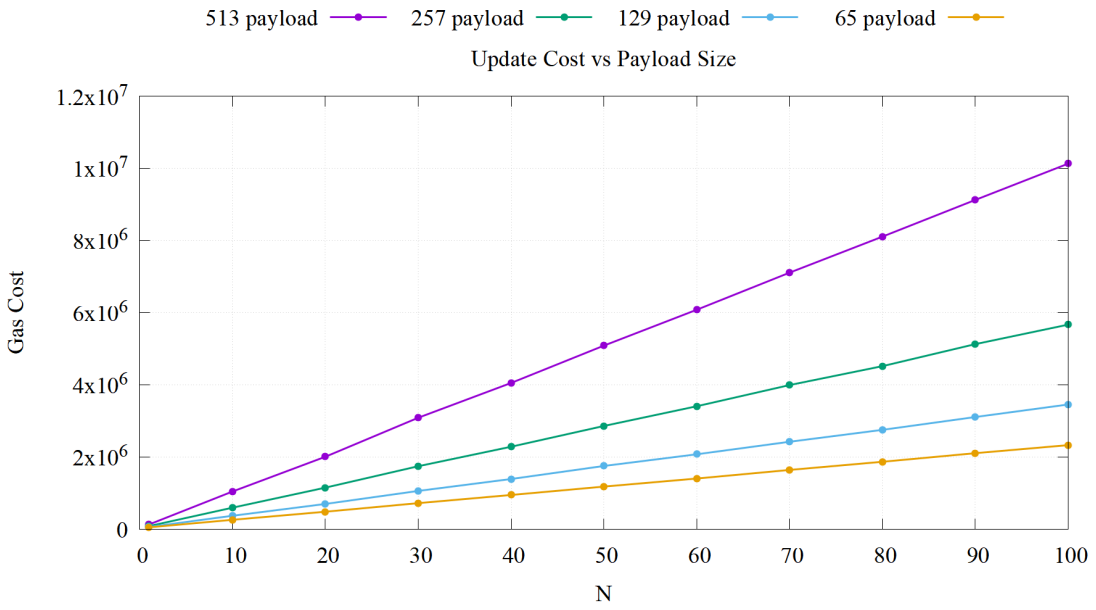

FIGURE 6.21: Update Cost

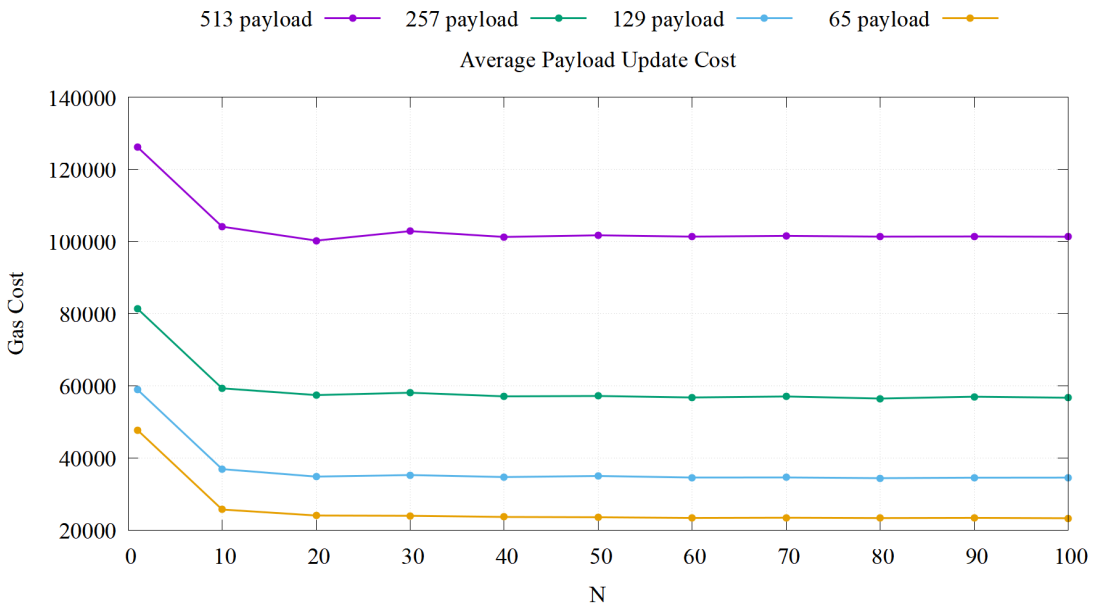

FIGURE 6.22: Average Update Cost

gas for a 513-byte payload, 56,692 gas for a 257-byte payload, 34,547 gas for a 129-byte payload, and 23,278 gas for a 65-byte payload.

6.7.3 Blockchain Read time

Figure 6.23 shows the total read time for reading a number of N payloads. The results show a linear relationship between read duration and the number of reads. Furthermore,

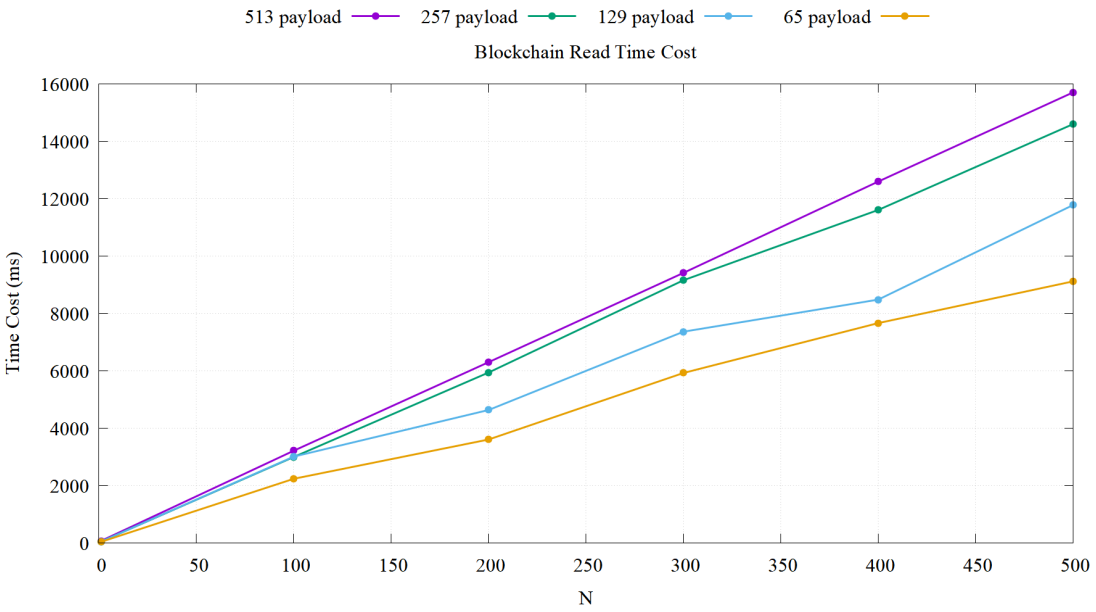

FIGURE 6.23: Time Cost to Read Payload

longer payloads also slow down read time, but not significantly. With $N = 500$, a 513-byte payload took 15700 ms to read, a 257-byte payload took 14594 ms, a 129-byte payload took 11773 ms, and a 65-byte payload took 9112 ms.

A comparison of on-chain gas consumption after quantizing the model outputs reveals a clear positive linear relationship between the total gas cost and the amount of data uploaded. In addition, the size of each data point is strongly correlated with gas consumption; larger payloads incur higher gas fees (6.21).

Furthermore, each smart contract invocation introduces a fixed overhead cost, regardless of the amount of data being uploaded. As shown in Figure 6.22, uploading a single data point results in the highest gas cost per data point due to this overhead. As the number of data points per transaction increases, the average gas consumption per data point decreases because the fixed cost of contract invocation is amortized across more data points. This effect is most pronounced when increasing the batch size from 1 to 10 data points, after which the reduction in average cost gradually stabilizes.

However, the number of data points that can be uploaded in a single transaction is constrained by the block gas limit. Under the same gas limit, smaller payloads (e.g., 65-byte embeddings) allow more data points to be stored per transaction, whereas larger payloads (e.g., 513-byte embeddings) incur higher per-item gas costs, thereby limiting the number of data points per batch upload.

To provide an intuitive understanding of on-chain storage costs, gas consumption is converted into an estimated monetary value. Assuming a gas price of 20 Gwei and an

ETH price of 3000 USD, the cost of storing a single 513-byte payload is approximately 27.55 USD, while a 65-byte payload costs approximately 9.50 USD.

When storing 100 data points in a batch, the cost per data point decreases to approximately 25.94 USD for 513-byte payloads and 6.87 USD for 65-byte payloads. This demonstrates that reducing the payload size significantly lowers on-chain storage costs, achieving up to a 73% reduction when comparing 513-byte and 65-byte embeddings.

Additionally, batch storing further reduces the average cost per data point by amortizing the fixed gas overhead of smart contract invocation. This effect is more pronounced for smaller payloads, indicating that combining dimensionality reduction with batch storage provides substantial improvements in cost efficiency.

It should be noted that these values are only illustrative, as gas prices and ETH market values are highly variable. Therefore, all primary analyses in this study are conducted in terms of gas consumption rather than monetary cost.

The testing of the blockchain also included testing the updating of existing payloads on the chain. Results showed that updating the payload consumed significantly less gas than storing the payload (6.21, 6.22).

While reducing the payload size significantly lowers on-chain storage cost, its impact on retrieval latency is comparatively limited. The blockchain read latency increases approximately linearly with the total number of stored payloads(6.23). While the size of individual payloads has some impact on the read time, this effect is relatively limited compared to the influence of the dataset size. This suggests that the overall retrieval latency is dominated not only by data transfer but also by fixed overheads such as smart contract execution and RPC communication.

6.8 Matching Simulation

6.8.1 FPR rate from simulation

Figure 6.24 shows the average similar count of 100 new images versus 2000 unique image payloads between two models. Since all the images are considered unique, any similar count will be seen as a false positive. The results show that the best-performing model in the simulation is DeiT-Small with a 512-dimensional output; its average similar count is 7.96, corresponding to a false-positive rate of 0.398%. The average similar count for ResNet50 with a 512-dimensional output is 23.87, corresponding to a false-positive rate of 1.193%. When models have a lower-dimensional output, ResNet-50 outperforms

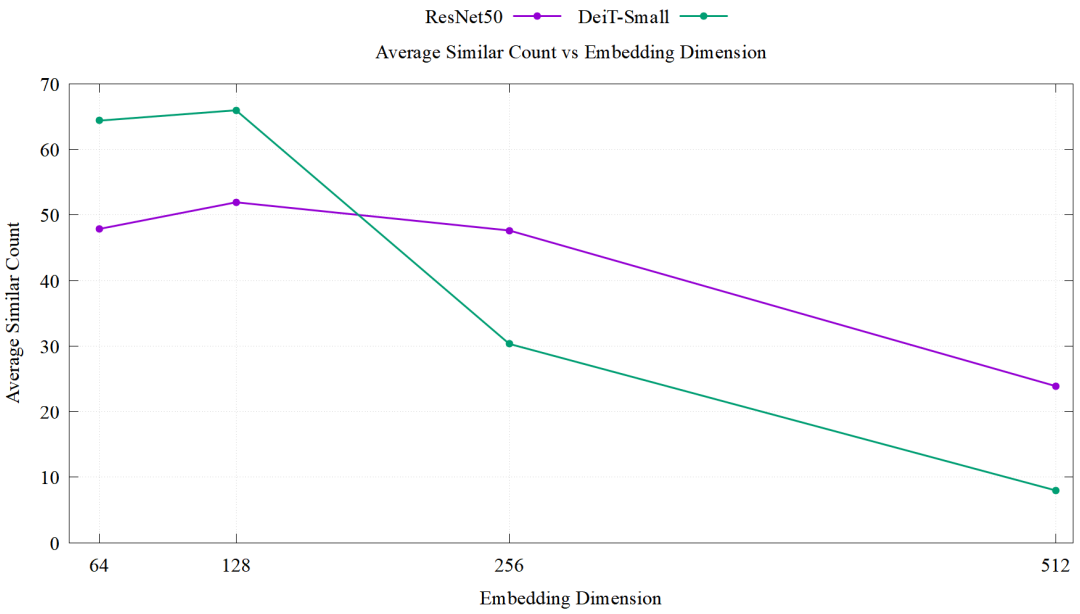

FIGURE 6.24: Average Similar Count

DeiT-Small, and it outperforms DeiT when the dimensionality is reduced to 128 or 64. The worst performance of both models occurred in the 128-dimensional output, with ResNet50 achieving an average similar count of 51.92 (2.596%) and DeiT-Small achieving 65.96 (3.298%).

In the simulation experiments, the impact of embedding dimensionality on model performance follows a trend consistent with previous evaluations. Both models achieve their best performance under the 512-dimensional output configuration, while the average number of false positives increases as the embedding dimension decreases.

However, unlike earlier results, DeiT-Small outperforms ResNet50 under high-dimensional configurations, but underperforms when the embedding dimension is reduced. As illustrated in Figure 6.24, the performance curve of ResNet50 is relatively smooth, whereas DeiT-Small exhibits a steeper trend, suggesting that DeiT-Small is more sensitive to changes in embedding dimensionality.

Notably, DeiT-Small demonstrates superior performance under the 512-dimensional configuration. In the 100 vs. 2000 matching scenario, it achieves an average of 7.96 false positives, corresponding to a false-positive rate of approximately 0.398%. In comparison, ResNet50 produces 23.87 false positives, with a false-positive rate of 1.19%. This indicates that, in high-dimensional settings, DeiT-Small achieves greater separability between positive and negative samples and provides stronger suppression of false positives.

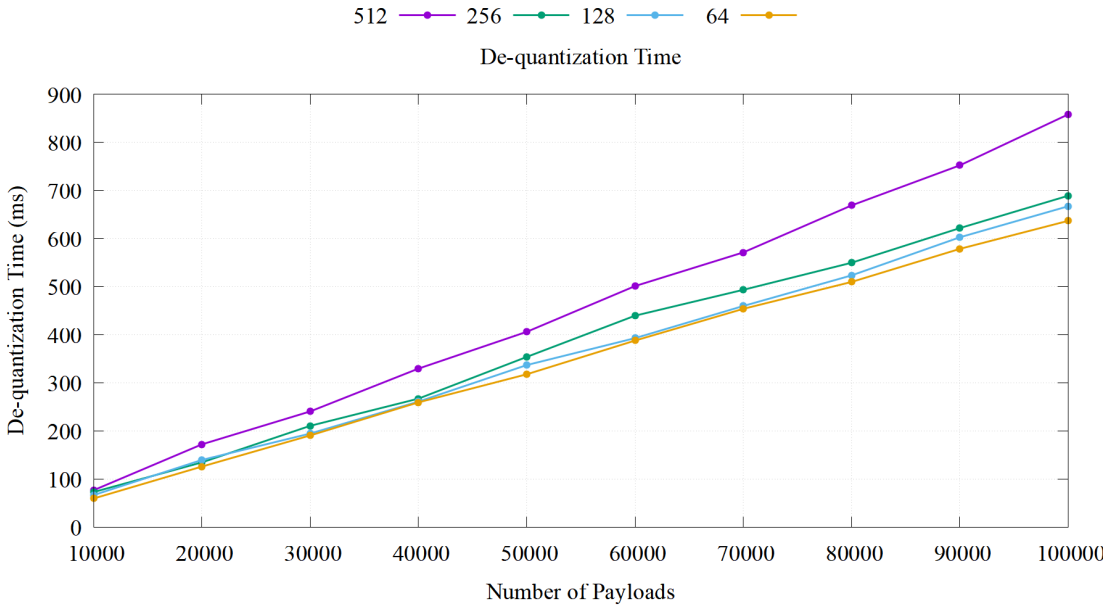

FIGURE 6.25: De-quantization Time Cost

Interestingly, both models exhibit their highest false-positive rates at the 128-dimensional configuration. DeiT-Small records an average of 65.96 false positives (3.298%), while ResNet50 yields 51.92 false positives (2.596%). Despite this increase, the overall performance in this dimension remains within an acceptable range, indicating that both models still maintain a reasonable degree of separability.

The inferior performance at 128 dimensions may be attributed to a suboptimal trade-off between feature compression and representation capacity. Compared with the 64-dimensional setting, 128-dimensional embeddings may retain more non-discriminative information, leading to greater overlap between positive and negative samples and, consequently, a higher false-positive rate.

6.8.2 Time Cost on Large-Scale Data

In the large-scale data simulation experiment, the method in the FPR test was still used, and 100 new query images were matched with existing images of different sizes based on similarity. The number of existing images was gradually increased from 10,000 to 100,000 to evaluate the system's performance under different data scales.

6.8.2.1 De-Quantization Time

Figure 6.25 shows the De-quantization time for different numbers and types of payload. This figure shows that the de-quantization time of image embeddings is approximately

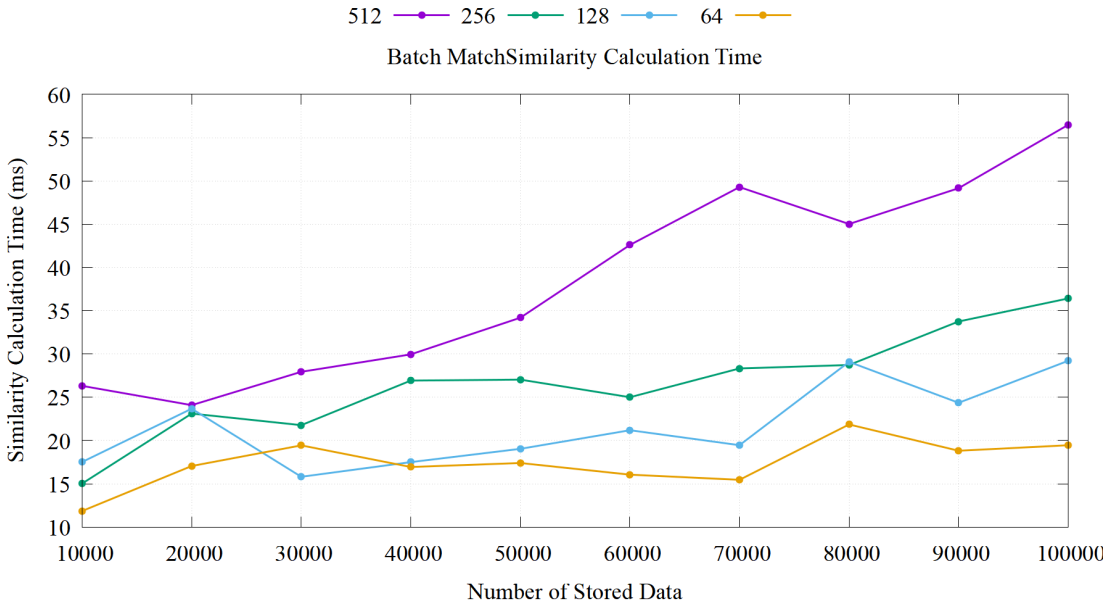

FIGURE 6.26: Batch-Match Similarity Calculation Time

linearly related to the data size. As the number of images to be dequantized increases, the overall processing time also increases linearly.

Furthermore, the dimensionality of the embedding has a relatively limited impact on dequantization time. Although lower-dimensional feature vectors are computationally less expensive, the overall time does not decrease proportionally with decreasing dimensionality. This is because the dequantization process is primarily affected by fixed overhead (such as payload decoding and tensor construction), rather than being entirely determined by dimension-wise computation. However, overall, the dequantization time is acceptable. Dequantizing 100,000 512-dimensional images takes 858.3 ms, while dequantizing 100,000 64-dimensional images takes 636.9 ms.

6.8.2.2 Similarity Calculation Time

Figure 6.26 shows the similarity computation time under different configurations. In the batch-match mode, the computation time increases approximately linearly with the size of the dataset. As the number of stored embeddings grows, the computation time increases accordingly.

Furthermore, embedding dimensionality affects the computational cost, where higher-dimensional embeddings require more processing time. For example, 512-dimensional embeddings consistently take longer than 64-dimensional embeddings. However, overall, the computation cost of similarity calculation in batch-match mode remains very low.

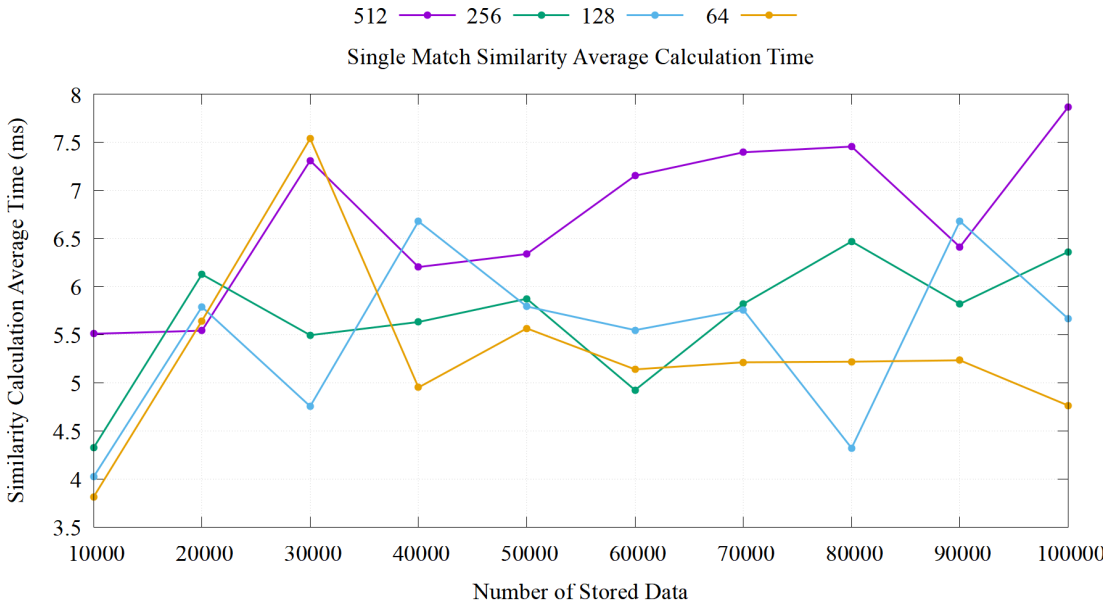

FIGURE 6.27: Single-Match Average Similarity Calculation Time

Even when comparing 100 query images against 100,000 stored embeddings with 512-dimensional representations, the total computation time is only approximately 56.5 ms.

Unlike batch-match, which performs a single global similarity computation, the single-match mode is designed to simulate real-time query scenarios. In this mode, each query image is processed independently and compared against all stored embeddings. This process is repeated until all query images are evaluated.

Therefore, to provide a more intuitive comparison, both the average per-query similarity computation time and the total computation time in the single-match mode are evaluated and compared with the one-time computation cost in the batch-match mode.

Figure 6.27 shows the average similarity computation time for a single query image against datasets of varying sizes. The results indicate that the per-query computation time remains relatively stable, typically ranging between 5 ms and 8 ms. In a single computation, the single-match mode is faster than the batch-match mode because it operates on a smaller matrix, resulting in lower per-run computational overhead.

However, when considering the total computation time (see Figure 6.28), the single-match mode is significantly slower than the batch-match mode. This is because single-match repeatedly performs similarity calculations for each query image, introducing substantial redundant computation.

When processing the same 100 query images as in the batch-match setting, the total similarity computation time of single-match reaches approximately 500–800 ms, which is significantly higher than the 56.5 ms required by batch-match. This demonstrates

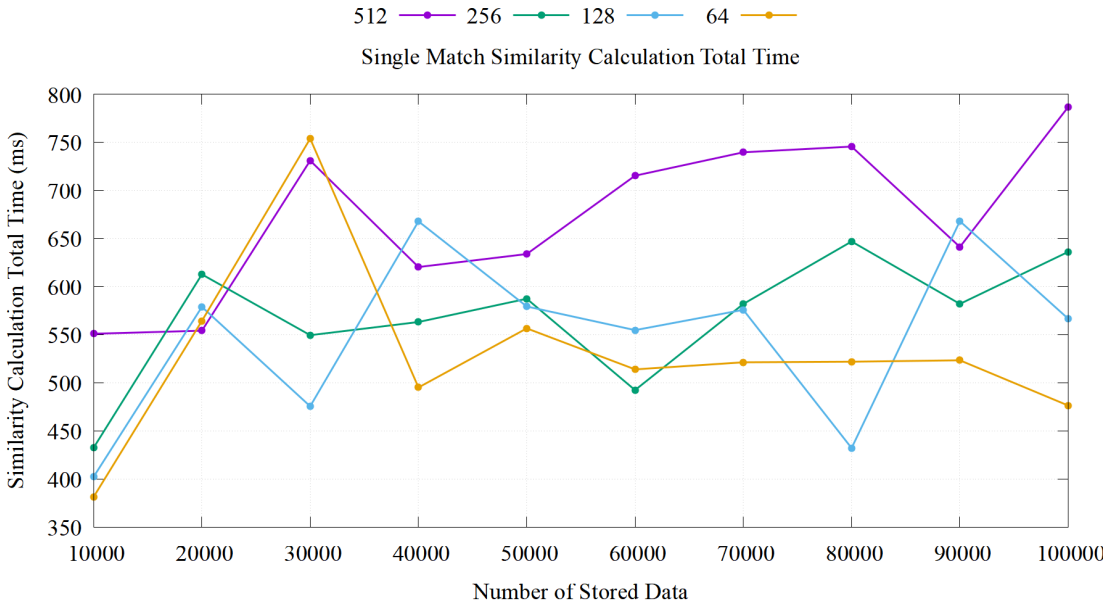

FIGURE 6.28: Single-Match Total Similarity Calculation Time

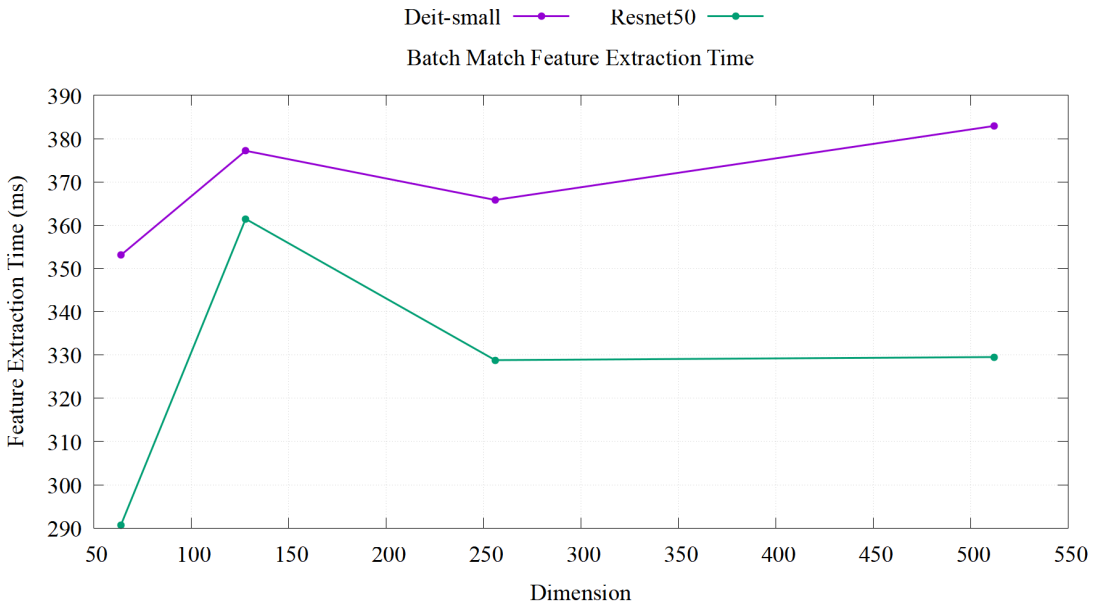

FIGURE 6.29: Batch-Match Feature Extraction Time

that although single-match is efficient for individual queries, the repeated computation makes it inefficient for large-scale scenarios, whereas batch-match achieves significantly better overall performance by avoiding redundant calculations.

6.8.3 Feature Extraction time

Figure 6.29 shows the time taken for feature extraction of images in Batch Match mode. These images were loaded into the existing database, and features were extracted

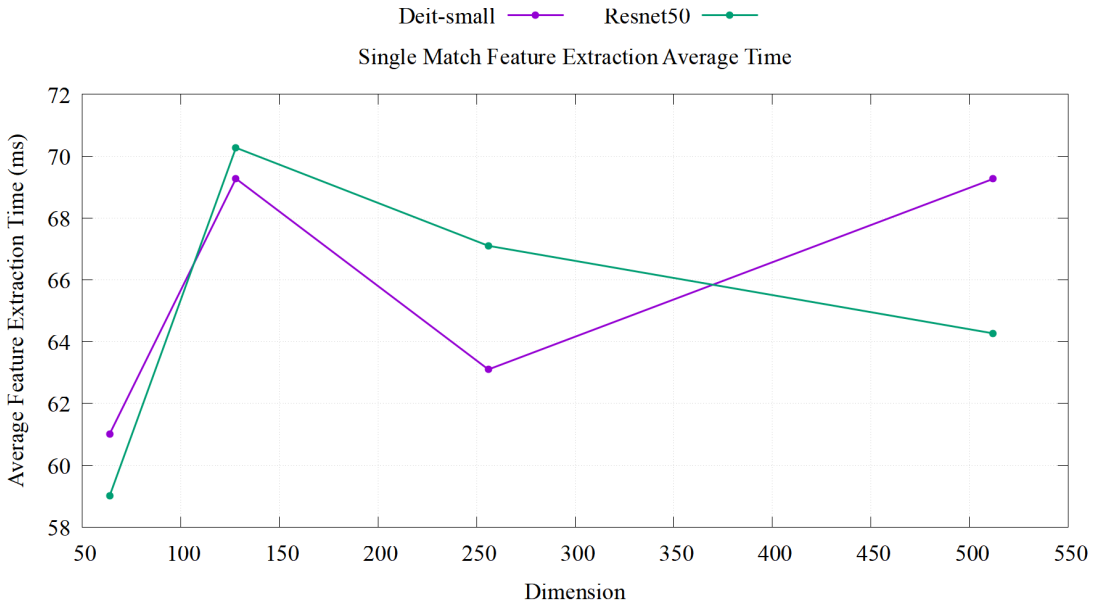

FIGURE 6.30: Single-Match Feature Extraction Average Time

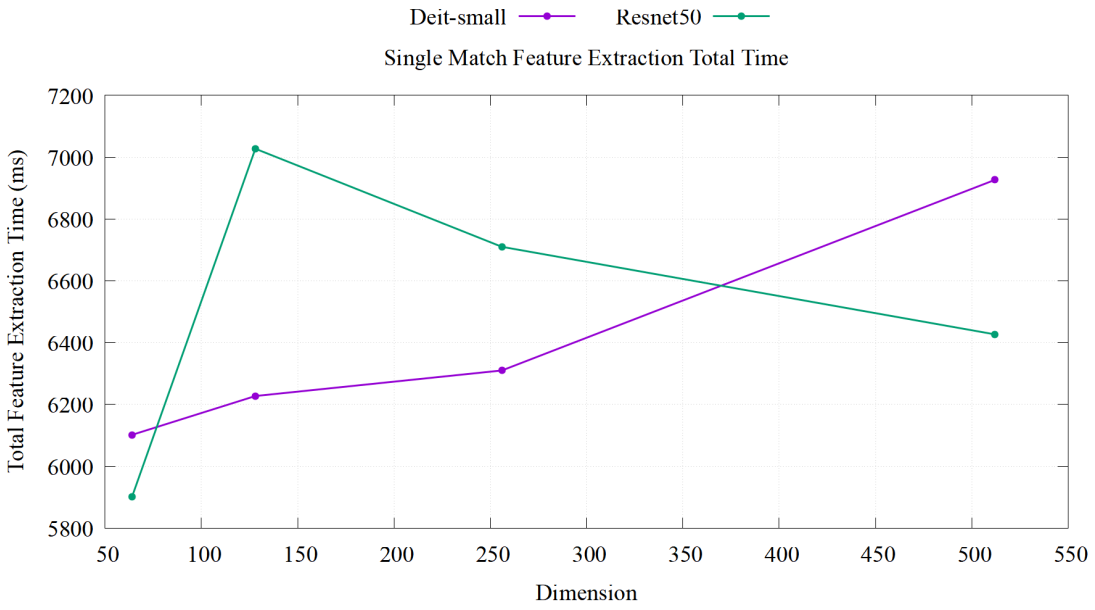

FIGURE 6.31: Single-Match Feature Extraction Total Time

in batches. The execution times of ResNet50 and DeiT-Small are similar, both around 300ms. Figure 6.31 shows the average time the model takes to extract features from a single image in Single-Match mode. ResNet50 and DeiT-Small perform similarly, each requiring approximately 60-70 ms. Furthermore, the model extracts features in a single pass, fewer than in batch mode, because it extracts features from only one image at a time. Figure 6.31 shows the total time taken to extract features from 100 images in single match mode. The time is significantly longer than in batch-match mode, reaching 600-700 ms.

The performance of the feature extraction and similarity calculation stages exhibits similar patterns. Batch-match, by processing more data at once, can make fuller use of hardware resources and reduce the overhead of repeated data loading and scheduling. In contrast, although single-match takes less time in a single feature extraction, it requires repeated loading, scheduling, and other system-related operations, resulting in higher overall cumulative overhead and a significantly longer total time than batch-match.

In conclusion, from a system performance perspective, batch matching is significantly more computationally efficient than single matching. It reduces redundant computation by processing multiple query tasks simultaneously, thereby significantly reducing overall computation time.

However, from a user experience perspective, batch-matching is not always the optimal choice. Its implementation typically relies on triggering mechanisms, such as building an upload pool in the system and setting triggering conditions (e.g., the number of data in the pool or time intervals). Only when these conditions are met will the system send the accumulated data in batches to the processing pipeline for similarity calculation.

Therefore, although batch-matching is more computationally efficient, its non-real-time processing introduces additional waiting time, impacting the user's immediate response experience. This reflects a typical trade-off in system design: real-time processing (single-matching) provides faster responses but incurs higher computational overhead, whereas batch processing (batch-matching) reduces resource consumption by improving computational efficiency but at the cost of increased user waiting time.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This research investigated the feasibility of combining deep visual models with blockchain for artwork similarity detection and ownership verification. First, convolution-based and transformer-based Siamese models were evaluated under different loss functions and distance metrics. Then, the output embeddings were compressed and quantized. Finally, the feasibility of storing quantized image features directly on-chain was analyzed, and a real-world simulation was performed.

The experimental results show that the ResNet50 model trained with the NT-Xent loss and Euclidean distance achieved the best overall performance among the evaluated settings, while the transformer-based DeiT-Small delivered competitive results and exhibited strong generalization. In general, NT-Xent loss consistently outperformed BCE loss across multiple evaluation metrics, including accuracy, AP, F1-score, and AUROC. This is because NT-Xent directly optimizes the relative structure of the embedding space, enabling the model to learn more discriminative feature representations for image-similarity tasks. By contrast, BCE formulates the problem as binary classification, which is less effective in preserving ranking quality and fine-grained similarity relationships. Moreover, within the NT-Xent framework, Euclidean and cosine distances showed stable performance and generally outperformed Mahalanobis distance.

The dimensionality reduction experiments further indicate that reducing the output dimension does not significantly affect ranking performance or pairwise comparison accuracy. However, the simulation results reveal that lower-dimensional embeddings tend to yield higher false-positive rates in large-scale matching scenarios. This suggests that while lower-dimensional representations are more storage-efficient, they may sacrifice robustness in practical deployment. In comparison, feature quantization introduced little

to no noticeable accuracy loss, demonstrating that it is a practical solution for reducing storage overhead without substantially degrading model performance.

From the blockchain perspective, the results confirm that directly storing quantized image features on-chain is feasible. Even for the largest tested payload size of 513 bytes, the on-chain storage cost remained within a manageable range. In the simulated deployment scenario, DeiT-Small with higher-dimensional outputs demonstrated stronger generalization, suggesting that the proposed system could reduce manual effort in artwork ownership verification and plagiarism screening. Overall, this study demonstrates that integrating visual similarity models with blockchain is a viable approach for building a secure and practical artwork verification system.

7.2 Future Work

There are several directions for future work. First, the proposed system can be evaluated on larger, more diverse art datasets to further examine its robustness and generalization in more realistic scenarios. Second, more advanced vision models may be explored to improve similarity detection performance while maintaining compact feature representations suitable for blockchain storage. Third, since lower-dimensional embeddings were observed to increase the false-positive rate in the final simulation, future research may focus on developing more effective projection and compression strategies that reduce storage costs without sacrificing robustness. Finally, the current system could be extended into a more comprehensive artwork verification platform by incorporating real-world blockchain deployment, ownership transfer mechanisms, and improved human-in-the-loop review processes.

Appendix A

Output example

A.1 scale clip table

This table shows the relationship between the model dimensions (ResNet50 and Diet-small) and the scale factor. "No" indicates that the clip limit was not triggered after scaling multiplied by 2^k, while "yes" indicates that it was. The model is tested using the test dataset.

TABLE A.1: Quantization Feasibility under Different k Values

Dimension	**k=15**	**k=16**	**k=17**	**k=18**
64	no	yes	yes	yes
128	no	no	yes	yes
256	no	no	yes	yes
512	no	no	no	yes

A.2 Payload Example

Here is an example of a 128-dimensional payload:

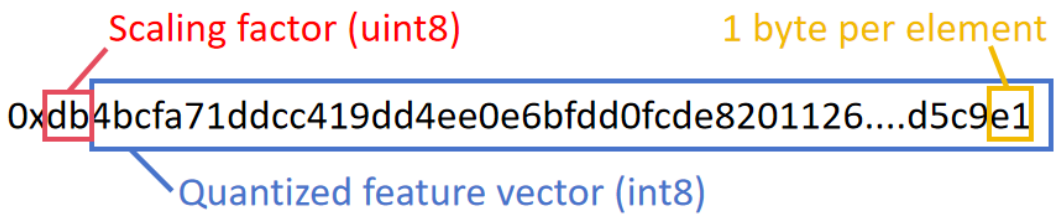

FIGURE A.1: Example of a 128-dimensional payload structure

Bibliography

[1] Nicolás Serrano and Alejandro Bellogín. Siamese neural networks in recommendation. *Neural Computing and Applications*, 35:1–13, 05 2023. doi: 10.1007/s00521-023-08610-0.

[2] Luisa Verdoliva. Media forensics and deepfakes: An overview. *IEEE Journal of Selected Topics in Signal Processing*, 14(5):910–932, 2020. doi: 10.1109/JSTSP.2020.3002101.

[3] Wei Chen, Yu Liu, Weiping Wang, Erwin M. Bakker, Theodoros Georgiou, Paul Fieguth, Li Liu, and Michael S. Lew. Deep learning for instance retrieval: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7270–7292, 2023. doi: 10.1109/TPAMI.2022.3218591.

[4] Yao Xu, Min Xia, Kai Hu, Siyi Zhou, and Liguo Weng. Style transfer review: Traditional machine learning to deep learning. *Information*, 16(2):157, 2025.

[5] Maria Trigka and Elias Dritsas. A comprehensive survey of deep learning approaches in image processing. *Sensors*, 25(2):531, 2025.

[6] Changhao Zhu, Junzhe Li, Ziyue Zhong, Cong Yue, and Meihui Zhang. A survey on the integration of blockchains and databases. *Data Science and Engineering*, 8:196 – 219, 2023. URL https://api.semanticscholar.org/CorpusID:258327026.

[7] Mohammad Jabed Morshed Chowdhury, Alan Colman, Muhammad Ashad Kabir, Jun Han, and Paul Sarda. Blockchain versus database: A critical analysis. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1348–1353, 2018. doi: 10.1109/TrustCom/BigDataSE.2018.00186.

[8] Jiayi Ma, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. Image matching from handcrafted to deep features: A survey. *International Journal of Computer Vision*, 129(1):23–79, 2021.

[9] Pin Wang, En Fan, and Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern recognition letters*, 141:61–67, 2021.

[10] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[12] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005. doi: 10.1109/CVPR.2005.202.

[13] Taehee Jeong. 4bit-quantization in vector-embedding for rag, 2025. URL https://arxiv.org/abs/2501.10534.

[14] Jikuan Xu, Jiamin Zhang, and Junhan Wang. Digital image copyright protection and management approach—based on artificial intelligence and blockchain technology. *Journal of Theoretical and Applied Electronic Commerce Research*, 20(2), 2025. ISSN 0718-1876. doi: 10.3390/jtaer20020076. URL https://www.mdpi.com/0718-1876/20/2/76.

[15] Alexander Savelyev. Copyright in the blockchain era: Promises and challenges. *Computer Law Security Review*, 34(3):550–561, 2018. ISSN 2212-473X. doi: https://doi.org/10.1016/j.clsr.2017.11.008. URL https://www.sciencedirect.com/science/article/pii/S0267364917303783.

[16] Elmahdi Bentafat, M Mazhar Rathore, and Spiridon Bakiras. Towards real-time privacy-preserving video surveillance. *Computer Communications*, 180:97–108, 2021.

[17] Elmahdi Bentafat, M Mazhar Rathore, and Spiridon Bakiras. A practical system for privacy-preserving video surveillance. In *International Conference on Applied Cryptography and Network Security*, pages 21–39. Springer, 2020.

[18] Muhammad Mazhar Ullah Rathore, Elmahdi Bentafat, and Spiridon Bakiras. Towards a scalable and privacy-preserving audio surveillance system. *IEEE Transactions on Audio, Speech and Language Processing*, 33:186–198, 2024.

[19] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram

Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[20] Ibomoiye Domor Mienye, Theo G Swart, George Obaido, Matt Jordan, and Philip Ilono. Deep convolutional neural networks in medical image analysis: A review. *Information*, 16(3):195, 2025.

[21] Aimina Ali Eli and Abida Ali. Deep learning applications in medical image analysis: Advancements, challenges, and future directions. *arXiv preprint arXiv:2410.14131*, 2024.

[22] Shivalila Hangaragi, Tripty Singh, and Neelima N. Face detection and recognition using face mesh and deep neural network. *Procedia Computer Science*, 218:741–749, 2023. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2023.01.054. URL https://www.sciencedirect.com/science/article/pii/S1877050923000546. International Conference on Machine Learning and Data Engineering.

[23] P Jonathon Phillips and David White. The state of modelling face processing in humans with deep learning. *British Journal of Psychology*, 2025.

[24] Yueyan Bian, Jin Li, Chuyang Ye, Xiuqin Jia, and Qi Yang. Artificial intelligence in medical imaging: From task-specific models to large-scale foundation models. *Chinese Medical Journal*, 138(06):651–663, 2025.

[25] R Angeline, Abhiram Suji Nambiar, K Samuel Jacinth, P Alan Christo, and Princeton Antony Joseph. Ai art authenticator: Deep learning image classification. In *International Conference on Smart Computing and Communication*, pages 107–119. Springer, 2024.

[26] Anas Laamouri and Nawal Sael. A comparative study on image recommendation using siamese neural network architecture. In *2025 5th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, pages 1–7, 2025. doi: 10.1109/IRASET64571.2025.11008010.

[27] C. Ranjeeth Kumar, Saranya N, M. Priyadharshini, Derrick Gilchrist E, and Kaleel Rahman M. Face recognition using cnn and siamese network. *Measurement: Sensors*, 27:100800, 2023. ISSN 2665-9174. doi: https://doi.org/10.1016/j.measen.2023.100800. URL https://www.sciencedirect.com/science/article/pii/S2665917423001368.

[28] Alexandre C. Moreaux and Mihai P. Mitrea. Blockchain asset lifecycle management for visual content tracking. *IEEE Access*, 11:100518–100539, 2023. doi: 10.1109/ACCESS.2023.3311635.

[29] Huang Zhou, Yongqiang Chen, Jiangchen Zhou, and Wenxuan Wang. Blockchain implementation for visual art copyright protection. In *2024 IEEE 24th International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*, pages 232–239, 2024. doi: 10.1109/QRS-C63300.2024.00039.

[30] Wei Ren Tan, Chee Seng Chan, Hernan E Aguirre, and Kiyoshi Tanaka. Improved artgan for conditional synthesis of natural image and artwork. *IEEE Transactions on Image Processing*, 28(1):394–409, 2018.

[31] Hossein Aboutalebi, Dayou Mao, Rongqi Fan, Carol Xu, Chris He, and Alexander Wong. Deepfakeart challenge: A benchmark dataset for generative ai art forgery and data poisoning detection. *arXiv preprint arXiv:2306.01272*, 2023.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[34] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[35] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[36] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check, 2020. URL https://arxiv.org/abs/2003.08505.

[37] Usha Ruby, Vamsidhar Yendapalli, et al. Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10), 2020.

[38] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

[39] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d89a66c7c80a29b1bdbab0f2a1a94af8-Paper.pdf.

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